BILKENT UNIVERSITY

CS 319

Object Oriented Software Engineering

ANALYSIS REPORT

10 March 2019

QUADRILLION

by The Group

Ana Peçini

Krisela Skënderi

Muhammad Hammad Malik

Teymur Bakhishli

Ünsal Öztürk

This report is submitted to GitHub repository in partial fulfilment of the requirements of CS319 course project.

Table of Contents

1. Introduction 3
2.Overview 3
3.Functional Requirements 6
4.Non-functional Requirements 8
5.Pseudo Requirements 9
6.System Requirements 10
7.System models 10
7.1 Use case model 10
7.2 Object and class model 16
7.3 Dynamic models 20
7.3.1 Sequence Diagrams 21
7.3.2 State Diagrams 27
7.3.3 Activity Diagrams 31
7.4 User interface 34
8. Conclusion 39
9. References 40

1. Introduction

Quadrillion is a one-player board game designed by SmartGames, a company that develops IQ puzzles with innovative game mechanics [1] that are designed to stimulate cognitive skills such as concentration, problem solving and logic [2]. What makes this board game stands out from other ones of the same category is that even with the constraint of using the same grid parts and puzzle pieces over and over again, there exist as many as 638 976 different start positions. [3]

The board of the game consists of 4 black and white magnetic grid pieces that can be rotated or flipped and then arranged together in any combination. The game includes 12 different puzzle pieces [2]. The aim of Quadrillion is to arrange the 12 different puzzle pieces in such a way that they fit perfectly in the grid without leaving any empty spaces and without overlapping any of the blocked positions on the grid, which are fixed positions indicated by the opposite colour to that of the grid. The original game comes with a booklet of grid arrangements that are sorted according to their difficulty level. Each arrangement suggested in the booklet can have at most one solution, but the player can come up with his own grid designs that, if following a few simple guidelines and specifications, are guaranteed to have at least one solution but may have up to thousands [3].

The Group's digitalized Quadrillion, called Gazillion, is a roguelike version that enhances the dynamics of the original game by adding new immersive features, while preserving the core mechanics of the board game. The features include:

- multiple modes with different goals and interactive maps
- player inventory, health and budget
- time constraints for different levels
- shop entity where different themes and power ups can be purchased, etc.

2. Overview

2.1 General Information

We propose a system that digitalizes Quadrillion, called Gazillion. The system will simulate the core aspects of Quadrillion gameplay, as well as extended functionality based on top of the game. The game will be a desktop based Java application.

There will be three game modes that revolve around providing a different experience of Quadrillion to the player. These modes are:

- Treasure Mode:

This mode is a “game within a game”. The user will be able to see a map, in which there are randomly generated games of Quadrillion. The levels are arranged in a 2- dimensional matrix, and the user can click on an unlocked level to play a round of Quadrillion. Once the level is completed within the time constraint, the levels that are connected to the completed level are unlocked, and the user may play those levels as well. The purpose of this game mode is to collect all Quadrillion pieces by completing levels on certain nodes on the map, and to solve a final puzzle of Quadrillion once all the pieces have been gathered.

- Ladder Mode:

The ladder mode is about completing as many levels as possible under a time constraint. The player will have twenty minutes to solve as many random Quadrillion puzzles as they can until the timer runs out. For each completed level, the player will be awarded coins, with which the player may buy power-ups.

- Level Mode:

In the Levels mode, the player can choose among pre-made and “difficulty labelled” levels and play them without a time constraint. The levels are initially locked and the player should play levels one by one to unlock harder levels. An initial completion of a level provides the player coins. The “time” and “health” mechanics are absent in this game mode. The purpose of this mode is to simply complete the round of Quadrillion, without any time or failure concerns. The player may then spend these coins at the shop for power-ups.

2.2 Game Mode Specifications

2.2.1 Treasure Mode

- Initially only the levels in the 4 corners of the map are available for the Player to play.
- The rest of the levels are locked and the player unlocks new levels by playing a level which is connected to the completed level.
- Initially, the player will have five health points.
- The player will have a time constraint to solve individual levels, and if is unable to, will lose health points.

- The player can gather power-ups by successfully completing levels or purchasing them in the shop.
- He may increase both the time remaining in a level and the number of health points by using power-ups.
- The treasure mode allows the player to use three power-ups.
- The time power-up gives player extra time (1 minute) during a Quadrillion game.
- The hint power up gives the player a hint regarding the locations of Quadrillion pieces in the map.
- The health power-up restores one health point to the player.
- The player may use the health and the hint power-up in the map screen, and the time power-up may be used during the game of Quadrillion.
- If the user runs out of health at any point the Quadrillion game is over and the Player loses. He has to restart the game from the beginning.
- Once the player has collected all the pieces, the player plays a final round of Quadrillion, which has a time constraint. If the player is able to win the final round of Quadrillion, the player is considered to have completed the treasure mode.

2.2.2 Ladder Mode

- The player is greeted by an “intermission” message upon entering the ladder mode, displaying the amount of time the player has for the next round and the number of rounds the player has won thus far.
- If the player is able to solve the Quadrillion level before the timer runs out, the player is again displayed the “intermission” message for fifteen seconds, which shows the number of levels completed thus far and the remaining time. This goes on until the user runs out of time.
- When the player runs out of time, the game of Quadrillion is pre-emptively terminated and the player is displayed for a final time the number of levels the user has completed. The user is awarded coins based on the number of levels completed.
- In this mode the player may only use the time power-up, as the health and hint mechanics are disabled. The player may use as many time power-ups as desired.

2.2.3 Level Mode

- In this mode, the player is able to choose from an array of premade levels.
- Initially, all the levels are locked but the first one.
- The player plays the first level, and unlocks the next one.
- The player is awarded five coins for the completion of every level.
- The levels do not have any form of time or health constraints.

3. Functional Requirements

The game will provide the following functionalities to the Player:

3.1 Playing a round of Quadrillion:

Quadrillion comes in 4 different modes, all of which include the basic gameplay of the traditional board game. The original game grid is composed of 4 4x4 grids arranged in a particular shape with fixed blocker positions. The 12 quadrillion pieces are provided to the Player. The common functionalities with our system are:

- View Quadrillion pieces and the game board with the blockers
- Place Quadrillion pieces on the game board
- Rotate Quadrillion pieces by 90 degrees, clockwise and counter-clockwise.
- Flip Quadrillion pieces (180 degrees)
- View the remaining time for the round
- View the remaining health, if included in the mode
- Use power-ups according to the mode
- Give up on the round / leave the game

Additionally, Player has a power up inventory, a health bar and coins. Initially Player's inventory is initialized with 5 power ups and 5 health units. Depending on the selected mode, a timer will keep track of the elapsed time for the Quadrillion puzzle from the moment when the Player enters the screen until the puzzle is completed. If the Player is not able to finish the round in time he may be punished by losing health points. The Player may use a "time" power up at any point during the round to increase remaining time. When the Player completes a level, he is awarded with virtual currency (coins) which serve to buy power ups, themes, and hints.

3.2 Game modes:

The system will provide the Player with a choice between 3 game modes in the Main Menu: Treasure mode, Ladder mode and Levels mode.

- Levels mode: Player can choose a level among the unlocked ones. The “time” and “health” mechanics are absent in this game mode. Completing one of these levels provides the Player with 5 coins.
- Ladder mode: The Player will be provided randomly generated and unlabelled games (in terms of difficulty) of Quadrillion one after another. The health mechanic is absent from this game mode. The Player may not use any power ups in this mode. The Player will be awarded 10 gold coins after each completed level.
- Treasure mode: The Player can click on an unlocked level on the map to play a round of Quadrillion. Once the level is completed within the time constraint, the levels that are connected to the completed level are unlocked, and the Player may play those levels as well. If the Player is not able to complete the level in time, the Player will have his/her health deducted by one unit health and has to replay. The round to be replayed will be on the same position on the map as before; however, the level is randomly generated. The purpose of this game mode is to collect all Quadrillion pieces (treasures) hidden in the map by completing levels, and to solve a final puzzle of Quadrillion once all the pieces have been gathered. The treasure mode allows the player to use three power-ups. The time power-up gives player extra time (1 minute) during a Quadrillion game. The hint power up gives the player a hint regarding the locations of Quadrillion pieces in the map. The health power-up restores one health point to the player. If the Player runs out of health at any point the Quadrillion game is over and the Player loses. The player may use the health and the hint power-up in the map screen, and the time power-up may be used during the game of Quadrillion.

3.3 Settings:

The system will provide the Player the ability to adjust certain settings related to the game. The Player may access this menu by clicking on the “Settings” button in the Main Menu. The provided functionalities are:

- Change the volume level

- Change the theme
- Mute the sound

3.4 Shop:

The system will provide the Player the possibility of buying power-ups and themes in exchange for coins. The Player may access the shop by clicking on the “Shop” button in the Main Menu. The shop provides the following functionalities:

- Buy “health” power up
- Buy “time” power up
- Buy “hint” power up
- Buy theme

3.5 How to Play:

The system will allow the Player to view the rules of the game and instructions in the form of an image. Player has to click on the “How to Play” button in the Main Menu.

3.6 Credits:

The Player will be able to see the developers, and the people who were involved in the development of the game by clicking on the “credits” button.

3.7 Exit:

The Player will be able to exit the application on demand by clicking on the button in the main menu. The Player is able to access the main menu at any given time.

4. Non-functional Requirements

4.1 Accessibility:

The game should be easy to distribute and download. Given a certain hardware specification it should not take more than 3 minutes to install the game.

4.2 Usability:

- The average Player should be acquainted with the basics of the game and the system in a time interval of no more than 30 minutes with or without reading the game manual. The game should also provide the Player full control over the Quadrillion pieces: the Player should be easily able to rotate, flip, and place pieces, and the controls for these

operations should be conveniently situated, so that the Player can perform these actions in under a second on average.

- The average time constraint for a level is 5 minutes. The game should provide the Player enough time to complete the levels, but not too much, in order to keep the game challenging.
- The game should provide the Player enough resources (i.e.) coins to reduce the challenge posed by the difficulty of the levels. Initially the Player gets awarded 5 coins per level. The amount increases as level difficulty increases.

4.3 Portability:

The system should be easily portable to other platforms, such as OSX and Linux. This will be achieved by developing the system on a platform which runs on all of these platforms.

4.4 Performance:

The system should be responsive to the Player's actions and should run smoothly to maintain the Player experience above a certain standard. These standards are:

- Smooth motion and animations: the frame rate should be at least 40 frames per second, or should be at least 24 frames per second, given a motion blur implementation.
- The system should maintain responsiveness by reducing latency between its logical calculations and raster/graphical operations to less than a fraction of a second. This is the case since some of the modes in the games have time constraints.
- The game should not load, launch, or take any action that will make the Player wait (e.g. generating random levels of Quadrillion) for more than a fixed amount of time. The game modes should take no more than 5 seconds to load after Player has made his choice.

5. Pseudo Requirements

- The system will be fully developed using Java.
- The system will be compiled into an independent .jar executable.
- The system will run on the Java Virtual Machine.
- The system will be independent of internet connectivity to function.

6. System Requirements

- The game must take up no more than 512 MB disk space.
- The game must run on any machine that supports JVM.
- The game must not consume the majority of the resources of a commercially available computer while it is running.

7. System models

In this section we provide the models of our Gazillion game system by means of different UML diagrams: use case, sequence, state and activity; as well as through user interface mock-ups.

7.1 Use case model

Gazillion game consists of one actor (Player), whose interactions with the system are depicted in the Use Case Diagram below. For each action in the diagram, the flow of events and entry/exit conditions are explained in detail in the tables below.

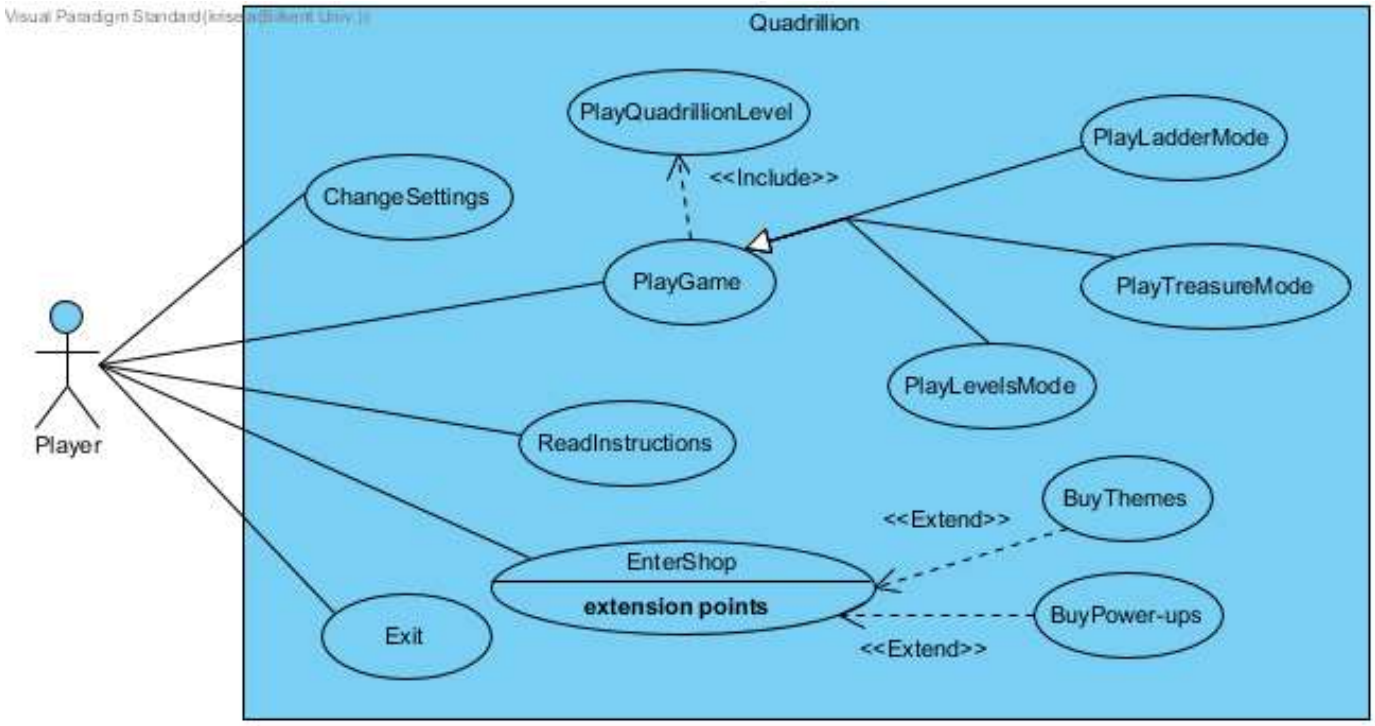

Figure 1. Use Case Diagram for Quadrillion game

USE CASE ReadInstructions	
Participating actors:	Player
Entry conditions:	Player is in the Main Menu and has pressed the Instructions button.
Exit conditions:	Gazillion provided Player with all the necessary information about the game rules, modes, objects and control keys.
Flow of events:	1. Player presses the Instructions button. 2. Game displays a window with the instructions. 3. Player scrolls down until he finds the part he is interested in and reads the instructions. 4. Player clicks on the Back button and returns to Main Menu.
Special requirements:	Player understands the language of the game.

USE CASE ChangeSettings	
Participating actors:	Player
Entry conditions:	Player is in the main menu and has pressed the Change Settings button.
Exit conditions:	• Configuration changes that the Player indicated have been applied. • Player returns to the main menu.
Flow of events:	1. Player presses the ChangeSettings button. 2. Game displays the settings options: Adjust volume, Change theme and Mute sound. A volume slider is and the available themes are also displayed. 3. Following events can happen in any order any number of times: 3.1 Player adjusts volume using the volume slider. 3.2 Player clicks on a theme to activate it. 3.3 Player mutes system sound by clicking on the corresponding button. 4. The game applies player's preferences immediately. 5. Player clicks on Back button to return to the Main Menu.
Special requirements:	Game must apply the changes specified by Player immediately.

USE CASE EnterShop	
Participating actors:	Player
Entry conditions:	• Player has pressed the Shop button in the main menu. • Player has enough coins to make the purchase.
Exit conditions:	• Player has the items he wanted to purchase. • Player can use them during the game in the suitable modes.
Flow of events:	1. Player presses the Shop button. 2. A window pops up and game provides player with 3 possibilities: Buy a hint, Buy a health potion and Buy extra time. *The options are displayed as buttons. Their respective prices are also shown.* 3. Player clicks on a button to make a purchase.

	4. The system checks if Player has enough coins. If so, system updates Player information to include the bought item/power up. 5. The system subtracts the correct amount from player's coins (0 is no purchase is made). 6. Steps 2-4 repeat until Player presses the Back button to go back to the previous state.
Special requirements:	Player's inventory and budget should be immediately updated when a purchase is made.

USE CASE Exit	
Participating actors:	Player
Entry conditions:	• Player wants to quit the game. • Player has pressed the Exit button in the Main Menu.
Exit conditions:	• Game state is not saved after Player leaves the game (ex. score, coins, power ups, level progress).
Flow of events:	1. Player presses the Exit button. 2. Game warns Player that the progress will be lost and prompts the Player to confirm the action. 3. Player confirms he wants to exit the game. 4. Game ends and game window closes.
Alternative Flow:	1. Player confirms he does not want to exit the game by pressing Close. 2. Game redirects Player to Main Menu.
Special requirements:	When a new game is opened after a previous game was closed, it starts from the beginning.

USE CASE PlayQuadrillionLevel	
Participating actors:	Player
Entry conditions:	• Player is in LevelMode and has chosen a level to play OR • Player is in TreasureMode and has chosen a level on the map OR • Player is in LadderMode.
Exit conditions:	• Player receives an award if he won the game (depending on the game mode) or his health is decreased if he lost in Treasure Mode. • Any power up that the Player used during the game is removed from his inventory. • Player is redirected to the previous window he was in before the level started.
Flow of events:	1. System displays a window with a game on it. 2. Player picks up a quadrillion piece and flips or rotates it in any position using the corresponding control keys. 3. Player tries to place the piece in the grid. 4. Game checks if the piece can be placed where the Player indicated. 4.1 If yes, Game places the piece in the grid.

	4.2 If not, Game puts the piece back in its place outside the grid. 5. Player can use any power up available in his inventory at any time (type of power ups depends on the game mode). 6. Steps 2-5 are repeated until the game ends: 6.1 Player loses the game if time runs out in a timed level. Player also loses if he clicks on Quit Game. 6.2 Player wins the game if all the pieces are placed on the grid on time. 7. Game updates Player information depending on the outcome: 7.1 Game awards Player if he won the game. Awards can be coins, power ups or puzzle pieces depending on the game mode. 7.2 Game decreases Player health by one unit if Player lost the level in Treasure Mode. 8. Player loses the Gazillion game if his health reaches 0. Otherwise he is redirected to the previous state: the window of the mode he was playing before the level started.
Special requirements:	Game must make sure that Player will not receive any award again by replaying the same level.

USE CASE PlayTreasureMode	
Participating actors:	Player
Entry conditions:	Player clicked on Play Game in the Main Menu and then Play Treasure Mode.
Exit conditions:	• Game saves the state and Player progress after Player goes back to the Main Menu. • Player inventory, budget and health remain the same as just before Player has exited Treasure Mode, regardless of which mode Player enters next.
Flow of events:	1. Player selects Play Treasure mode option from Main Menu. 2. Game displays a treasure map with levels all over it. Initially only the levels at the corner are available for Player to play. 3. Player chooses a level from the available ones. 4. Game initiates *timed* PlayQuadrillionLevel Use Case. 5. Player information is updated according to last level's outcome: 5.1 Player loses one unit of health if he loses the level. If the health reaches 0 Player loses the game completely. 5.2 If Player wins the level he gets awarded: any Power Up, coins or a puzzle piece (treasure) for the "game inside the game". 6. If player won last level, Game unlocks the adjacent levels in the map. 7. Player can activate hints to get information about the location of the puzzle pieces for the "game inside the game" in the treasure map. 8. Game displays a message with direction information. 9. Player can check the puzzle pieces he already found by clicking on the Treasure Inventory button.

	10. Game displays a window with a Gazillion level on it, in which only the pieces that the Player has found are shown. 11. Player can place the pieces inside that grid just like in a normal level. Player wins Gazillion if he can complete this special level. 12. Game always records the changes Player makes in this grid. 13. Player clicks exit to go back to the Main Menu. 14. Game saves the state of the Treasure Mode and Player information.
Special requirements:	None

USE CASE PlayLevelsMode	
Participating actors:	Player
Entry conditions:	Player clicked on Play Game in the Main Menu and then Play Levels Mode.
Exit conditions:	• Game saves the state and Player progress after Player goes back to the Main Menu. • Player inventory, budget and health remain the same as just before Player has exited Levels Mode, regardless of which mode Player enters next.
Flow of events:	1. Player selects Play Levels Mode option from Main Menu. 2. Game displays a map with numbered levels all over it, labelled according to difficulty. Initially only the first level is available. 3. Player chooses the level he wants to play. 4. Game initiates *untimed* PlayQuadrillionLevel use case. 5. Player information is updated according to last level's outcome: 5.1 If Player loses a level he does *not* lose health. 5.2 If Player wins the level he gets awarded either a Power Up or coins. 6. If Player won last level, Game makes the next level available to the Player if it was previously locked. 7. Player clicks on Back to Main Menu to exit LevelsMode. 8. Game saves Player information and his progress in Levels Mode.
Special requirements:	Player may not use any Power up in this mode and health is not affected.

USE CASE PlayLadderMode	
Participating actors:	Player
Entry conditions:	Player clicked on Play Game in the Main Menu and then Play Ladder Mode.
Exit conditions:	Player inventory, budget and health remain the same as just before Player has exited Ladder Mode, regardless of which mode Player enters next.
Flow of events:	1. Player selects Play Ladder Mode option from Main Menu. The overall time he has to play Ladder Mode is initialized to a predefined amount. 2. Game randomly initiates *untimed* PlayQuadrillionGame use case.

	3. Player information is updated according to last level's outcome: 3.1 If time runs out Ladder Mode is over. Player is shown his final score (number of levels he passed in the time limit.) 3.2 If Player wins the level he gets awarded coins. Game initiates the next *untimed* PlayQuadrillionLevel use case directly. 4. Steps 2-3 are repeated until time runs out. 5. Player clicks on Back to Main Menu to exit LadderMode. 6. Game saves Player information but not Player progress.
Special requirements:	• Every time Player enters this mode time and score are reset. • Player may not use any Power up in this mode and health is not affected.

7.2 Object and class model

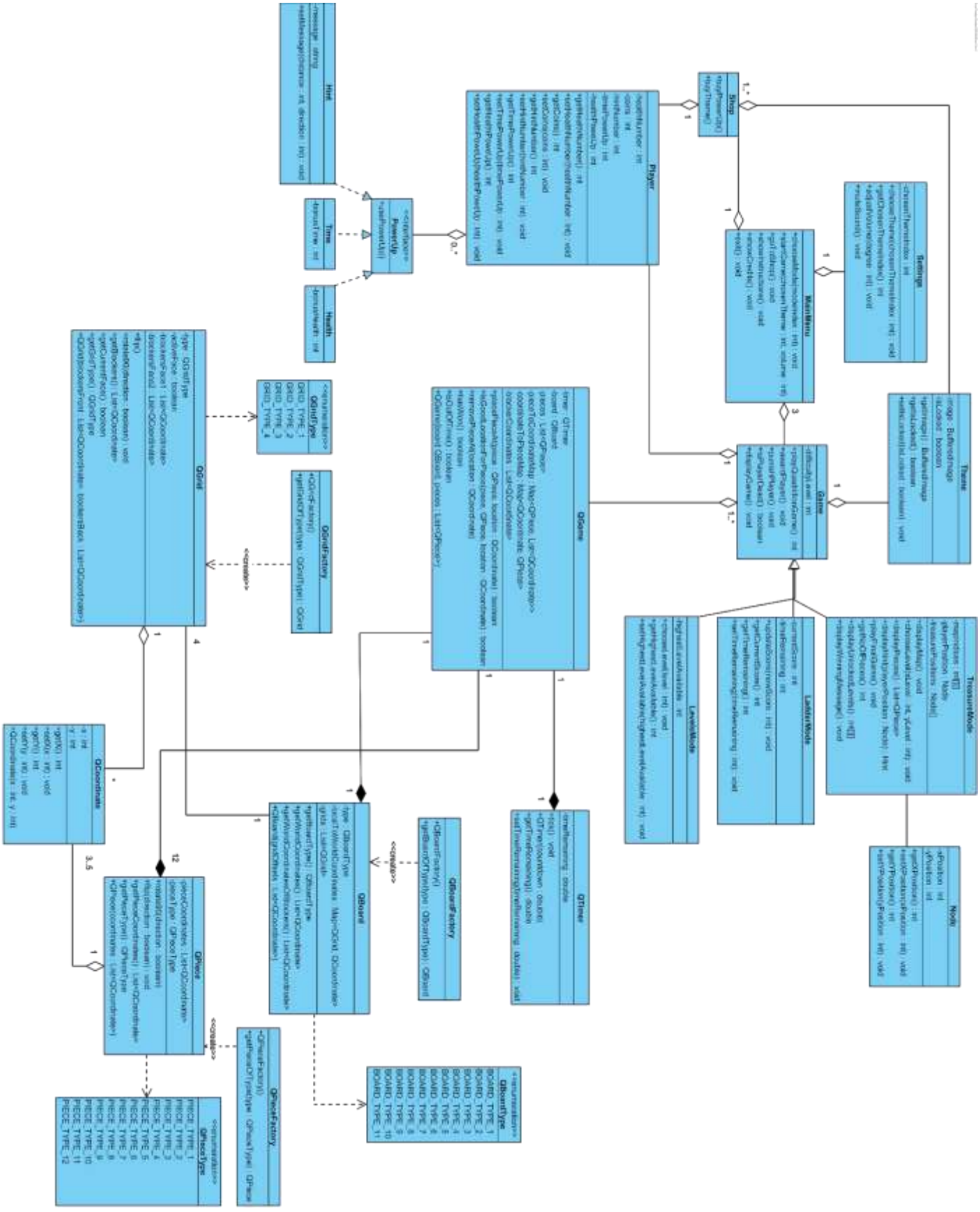

Figure 2. Object and class model for Quadrillion application

- MainMenu

MainMenu is the main interface of our program from where the player can choose the mode of the game (Treasure mode, Ladder mode, Levels mode), can go to the Shop, to Settings, to Instructions page or Credits page, start the game with the chosen settings passed as parameters and exit the application. MainMenu has three instances of Game (one for each mode), an instance of Shop and an instance of Settings.

- Settings

Settings will allow the player to change themes and volume level. Settings class has one attribute of integer type which represents the index of the chosen theme stored in an array of themes. The theme is changed by simply changing the index.

- Shop

Shop class will have a list of themes which will initially be locked and an instance of the Player. As the player progresses and gets rewards, the Player can purchase more themes. The player can also buy power ups. The method buyPowerUps() increases the player's hintNumber, timePowerUp or healthPowerUp according to his choice. This power ups can be used later during the game.

- Theme

A theme is represented by an attribute of type BufferedImage and an attribute of type boolean which determines whether the theme is locked or not. The attributes also have the corresponding setters and getters.

- Player

A player has attributes that indicate his number of remaining life and total amount of coins. Player has also the attributes that represent the number of power ups he has gathered throughout the game but has not used yet. This class has a list of instances of the PowerUp interface, by which it can call the method to use the available power ups.

- Hint, Time, Health

These classes implement the PowerUp interface whose main method is usePowerUps(). In Hint subclass, this method returns a message according to the position of the player in the map (Hints are available only on Treasure mode) with respect to the nearest treasure piece. In Time

subclass, this method increases time by an amount of timeBonus and in Health subclass, the method increases player's healthNumber by a unit amount of bonusHealth.

- Game

Game is an abstract class which specializes in three subclasses: TreasureMode, LadderMode, LevelsMode. The common attribute for all modes is the difficulty level of type integer. The common behavior of the classes is generalized by the abstract methods playQuadrillionGame(), awardPlayer(), punishPlayer(), isPlayerDead() which are then specialized to concrete methods is the subclasses. Game has an instance of Theme, which is the theme the Player has chosen or default theme. Game also has an instance of Player, from where it accesses player's healthNo and coins and updates them. Moreover, Game has 1 or more instances of QGame that are called when the method playQuadrillionGame() is invoked.

- TreasureMode

The map in the treasure mode is represented by a double array of integers, where 1 are the visited entries by the player and 0 are the rest. PlayerPosition is a Node object with x and y coordinates and treasurePositions is an array of Nodes whose coordinates are the locations of the pieces. When player chooses a level by clicking on a specific position of a map, the position is translated into matrix entries and if the level is unlocked the mapIndices is updated and a new QGame is instantiated. The method displayPieces returns a list of QPiece type with all the pieces collected by the player so far. displayHint is called when the player uses hint power ups and TreasureMode passes the player position as a parameter to generate a relevant hint. If the player has collected all 12 pieces, he can click to play the final game, which again calls a QGame. If the player wins the final game, the treasure mode progress is restarted.

- LadderMode

The Ladder mode keeps track of the current score and the time remaining in two distinct attributes, with their respective getter and setter methods.

- LevelsMode

In Levels mode, we store the highest available level so far and we continually update the level according to the player's progress.

- QCoordinate:

QCoordinate is a class which defines a two-tuple representing a coordinate on a discrete grid (i.e. a grid with only integer coordinates), with the necessary getters and setters.

- QGrid:

QGrid defines one of the four Quadrillion grids, each in its local coordinate system. Since a grid can only be 4x4, it does not explicitly store any QCoordinate objects regarding to the topology of the grid. Rather, it only stores the places on which no piece may be placed (blockers), on both faces of the grid. The class provides methods for the rotation and flipping of the board, along with the necessary getters.

- QPiece:

QPiece defines a Quadrillion piece in a local coordinate system by explicitly storing the coordinates of the piece. It provides methods for flipping, and rotation, and the necessary getters.

- QBoard:

QBoard defines a full game board, which consists of the spatial combination of four QGrids. For each grid, the class stores the offset of the local coordinate system in a map, list of the grids, and the type of the topology of the board. It provides the necessary getters, and a method to "materialize" the entire grid by building a list of coordinates. To build the list of coordinates, the grid coordinates are explicitly generated and for every grid, the offset of that grid is added to all coordinate objects, and all of them are appended to the list. The class also provides a method for expressing the "blocker" locations in world coordinates.

- QTimer:

QTimer is a simple timer class which keeps track of time elapsed by subtracting a fixed amount from the time remaining at every call of the tick() method it provides. Has the necessary getters and setters. Runs on a separate thread.

- QGame:

QGame defines a game of Gazillion. It consists of 12 QPiece and 1 QBoard instances. Keeps track of where the pieces are, and the piece hosted by a particular coordinate. Provides methods for piece manipulation via aggregation with QPiece and QBoard instances. Provides methods that check whether the game is considered to be complete, or time has run out.

Enumerations:

- QPieceType:

Enumerates the types of pieces as given in the Quadrillion game manual. There are twelve distinct pieces.

- QGridType:

Enumerates the types of grids as given in the Quadrillion game manual. There are four distinct grids.

- QBoardType:

Enumerates the types of boards that can be created using four grids. There are eleven official positions, and they are guaranteed to have at least one solution.

Factory Classes:

- QPieceFactory:

Constructs specific instances of QPiece by specifying coordinates according to the type (enumeration) of the QPiece, as present in the game.

- QGridFactory:

Constructs specific instances of QGrid by specifying the coordinates of the "blocked" coordinates according to the type of the QGrid, as present in the game. A grid may be uniquely defined by the location of its blockers on each face.

- QBoardFactory:

Constructs specific instances of QBoard by specifying the offsets of the QGrid instances to express them in "world" or "board" coordinates. There are eleven configurations for these offsets in the game manual.

7.3 Dynamic models

This section contains dynamic model diagrams (sequence, state and activity) for the major use cases described in the previous section.

7.3.1 Sequence Diagrams

These are the sequence diagrams that depict the flow of events of the main functionalities in our program:

- **Settings**

After Player presses the Settings button in the Main Menu, Settings window pops up. The Player is presented with three possibilities: Adjust volume (through a volume slider), Mute Sound (button) and Change Theme (a list of available themes to select from). If Player adjusts the volume through the slider the system immediately sets the volume to the amount specified. If Player presses Mute Sound the system sound is immediately muted. If player clicks on an available theme, the information about that theme is passed to the Game class through the Main Menu and whenever Player plays the next game, the theme is changed.

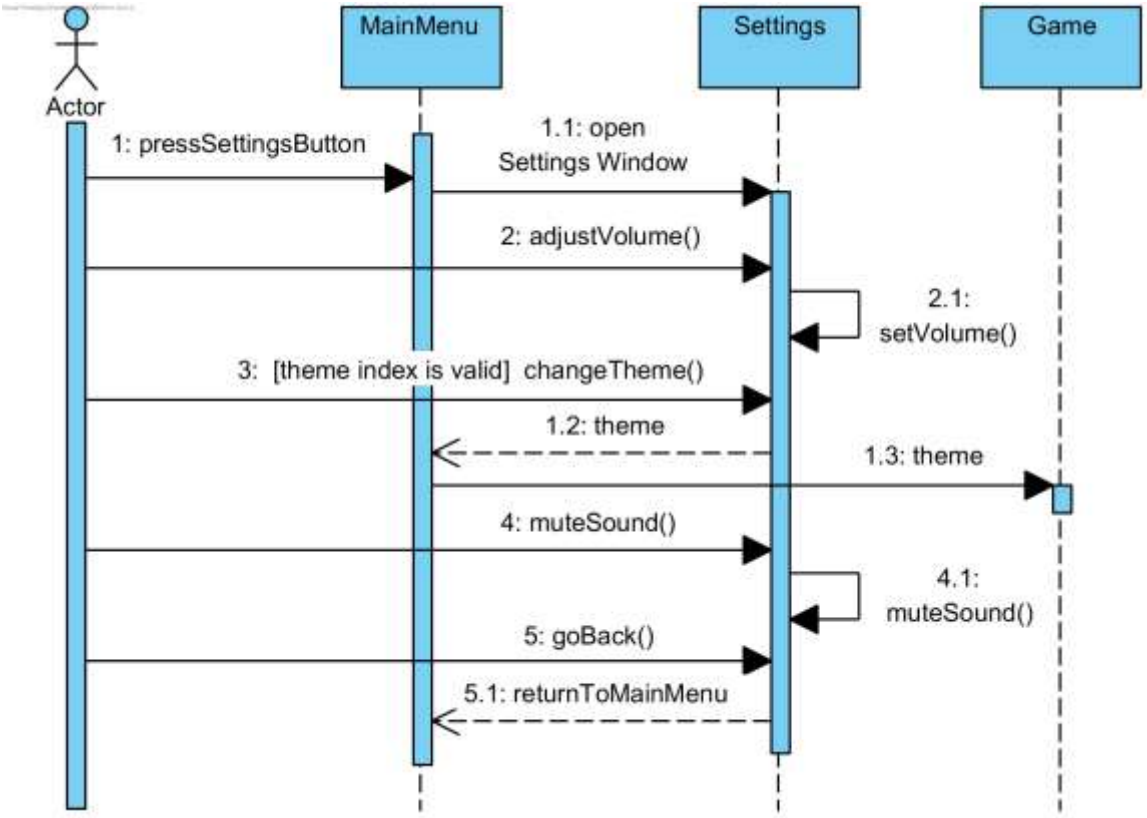

Figure 3. Change Settings Sequence

- QGame (basic Gazillion Game)

When the player starts playing the game the QGame instantiates a QBoard and 12 QPiece instances. If the game is in Treasure or Ladder mode, a QTimer instance is also created. During the game, the player chooses one piece type, which he can rotate or flip and then chooses a location in the board to locate the piece. The game first checks if the position is available (i.e. the entire piece fits inside the bounds of the board and it is not placed over a blocker). Only if the position is available, the player can place the piece, and the game updates the corresponding mappings between pieces and coordinates.

A player also can remove a piece from the board. The mappings are updated after each movement on the game. Meanwhile, the QTimer is running on the background and it continuously notifies the game about the remaining time. If the player chooses to use time power ups, the QTimer increases the remaining time. If the time is up or the player has won, the game is completed.

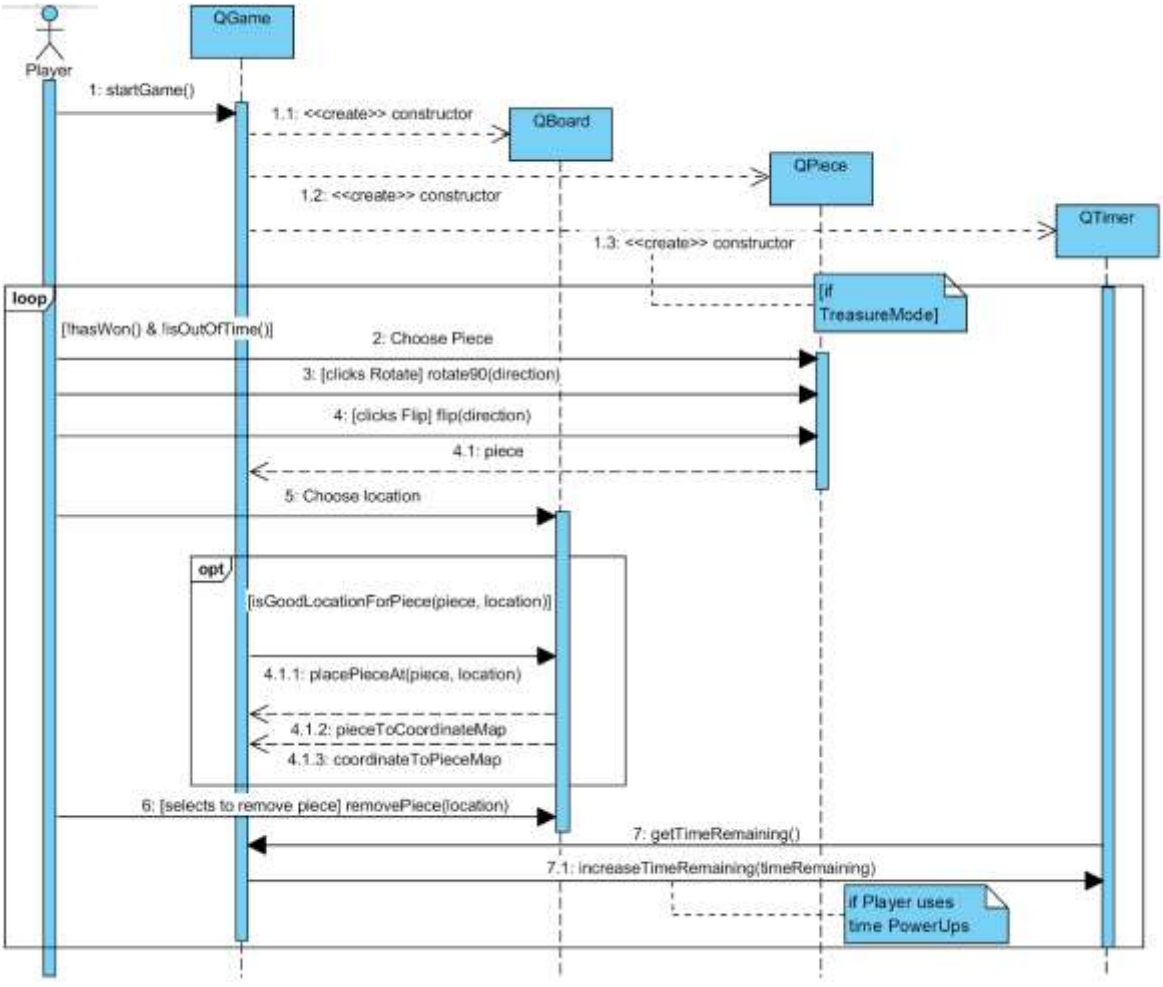

Figure 4. Sequence Diagram for QGame

- Levels Mode

When Player chooses Levels Mode from the Main Menu, the Levels Mode menu pops up. This mode contains many numbered levels, which are labelled according to their difficulty level. When Player chooses a level, LevelsMode class initializes a QGame instance, which is a Quadrillion game with random board positions but a specified difficulty level. The Player continues to play the basic Quadrillion game until he wins or quits. After beating a level, the Player is awarded coins and unlocks the next level. Player is not awarded for completing a level he has already completed before.

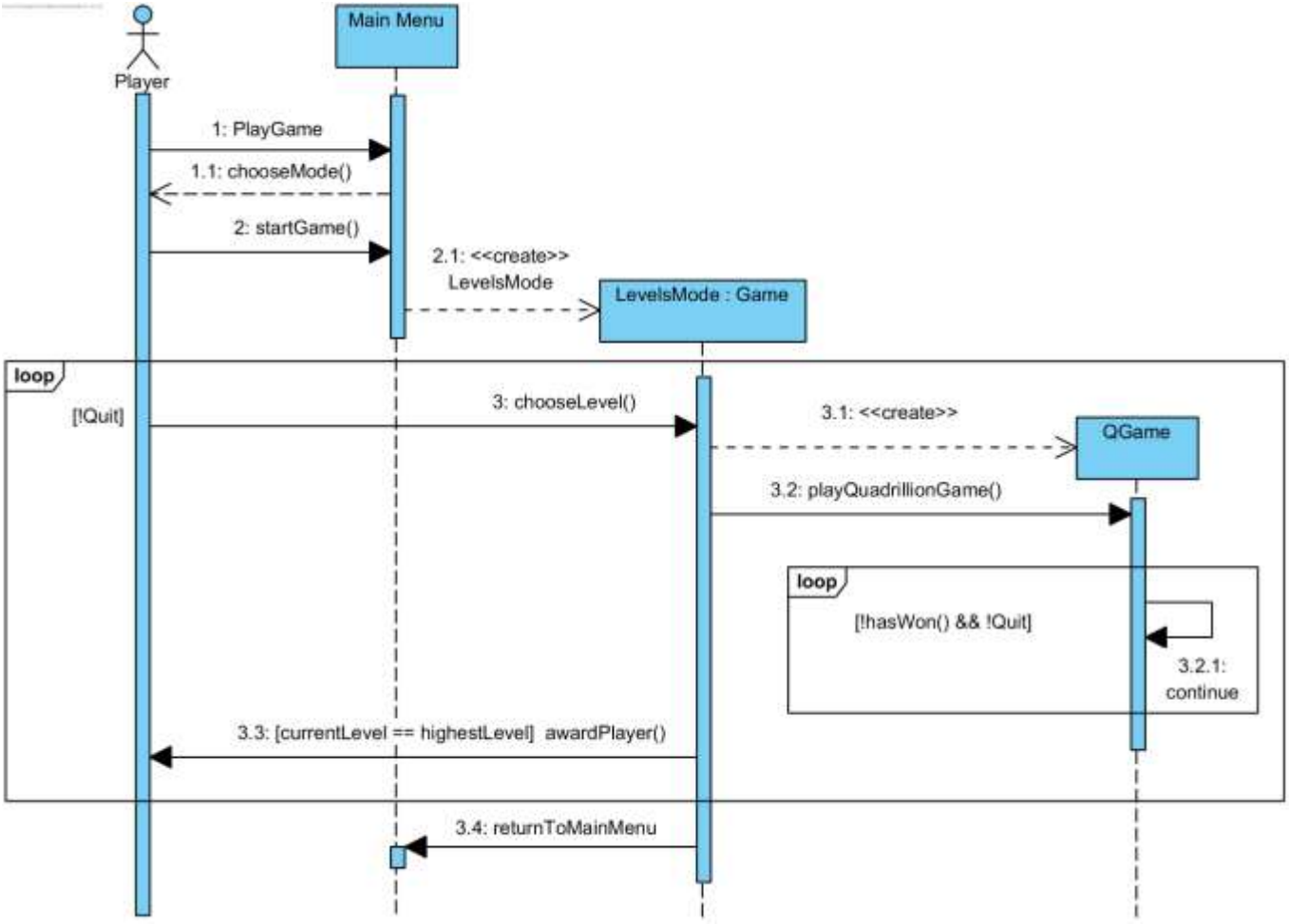

Figure 5. Sequence Diagram for Levels Mode

- Treasure Mode

After Player chooses the Treasure Mode from the Main Menu, the Treasure Mode menu pops up and a treasure map (image) is displayed. There are locked levels all over the map. While the Player has not lost the game (his health bar is not empty) has not won the game yet or has not pressed the Back Button to the Main Menu, he can select an unlocked level and play the random Quadrillion game generated by it. Player can use his power ups during the game.

If Player wins, he gets awarded a Power Up, health unit or coins and he can choose which direction he would like to pursue on the treasure map (which level from the adjacent levels he would like to play next). If Player loses the level, a unit of health is subtracted. If Player presses Hint Button, a Hint about the nearest treasure position (quadrillion piece) is displayed ("You are close", "Too far" ...) given the Player has enough Hint power ups. If Player presses the Display Pieces button, a window with the pieces he has collected so far pops up. Only if he has gathered the 12 pieces can the Player play the Final Game. He can win the ultimate Gazillion game only after beating this last level.

- Ladder Mode

Upon selecting Ladder mode among the available modes for the game, Player is presented the menu for the Ladder mode. Shortly, a QGame is created randomly and the manager class for the ladder mode is notified of the creation. When the QGame is completely instantiated, it is displayed to the Player, and the Player may now play a game of Quadrillion. The Player may select, place, remove, rotate and flip pieces at any time while the timer is still running. After every action, the QGame is refreshed, i.e. rerendered. This goes on until either the Player runs out of time or completes the level. If the Player runs out of time, the QGame instance is closed, and the Player is shown their current score, and the Ladder mode ends. Otherwise, the score of the Player is incremented by one and a new QGame is constructed, and the Player plays another game of Gazillion. This goes on until the Player loses.

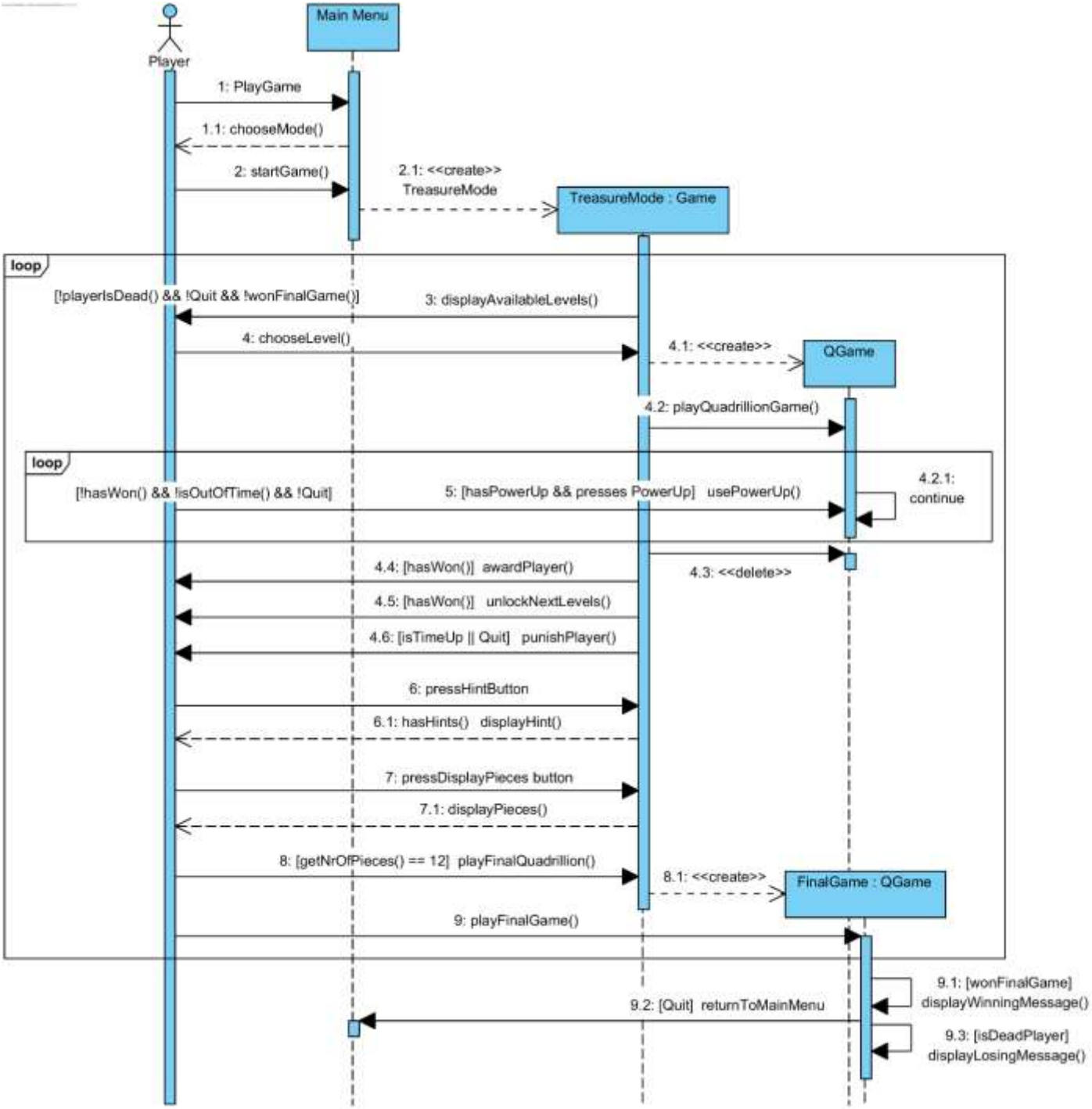

Figure 6. Sequence Diagram for Treasure Mode and Winning/Losing Scenarios

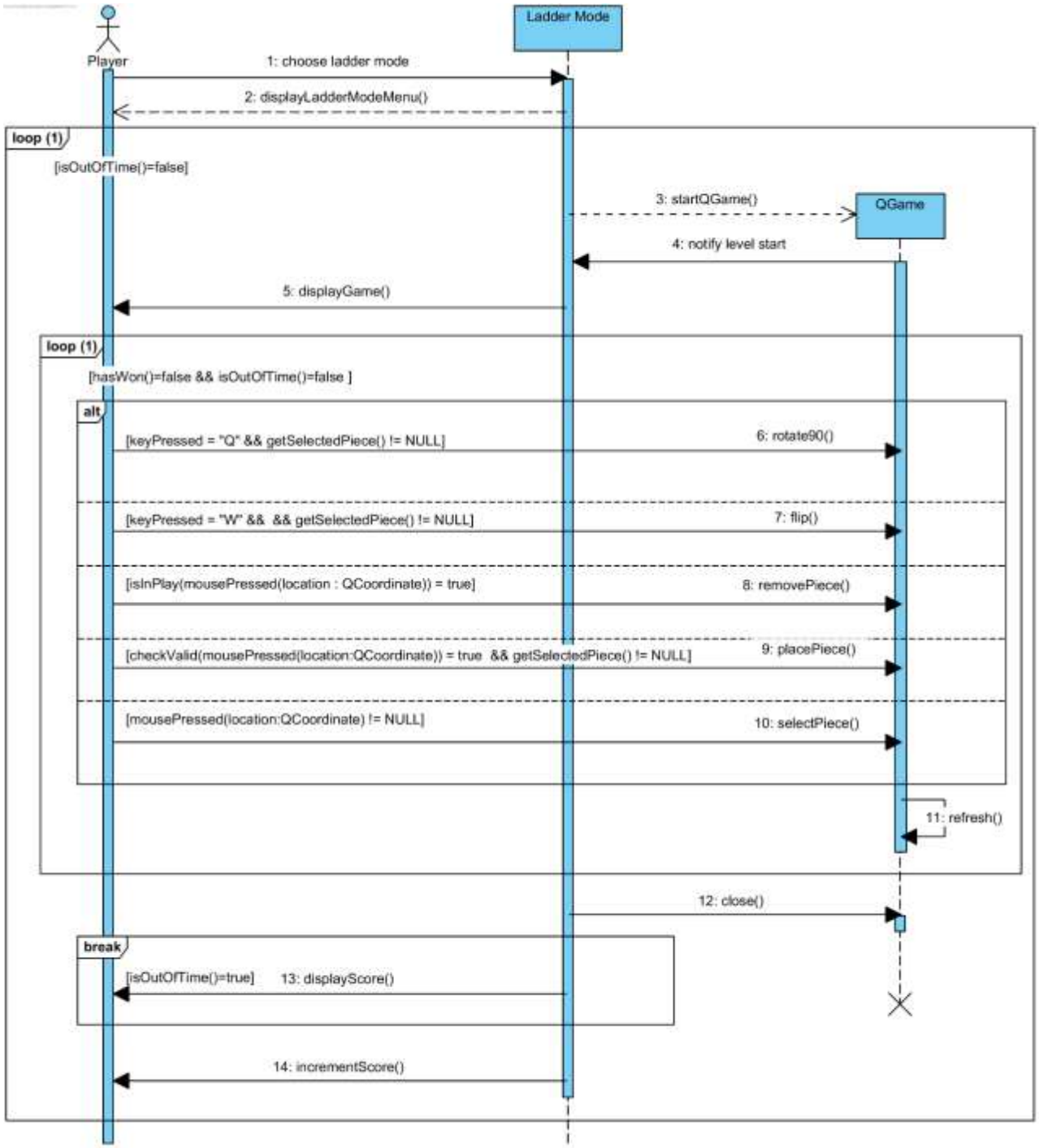

Figure 7. Sequence Diagram for Ladder Mode

7.3.2 State Diagrams

Below are provided the state diagrams for each mode of Gazillion and QPiece object.

- QPiece (Gazillion piece)

The following is the state diagram for a QPiece object. When an instance of QGame is launched, the QPiece is, always, considered to be in the state of "Out of the board". If the Player clicks on a piece, anywhere, the QPiece state is set to "selected". When a QPiece is selected, the Player may rotate, flip, or attempt to place the QPiece object. Rotations and flips do not change alter the state of the QPiece, and the QPiece is considered to be selected after rotations and flips. In the case of an attempt in placement, if the Player fails to appropriately place the object on the board, the QPiece instance will be set as out of the board. If placed appropriately, the QPiece transitions into a state of being on the board / in play. If the final end-game state has not yet been reached, if a QPiece on the board is selected by the Player via a mouse click, it is marked as Selected. The "meta" final condition for a QPiece object is when all pieces are in the state of "on the board", i.e. the puzzle has been completed.

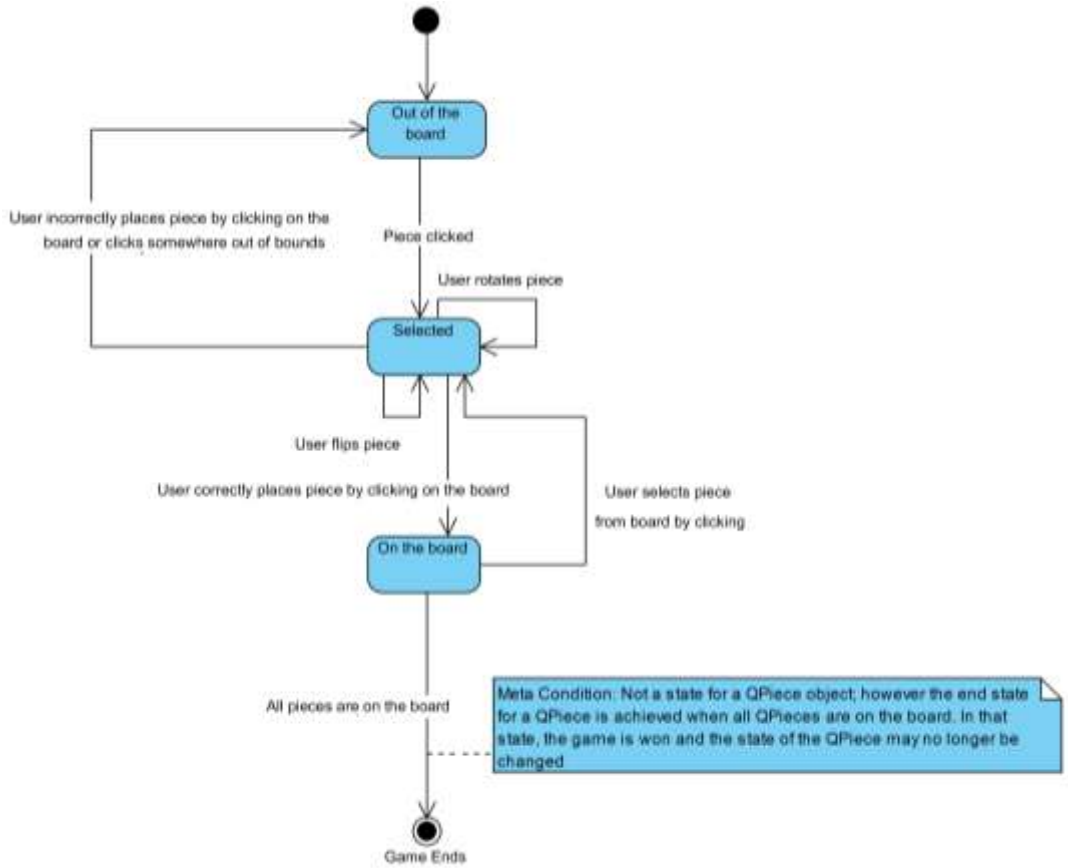

Figure 8. State Diagram for QPiece Class

- Levels Mode

The following is the state diagram for the level mode. Upon starting the level mode, the game enters a Level Selection Menu state. The state is retained until the Player provides an input. If the Player selects exit, a final state of "Exit Level Mode" is achieved, and this is the only way of leaving the level mode. Or, the Player may select an available level to play, and the game transitions into the game of Quadrillion state. In this mode, since there is no time constraint, the Player will be in this state until the Player wins or decides to quit the game. If the Player quits the game, the game transitions back into the level selection menu state, and waits for input. If the Player wins the game by playing it, the game enters the state of "Victory and Award display", in which the awards to the player are displayed. When the display is dismissed, the game promptly updates the available levels, and from that state, transitions back into the level selection menu state.

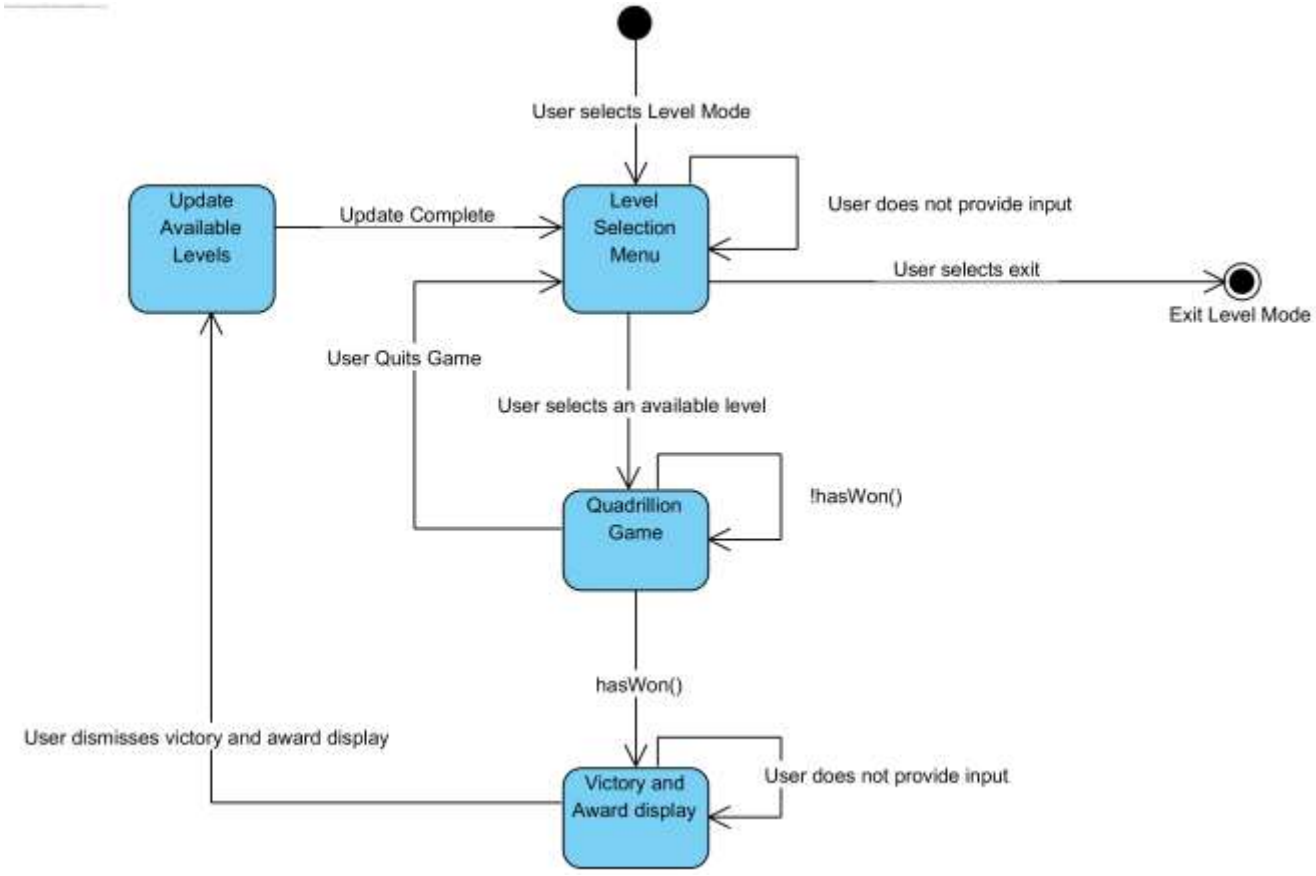

Figure 9. State Diagram for Levels Mode

- Treasure Mode

The following is the state diagram for the treasure mode. Upon starting the treasure mode, the player will be shown the map of the treasure mode, which is represented by the state "Level Selection and power-up use on the Map". The Player is free to use any power up in this state. If the Player uses a health power up, an animation will play indicating that a power up has been used, and the game will be in the state of "health increment animation". When the animation ends, the game will go back to its previous state. Using a hint power up results in a similar outcome. The game will transition into a "Hint display" state when the Player uses a hint power up. The game stays in that state until the Player dismisses the display panel, and when the Player does so, the game will return to its former state.

The other main action that the Player can take in the map is selecting a level to play by clicking on one of the levels. When this action is taken, the game enters the loading state, and when the loading is finished, the game enters the "Game of Quadrillion" state. The game stays in this state until the Player either runs out of time or wins. If the Player wins, the game enters a state of animation and award display. After this animation ends, if the number of collected pieces by the Player is less than 12, the game reverts back to the Level Selection and power-up use state. If the Player has collected all 12 pieces, the game can enter the state of "Final QGame", in which the Player is to play a final round of Quadrillion. Similar to the Game of Quadrillion state, the Player plays the game until the Player wins or runs out of time. If the Player wins, a final state of Victory is achieved. Otherwise, the Player loses the game and gets a Game over. Going back to the Game of Quadrillion state, if the Player runs out of time in that state or ends the game, the Player is deducted one health, and depending on the number of health points remaining, the game either transitions into a state of game over, or back to the Level Selection and power-up use state. If the Player quits while playing a game of Quadrillion, it is considered a loss and the game transitions into the health decrement animation state. If the Player decides to give up on the game in the Level Selection and power-up use, the game transitions into the Game over state.

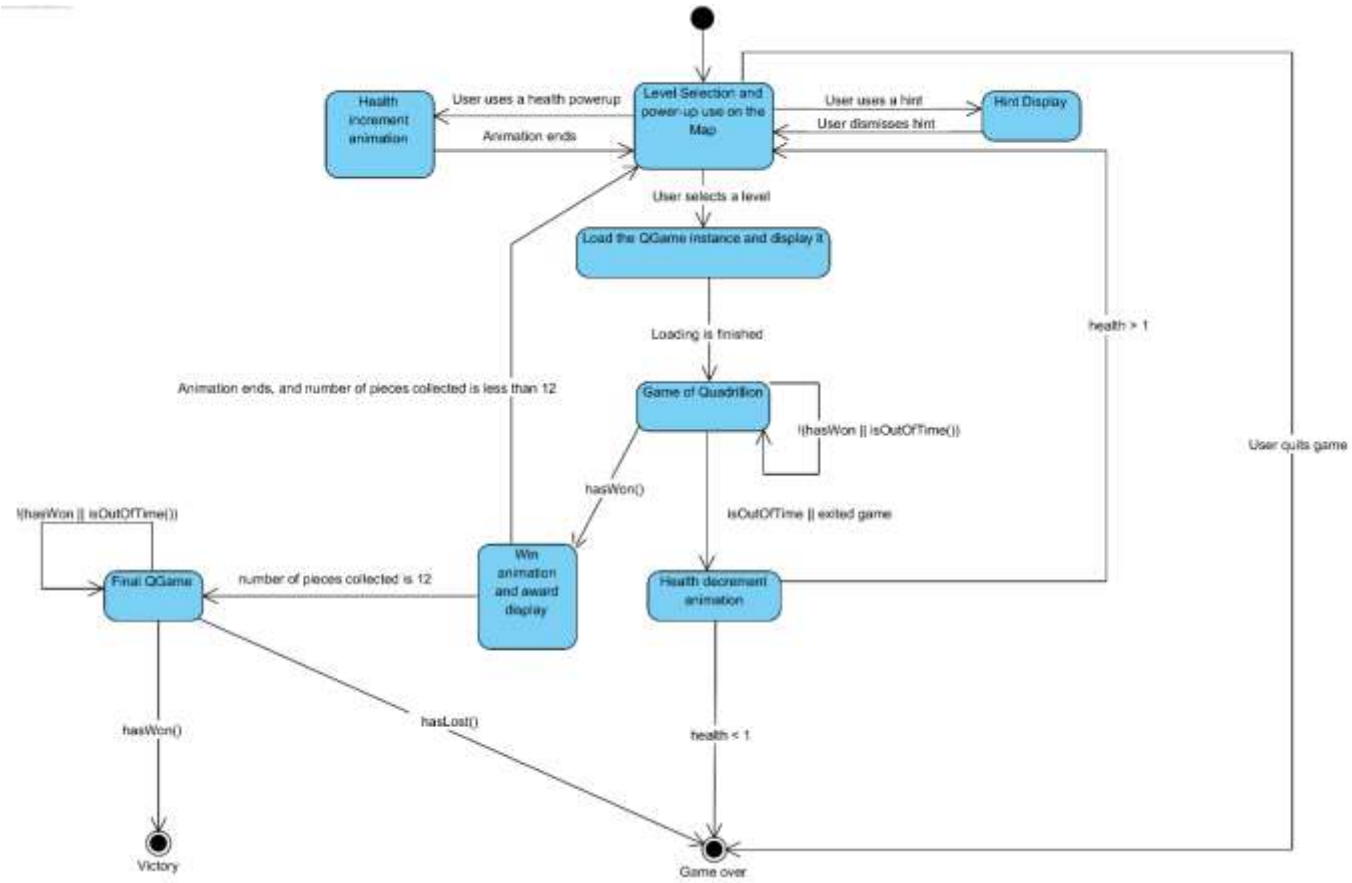

Figure 10. State Diagram for Treasure Mode

- Ladder Mode

The following is the state diagram for the Ladder mode. Upon starting the Ladder mode, the player is greeted by the game mode, and the game automatically transitions into a state of "Quadrillion Game". The state is retained until either the Player wins or runs out of time. If the Player wins, the score of the Player is incremented, and the Player is given an intermission between two levels of Quadrillion, as indicated by the Score Increment and Round Intermission state. The game loads another game of Quadrillion when this state is terminated internally by the game logic. Whenever the Player runs out of time in the state of Quadrillion game or decides to quit the game, the Player is displayed their final score, and when this display is dismissed, a final state of "Exit Ladder Mode" is achieved.

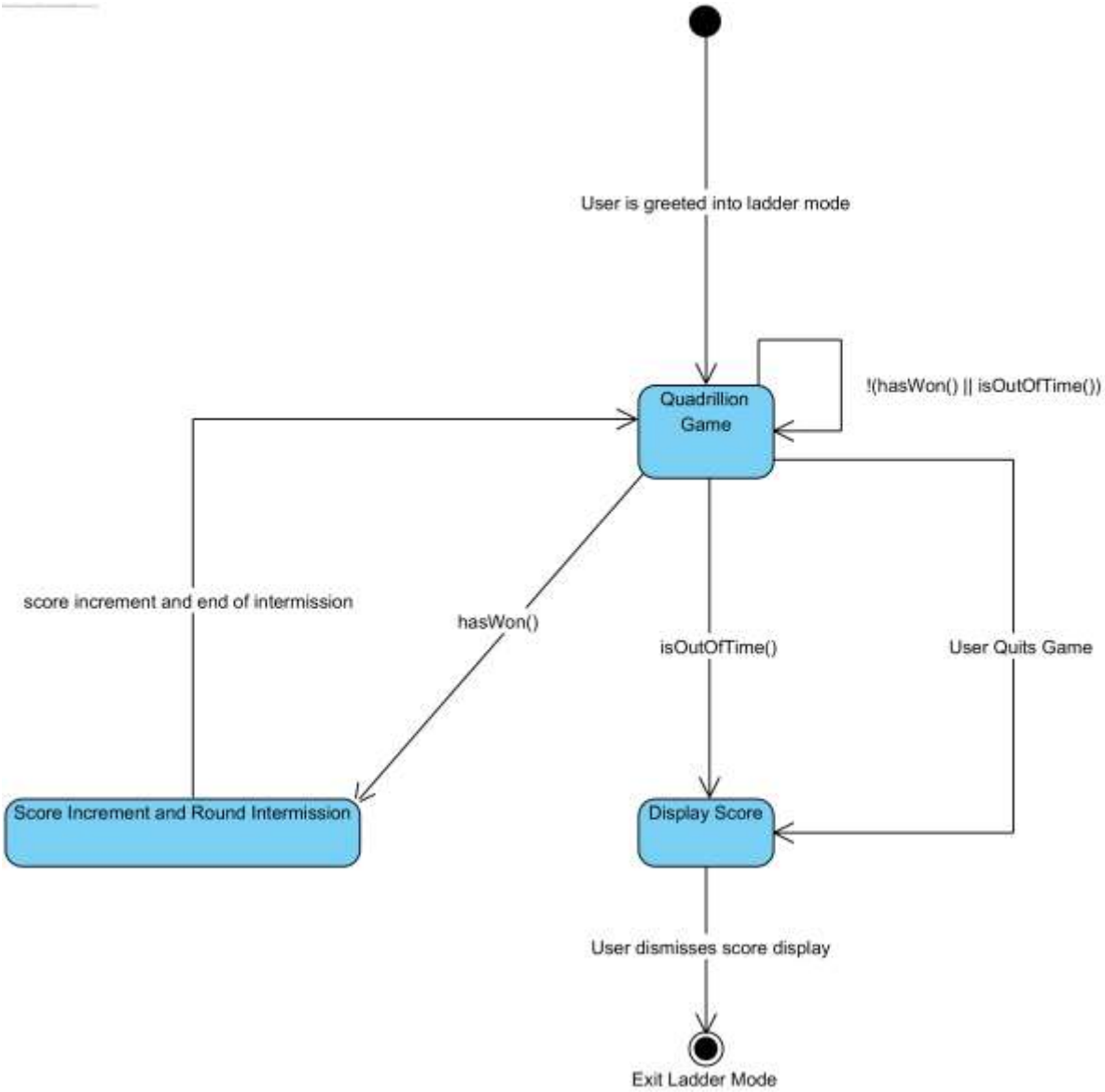

Figure 11. State Diagram for Ladder Mode

7.3.3 Activity Diagram

The following is the activity diagram for the gameplay system. The possible inputs that the Player can provide to manipulate the pieces are: rotating a piece, flipping a piece, selecting a piece, placing a piece on the board, or removing a piece from the board. After the player selects one of these options, the Game instance attempts to locate the piece on which the Player action will be performed, i.e. it translates the GUI to game logic. For rotations, flips, and selections, Game tries to determine which piece is the currently selected piece. Then, according to the Player action, the QGame instance performs that action on the piece, and updates the game state.

For piece placement and removals, the system first locates the piece that will be moved onto or out of the board. For piece removal, the QGame instance simply moves the piece out of the board and updates the game state. However, for the placement of a piece, the system first has to check whether the location of the placement is appropriate. This is done by correlating the coordinates of the piece (by converting its coordinates from its local coordinate system to its world coordinates) and the coordinates of the section of the board on which the placement is requested. This is done by querying the two objects in parallel, and after the necessary messages have been passed, the system decides whether to put the piece on the board or not. If the piece is placed on the board, the game state is updated. If the piece is not eligible for placement at that particular position, the Player is notified, and the game state is updated again. After the game state is updated, the QGame logic checks whether the player has won the game by looking at the number of empty and fillable coordinates; as well as the remaining time of the Player. If the Player has not yet run out of time or won the game, the control is given back to the UI, which waits for another Player input. Otherwise, the game prompts the Player regarding the end of the game (victory or timeout), and the activity is terminated. The other instance at which the player may terminate the game is when the UI is awaiting input: if the Player clicks on the necessary button on the UI, the game is finished, without any display of endgame messages.

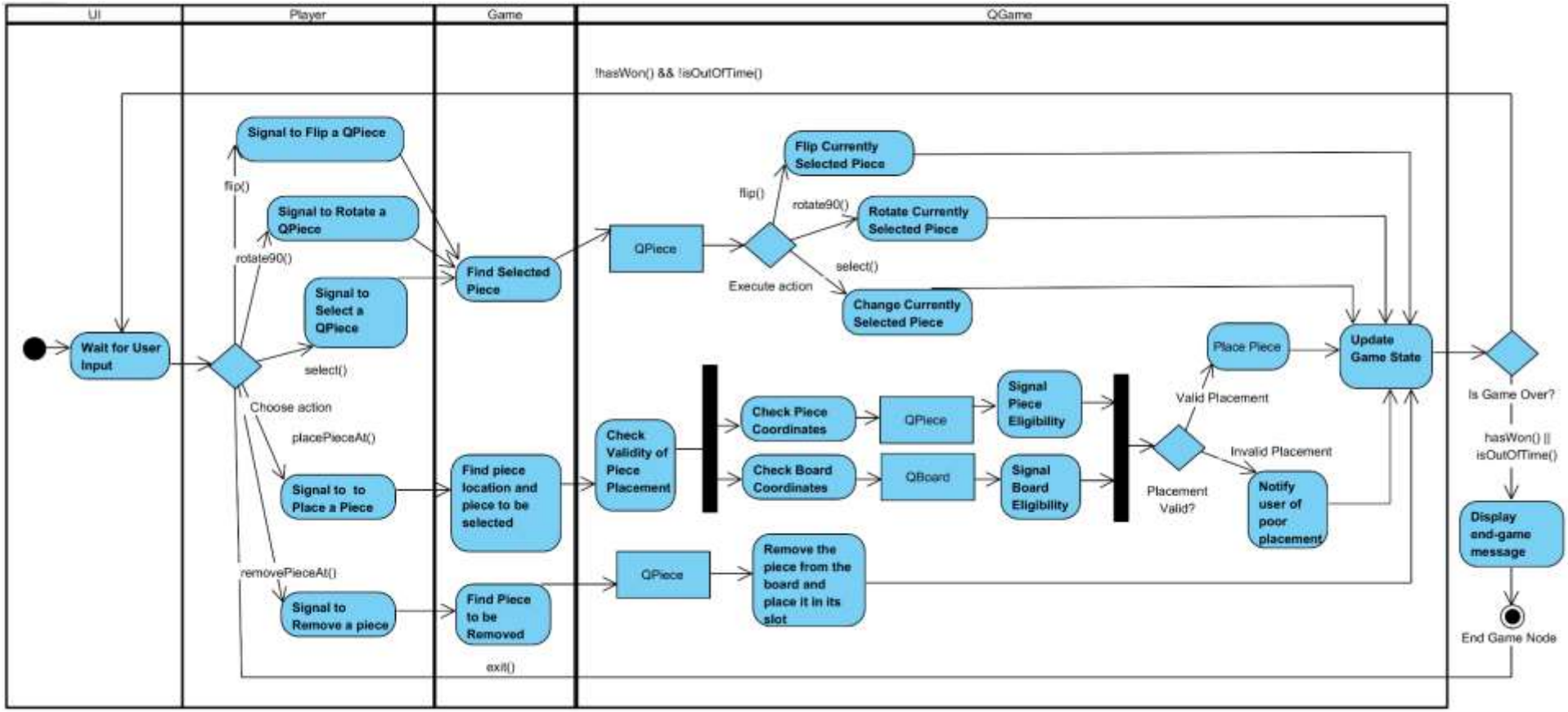

Figure 12. Activity Diagram for round of Quadrillion.

7.4 User interface

We used the online Mock+ tool to create the mockups for the UI.

7.4.1 Main Menu

In the Main Menu which pops up when you start the game, there are 6 options for the player: Play Game is the option to start the game, clicking on it directs the user to the Modes Screen page. Shop opens the shop where user can buy themes, hints, health and time powers ups. Instructions informs the player about the game, gives a brief description about the different modes and how to play each. Clicking on the Settings button opens the settings for themes and sound. The exit button quits the game by closing the application.

Figure 13. Main Menu mockup

7.4.2 Settings Screen

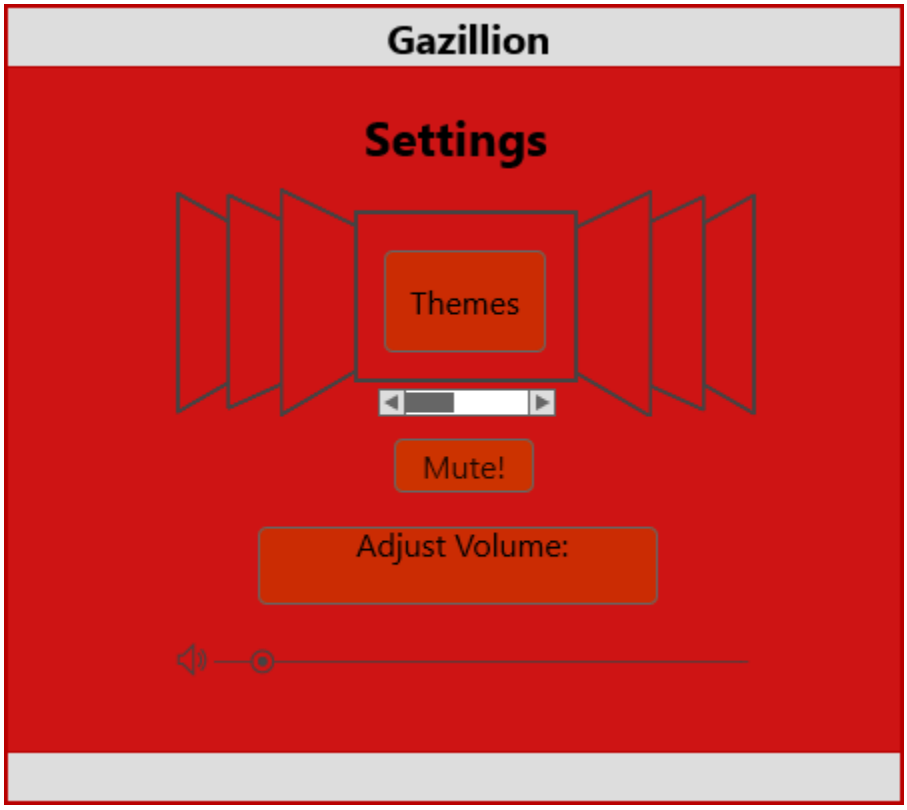

Figure 14. Settings Screen Mockup

7.4.3 Credits Screen

Figure 15. Credits Screen Mockup

7.4.4 Modes Screen

Upon clicking the “Play” button on the Main Menu, the user is shown the different modes of the game to choose from.

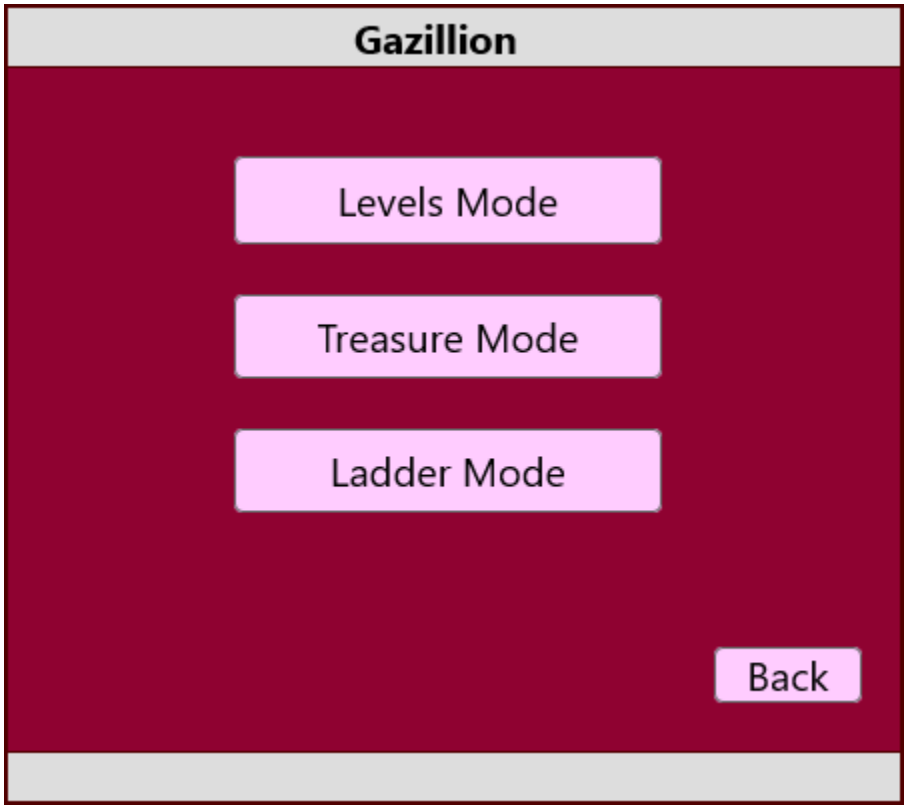

Figure 16. Modes Screen Mockup

7.4.5 Game Play

This is the Game Play screen of the Quadrillion game. On the top left, the health bar is shown with an option to get hints on the right of it and to use coins at the top. The Game board is in the middle left with the pieces on the right in the middle. On the bottom of the screen the remaining time is displayed, and there is an option to get time power ups next to it.

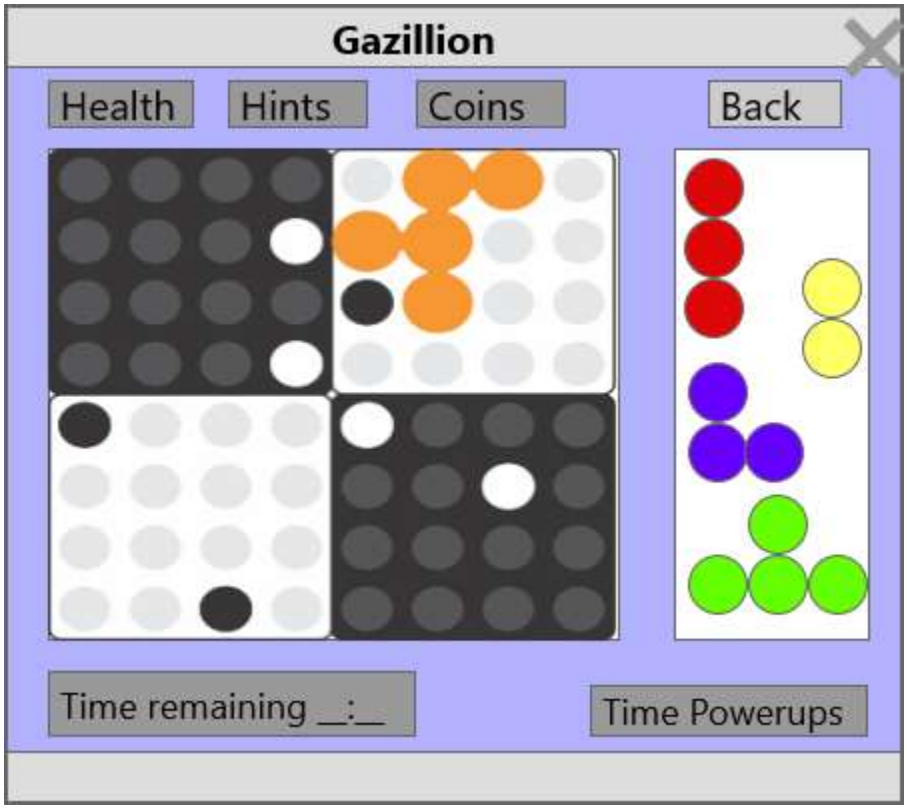

Figure 17. Game Play Mockup

7.4.6 Level mode

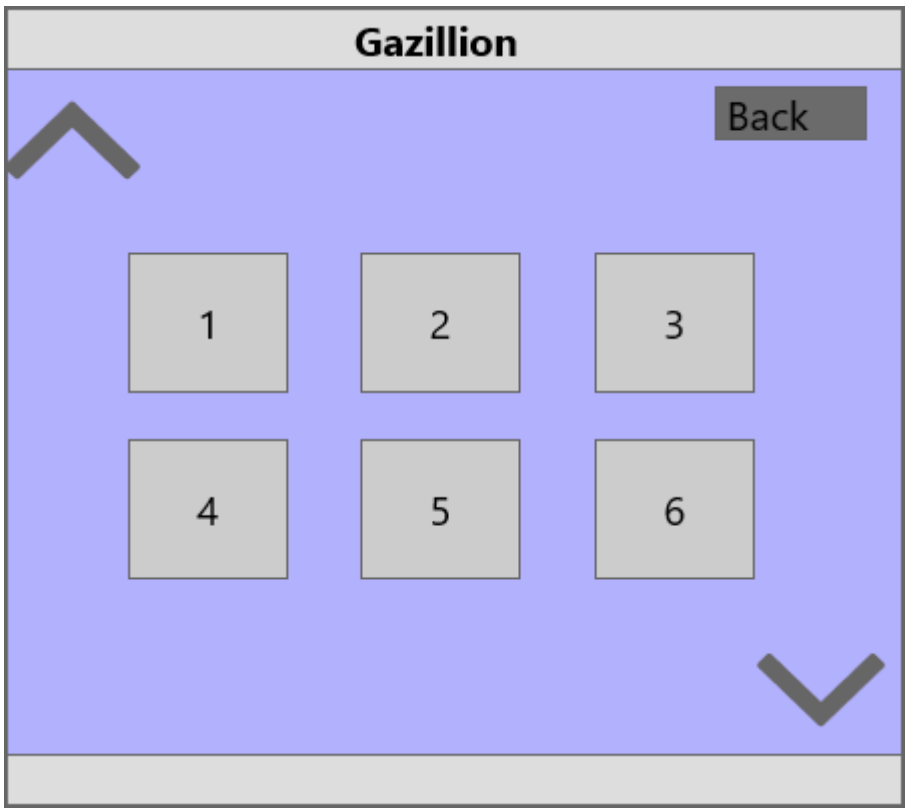

Figure 18. Levels Mode Mockup

This is the screen which appears when the user chooses the “Levels Mode”. 6 levels are shown on the screen with an option to see more with the arrows at the top left and the bottom right of the screen. By clicking on a particular level number, the game play screen shown above opens with the configuration of that specific level.

7.4.7 Ladder Mode

This is the intermediate screen which appears in the Ladder mode. It includes an extra option on the bottom right of the screen, “Give up!” In the middle of the screen, the progress of the user in terms of rounds completed is shown.

Figure 19. Ladder Mode Mockup

7.4.7 Treasure Mode

Upon choosing the treasure mode, this screen is displayed. On the top of the screen, the user’s health bar and options to get hints and use coins are displayed along with the back button to go back to the previous screen i.e the mode screen. On the right side of the screen the collected pieces are displayed for the user.

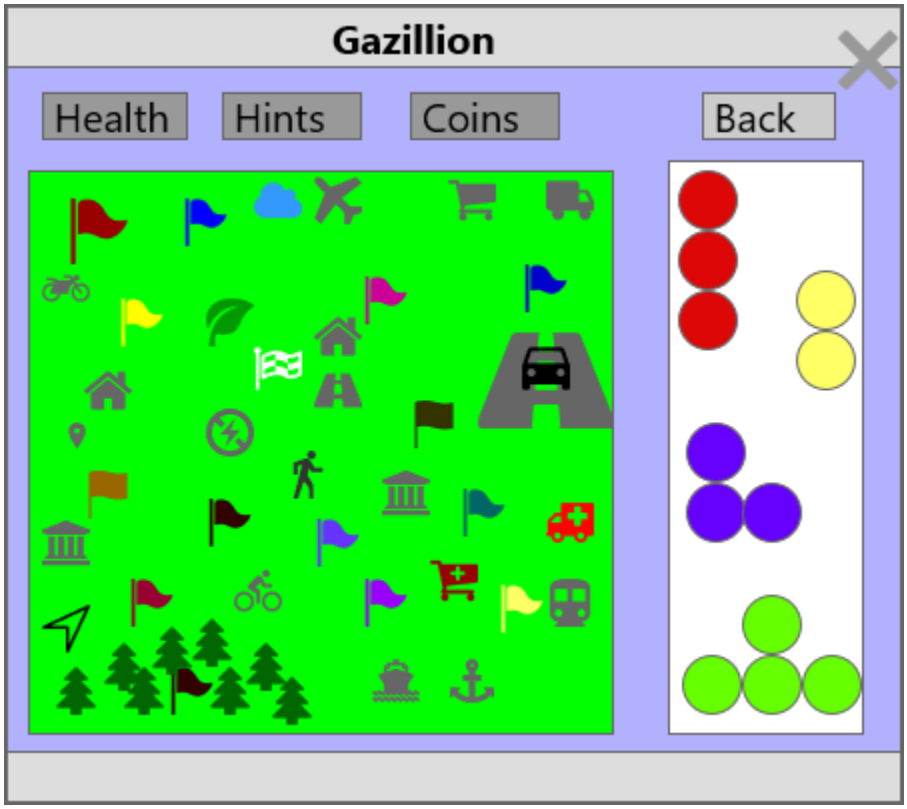

Figure 20. Treasure Mode Mockup

8. Conclusion

In this report, we have analysed the initial design of our game "Quadrillion". The report includes the requirement analysis and the system models.

The analysis consists of a summary of the functional and non-functional requirements of the project. Regarding the functional requirements, we have explained the behaviour and functionality provided to the player. The non- functional requirement contains performance-wise features of the application like accessibility, usability, portability, gameplay stability etc.

Moreover, the report depicts the overall system model by making use of diagrams such as use case diagrams, sequence diagrams, class, activity and state diagrams. Each diagram illustrates a specific aspect of the project functionality. Hence, this report serves as a foundation for the latter implementation and design of the project.

9. References

[1] “Discover Our SmartGames Collection.” *IQ Puzzler Pro - SmartGames,* https://www.smartgames.eu/uk/our-smart-story. [Accessed: Mar 10, 2019].

[2] “Quadrillion.” www.smartgames.eu/uk/one-player-games/quadrillion. [Accessed: Mar 05, 2019].

[3] “QUADRILLION - keep on playing forever”. www.smartgamesandpuzzles.com/inventor/Quadrillion.html. [Accessed: Mar 05, 2019].