Bilkent University

Department of Computer Engineering

CS-319 Project

Quadrillion

Analysis Report

Group No: 1H
Group Name: COGENE

Osman Orhan Uysal

Samet Özcan

Mehmet Alper Karadağ

Ziya Erkoç

Talha Murathan Göktaş

Contents

Introduction **4**

Current System **4**

Overview **4**

Piece 4
Ground 4
Level 4
Hint 4
Functional Requirements 4
Play Game 4
Casual Game 5
Ranked Game 5
Compose Level 5
Select Level 5
Buy/Use Hint 5
Non-functional Requirements 5
Usability 5
Reliability 5
Performance 6
Supportability 6
System Models 6
Use-Case Model 6
Object and Class Model 17
Drawable Class 17
Piece Class 17
Ground Class 17
Level Class 18
GameComponent Class 18
Page Class 18
PlayGame Class 18
PlayRanked Class 18
MainMenu Class 18
ComposeLevel Class 18
Login Class 18
SelectLevel Class 18
FetchLevel Class 18
Screen Class 19
User Class 19
Record Class 19
Dynamic Models 20
Sequence Diagram 20
Play Casual 20
Create Level 21

Play Ranked 22
Compose Level 23
State Diagram 24
Activity Diagram 25
User Interface 26

Improvement Summary **30**

References & Glossary **30**

1. Introduction

Quadrillion is a puzzle-board game in which the player needs to fill all the vacancies on the board using prefabricated pieces. Game consists of 4 grounds each of which can be combined in a lot of way to create the whole board. Each ground has two or three places which are blocked in a way that the marbles of the pieces cannot be placed there. Grounds are also double sided which indeed increases the combination possibilities.

In this project, we aimed to digitalize the Quadrillion game to make it more attractable and fun by adding features that it is not possible to add to the board game version.

2. Current System

Board version of the Quadrillion contains the pieces and 4 grounds. It allows players to attach grounds together to create different combinations. It contains a manual which includes around 60 challenges with solutions.

3. Overview

3.1. Piece

A piece is the main component of game which consists of marbles that are tied together and player can drag it around. player aims to find an available spot on the ground for each piece in order to complete the game. There are 12 different pieces in the game. A Piece can be rotated 90-degree clockwise, flipped along the y-axis and moved around.

3.2. Ground

A ground represents 4x4 ground where some locations are not available. Combination of these 4x4 grounds results in a level. Like Piece, Ground can be flipped around (front-side or back-side of the ground can be used), moved around and rotate 90-degree clockwise.

3.3. Level

Level is a combination of 4 grounds. In our game, there are predefined levels which already exist in the manual of the board version as well as player created levels.

3.4. Hint

If player stucks at some point there will be hint provided for him/her. Hint basically will reveal the location of one of the pieces. Player will be given some amount of hint tokens which will decrease as player requests hint. By paying small amount of fee player will be able to get extra hints. Player should use their hints wisely since at some point they may be scarce.

4. Functional Requirements

4.1. Play Game

Player can play either ranked game or casual game.

4.1.1. Casual Game

In this mode, player selects either one of the predefined levels or player created levels. Player drags pieces and drops them on the grounds in order to fill all the vacancies on the grounds. In the meantime, number of moves he/she made and time elapsed for the completion of the level is tracked so that player can compete with himself/herself. In case player gets stuck, hint will be provided to him/her if he/she has enough hints available. Otherwise, he/she can buy hints. Player's performance in casual game is not added to the leaderboard of the level.

4.1.2. Ranked Game

Unlike casual mode, in ranked mode, the player will be randomly assigned to a level. Again, player can drag & drop the pieces to complete the level. However, he\she will not be able to request a hint. If player successfully completes the level, based on the time it took to solve the level and the number of moves he made, he will be ranked on the leaderboard of the level.

4.2. Compose Level

Player can create levels by moving, flipping and rotating the grounds. Then he/she can hit the submit button so that the level can be checked for validity. The levels are accepted after checking if the level is a valid combination of grounds, has a valid solution and this combination has not been created before.

4.3. Select Level

Both predefined levels and player defined levels are displayed on level selection screen. Player can see the leaderboard of the level before playing it.

4.4. Buy/Use Hint

Players are given the ability to use and buy hints to proceed through the game. When player is stuck at a level in casual mode, he/she can use one of his/her hints to reveal the position of one of the pieces. Players are given some amount of hints initially and they are expected to use them wisely; that is, they should not waste them. If they run out of hints, they can buy some by entering their credit card information.

5. Non-functional Requirements

5.1. Usability

- Players should be able to drag the pieces by holding it from any of its parts.
- %80 of the players should be able to play the game without looking at the instructions.

5.2. Reliability

- If played 100 times, it should behave unexpectedly at most 10 times.

5.3. Performance

- All database related actions (i.e. signing in, signing up, uploading a level, opening a level) should at most take 4 seconds.
- Hint purchase transaction should at most take 10 seconds.

5.4. Supportability

- Hashed versions of the player passwords should be kept in database.
- Credit-card information should not be stored in the database.

6. System Models

6.1. Use-Case Model

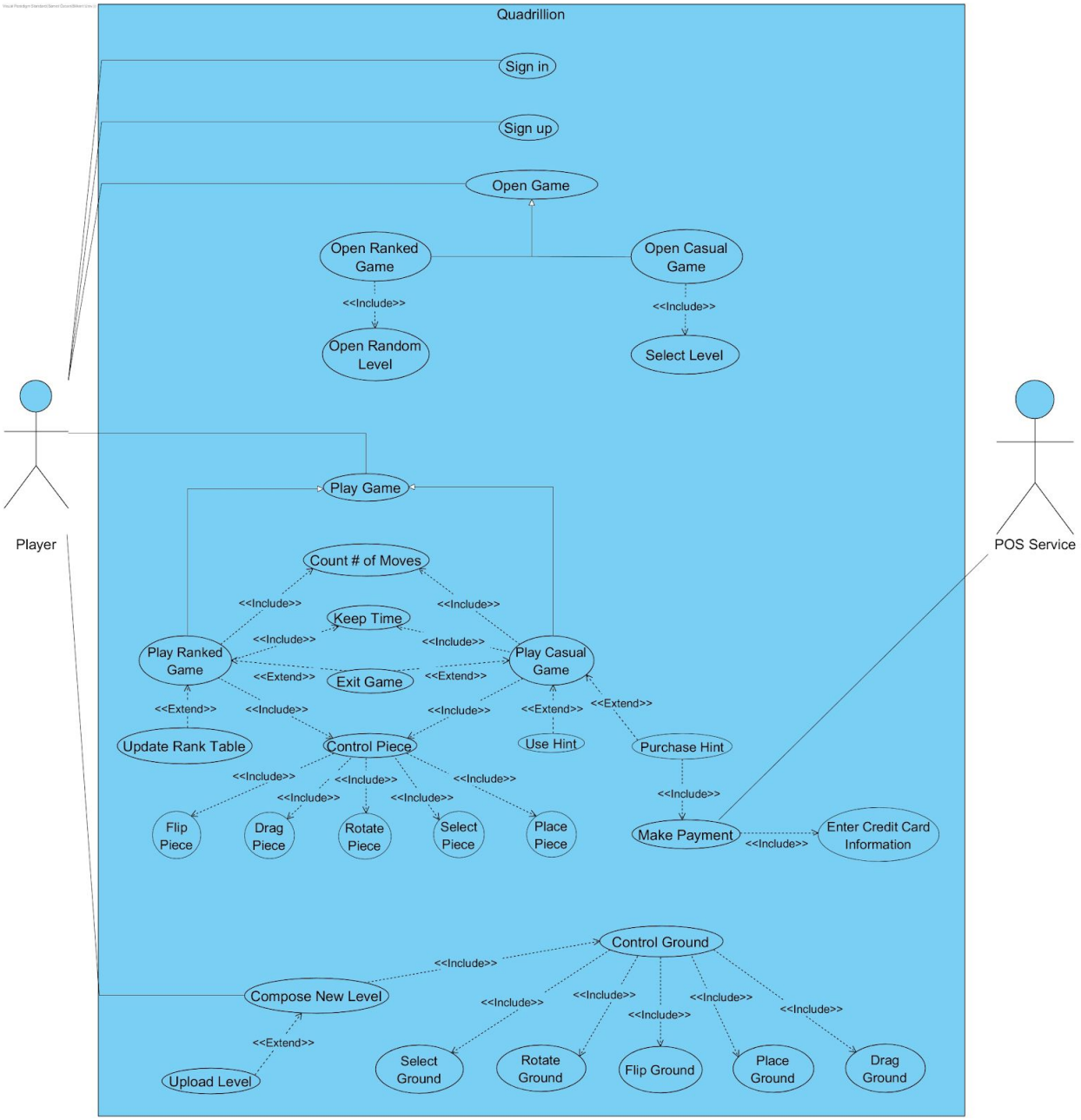

Figure 1: Use Case Diagram

Use case name **Sign in**
Participating actor
●Player
Entry condition
●Player opens the game and pushes the sign in button
Flow of Events
●Player enters his/her id
●Player enters his/her password
Exit condition
●Player sign in to the system successfully or gives up
Special requirement
●Only 3 attempts are allowed in a minute

Use case name **Sign up**
Participating actor
●Player
Entry condition
●Player opens the game and pushes the sign up button
Flow of Events
●Player enters id
●Player enters password
Exit condition
●Player sign up to the system by entering valid id and password or gives up

---

Use case name **Open Game**
Participating actor
●Player
Entry condition
●Player signs in successfully
Exit condition
●Player makes his/her decision

Use case name **Open Ranked Game**
Participating actor
●Player
Entry condition
●Player decides to play a ranked game

Flow of events

●Player pushes the Play Ranked button

Exit condition

●Ranked Game style is selected

Use case name **Open Random Level**

Participating actor

●Player

Entry condition

●Player selects Ranked Game style

Flow of events

●Player is assigned to a random level

●Best scores of the level is displayed

Exit condition

●Player starts to play the Ranked Game

Special Requirement

●Player is assigned to a random level in order to hinder cheating

---

Use case name **Open Casual Game**

Participating actor

●Player

Entry condition

●Player decides to play a casual game

Flow of events

●player pushes the Play Casual button

Exit condition

●Casual Game style is selected

Use case name **Select Level**

Participating actor

●Player

Entry condition

●Player selects Casual Game style

Flow of events

●The existing levels are displayed on the screen

●Player selects one of the levels

●The best scores of the selected levels are displayed

Exit condition

●Player starts to play the Casual Game

Use case name **Play Game**

Participating actor

●Player

Entry condition

●Player either selects a level in Casual Game mode or s/he is assigned to a random level in Ranked Game mode

Flow of events

●The game starts

Exit condition

●Player finishes the level successfully or s/he exits

Use case name **Play Ranked Game**

Participating actor

●Player

Entry condition

●The Ranked Game starts as a result of the player's decision

Flow of events

●The board and pieces are created and displayed on the screen, time counter and move counter are started and the data related to the player actions and needed start to be kept

Exit condition

●Player finishes the level successfully or s/he exits

Use case name **Update Rank Table**

Entry condition

●Player finished the Ranked Game successfully and breaks some records of the leaderboard for that level

Flow of events

●Player's name and his/her score is placed on the Rank Table accordignly and the last score and its owner is dropped from the table

Exit condition

●The Rank Table is updated

Use case name **Count Number of Moves**

Entry condition

●Either Ranked Game or Casual Game is being played

Flow of events

●Counter for number of moves made is increased by one and the display of the counter is updated accordingly

Exit condition

●Player finishes the level or s/he gives up

Use case name **Keep Time**

Entry condition

●Either Ranked Game or Casual Game starts

Flow of events

●The time passed since the start of the game is counted and displayed on the game screen

Exit condition

●Player finishes the level or s/he gives up

Special requirements

●The time is paused during the payment process

Use case name **Make Payment**

Participating actors

●Player, POS Service

Entry condition

●Player wants to get hint(s) during level selection

Flow of Events

●Player selects how many hints is s/he going to purchase

●Player enters the information required for the purchase process

●Player approves the purchase process

Exit condition

●Hint(s) are purchased successfully

Use case name **Exit Game**

Participating actor

●Player

Entry condition

●Player gives up the game

Flow of events

●Player pushes the Exit Button

Exit condition

●When player approve the exit from the game the game is ended without success and the player is redirected to main menu

---

Use case name **Control Piece**

Participating actor

●Player

Entry condition

●Player plays the game

Flow of Events

●Player tries to fill the blanks on the board by selecting, flipping, rotating, dragging and placing the pieces

Exit condition

●Player makes an attempt to place a piece onto board

Use case name **Flip Piece**

Participating actor

●Player

Entry condition

●Player tries to place any of 12 pieces onto board

Flow of Events

●Player decides to flip a piece

●S/he left clicks on it

Exit condition

●The piece is flipped

Use case name **Drag Piece**

Participating actor

●Player

Entry condition

●Player tries to place any of 12 pieces onto board

Flow of Events

●Player decides to drag a piece

●S/he left clicks on the piece and moves the mouse while continuing to click

Exit condition

●Player stops to left clicking on the piece thinking that the location of piece is fine on the board or s/he gives up his/her attempt

---

Use case name **Rotate Piece**

Participating actor

●Player

Entry condition

●Player tries to place any of 12 pieces onto board

Flow of Events

●Player decides to rotate a piece

●S/he right clicks on the piece

Exit condition

●The piece is rotated by the angle of $\pi/2$ radian.

●Player can repeat the events till s/he thinks that the piece is suitable to place in that position

---

Use case name **Select Piece**

Participating actor

●Player

Entry condition

●Player tries to place any of 12 pieces onto board

Flow of Events

●Player decides on one of the pieces to make a placement attempt

●S/he right clicks on the piece and continue to click in order to drag it

Exit condition

●Player makes his/her attempt

Use case name **Place Piece**

Participating actor

●Player

Entry condition

●Player tries to place any of 12 pieces onto board

Flow of Events

●Player decides to make a placement attempt after dragging the piece s/he selected onto the place

●Player stops to right click on the piece

Exit condition

- The piece is placed on the board if the place is already empty, and the piece is close enough to a possible place or it continues to stay where it is left

---

Use case name **Play Casual Game**

Participating actor

●Player

Entry condition

●The Casual Game starts as a result of the player's decision

Flow of events

- The board and pieces are created and displayed on the screen, time counter and move counter are started and the data related to the player actions and needed start to be kept

Exit condition

●Player finishes the level successfully or s/he exits

Use case name **Use Hint**

Participating actor

●Player

Entry condition

●Maybe the player faces a difficulty and need a hint in order to continue to play the game

Flow of events

●Player clicks on the Use Hint button

●Suitable piece and place is indicated to the player according to already done placement

Exit condition

●Player is free to follow to instruction of hint but it is guaranteed that there exist a solution possible after the instruction is followed

Special Requirement

●Player must already have hints available or a screen for hint purchase is displayed

●Hint feature is only available in Casual Game Mode

Use case name **Purchase Hint**

Participating actor

●Player

Entry condition

●Player decides to purchase hint(s) and either clicks on the Purchase Hint button on the Main Menu or during the Casual Game

Flow of events

●Player selects how many hint(s) s/he wants to purchase

Exit condition

- Player pushes on the Continue to Payment button

Special Requirement

- Player can use the hint(s) s/he purchased in only Casual Game Mode

---

Use case name **Make Payment**

Participating actor

●Player, POS Service

Entry condition

●Continue to Payment button is pushed

Flow of events

●Player enters required credit card information

●Player pushes on the OK button

Exit condition

- Either player purchases the ordered hints successfully by entering valid credit card information in which there exist enough amount or s/he closes the window

Special Requirement

- Player can make only one attempt in a minute, and at most 3 attempts in 10 minutes

Use case name **Enter Credit Card Information**

Participating actor

●Player, POS Service

Entry condition

●Player tries to purchase hint(s)

Flow of events

●Player enters name on the credit card

●Player enters the surname on the credit card

●Player enters the credit card no

●Player enters the CVC on the credit card

●Player enters the expiry date of the credit card

Exit condition

●All required information is given to the corresponding blanks before pressing OK button (best case)

Use case name **Compose New Level**

Participating actor

●Player

Entry condition

●Player pushes to the 'Compose New Level' button on the main menu

Flow of Events

●Player tries to compose a new level by selecting, rotating, flipping, dragging and placing the 4 grounds.

Exit condition

●Player uploads the level to the database successfully or s/he gives up

Special Requirement

●Upload is approved only if the newly created level is complete and it is not similar with any of the already existing levels

---

Use case name **Upload Level**

Participating actor

●Player

Entry condition

●Player pushes on the Upload Level button

Flow of Events

●The newly created level is uploaded to the database after it is concluded by the program that it is complete and not similar with any of the already existing levels in the database

Exit condition

●Upload is done successfully

Use case name **Control Ground**

Participating actor

●Player

Entry condition

●Player starts to compose a new level

Flow of Events

●Player tries to create a complete new level by selecting, dragging, flipping, rotating and placing all boards

Exit condition

●Player makes and attempt to place a ground

Use case name **Flip Ground**

Participating actor

●Player

Entry condition

●Player tries to place any of 4 grounds

Flow of Events

●Player decides to flip a ground

●S/he left clicks on it

Exit condition

●The ground is flipped

Use case name **Drag Ground**

Participating actor

●Player

Entry condition

●Player tries to place any of 4 grounds

Flow of Events

●Player decides to drag a ground

●S/he left clicks on the ground and moves the mouse while continuing to click

Exit condition

●Player stops to left click on the ground thinking that the location of the ground is fine or s/he gives up his/her attempt

---

Use case name **Rotate Ground**

Participating actor

●Player

Entry condition

●Player tries to place any of 4 ground

Flow of Events

●Player decides to rotate a ground
●S/he right clicks on the ground
Exit condition
●The ground is rotated by the angle of **π**/2 radian.
●Player can repeat the events till s/he thinks that the ground is suitable to place in that position

---

Use case name **Select Ground**
Participating actor
●Player
Entry condition
●Player tries to place any of 4 grounds
Flow of Events
●Player decides on one of the grounds to make a placement
●S/he right clicks on the ground and continue to click in order to drag it
Exit condition
● Player makes his/her attempt

Use case name **Place Ground**
Participating actor
●Player
Entry condition
●Player tries to place any of 4 grounds
Flow of Events
●Player decides to make a placement attempt after dragging the ground s/he selected
●Player stops to right click on the ground
Exit condition
●The ground is placed if the place is already empty or it continues to stay where it is left

6.2. Object and Class Model

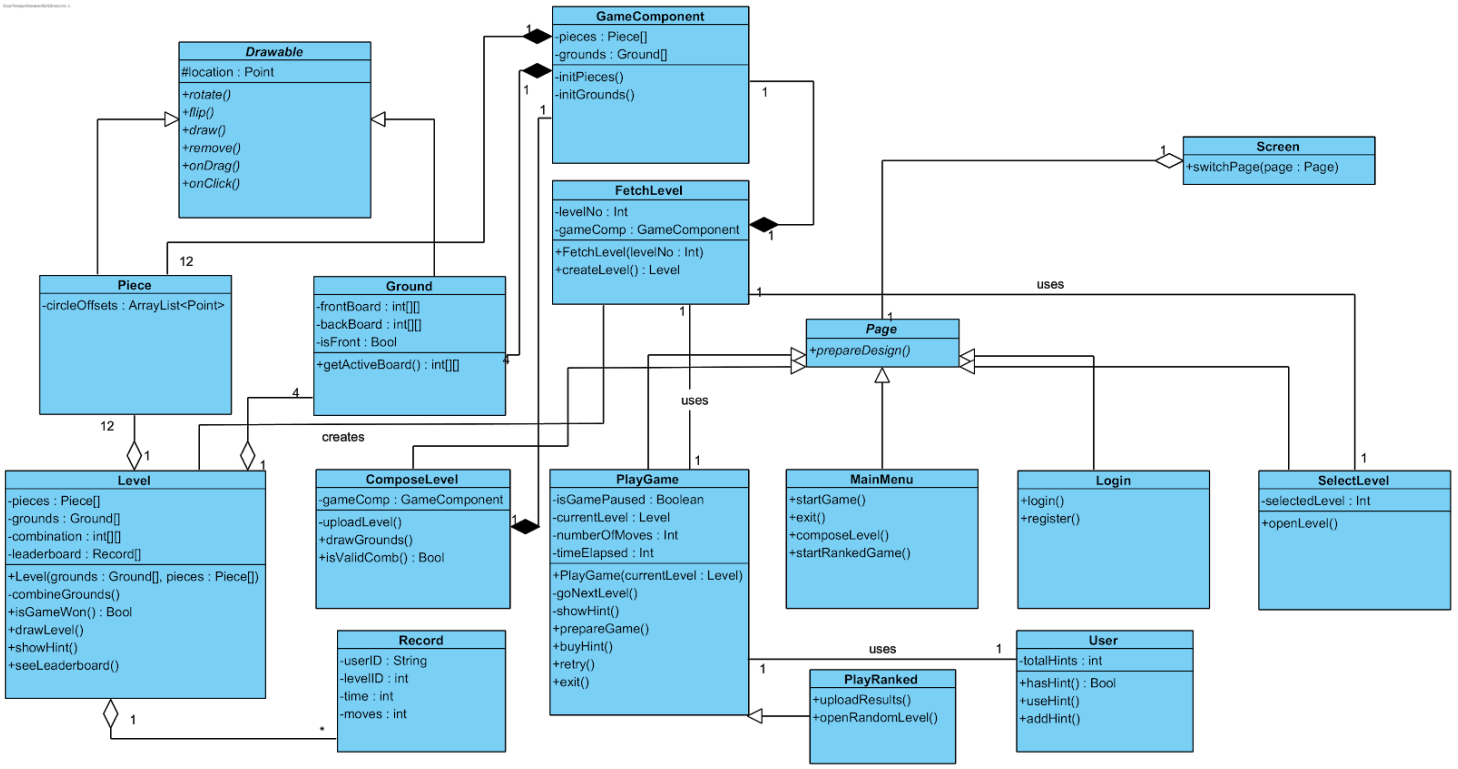

Figure 2: Class Diagram

6.2.1. Drawable Class

This class is defined to represent the main components of the game; namely, Piece and Ground which are subclasses. It has only one attribute.

1. location : Location of the component on the screen.

6.2.1.1. Piece Class

This represents pieces of the Quadrillion game which consist of marbles that are tied to each other in various ways. This class has one attribute:

1. circleOffsets - It holds the information of how marbles are positioned thereby defining the structure of the piece.

6.2.1.2. Ground Class

This class represents ground (there are 4 of them in Quadrillion). Each ground has two faces which have different occupied* places. There are three attributes:

1. frontBoard - It holds the information of which cells of the front side of ground is occupied.
2. backBoard - It holds the information of which cells of the back side of ground is occupied.
3. isFront - It checks whether front face is used or not.

* “occupied place” refers to a cell on the ground being closed so that no marble can be put there.

6.2.2. Level Class

This class is responsible for holding information about the level such as how grounds are oriented and how they are connected to each other. It has 4 attributes:

1. pieces - The pieces that will be used in the level.
2. grounds - The grounds that will be used in the level.
3. combination - The matrix that will be built by using grounds (grounds). It brings all grounds together an acts as a single board.
4. leaderboard - It is an array of records and it holds information about the best performers of the level.

6.2.3. GameComponent Class

Since pieces and grounds are all prefabricated, this class will be responsible for fabricating them. It holds *pieces* and *grounds* it fabricated as attributes.

6.2.4. Page Class

This class holds the layout of the user interfaces. In the project, there are 6 scenes that are subclasses of this class and that player can interact with.

6.2.4.1. PlayGame Class

Player actually plays the game here. This class has four attributes:

1. isGamePaused - It checks whether game is paused.
2. currentLevel - It holds the level currently being played.
3. timeElapsed - It counts the time elapsed until sloving the puzzle.
4. numberOfMoves - It counts the number of times player makes a move.

6.2.4.1.1. PlayRanked Class

This class extends PlayGamce class. Player can also play the game competitively. numberOfMoves and timeElapsed are uploaded in order to rank the player.

6.2.4.2. MainMenu Class

This class represents the Main Menu where player can start or exit the game.

6.2.4.3. ComposeLevel Class

This class represents the level composition page. By merging predefined grounds in various ways, players can create new levels and upload it. It has single attribute:

1. gameComp: It will be used to draw Grounds (grounds).

6.2.4.4. Login Class

Players can register and login to the system through this page.

6.2.4.5. SelectLevel Class

This class is responsible for selection of level to be played. It holds the level selected by the player as an attribute; namely, *selectedLevel*.

6.2.5. FetchLevel Class

This class fetches the information of a level from database or a text file and creates a new level. It has two attributes:

1. levelNo - Id of the level that will be fetched

2. gameComp - It holds the copies of game componentes that are Pieces and Grounds and will be used to create level.

6.2.6. Screen Class

This class is responsible for creating main window of the game and switching between pages.

6.2.7. User Class

This class holds number of hints available for the player and regulates the hint addition and subtraction.

6.2.8. Record Class

This class represents records in the leaderboard. It holds how many moves player made and how much time it took for him/her to solve the level as well as unique user number and level number.

6.3. Dynamic Models

6.3.1. Sequence Diagram

6.3.1.1. Play Casual

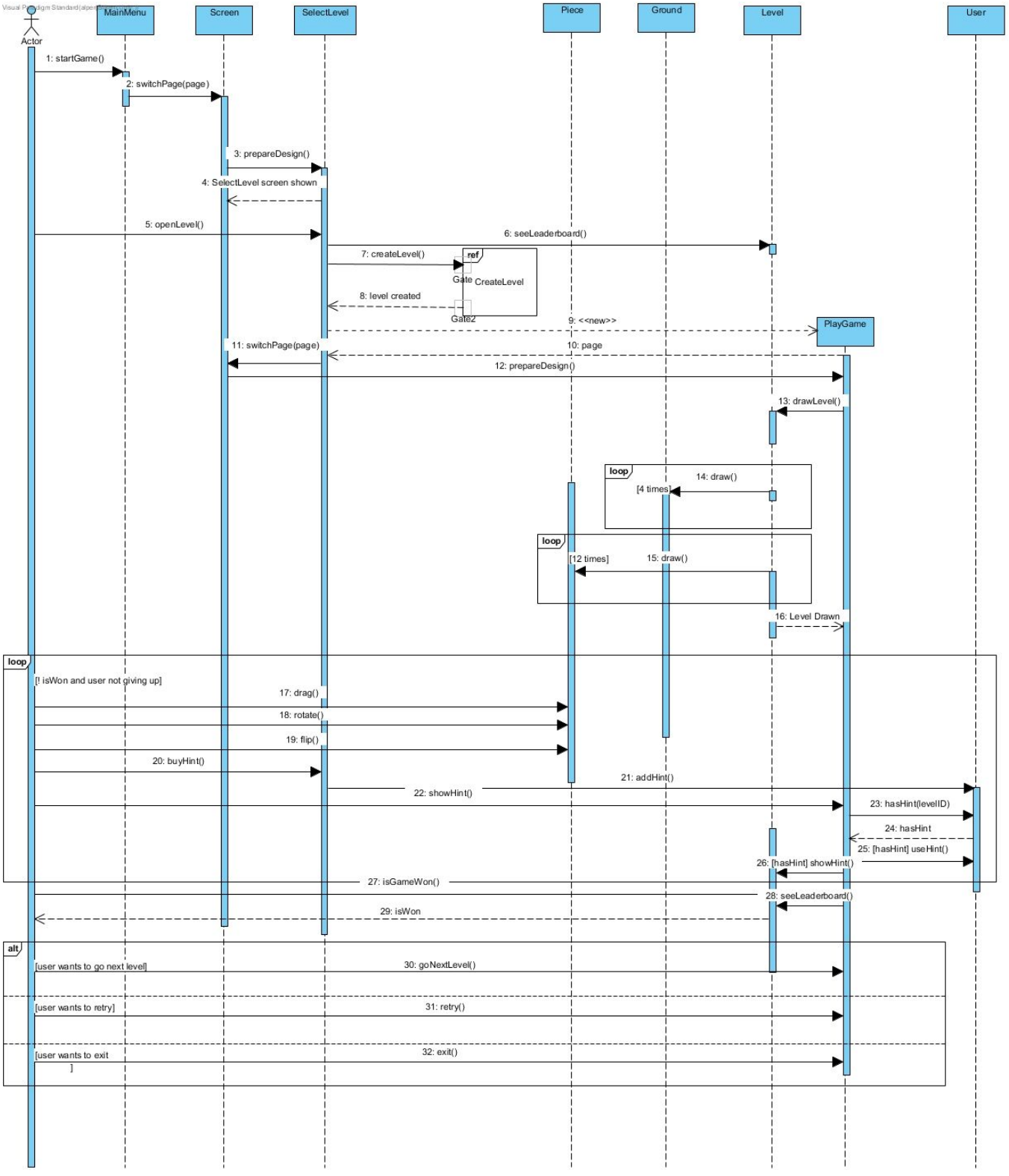

Figure 4: Sequence diagram for Casual game

Player starts playing casual game and level selection screen shows up in front of him/her. Then, he/she opens a level using Select Level class. Select Level class then creates level by communication with CreateLevel sequence diagram. Created level is returned back to the Select Level class. Select Level class creates new PlayGame class that is the environment to play the game and sends it to Screen class to show it on the screen. Screen class says PlayGame class to prepare itself for the gameplay. PlayGame class commands all Pieces and Grounds to be drawn on the screen and ready to be played. Then, player constantly drags and clicks pieces to complete the level. He/she can use hints. If user wants, he/she can buy hints. Then, the number of hints in the User class is updated. Whether or not he completed the level is controlled repeatedly. If he completes or gives up, leaderboard is shown to him. Finally, he can either retry or proceed to the next level.

6.3.1.2. Create Level

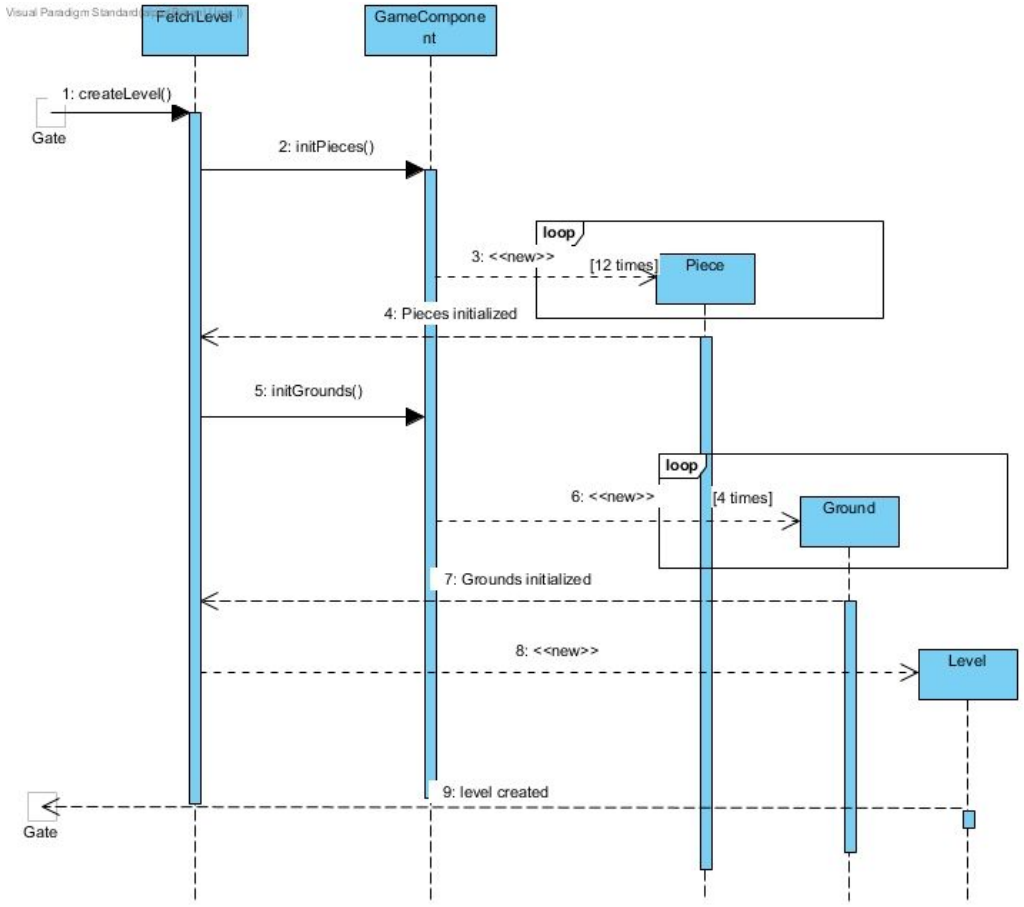

Figure 6: Sequence diagram for Create Level

When the level to be played is determined createLevel method of the FetchLevel class is called. Then initPieces and initGrounds methods of the GameComponent class is called so that 12 pieces and 4 grounds of the corresponding level can be created. After all the pieces and grounds are created an instance of Level class is created with given pieces and grounds. Finally created level is returned to the caller of createLevel.

6.3.1.3. Play Ranked

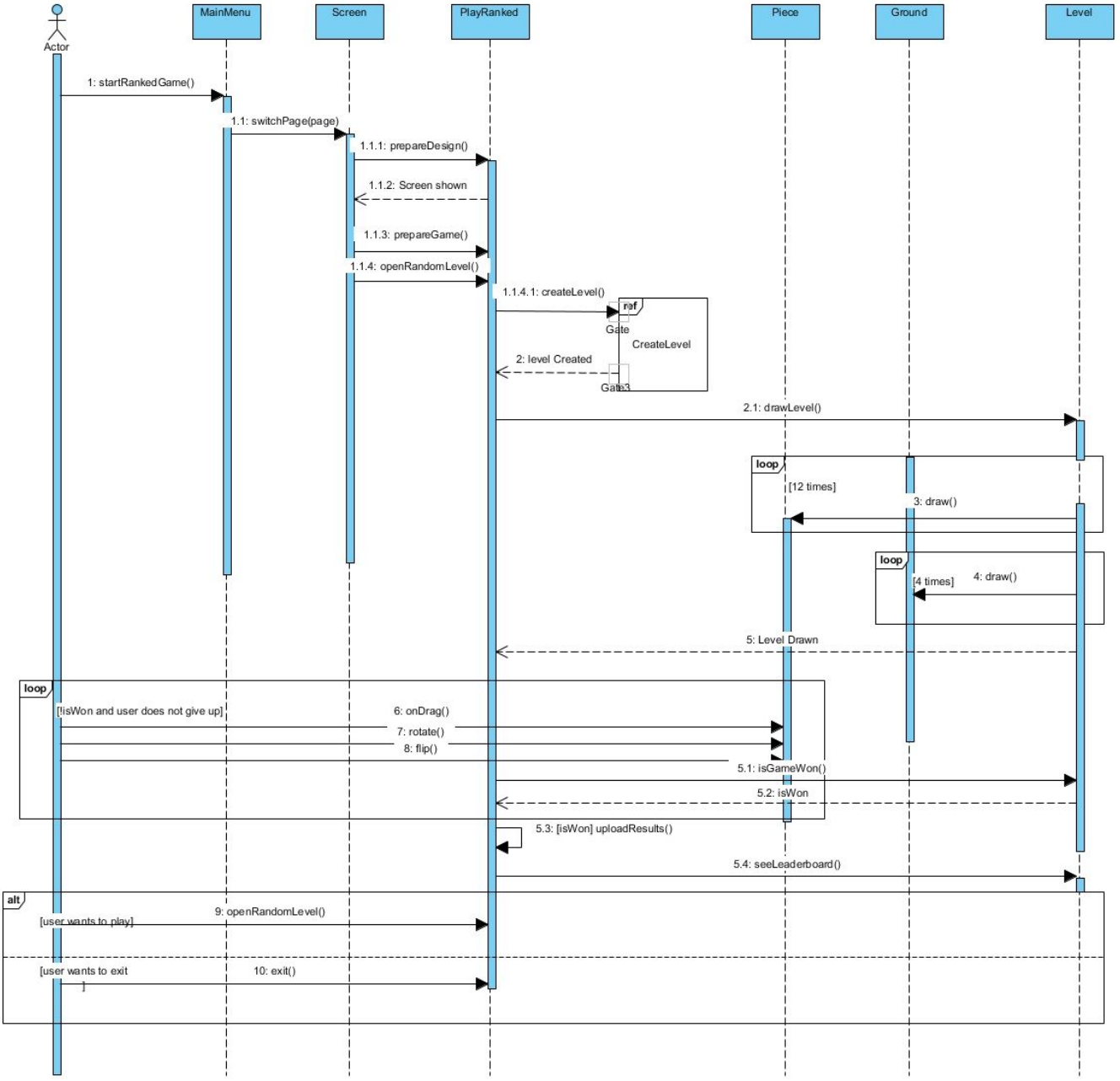

Figure 5: Sequence diagram for Ranked Game

Player starts ranked game through MainMenu class. MainMenu tells Screen class to show Ranked Game screen. Screen induces PlayRanked class to prepare itself and open a random level since player will be ranked based on the performance. Then, PlayRanked class creates level through Create Level sequence diagram. After that, PlayRanked class tells all grounds and draws to be drawn on the screen. Unless player gives up or completes the level, he/she can move pieces to complete the level. Constantly game board is checked to control whether level is completed. If level is completed, then player results are uploaded for ranking purposes. Finally, player sees the leaderboard and then he/she either plays new level or exits.

6.3.1.4. Compose Level

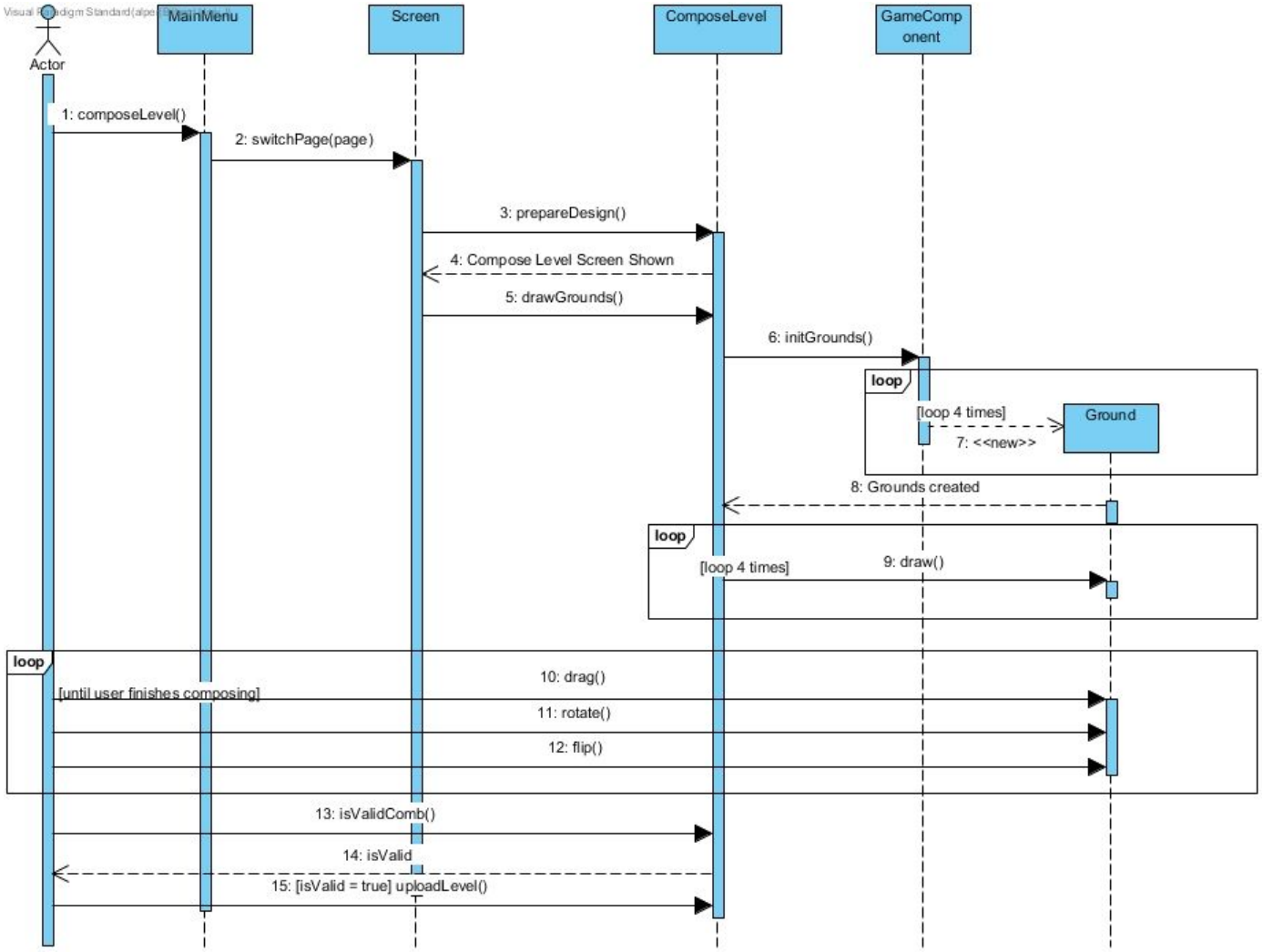

Figure 7: Sequence diagram for Compose Level

Player can access compose level screen through main menu by pressing Compose Level button. Pressing the button triggers the switchPage method of Screen object. Then Screen object requests the specific design of the compose level screen by prepareDesign method. After changing the screen grounds that will be shown in the screen are requested from ComposeLevel class. With the help of GameComponent's initGrounds method 4 grounds that every level must have are created. Then these grounds are drawn to the screen with a loop. When the compose level screen is set, player can try new combinations with grounds by dragging them on screen and clicking to flip it. Finally, player can click on submit button which will first check the validity of the composed level then check if the level already exists. If everything is alright the level will be added to the game.

6.3.2. State Diagram

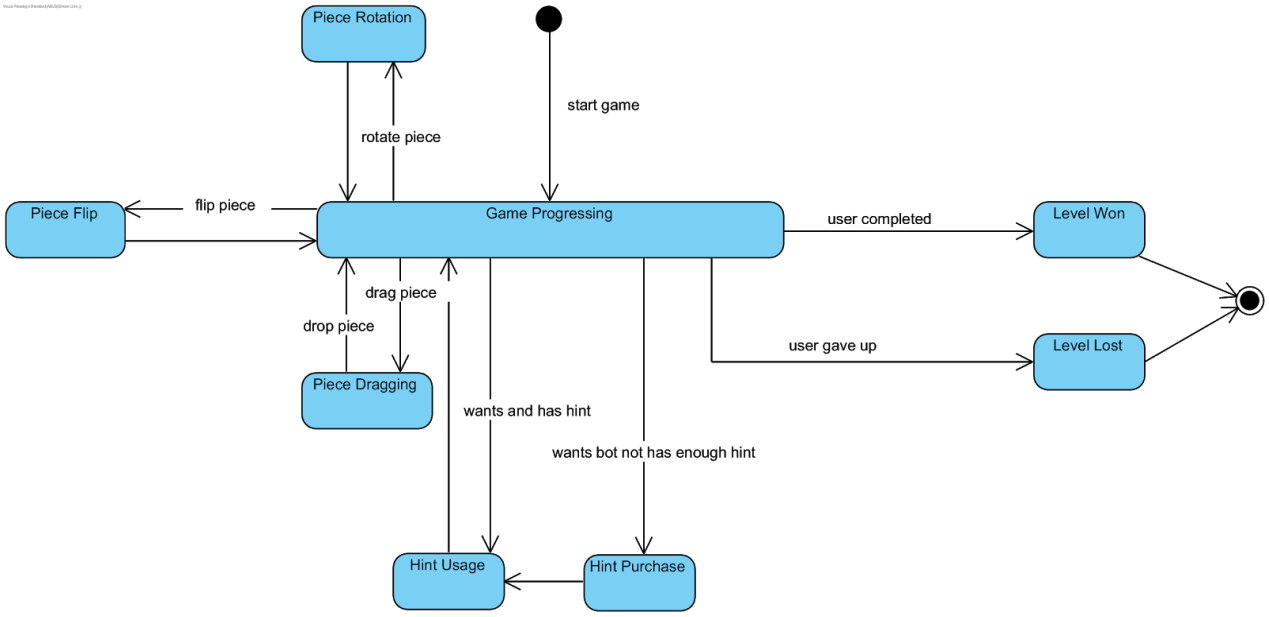

Figure 8: State diagram for Winning and Losing condition

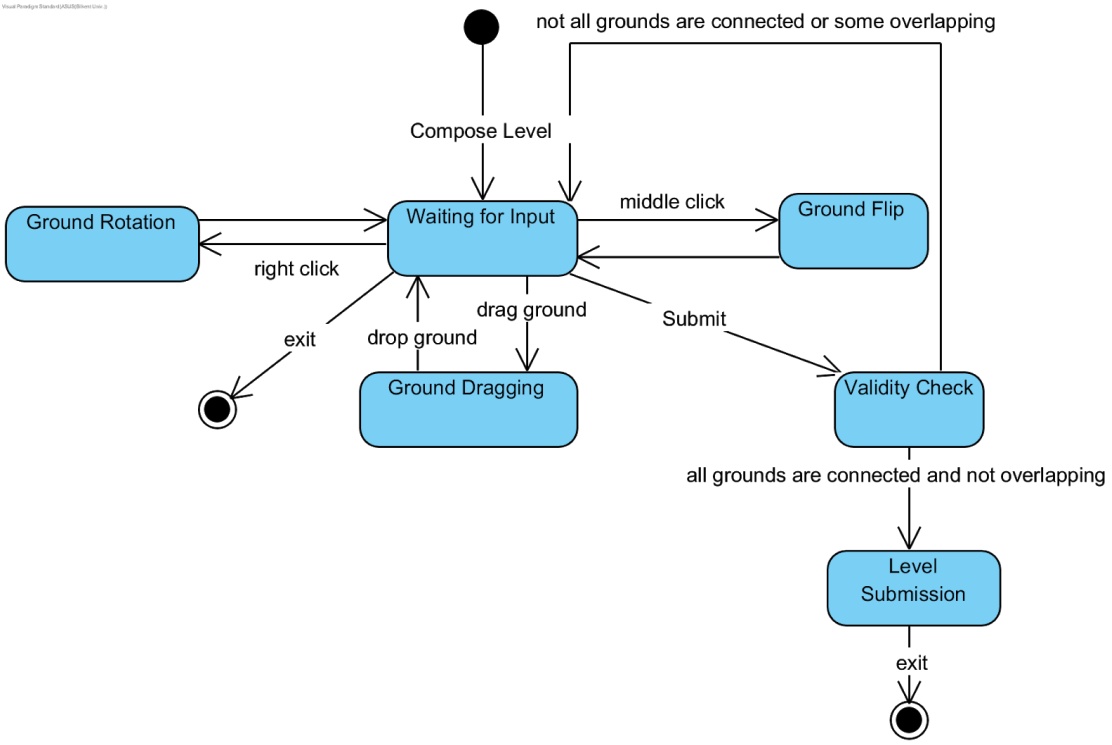

Figure 9: State diagram for Compose Level

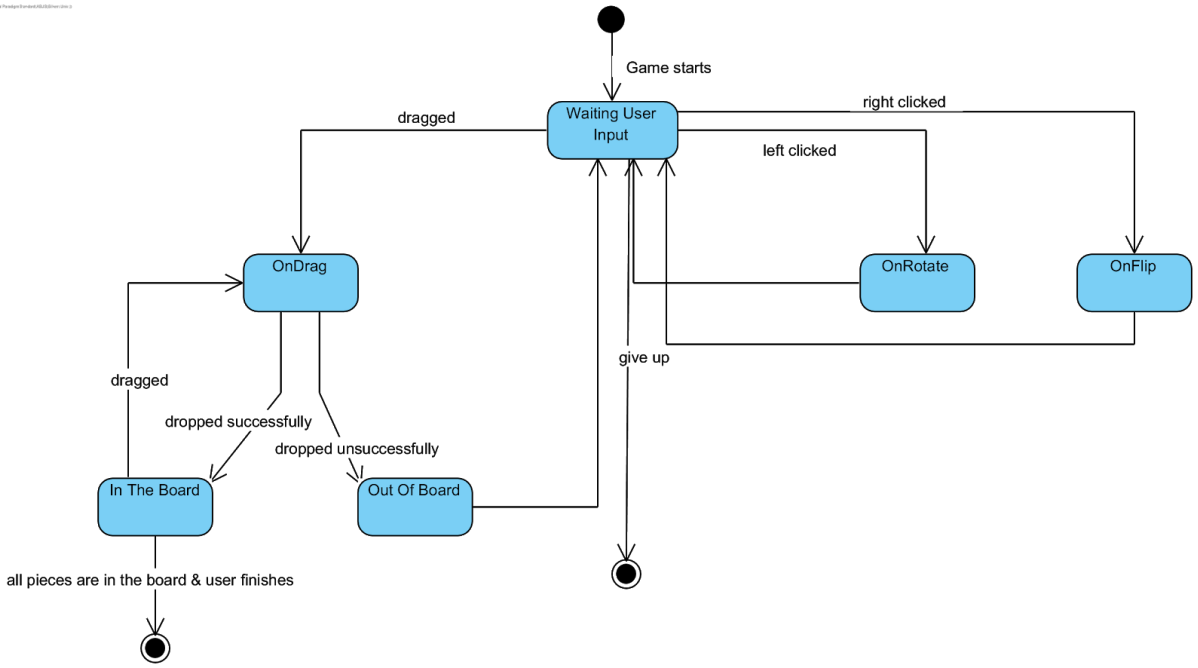

Figure 10: State diagram for Piece objects in game

6.3.3. Activity Diagram

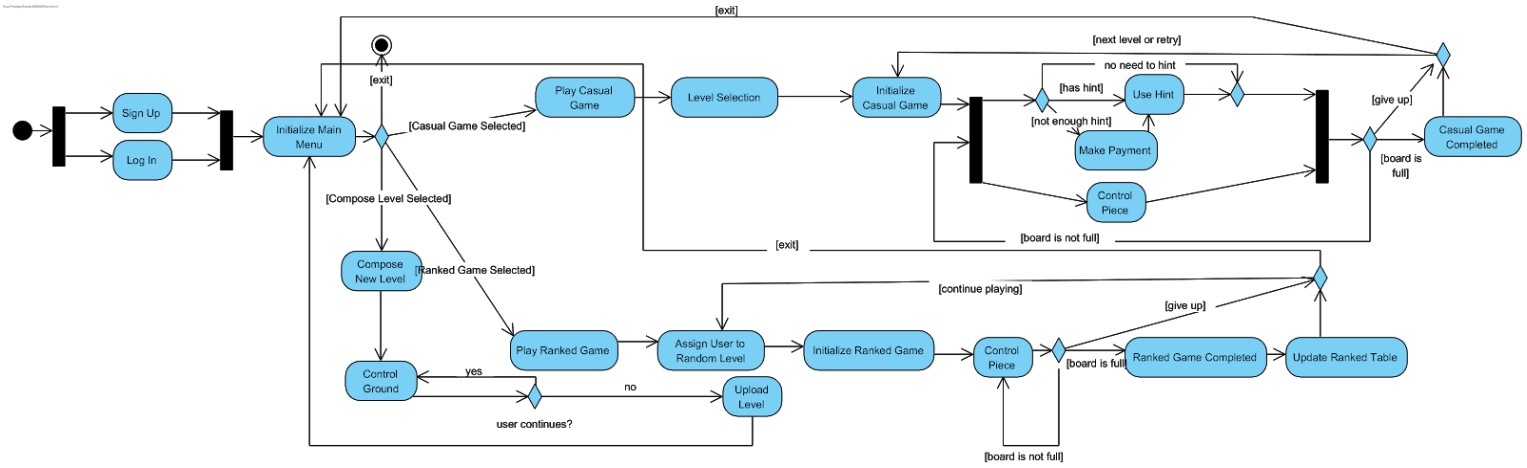

Figure 11: Activity diagram

At the beginning, player signs up or logs in. Then, main menu screen shows up. There are 4 choices from here:

1. Play Casual Game: In that case Level Selection screen is shown and here player has the only chance to buy hints. After player selected the level, game is initialized. He repeatedly, modifies the piece (by rotating, flipping and moving) and he can use hint in the meantime. When board is full game is completed. Player can give up playing the level whenever s/he wishes. Then, s/he can play again or quit to the main menu.
2. Play Ranked Game: First, player is assigned to a random level then, game is initialized and area shows up. Then player can modify pieces until board is full or he/she gives up. Finally, he/she can either continue playing or return to the main menu. If s/he successfully completes the level, his/her rank is uploaded to the server.

3. Compose New Level: Player can repeatedly modify grounds (rotate, flip, move) to prepare his/her own unique level. Then, s/he can attempt to upload the level. This activity includes checking for the validity of the level. If the level is valid upload process is successful.

4. Exit: Player can exit the game.

6.3.4. User Interface

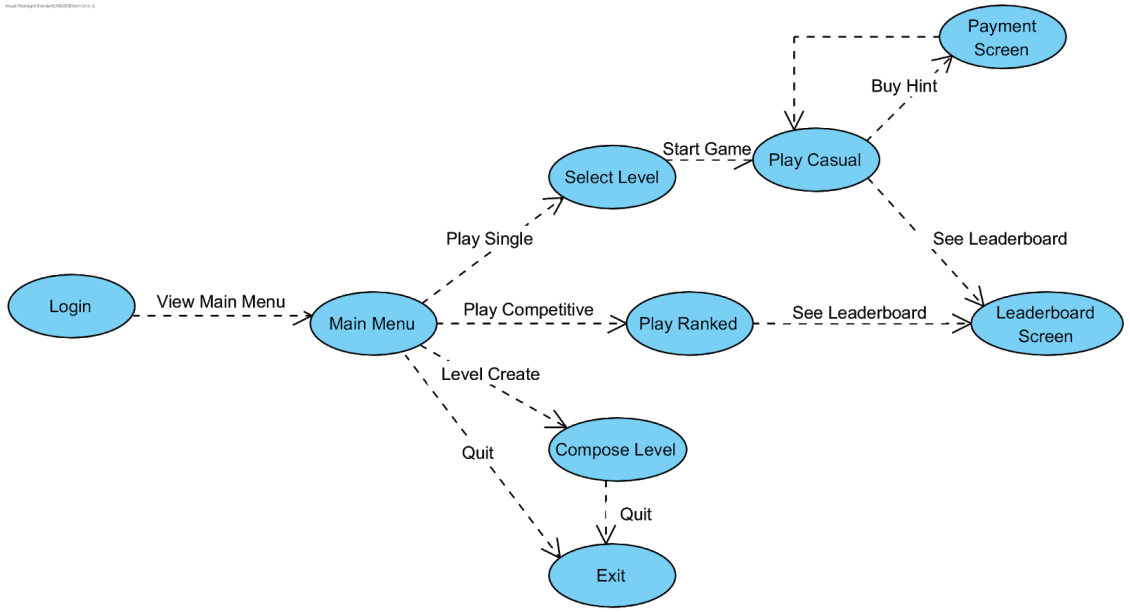

Figure 12: Navigational Path

Figure 13: Login Screen

Figure 14: Main Menu Screen

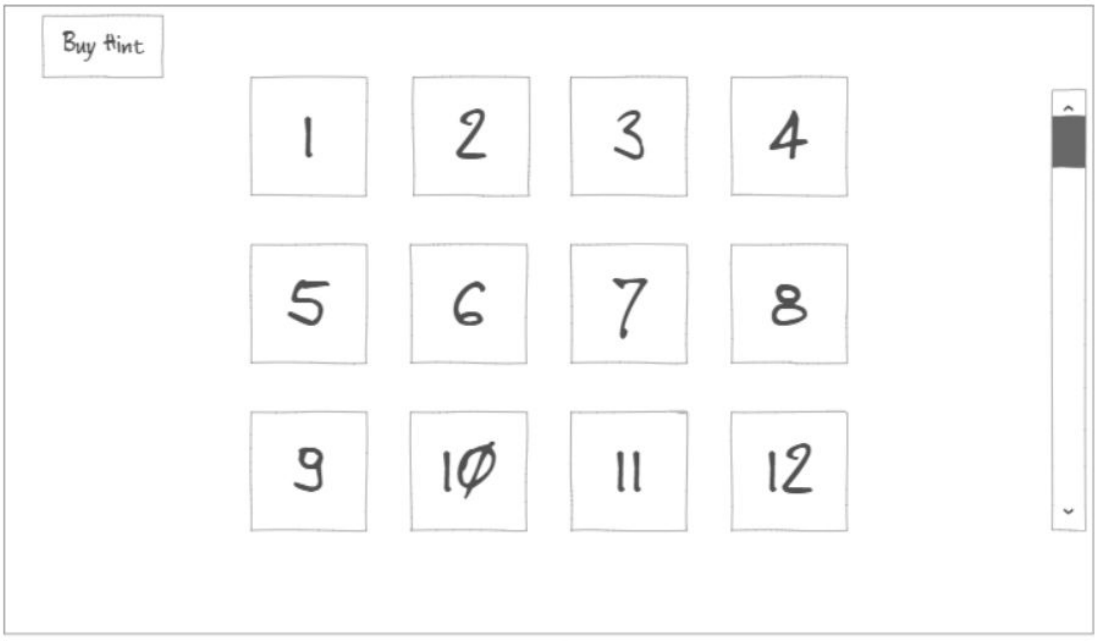

Figure 15: Select Level Screen

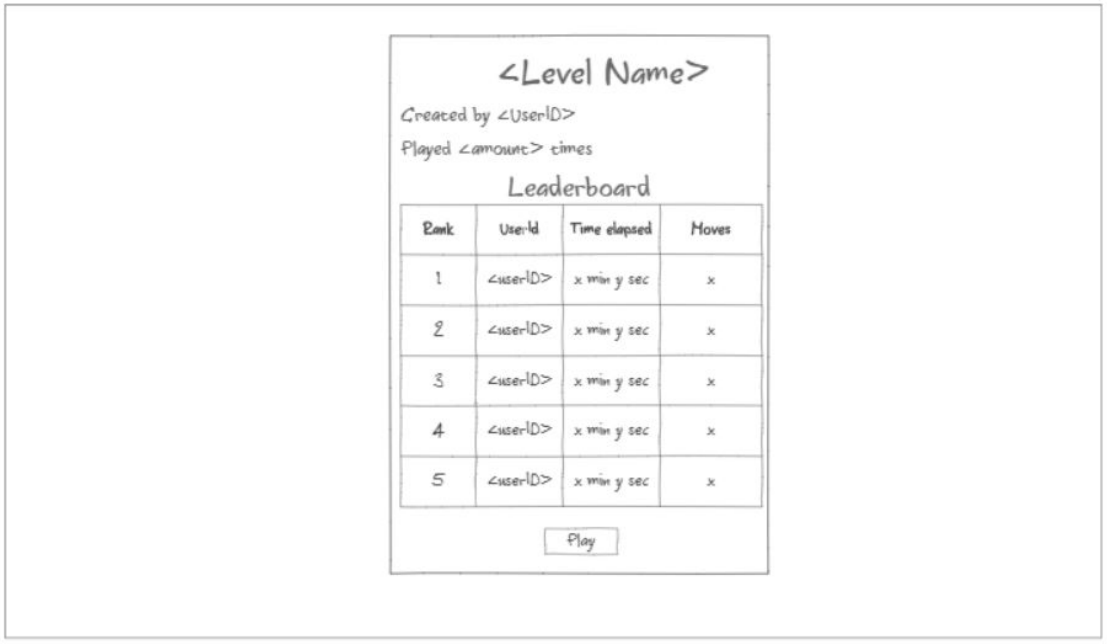

Figure 16: Leaderboard Screen (Pop-up window)

Figure 17: Hint Purchase Screen (Pop-up window)

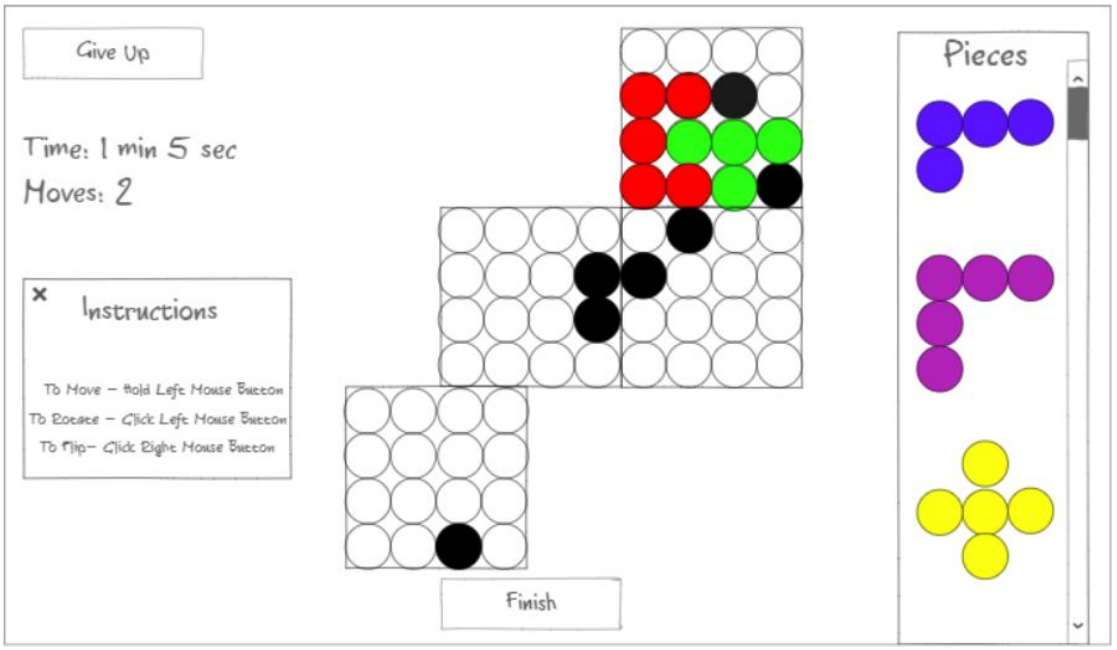

Figure 18: Play Casual Screen

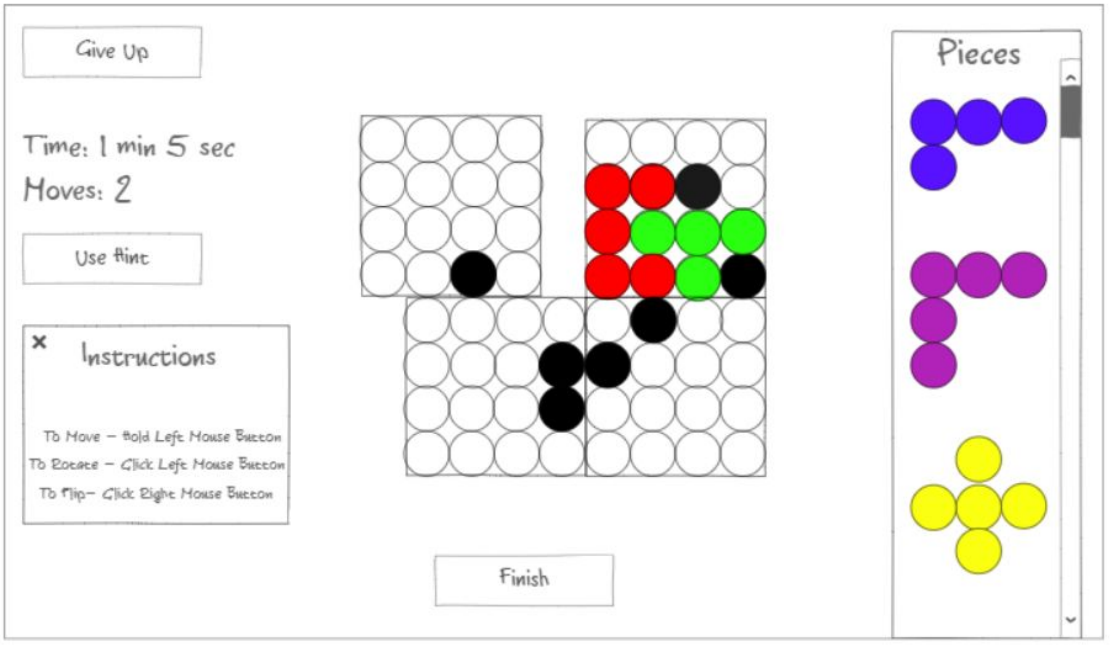

Figure 19: Play Ranked Screen

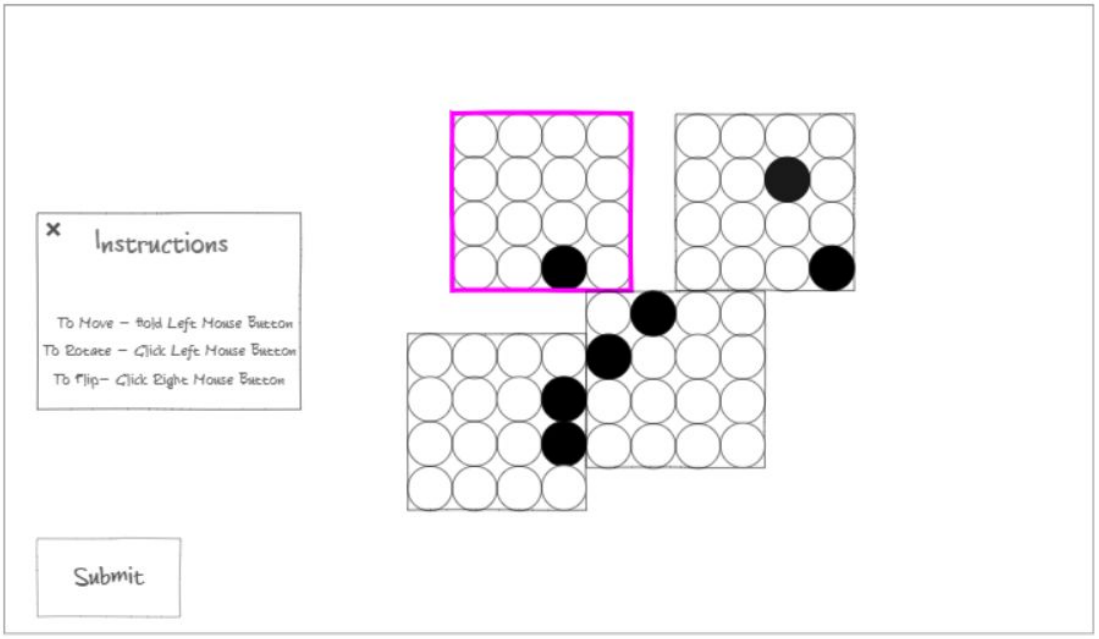

Figure 20: Compose Level Screen

7. Improvement Summary

- Non-functional requirements are replaced with testable/quantifiable requirements.
- Use case diagram, state diagrams, sequence diagrams and activity diagrams are slightly improved.

8. References & Glossary

Ground: It is a 4x4 sub-board.
Board: It is the combination of 4 Grounds (4x4 sub-boards).

http://www.smartgamesandpuzzles.com/inventor/Quadrillion.html