|  |
| --- |
| **BILKENT UNIVERSITY**  **COMPUTER ENGINEERING DEPARTMENT** |
| **CS 319 – OBJECT ORIENTED SOFTWARE ENGINEERING**  **SECTION 1 – GROUP 7**  **PROJECT TITLE: CRAZY SHOOTER**  **ANALYSIS REPORT** |
|  |
| TURAN KAAN ELGİN 21302525  ONUR UYGUR 21202165  NİHAT ARDA ŞENDUR 21101240  BENGÜ AKBOSTANCI 21100630 |
|  |
|  |

|  |
| --- |
|  |

Contents

[**1.** **Introduction** 3](#_Toc433649682)

[**2.** **Requirement Analysis** 3](#_Toc433649683)

[**2.1** **Overview** 3](#_Toc433649684)

[**2.1.1** **Gameplay** 4](#_Toc433649685)

[**2.1.2** **Leveling** 4](#_Toc433649686)

[**2.1.3** **Screen Elements** 4](#_Toc433649687)

[**2.1.4** **Game Store** 5](#_Toc433649688)

[**2.2** **Functional Requirements** 5](#_Toc433649689)

[**2.2.1** **Login** 5](#_Toc433649690)

[**2.2.2** **Play Game** 5](#_Toc433649691)

[**2.2.3** **View High Scores** 6](#_Toc433649692)

[**2.2.4** **View Help** 6](#_Toc433649693)

[**2.2.5** **Purchase Special Balls and Lives** 6](#_Toc433649694)

[**2.3** **Nonfunctional Requirements** 7](#_Toc433649695)

[**2.3.1** **Proper Animations** 7](#_Toc433649696)

[**2.3.2** **Response Time** 7](#_Toc433649697)

[**2.3.3** **User-friendly Interface** 7](#_Toc433649698)

[**2.3.4** **Extendibility** 7](#_Toc433649699)

[**2.4** **Constraints** 7](#_Toc433649700)

[**2.5** **Scenarios** 8](#_Toc433649701)

[**2.5.1** **Login** 8](#_Toc433649702)

[**2.5.2** **Create Account** 8](#_Toc433649703)

[**2.5.3** **Start Game** 8](#_Toc433649704)

[**2.5.4** **Play Game #1** 8](#_Toc433649705)

[**2.5.5** **Play Game #2** 8](#_Toc433649706)

[**2.6** **Use Case Models** 9](#_Toc433649707)

[**2.6.1** **Login** 10](#_Toc433649708)

[**2.6.2** **LoginAsGuest** 11](#_Toc433649709)

[**2.6.3** **CreateNewAccount** 11](#_Toc433649710)

[**2.6.4** **DeleteAccount** 12](#_Toc433649711)

[**2.6.5** **StartGame** 13](#_Toc433649712)

[**2.6.6** **PlayGame** 13](#_Toc433649713)

[**2.6.7** **ViewHelp** 15](#_Toc433649714)

[**2.6.8** **Pause** 15](#_Toc433649715)

[**2.6.9** **ViewHighScores** 16](#_Toc433649716)

[**2.6.10** **PurchaseItem** 16](#_Toc433649717)

[**2.7** **User Interface** 18](#_Toc433649718)

[**2.7.1** **Navigational Path** 18](#_Toc433649719)

[**2.7.2** **Screen Mock-ups** 19](#_Toc433649720)

[**2.7.2.1** **Login** 19](#_Toc433649721)

[**2.7.2.2** **Main Menu** 23](#_Toc433649722)

[**2.7.2.3** **Special Balls** 28](#_Toc433649723)

[**3.** **Analysis** 28](#_Toc433649724)

[**3.1** **Object Model** 28](#_Toc433649725)

[**3.1.1** **Domain Lexicon** 28](#_Toc433649726)

[**3.1.2** **Class Diagram** 29](#_Toc433649727)

[**3.1.2.1** **Entity Classes** 29](#_Toc433649728)

[**3.1.2.2** **Boundary Classes** 31](#_Toc433649729)

[**3.1.2.3** **Control classes** 31](#_Toc433649730)

[**3.2** **Dynamic Models** 32](#_Toc433649731)

[**3.2.1** **Activity Diagrams** 32](#_Toc433649732)

[**3.2.1.1** **Login** 32](#_Toc433649733)

[**3.2.1.2** **Play Game** 33](#_Toc433649734)

[**3.2.2** **State Chart Diagram** 34](#_Toc433649735)

[**3.2.3** **Sequence Diagrams** 35](#_Toc433649736)

[**3.2.3.1** **Login** 35](#_Toc433649737)

[**3.2.3.2** **Create Account** 36](#_Toc433649738)

[**3.2.3.3** **Start Game** 37](#_Toc433649739)

[**3.2.3.4** **Play Game #1** 38](#_Toc433649740)

[**3.2.3.5** **Play Game #2** 40](#_Toc433649741)

[**4.** **Conclusion** 43](#_Toc433649742)

1. **Introduction**

We will implement a video game which is called Crazy Shooter. It will be an alternative version of the games called Zuma and Luxor which were created and developed for IOS, Android and Desktop. The main aim of this game will be to destroy a sequence of balls, which is moving in a maze around a ball shooter, before it reaches to a hole. We have influenced from the following link in the project selection.

<http://www.popcap.com/games/zumas-revenge/online>

Crazy Shooter will have some different properties than those previous games. We have worked for creating ideas of bonuses, special ball types and different level styles.

In the game, the actual goal will be to destroy the sequence by combining balls with same colors. For this goal, players will throw balls to the sequence by controlling the ball shooter. The game will be composed of ten levels and difficulty of the game will increase level by level. Whenever players complete a level successfully, they will have a chance to play the next level. Each player will also have a unique account which keeps their progress in the game.

Crazy Shooter will be a Desktop application and flow of the game will be managed by using mouse and some keyboard shortcuts. The game may be improved for mobile environment in the future.

1. **Requirement Analysis**
   1. **Overview**

Crazy Shooter is a kind of Desktop video game. It is straightforward to play and enjoyable. In the game, there is a ball sequence moving in a maze, which is composed of different colors of balls and players’ purpose is to destroy the sequence by making combinations of balls with same colors. For this purpose, they control a shooter which throws the balls from the middle of the screen. If they are not able to destroy the sequence on time, it will go through a hole and they lose. Every successful trial helps players to finish the level faster and their scores will be recorded according to the times they finish the level. Crazy Shooter provides some opportunities for players to enjoy the game more. Sometimes a coin appears in the corner of the screen and it is a gift for players for purchasing some special items from the store of the game. (See Section 2.1.4) Besides, at the beginning of the game, players encounter a menu which directs them to create their accounts with their usernames and passwords and their names can appear in the high score table of the game. The aim is to play all levels successfully and complete them as much as fast.

* + 1. **Gameplay**

Players use the mouse to play the game. Mouse is dedicated to rotate the shooter and throw the balls. Also, when a coin appears in the screen, players have to click on it to get it. However, keyboard is also assigned for some specific purposes such as some shortcuts. For example, whenever players want to freeze the game, it will be sufficient to press the “Escape” button from the keyboard. Also, they use the menus and login screens of the game with both mouse and keyboard.

* + 1. **Leveling**

Crazy Shooter has 10 different levels. Difficulty increases level by level such that the ball sequence has more balls and move faster in each level. Players do not have a chance to access all levels such that whenever they complete one level successfully, they have a chance to play the next level. In the beginning of the game, they are able to choose any level they have come before to start with. Indeed, whenever players finish a level, they can play that level again to improve their high scores. Game has three checkpoints which are at the end of 3rd, 6th and 9th levels. When players lose all of their lives, they start from the last checkpoint. 10th level does not have a condition with lives; however it will be more difficult to play according to the first 9 levels as it is the final level.

* + 1. **Screen Elements**

Balls are designed with 5 different colors. Colors are blue, red, yellow, green and white. For destroying some parts of the sequence, users must combine 3 or more balls with same color. Moreover, 3 types of special balls are assigned for private functions. They are called as freeze ball, bomb ball, and back ball. These balls have colors as other balls, but they have some identifiers like a letter or symbol on them which can be seen in the User Interface part. (Figure 2.7.16) Freeze balls have an ability to freeze the sequence and timer for a while. Bomb balls can destroy more balls than the others by creating an explosion. Back balls are designed to rewinding the sequence. These types of special balls must be combined with 3 or more balls with the same color for using their abilities. If players throw them to wrong parts of the sequence, they lose their abilities. These specific balls make the game more enjoyable.

Sequences include balls with different colors. Whenever players make a successful trial, a number of balls in a sequence are diminished. If players make an unsuccessful trial, the ball they shoot is also added to the sequence, so the level becomes harder to pass.

There is a maze in the level screen which covers the ball sequence. At the end of the maze, there is a hole. If the sequence enters the hole, players lose the level.

In a level, a coin appears in the screen with a random value from 1 to 100 at random times. It remains in the screen for 5 seconds and for gaining it, players must realize it and click on it at that time interval. If they are able to take it, the game adds it to their accounts and they can use it in the game store.

* + 1. **Game Store**

There is a page in the game called game store from which players are able to purchase special balls and lives by using their coins. When they purchase a special ball, game adds it to their accounts, and they become able to use it during the game whenever they want. They can use them by clicking on a shortcut and the ball of the shooter becomes that type. When they purchase a live, they can decrease their chance of falling to the last checkpoint, however lives are more expensive than the balls, because they are more valuable.

* 1. **Functional Requirements**
     1. **Login**

There are two options for entering to the game. One of them is having an account, and the other is entering as a guest. Players are able to create their accounts by specifying their usernames and passwords, and the system gives them a unique id. Later, when they enter their ids and passwords, they can play the game, and the system saves their progress. Usernames are saving their names to the high score table as previously told. If they want to play the game one time for trial, they can choose the option of login as a guest, and the system does not save their progress and they are not able to use the store.

* + 1. **Play Game**

Crazy Shooter is an alternative version of the games called Zuma and Luxor. The game is like the following: There is a shooter in the middle which has abilities to rotate and throw a ball. The purpose is to destroy a ball sequence, which is moving in a maze around the shooter, by throwing balls to it. The balls have different colors and when three or more balls with the same color come together, they are destroyed. At the end of the maze, there is a hole, and players must destroy the sequence before it reaches to the hole.

During the game, shooter will have some special balls which are bomb, back and freeze balls. Bomb balls have the ability to destroy more balls, so they have a greater impact. Back balls have the ability to make the sequence rewind for a while, so it makes it be more away from the hole. Freeze balls freeze the sequence and timer for a while such that players can destroy the sequence while it is not moving. These balls come either at random, or players can purchase them from the game store and use it whenever they want during the level.

Also during the game, a coin will appear at the corner in the screen periodically with a random value through 1 to 100. However, the chance of its value to be larger than 50 is less than the chance to get smaller values. The coin is small, and its time of appearance is about 5 seconds. If players can realize it and click on it on time, they can earn it for using in the store.

The game has 10 levels and there are checkpoints in 3rd, 6th and 9th level. Players will have 3 lives and when they use all of them, they begin from the last checkpoint, so their progress in the levels which are after the checkpoint is deleted. When they are starting the game, there is a level table which they can select a level to start from. However, it must be a level which they have come before in their progress. Also, there is a timer in each level for recording the time players finish it for keeping the high scores.

* + 1. **View High Scores**

The system keeps the scores inversely proportional to time as (1 / time) \* 10000. There is one high score table for all levels, and the rank is like the following: First, it groups the players according to their progress. (Players who have passed Level 2 are higher than the ones who have passed Level 1.) Then, it groups them according to their scores. Players are able to view the first five scores and their ranks. They can select the option from the Main Menu to see the score table.

* + 1. **View Help**

Players are able to get help about the game during the level. When they select that option, a frame appears which have the instructions about the game.

* + 1. **Purchase Special Balls and Lives**

There is an option called “Store” in the Main Menu which directs the players to a store from which they can purchase special balls (bomb, back and freeze) and also lives. When they purchase a ball, they can make the ball of the shooter be that kind of special ball during the game. However, they must have enough coin to buy them. Also, they can purchase lives for increasing their chances of losing, but lives are more expensive than the balls.

* 1. **Nonfunctional Requirements**
     1. **Proper Animations**

In the game, the ball sequence moves in the maze, and the shooter has an ability to rotate. These animations should be proper such that they should occur in enough speed without any errors. Also, when shooter throws a ball, the ball must reach to the target without changing its direction. We will make these animations as proper as possible by using some mathematical libraries and classes if necessary.

* + 1. **Response Time**

Response time of the game for user inputs should be as fast as possible. Some examples for this are creating an account, finding an account, throwing a ball and so on. Especially in concurrent events, the time interval between the events should not be seen by the players such that updating the ball sequence and creating a sound after throwing a ball. Fulfilling this requirement is important for managing the performance criteria of the game.

* + 1. **User-friendly Interface**

We will try to make the user interface of the game as user-friendly as possible such that players can be able to play the game easily. Adding some shortcuts to the game is an example for properties of the interface. User-friendliness of the interface is an important criterion for usability of the game.

* + 1. **Extendibility**

The game should be extendible such that developers should easily update it. For that purpose, we will separate the project into some parts which have separate functions such that updating the code will require only some of the classes to be changed. This requirement is important for the adaptability of the game for new application domain concepts.

* 1. **Constraints**
* The game must be a Desktop application.
* Implementation language must be Java.
* The game must not be an open-source product.
  1. **Scenarios**
     1. **Login**

**Scenario:** Player enters his id and password to the login screen and confirms his inputs. Then, System searches for the player from its database and finds him. Then it gets his score, progress and amount of coins and opens the game accordingly.

* + 1. **Create Account**

**Scenario:** Player selects “Create Account” option from the login screen. Then, System opens the create account screen. Player enters his username and password and confirms his inputs. Then, System adds his account to the database, and it creates a confirm frame which displays the confirmation message and player id. Player presses the confirm button to confirm his account.

* + 1. **Start Game**

**Scenario:** Player selects the “Start Game” option from the Main Menu. Then, System opens the Level Menu. Player selects the level which he wants to start. System opens the particular level and creates the screen.

* + 1. **Play Game #1**

**Scenario:** Player selects the “Start Game” option from the Main Menu and System does the previous “Start Game” scenario. Then, System starts the timer of the level. After that, the game loop starts. In the loop, at first Player rotates the shooter with mouse for determining where it will throw the ball. Then, he throws the ball by clicking on the screen. System updates the ball sequence and sees that there is a hit, so it creates a hit sound depending on the type of ball. Then, it checks the hole for if it contains a ball or not and it sees that it is empty. It checks if there are more balls in the sequence and it sees that the sequence is not fully destroyed yet. Then, the shooter prepares the next ball to throw and the sequence slides. At the end, System updates the level screen accordingly.

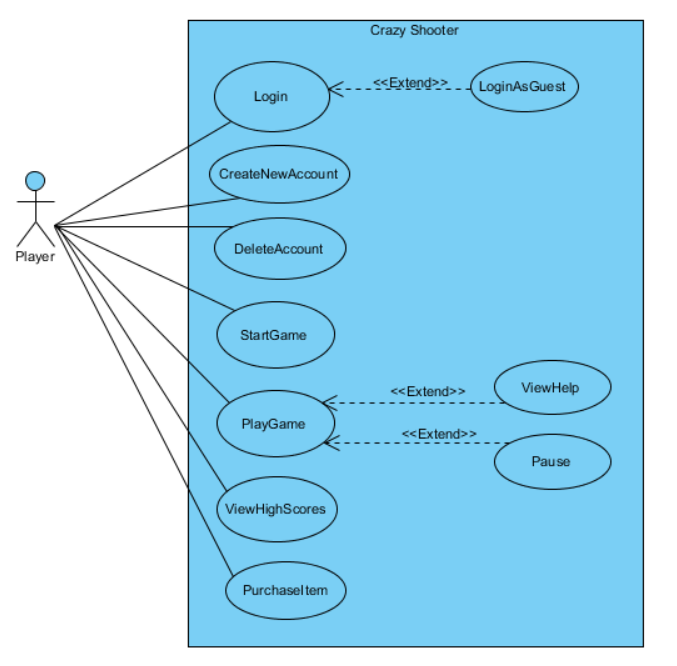
* + 1. **Play Game #2**

**Scenario:** Player selects the “Start Game” option from the Main Menu and System does the previous “Start Game” scenario. After that, the game loop starts. Then, System starts the timer of the level inside the loop since it will use a freeze ball. Player rotates the shooter. After that, Player changes the type of the ball, which is to be thrown, to a freeze ball. After updating the sequence and seeing the hit, System creates a freeze sound. It freezes the timer and updates the screen. Then, System creates a coin in the screen and gives it a random value. It starts the coin timer and after a while it checks if the timer ends or not. Player clicks on the coin and System creates a coin sound. System adds the value of the coin to the account of player and it removes the coin from the screen. Then, it returns to the beginning of the loop and starts the timer again.

**Notes:** There are not separate scenarios for bomb and back balls, since the only difference of them is their sounds and update of ball sequences and also the back ball does rewinding. Also in the scenarios, some activities must seem to happen at the same time such that preparing the ball of the shooter and updating the sequence. So, users must not be able to see the time intervals between them. (See Section 2.3.2)

* 1. **Use Case Models**

The use case diagram of the project is the following:



**Figure 2.6.1 Use Case Diagram**

* + 1. **Login**

**Use case name:** Login

**Participating actor:** Player

**Stakeholders and interests:**

* Player wants to login to the game.
* System makes him login to the game.

**Pre-conditions:**

* System keeps accounts of players.
* Player must be in the login screen.

**Post condition:** Player logins to the game.

**Entry condition: -**

**Exit conditions:**

* Player confirms his id and password.
* Player selects “Login as a Guest” option.
* Player selects “Create New Account” option.
* Player selects “Delete Account” option.

**Main flow of events:**

1. System displays the login screen by default.
2. Player enters his id and password to login.
3. System confirms player’s information and opens player’s account.

**Alternative Flows:**

1. System gives a warning if id or password is wrong.
2. If Player wants to create a new account:
   1. Player selects “Create New Account” option.
   2. System displays the “Create New Account” page.
3. If Player wants to login as a guest:
   1. Player selects “Login as a Guest” option.
   2. System opens the game without taking any record.
4. If Player wants to delete his account:
   1. Player selects “Delete Account” option.
   2. System opens “Delete Account” page.
      1. **LoginAsGuest**

**Use case name:** LoginAsGuest

**Participating actor:** Player

**Stakeholders and interests:**

* Player wants play the game without having an account.
* System makes Player login to the game without keeping any record.

**Pre-condition:** Player must be in the login screen.

**Post-condition:** System opens Level 1.

**Entry condition:** Player selects the “Login as a Guest” option from the login page.

**Exit conditions:**

* System opens Level 1.
* Player selects “Back to Login Page” to return to login page.

**Main flow of events:**

1. Player presses the “Login as a Guest” button in the Main Menu.
2. System opens Level 1.

**Alternative flows:**

1. If player wants to return to login page:
   1. Player selects the “Back to Login Page” option.
   2. System displays the login page.
      1. **CreateNewAccount**

**Use case name:** CreateNewAccount

**Participating actor:** Player

**Pre-condition:** Player must be in the login screen.

**Post-condition:** System creates a new account.

**Entry condition:** Player selects “Create New Account” option from Login page.

**Exit conditions:**

* System creates a new account.
* Player selects “Back to Login Page” option to return to login page.

**Main flow of events:**

1. System displays “Create New Account” page.
2. Player enters his information.
3. System creates the account.

**Alternative flows:**

1. System gives warning if player gives inadequate information.
2. If player wants to go back to login page:
   1. Player selects “Back to Login Page” option.
   2. System opens the login page.
      1. **DeleteAccount**

**Use case name:** DeleteAccount

**Participating actor:** Player

**Stakeholders and interests:**

* Player wants to delete his account from the system.
* System deletes Player’s account.

**Pre-conditions:**

* Player must have an account.
* Player must be in the login screen.

**Post-condition:** System deletes the account.

**Entry condition:** Player chooses “Delete Account” option in the Login screen.

**Exit conditions:**

* System creates the account.
* Player chooses “Return to Login Page” option.

**Main flow of events:**

1. Player presses “Delete Account” button in the login screen.
2. System opens the delete account screen.
3. Player enters his id and password and confirms them to delete his account.
4. System deletes Player’s account.

**Alternative flows:**

1. System gives a warning message if the id or password is not true.
2. If player wants to go back to login page:
3. Player selects “Back to Login Page” option.
4. System opens the login page.
   * 1. **StartGame**

**Use case name:** StartGame

**Participating actor:** Player

**Stakeholders and interests:**

* Player wants to start the game.
* System opens a particular level which Player chooses to play.

**Pre-condition:** Player must be in Main Menu.

**Post-condition:** System opens a particular level which player has chosen.

**Entry condition:** Player selects the “New Game” option from Main Menu.

**Exit conditions:**

* Player selects a level.
* Player returns to Main Menu.

**Main flow of events:**

1. System opens a menu from which Player can select a level.
2. Player selects a level.
3. System opens the level.

**Alternative flow:**

1. Player selects the option of returning to Main Menu.
2. System opens the Main Menu.
   * 1. **PlayGame**

**Use case name:** PlayGame

**Participating actor:** Player

**Pre-condition:** Player must be in a level.

**Post-condition:** Player wins or loses the level.

**Entry condition:** System opens a level.

**Exit conditions:**

* Player selects “Return to Main Menu” option.
* Player selects “Return to Login Page” option.

**Main flow of events:**

1. Player starts from the particular level.
2. Player plays the level until he manages to destroy the sequence.
3. System opens the new level.
4. Player starts the next level.

*Player repeats first 4 steps until he finishes the game or he loses all of his lives.*

1. System takes a record of Player’s score.
2. System returns to Main Menu.

**Alternative flows:**

2A. Player tries to destroy the ball sequence.

2A.1. Player rotates the shooter to adjust its direction.

2A.2. Player either changes the type of the ball of shooter firstly or directly throws it.

2A.3. The ball either destroys a part of the sequence or appends to it.

2A.3.1. If it is a bomb ball, it destroys more balls from the sequence.

2A.3.2. If it is a back ball, it rewinds the sequence for a while.

2A.3.3. If it is freeze ball, it freezes the sequence and timer for a while.

*Player repeats the previous steps until he destroys all of the sequence.*

2A.4. Player completes the level or loses it.

2A.4.1. If Player completes a group of 3 levels, System gives him a checkpoint.

2A.4.2 If Player loses all of his lives, System makes his progress return to previous checkpoint.

2B. Player collects coins during the game.

2B.1. A coin appears in the screen with a random value through 1 to 100.

2B.2. Player clicks on the coin.

2B.3. System removes the coin from the screen and adds it to Player’s account.

*Previous steps are repeated at random times in the game.*

* If Player selects “View Help” option, System displays a help frame.
* If Player selects “Pause” option, System stops the game and displays a pause menu.
  + 1. **ViewHelp**

**Use case name:** ViewHelp

**Participating actor:** Player

**Stakeholders and interests:**

* Player wants help about playing the game.
* System displays the help manual in a frame.

**Pre-condition:** Player must be in a level.

**Post-condition:** System displays help manual for Player.

**Entry condition:** Player selects “View Help” option during the level.

**Exit condition:** Player closes the help frame.

**Main flow of events:**

1. System displays the help frame.
2. Player closes the frame after reading the instructions.
   * 1. **Pause**

**Use case name:** Pause

**Participating actor:** Player

**Stakeholders and interests:**

* Player wants to stop the game at any given time while the game is continuing.
* System stops the game until Player wants to continue the game.

**Pre-condition:** Player must be in a level.

**Post-condition:** System continues the game.

**Entry condition:** Player selects “Pause” option during a level.

**Exit conditions:**

* Player selects “Continue” option to return to the level.
* Player selects “Return to Main Menu” option.

**Main flow of events:**

1. System stops the level.
2. After a while, Player chooses to continue to the level.
3. System opens the level again.

**Alternative flows:**

1. Player chooses to return to Main Menu.
   1. System exits level and opens the Main Menu.
2. Player chooses to toggle the sound of the game.
   1. System toggles the sound.
      1. **ViewHighScores**

**Use case name:** ViewHighScores

**Participating actor:** Player

**Stakeholders and interests:**

* Player wants to view the high scores and also see his rank in the game.
* System displays the high score table and Player’s rank.

**Pre-condition:**

* System must keep the high scores in a table and also the rank of Player.
* Player must be in Main Menu.

**Post-condition:** System displays the high score table and Player’s rank.

**Entry condition:** Player chooses “View High Scores” option.

**Exit condition:** Player returns to the Main Menu.

**Main flow of events:**

1. Player presses “View High Scores” option.
2. System displays the score table.
3. Player returns to Main Menu.
   * 1. **PurchaseItem**

**Use case name:** PurchaseItem

**Participating actor:** Player

**Stakeholders and interests:**

* Player wants to purchase an item from the game store.
* System responds to Player’s request in the store.

**Pre-condition:** Player must be in Main Menu.

**Post-condition:** System updates Player’s account by adding the new items.

**Entry condition:** Player chooses “Store” option from Main Menu.

**Exit condition:** Player returns to Main Menu.

**Main flow of events:**

1. Player presses to “Store” button in Main Menu.
2. System displays the game store.
3. Player clicks on an item which he wants to purchase.
4. System checks Player’s coin amount.
5. If there are enough coins, System gives the item to Player and takes coin from him.

*Steps 3-5 are repeated until Player wants to exit the store.*

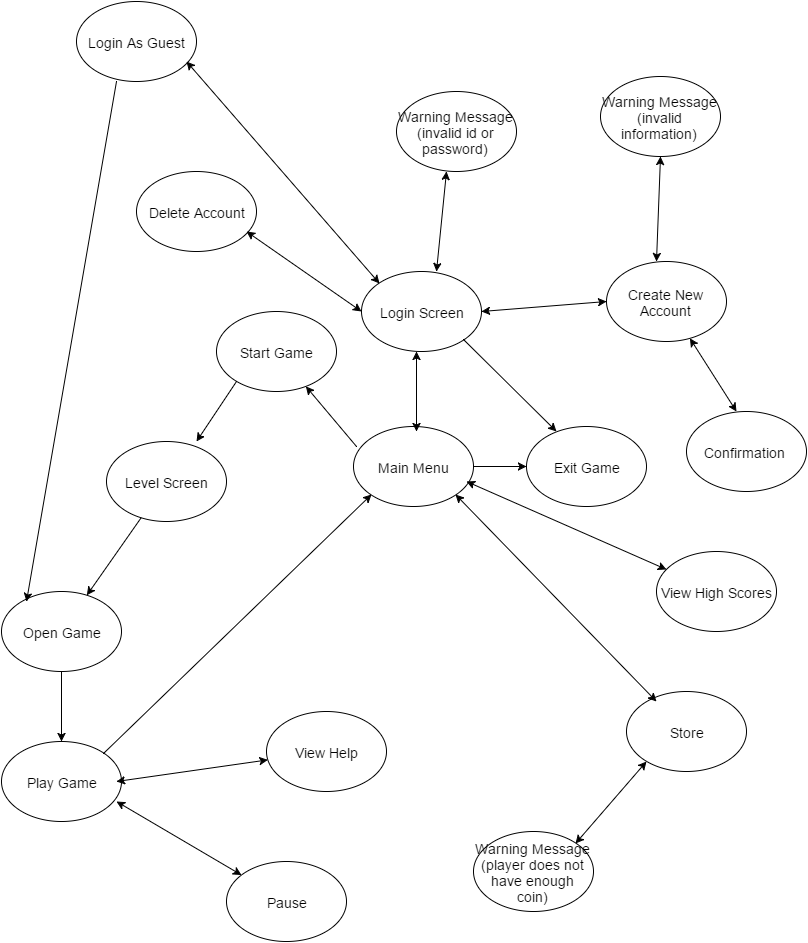
1. System returns to Main Menu.

**Alternative flows:**

1. If there are not enough coins, System gives a warning message.

Use cases in Main Menu and login screen have also an exit condition of quit game, but it is not included in the textual descriptions for clarity.

* 1. **User Interface**
     1. **Navigational Path**



**Figure 2.7.1 Navigational Path**

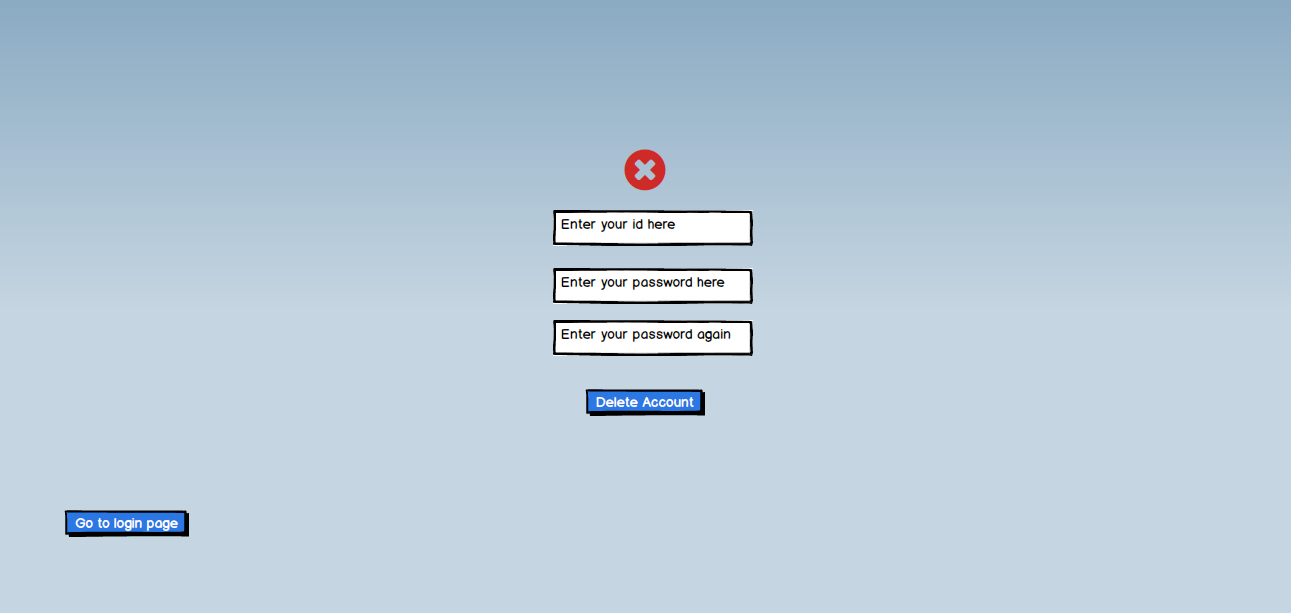
* + 1. **Screen Mock-ups**
       1. **Login**

When the program begins to run, player sees the login screen. Login screen displays five options to player which are exit, delete account, create new account, login as guest and login. (Figure 2.7.2)

**Figure 2.7.2 Login Screen**

**Exit:** If Player selects exit button, the game window closes and Player exits from the application.

**Delete Account:** If Player selects to delete his account, he will be asked to enter his id and password to delete the existing account. (Figure 2.7.3) After confirming his account, the account will be deleted by clicking on the “Delete Account” button. Player can return to login screen by clicking “Go to Login Page” button.



**Figure 2.7.3 Delete Account Screen**

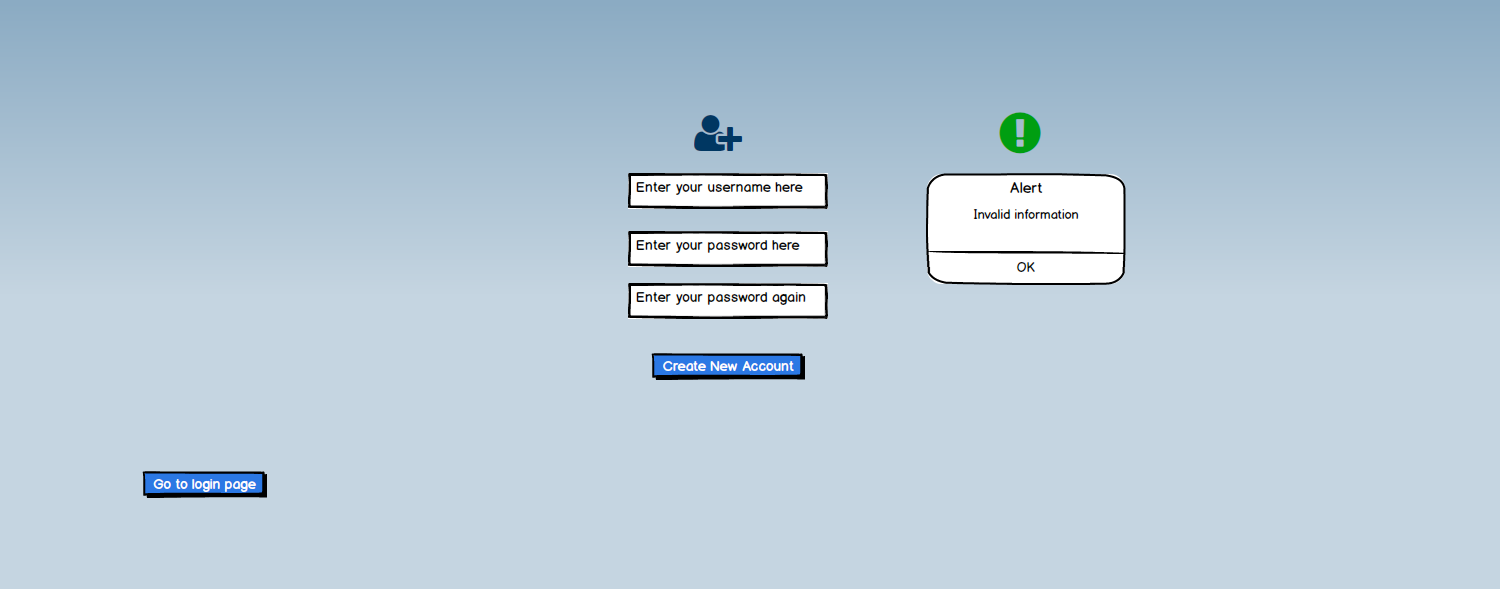
**Create New Account:** If Player selects to create new account, he will be asked to enter his username, which will be used in the game, and password to create a new account. (Figure 2.7.4)

If Player does not fill those areas which are required to create a new account, program displays a warning message on the screen. Also same warning message appears on the screen if he does not type his password correctly for the second time. (Figure 2.7.5)

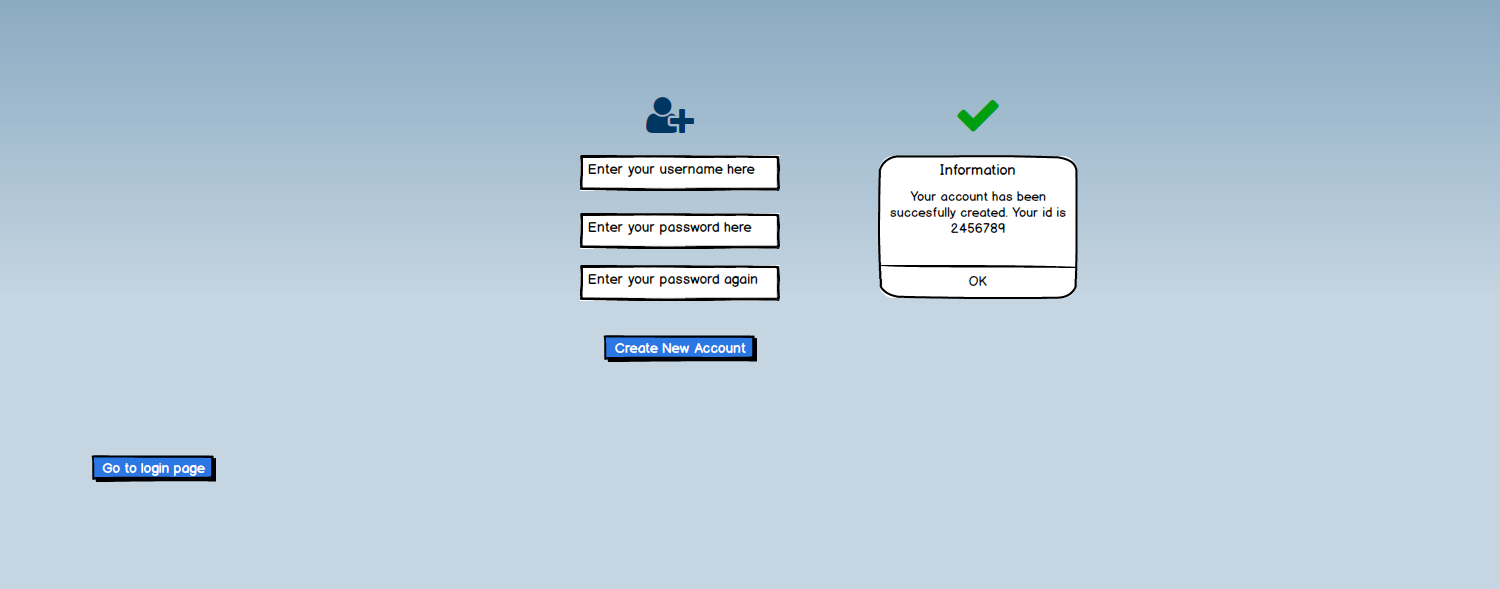
If Player fills the required areas successfully and clicks on “Create New Account” button, a confirmation message appears on the screen. (Figure 2.7.6) Player can go to login screen by clicking on “Go to Login Page” button.



**Figure 2.7.4 Create Account Screen**



**Figure 2.7.5 Warning for Create Account**



**Figure 2.7.6 Confirmation Message for Create Account**

**Login as Guest:** If Player selects login as guest, program opens the first level of the game without holding any record for the player.

**Login:** If Player has an account, he can login by entering his id and password on the login screen. If Player types his id or password incorrectly or leaves one of these areas empty, program displays a warning message on the screen. (Figure 2.7.7)



**Figure 2.7.7 Warning Message for Login**

* + - 1. **Main Menu**

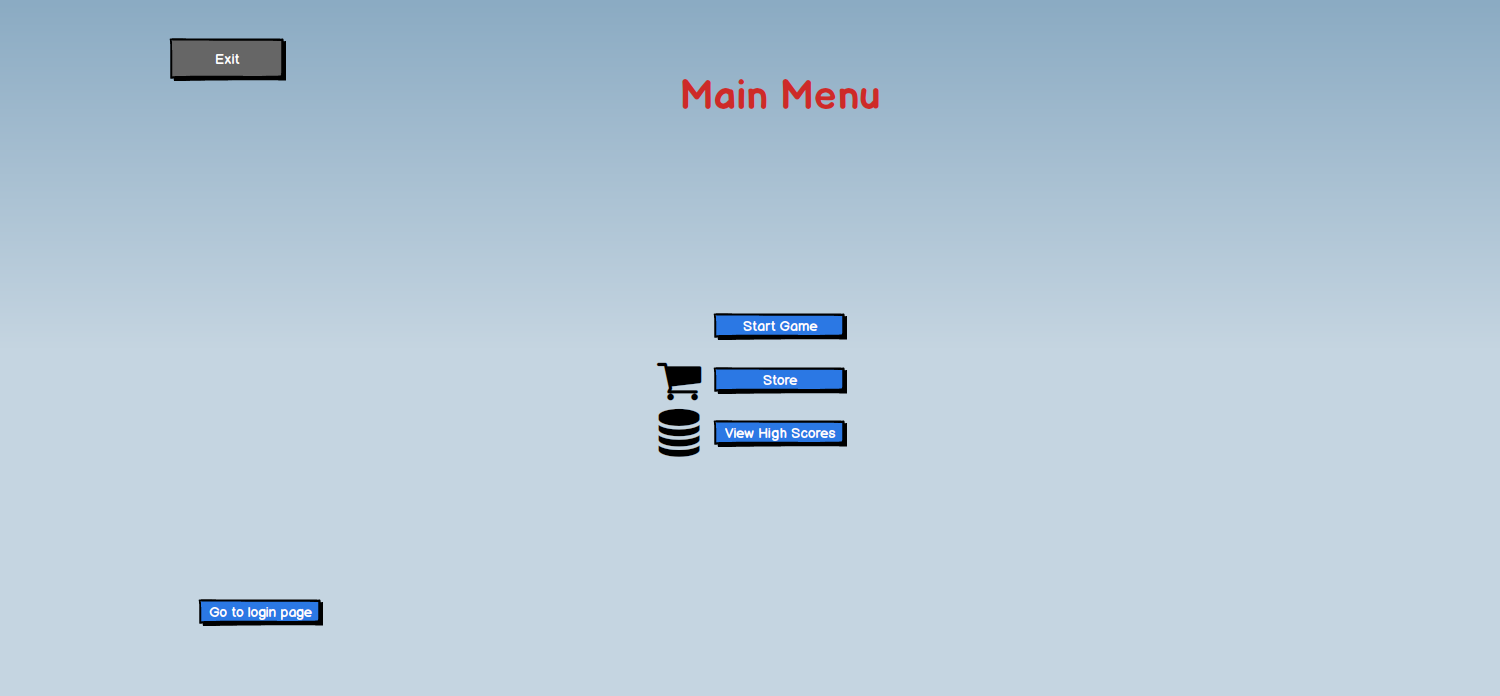
After logging in with an account, Player sees the Main Menu screen which contains five options. (Figure 2.7.8)

**Exit: “**Exit” button stops the application and closes the game window.

**Go to Login Page: “**Go to Login Page” button makes Player go to login screen in order to log in with a different account.

**View High Scores:** Shows the best five scores in the game and Player’s rank. Player can return to Main Menu by clicking on “Go to Main Menu” button. (Figure 2.7.9)

**Store:** Store contains three special balls, namely bomb ball, freeze ball and back ball, and life to purchase. Player’s coin is displayed at the top of the screen. If player has enough money, the items can be bought by clicking on the “Purchase” button. (Figure 2.7.10)



**Figure 2.7.8 Main Menu**

****

**Figure 2.7.9 High Score Screen**



**Figure 2.7.10 Store**

If Player’s coin is not enough to purchase the item, a warning message appears on the screen. (Figure 2.7.11) Player can go to Main Menu by clicking on “Go to Main Menu” button.

 **Figure 2.7.11 Warning Message for Insufficient Coin**

**Level Menu:** If Player selects the start game option in Main Menu, Level Menu is shown on the screen. The buttons with orange color show the completed levels by the player so far and the buttons with gray color show the levels which have not been completed yet. Player can select only the completed levels to play. (Figure 2.7.12) Player can go to Main Menu by clicking on “Go to Main Menu” button.



**Figure 2.7.12 Level Menu**

**Level Screen:** Figure 2.7.13 is a screenshot of the Zuma game which also represents our level screen with slight modifications. When Player selects the level from the Level Menu, level screen appears. Player’s coin amount is shown at the top left corner of the screen. Timer and pause buttons are shown at the top right corner of the screen. (Figure 2.7.13) There will also be a shortcut for the pause option as in all games.

**Pause Menu:** If Player clicks on the pause button, game stops and Pause Menu appears on the screen. (Figure 2.7.14) Pause Menu has three options: Player can return to Main Menu by clicking on “Go to Main Menu” button or login page by clicking on “Go to Login page” button. Sound can be turned on and off by the switch.

**View Help:** Player can obtain information about the game by clicking on the “Help” button in the place of “Zuma” text in the figure at the top of the screen. If Player clicks on it, help frame appears on the screen. It contains a brief description about how to play the game. (2.7.15)



**Figure 2.7.13 Level Screen**



**Figure 2.7.14 Pause Menu**



**Figure 2.7.15 View Help**

* + - 1. **Special Balls**

Special balls in the game are listed below with their images in Figure 2.7.16.



Bomb Ball

Backward Ball

Freeze Ball

**Figure 2.7.16 Special Balls**

1. **Analysis**
   1. **Object Model**
      1. **Domain Lexicon**

There are some terms about the application domain which may be ambiguous for the readers. Their definitions are the below. Some of them were taken from the Oxford dictionary.[[1]](#footnote-1)

* **Level:** Steps of the game which players have to complete individually.
* **Live:** A chance to lose a level.
* **Store:** A place in the game which players can purchase special balls and lives.
* **Coin:** It is for players to purchase special balls from the store.
* **Maze:** A complex network of path.
* **Checkpoint:** A point in the progress where players come back when they lose their lives.
* **Shortcut:** Alternative route which is shorter than other.
* **Rewinding:** Being wound back to the beginning.
  + 1. **Class Diagram**

Class diagram of the project is in the next page.

* + - 1. **Entity Classes**

Entity classes represent the models in the game which are controlled by other classes. Their purpose is to provide abstractions for basic elements of the game.

There is an abstract class called ScreenElement which keeps the screen coordinates and draw operation which are the general requirements of all screen element classes. The classes which extend from it are the following: Ball, BallSequence, BallShooter, Maze, Hole, and Coin. Ball class represents a ball with a color and ball type. There are four ball types which are enumerated as the following: Plain, Bomb, Back and Freeze. Their specific properties are told in the Requirements Specification part. BallSequence class keeps a sequence of balls which moves throughout the maze and is destroyed by the players. It has a slide operation which is called in each step of the level and a rewind operation which is for back balls. BallShooter class represents the shooter which is controlled by the players to throw balls to the sequence. It also keeps a ball instance which is the ball to be thrown in the current step. It has functionalities for determining the ball type and set a random color for the ball. Maze class keeps the maze which covers the sequence. There is also a Hole class separate from the Maze for controlling if there is a ball in the hole or not in each step. Coin class represents the coins of the game which have different values and players should catch them for purchasing special balls from the store.

There is a Game class which holds the players and their information. The Player class is for keeping the identities of the players and their progress in the game.

**Figure 3.1.1 Class Diagram**

* + - 1. **Boundary Classes**

Boundary classes are for making the interactions between the players and the game by providing user interfaces.

There is a Menu class which is the generalization of the menus in the game. It is for adjusting common properties of menus such as their sizes, back buttons, listeners and so on. There are four menus in the game: MainMenu, StoreMenu, PauseMenu, and LevelMenu. MainMenu contains the options of starting the game, entering to the store, and quit game. StoreMenu contains the items which players can purchase. LevelMenu is for selecting which level player chooses for starting the game. PauseMenu appears during the level which users can exit or resume the level or toggle the sound of the game. Also from the menus, users are able to return back to MainMenu, previous screen and login screen; however those are not included for clarity.

There are some screens in the game which are represented by the following classes: LoginScreen, HighScoreScreen, LevelScreen, and HelpScreen. Players login to the game by LoginScreen. They can see high scores of the game and their own ranks among the scores from HighScoreScreen. For playing the game, they interact with LevelScreen. During the game, they get help about the level from HelpScreen.

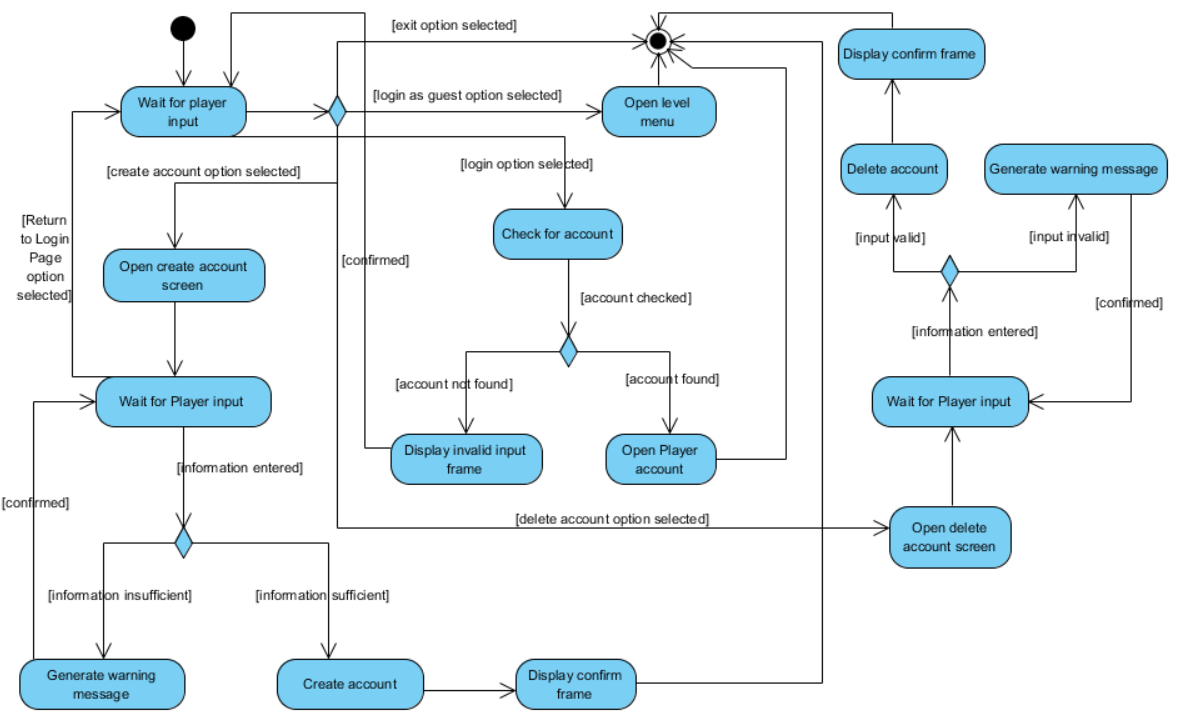
* + - 1. **Control classes**

The purpose of control classes is to control the operations in the game, such as updating a view or model.

GameManager class controls the transitions between screens. Also it handles the login requests comes from the login screen by interacting with the Game class which keeps the player accounts.

LevelManager class serves as the main control class while playing the game and it handles the requests of LevelScreen. It interacts with three other control classes which are SoundManager, TimerManager and CollisionManager. SoundManager handles the sound effects of the game which are created in collisions. TimerManager deals with the timer of the level which flows during the level and determines the score of the player. The class also handles the time of appearance of a coin in the screen by keeping a timer for the coin. On the other hand, it has an operation for freezing the time for the freeze balls. CollisionManager determines if there is a hit or not, and it updates the ball sequence accordingly.

* 1. **Dynamic Models**
     1. **Activity Diagrams**
        1. **Login**

****Activity diagram for “Login” use case is the following:

**Figure 3.2.1 – Activity diagram for “Login”**

There are four cases:

1. If Player chooses to login directly:

System takes his id and password as inputs, checks for an account with the given id and password; then if they are valid, it opens the account. If they are invalid, it displays a warning frame and waits for an input again.

1. If Player chooses to login as a guest:

System opens the level menu directly.

1. If Player chooses to create a new account:

System takes his username and password as inputs. If Player’s input is enough, System creates an account and displays a confirm frame and in that frame, it also displays Player’s id. If input is not enough, System generates a warning message and waits for an input again. While System is waiting for an input, Player also has an option to return to Login page.

1. If Player choose to delete his account:

System takes his id and password as inputs and checks for an account. If they are valid, it deletes the account and displays a confirm frame. If they are not valid, it displays a warning message and waits for an input again.

* + - 1. **Play Game**

****Activity diagram of “Play Game” use case is the following:

**Figure 3.2.2 – Activity diagram of “Play Game”**

At first, System opens a particular level which Player requests to play. Then, it initializes the level map (level screen). For triggering the level, it starts the timer and slides the ball sequence. Then, it rotates the shooter according to the player input. If Player desires to change the ball type, System changes the ball type. System throws the ball according to the input. After that, it checks if there it is a hit or miss.

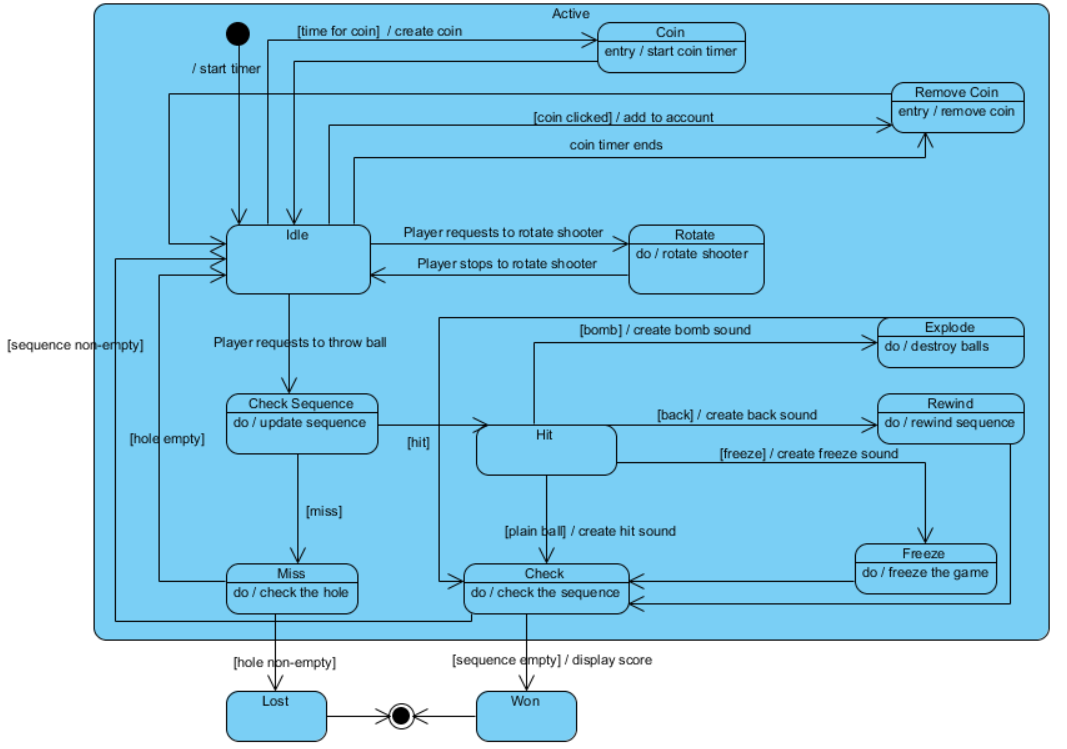
1. If it is a hit:

It concurrently creates a hit sound and updates the sequence according to the ball type. At first, it destroys the balls regularly. Then, if the ball is a back ball, it rewinds the sequence. If it is a bomb ball, it destroys more balls. If it is freeze ball, it freezes the sequence and timer. After that, it checks the sequence that if it is empty or not. If it is empty, that means the level is over, so it displays the score. If it is not empty, it loops the procedure again.

1. If it is a miss:

If the ball is a special ball, System makes the ball a plain ball. Then, after adding the ball to the sequence, it checks the hole that it contains any ball or not. If it contains any ball, that means Player loses. If it is empty, it loops the procedure again.

* + 1. **State Chart Diagram**

****

**Figure 3.2.3 State Chart Diagram of LevelManager class**

There are three main states: Active, Won, and Lost. System is in the Active state if the level continues. At first, it is in the Idle state. If Player requests to rotate the shooter, it goes to Rotate state and comes back when the process is done. If Player throws a ball, it goes to Check Sequence state and updates the sequence.

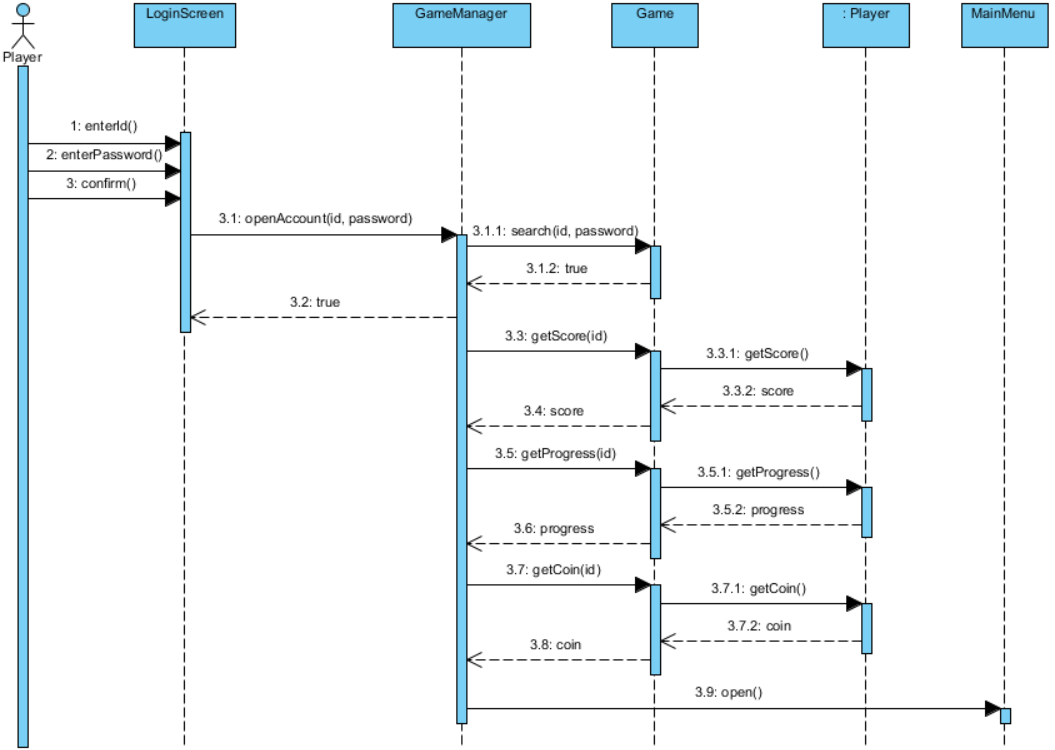
If there is a hit, it goes either directly to Check state for checking the sequence or firstly goes to a state belongs to one of the special balls. Also during the transition, it creates a hit sound depending on the type of ball. If the sequence is empty, it goes to Won state, if not it loops back.

If there is a miss, it checks the hole and if it contains a ball, it goes to Lost state; else, it loops back.

Also, from the Idle state, System goes to Coin state when there is time for creating a coin. As an entry condition, it starts the coin timer. From the Idle state, when Player clicks on the coin or coin timer ends, it goes RemoveCoin state and removes the coin from the screen. If there is a click; during the transition, it adds the coin to Player’s account.

* + 1. **Sequence Diagrams**
       1. **Login**

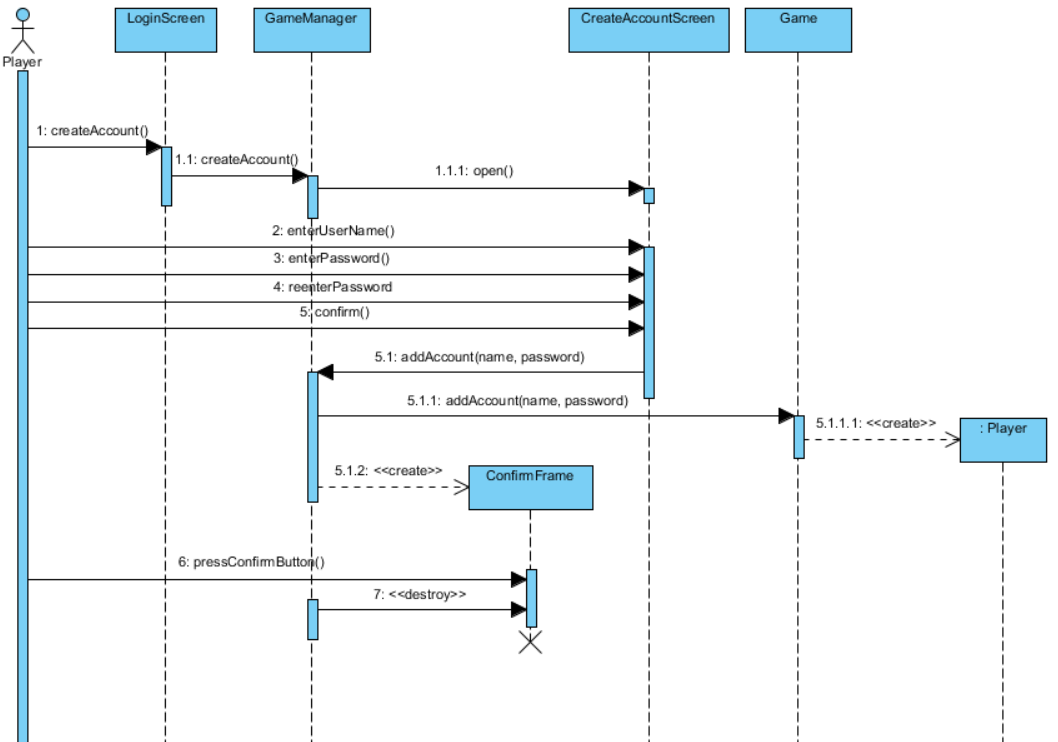
The following sequence diagram corresponds to the “Login” scenario in the Scenarios part of the report.

**Figure 3.2.4 Sequence Diagram for “Login” scenario**

In the scenario, LoginScreen class represents the boundary class which is responsible for user interaction. GameManager is the control class which is responsible from player accounts. It searches for the particular account using the Game class which is an entity class that holds the players. If Player exists, it gets the information about the player from the Game class and Game class takes from the Player class. At the end, GameManager class opens the Main Menu since it is responsible for transition between the pages in the game.

* + - 1. **Create Account**

The following sequence diagram corresponds to the “Create Account” scenario in the Scenarios part of the report.

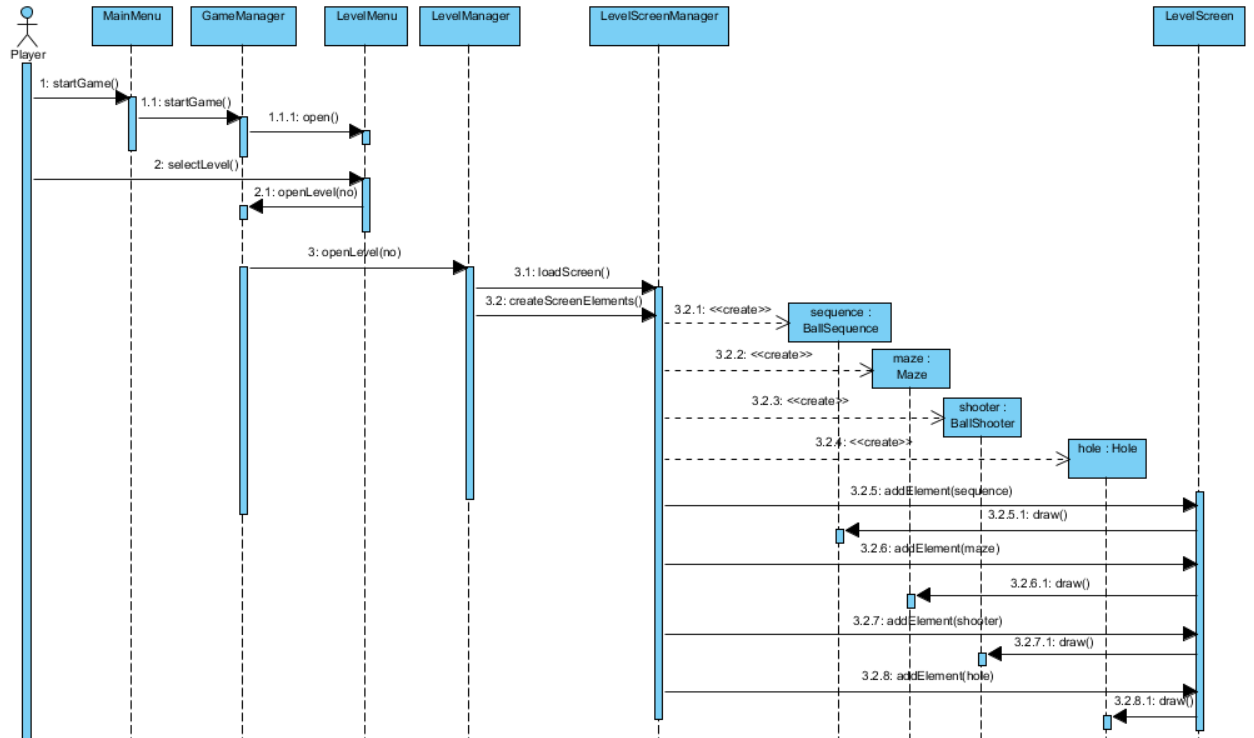


**Figure 3.2.5 Sequence Diagram for “Create Account” scenario**

Again, LoginScreen is the boundary class for user interaction. However, Player also interacts with CreateAccountScreen which is a boundary class too. As a control class, GameManager handles the transition between the Login screen and Create Account screen and adds the account to Game. Then, Game class creates a Player object for the new player. After that, GameManager creates a confirm frame which is destroyed after Player presses the confirm button.

* + - 1. **Start Game**

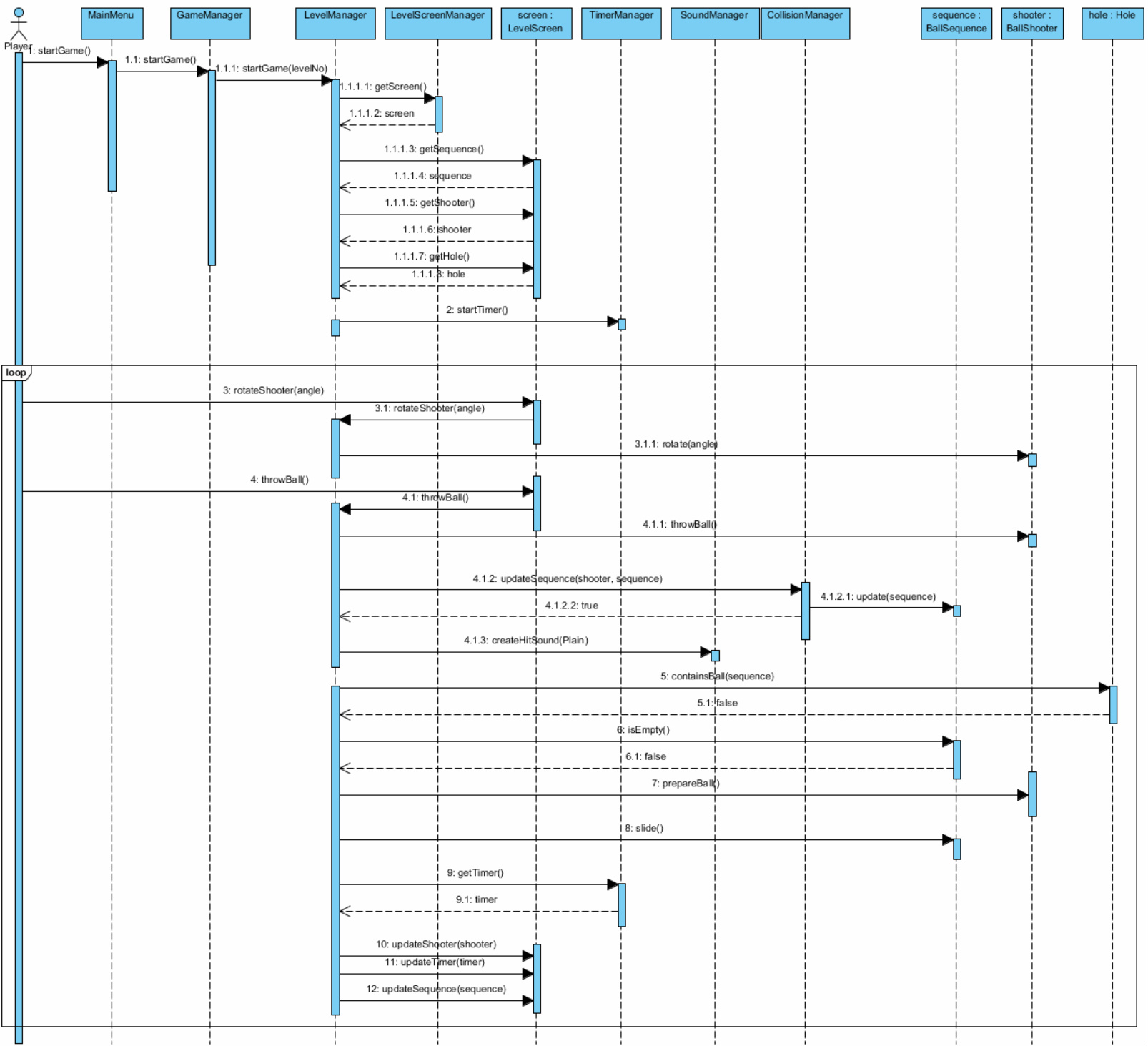
The following sequence diagram corresponds to the “Start Game” scenario in the Scenarios part of the report.

**Figure 3.2.6 Sequence Diagram for “Start Game” scenario**

In this scenario, MainMenu class serves as the boundary class from which Player chooses to start the game. From LevelMenu, Player selects a level to play and LevelManager opens the level by the control of GameManager. Then, LevelScreenManager initializes the LevelScreen by creating its BallSequence, Maze, and BallShooter. Then, LevelScreen class draws them to the screen. Here, LevelMenu and LevelScreen are boundary classes which Player can interact with. LevelManager and LevelScreenManager are control classes which controls the operations of boundary and entity classes. BallSequence, Maze and BallShooter are entity classes which are some basic elements of the game.

* + - 1. **Play Game #1**

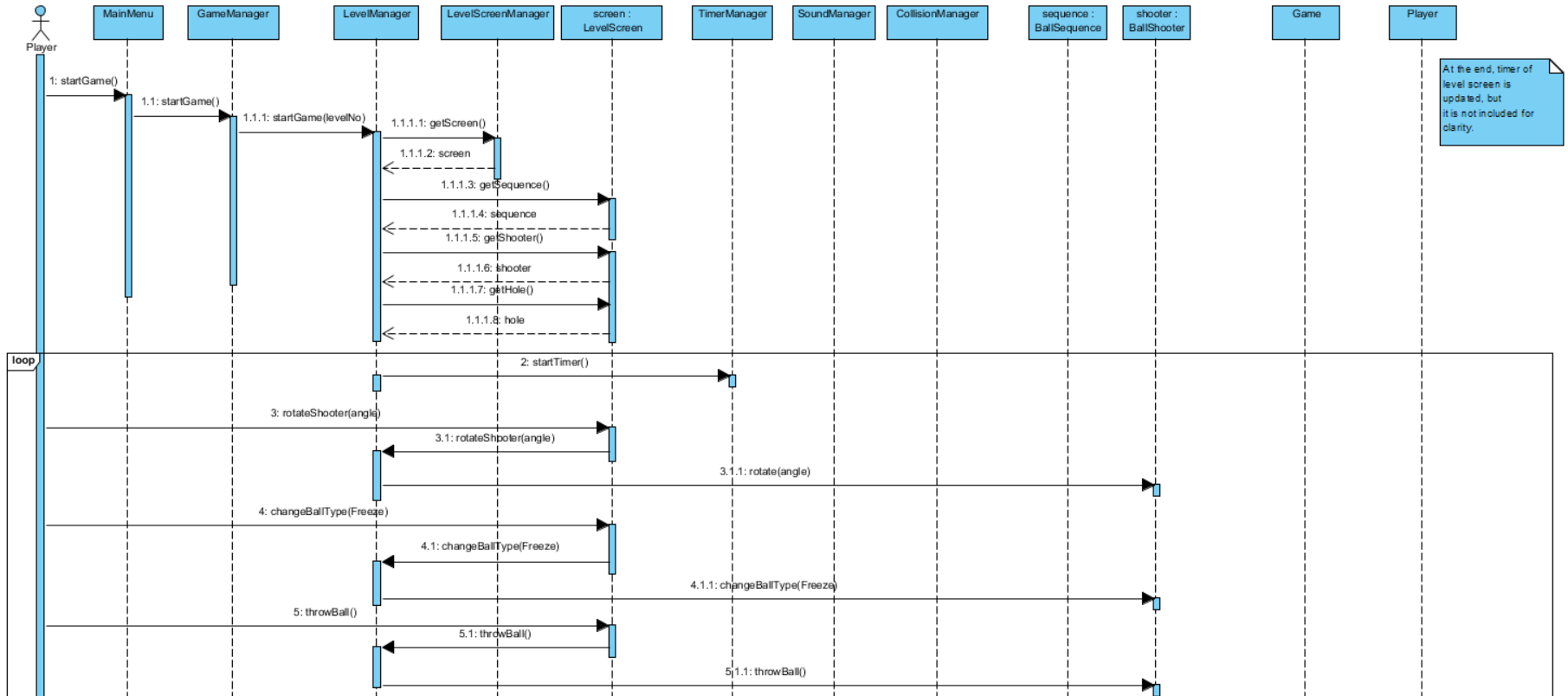
The following sequence diagram corresponds to the “Play Game #1” scenario in the Scenarios part of the report.

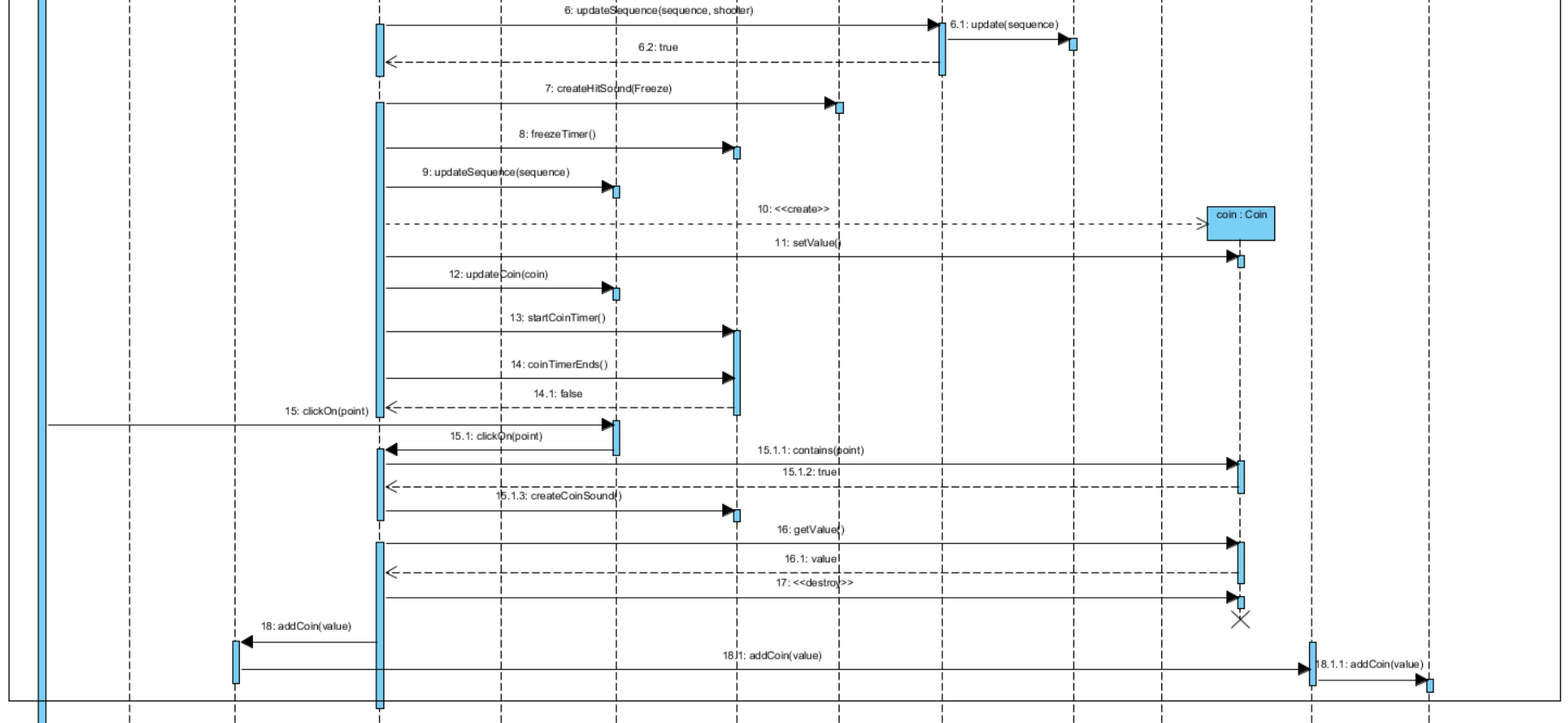
**Figure 3.2.7 Sequence Diagram of “Play Game #1” scenario**

After the previous “Start Game” scenario, LevelManager gets the screen elements from LevelScreen. LevelManager is the main control class for the level play. There are also three control classes associated with it: TimerManager, SoundManager, and CollisionManager. At first, TimerManager starts the timer of the level. Then, the game loop begins. Player requests to rotate shooter and throw ball by interacting with LevelScreen and LevelManager calls the methods of BallShooter for those. Then, CollisionManager updates the sequence accordingly, and if there is a hit, SoundManager creates a hit sound depending on the ball type. Then, LevelManager checks the Hole and BallSequence for determining if Player wins or loses. If the game continues, BallShooter prepares the new ball and LevelManager slides the sequence, and updates the screen elements accordingly.

* + - 1. **Play Game #2**

The following sequence diagram corresponds to the “Play Game #2” scenario in the Scenarios part of the report.



**Figure 3.2.8 Sequence Diagram for “Play Game #2” Scenario**

This scenario differentiates from the first one such that it is about bonus properties of the game. Since it will use freeze ball, it starts the timer inside the loop. Player requests the change the ball type from LoginScreen and LevelManager requests it from BallShooter. After updating the sequence, TimerManager freezes the time. Then, LevelManager creates a coin in LevelScreen and gives it a random value. Then, TimerManager starts the coin timer, and after a while LevelManager checks if it ends or not. Player clicks on a point, and LevelManager checks that if it belongs to the coin or not. Then, if it belongs to the coin, LevelManager gets the value of it and GameManager adds it to Player’s account by using Game. Also, LevelManager destroys the coin when since it is done with it after getting its value.

1. **Conclusion**

In conclusion, we organized our analysis report in order to show how a Desktop game which is called “Crazy Shooter” will be designed and implemented. The report has two main sections which are called as “Requirement Analysis” and “Analysis”. Requirement analysis part includes seven topics which are overview, functional requirements, nonfunctional requirements, constrains, scenarios, use case models and user interface. In this section, the necessary requirements for the game are considered and they are fulfilled. We tried to make the requirements verifiable such that there should not be an ambiguity about those for designing the system more efficiently. Moreover, the properties and rules of the game are defined in the overview part and the appearance and structure of the game are determined and supported by the screen mock-ups. Also, navigational path in the user interface section is for demonstrating the usability of the program. In addition to these, use case diagram depicts the functions of the game from the view of the external actors which are the players.

The analysis section includes two topics which are object model and dynamic model. In the object model section, the class diagram is used to describe the static structure of the system. On the other hand, dynamic model part consists of activity diagrams, state chart diagram and sequence diagrams which represent the dynamic structure of the game. There are two activity diagrams and they demonstrate the dynamic behavior of use cases called “Login” and “Play Game”. In addition to those, there is a state chart diagram which models the behavior of the control class for level play called LevelManager. Five sequence diagrams indicate the dynamic behavior of scenarios for creating account, login, and starting and playing the game.

To summarize, we organized an analysis report so that we can benefit from that while designing and implementing our project. Also by writing this report, we had a chance to shape the solution domain of the game. By using the contents of the report, we can prevent facing any problem and we can design and implement our game more efficiently.

**References**

* Bruegge, Bernd, and Dutoit, Allen H., *Object-Oriented Software Engineering,* Pearson Education Limited: 2014. Print.
* <http://www.popcap.com/games/zumas-revenge/online> (Development of idea)
* <http://www.oxforddictionaries.com/definition/> (Domain Lexicon)
* <http://dictionary.reference.com/browse/checkpoint> (Domain Lexicon)

1. <http://www.oxforddictionaries.com/definition/> [↑](#footnote-ref-1)