  
  
  
  
CS 319 - Object-Oriented Software Engineering  
Analysis Report  
  
  
Survivor  
Group 7  
Defne Demirtürk

Sevde Şahin

Nihat Atay

Yağızhan Güler

Table of Contents

[1. Introduction 1](#_Toc433820307)

[2. Requirement Analysis 2](#_Toc433820308)

[2.1 Overview 2](#_Toc433820309)

[2.1.1 List of Bonuses 3](#_Toc433820310)

[2.1.2 List of Penalties 3](#_Toc433820311)

[2.1.3 List of Baits 4](#_Toc433820312)

[2.1.4 List of Cursors 4](#_Toc433820313)

[2.1.5 List of Other Objects 5](#_Toc433820314)

[2.2 Functional Requirements 5](#_Toc433820315)

[2.3 Non-functional Requirements 5](#_Toc433820316)

[2.4 Constraints 6](#_Toc433820317)

[2.5 Scenarios 6](#_Toc433820318)

[2.5.1. View High Scores 6](#_Toc433820319)

[2.5.2. View Credits 7](#_Toc433820320)

[2.5.3. Play Game 8](#_Toc433820321)

[2.5.4. Change Settings 9](#_Toc433820322)

[2.5.5. View Help 10](#_Toc433820323)

[2.5.6 View Credits 11](#_Toc433820324)

[2.6 Use-Case Models 12](#_Toc433820325)

[2.7 User Interface 13](#_Toc433820326)

[2.7.1 Navigational Path 17](#_Toc433820327)

[2.7.2 User-Friendly Interface 17](#_Toc433820328)

[3. Analysis 18](#_Toc433820329)

[3.1 Object Model 18](#_Toc433820330)

[3.1.1 Domain Lexicon 18](#_Toc433820331)

[3.1.2 Class Diagrams 18](#_Toc433820332)

[3.2 Dynamic Models 20](#_Toc433820333)

[3.2.1 State Chart 21](#_Toc433820334)

[3.2.2 Sequence Diagrams 22](#_Toc433820335)

[4. Conclusion 27](#_Toc433820336)

**Table of Figures**

Figure 1: Accelerate Cursor

Figure 2: Extra Life Span

Figure 3: Expand Cursor

Figure 4: Rock

Figure 5: Bomb

Figure 6: Yellow Banana

Figure 7: Green Banana

Figure 8: Blue Banana

Figure 9: Red Banana

Figure 10: Regular Cursor

Figure 11: Red Cursor

Figure 12: Blue Cursor

Figure 13: Life Span

Figure 14: Use Case Diagram

Figure 15: Main menu window

Figure 16: In game window

Figure 17: Settings window

Figure 18: Help window

Figure 19: Credits window

Figure 20: Navigational Path

Figure 21: Class Diagram

Figure 22: Activity Diagram

Figure 23: Sequence Diagram (Start Game)

Figure 24: Sequence Diagram (Play Game)

Figure 25: Sequence Diagram (Change Settings)

# 1. Introduction

Survivor is an action game basically evolved from DXBall that we decided to develop. We plan to develop our game with different features and images that we have seen from DXBall.

In Survivor, we are going to use new features as well. In the game, the aim of the user is to catch baits fall from top. By this, the user will be able to increase points and so level up. However, the baits that fall out will include rocks, which will decrease the players points, and bombs which will cause the player to loose 1 life span. Also the baits fall out will include extra life span, which will give extra playing chance to user, and extra speed, which will give extra speed to cursor for a few seconds.

Apart from these, there will be enemy against user. This immobile machine will be sending ray to curser and user will try to escape. If this ray hits curser, user will lose 1 life span to play game. Furthermore, the game will be a desktop application and will be controlled by a mouse or the keyboard.

# 2. Requirement Analysis

## 2.1 Overview

## 

The game consists of five levels where the colors of bananas are changed. For each level there will be 10000 points limit. Purpose of this game is completing all the levels without losing all lives. Basically, the game is about collecting falling baits by the cursor to achieve goal points. While the player playing the game there can be various bonuses as well as the rocks and bombs. The bonuses provide advantages to player while rocks make the player lose points. Moreover, collected bombs cause to lose life span.

During the game, player can pause and carry on game at any time.

There will be high score system which will keep the highest 10 scores with players’ name. Player gains according to the type of bananas. Different types of bananas give different amount of points.

Player will be able to control cursor using keyboard.

Player is allowed to change game background and cursor using the setting menu. There will be three different background and cursor options. Also user can change game sound as on and off.

### 2.1.1 List of Bonuses

 Figure 1: Accelerate cursor

It accelerates the cursor.

 Figure 2: Extra life span

It gives extra life span to user.

 Figure 3: Expand cursor

It expands the cursor.

### 2.1.2 List of Penalties

 Figure 4: Rock

It causes 5000 point loss.

 Figure 5: Bomb

It causes 1 life span.

### 2.1.3 List of Baits

C:\Users\Nihat\Desktop\mockups\banana 1.pngFigure 6: Yellow Banana

It gives 250 points to player.

C:\Users\Nihat\Desktop\mockups\banana 2.png Figure 7: Green Banana

It gives 500 points to player.

C:\Users\Nihat\Desktop\mockups\banana 3.png Figure 8: Blue Banana

It gives 1000 points to player.

C:\Users\Nihat\Desktop\mockups\banana 4.png Figure 9: Red Banana

It gives 2500 points to player.

### 2.1.4 List of Cursors

C:\Users\Nihat\Desktop\mockups\Cursor.pngFigure 10: Regular Cursor

This is the default cursor in game.

C:\Users\Nihat\Desktop\mockups\cursor 2.pngFigure 11: Red Cursor

Player can choose this cursor in settings menu.

C:\Users\Nihat\Desktop\mockups\cursor 3.png Figure 12: Blue Cursor

Player can choose this cursor in settings menu.

### 2.1.5 List of Other Objects

C:\Users\Nihat\Desktop\mockups\Life Span.pngFigure13: Life Span

Next to this object, user can see how many life span s/he has.

## 2.2 Functional Requirements

* User will be able to control the cursor with mouse.
* User can change the settings of the game. Such as:

Change the background

Change the sounds

* User will be able to access the help menu that consists of information about how to play the game and items.
* User can see the high scores of top 10 players.
* The game shall offer numerous bonuses and penalties.
* The user shall be allowed to pause the game and then resume.

## 2.3 Non-functional Requirements

* Control mechanism of the game shall have short response time that allows the player to play with minimal delay.
* Concepts of the levels in the game are easy to understand and react to.

## 2.4 Constraints

* The game will be implemented in Java.

## 2.5 Scenarios

### **2.5.1. View High Scores**

**Use Case Name**: View High Scores

**Primary Actor**: Player

**Event Flow**:

-Player wants to see top ten scores with player names.

-System shows the list containing top ten scores with player names.

**Pre-conditions**: System keeps records of top ten scores.

Post-condition: -

**Entry Condition**: Player selects “View High Scores” option from Main Menu.

**Exit Condition**: Player selects “Back” option to return to Main Menu.

**Success Scenario Event Flow**:

1. System displays top ten scores with player names.

Alternative Flows:

-If player desires to return to Main Menu at any time:

1) Player selects “Return to Main Menu” button to return to Main Menu.

2) System displays Main Menu.

### **2.5.2. View Credits**

**Use Case Name**: View Credits

**Primary Actor**: Player

**Event Flow**:

- Player wants to learn names of people and their contact information created Crazy Ball.

- System displays contact information of software developers with their names.

**Pre-conditions**: Player should be in the Main Menu window.

Post-condition: -

**Entry Condition**: Player selects “View Credits” from Main Menu.

**Exit Condition**: Player selects “Back” to return to Main Menu.

**Success Scenario Event Flow**:

1. System displays contact information and names of people who developed Crazy Ball.

Alternative Flows:

- If player desires to return main menu at any time:

1) Player selects “Return to Main Menu” button to return main menu.

2) System displays Main Menu.

### **2.5.3. Play Game**

**Use Case Name**: Play Game

**Primary Actor**: Player

**Event Flow**:

-Player aims to complete all levels and make highest score.

-System keeps the score of the Player.

**Pre-condition**: For first running, game settings are set as default. If Player changes game settings, adjusted settings will be saved and used by System.

**Post-condition**: If Player gets a score which is high enough to be in high score list, high score list will be updated by System.

**Entry Condition**: Player selects “Play Game” button from Main Menu.

**Exit Condition**: Player selects “Return to Main Menu” from the game screen.

**Success Scenario Event Flow**:

1. Game is started by the System.

2. Player starts playing from first level.

3. Player plays the level until the time limit is reached.

4. System takes the player to the next level.

5. Player starts playing next level.

Player repeats the steps 3 – 5 until all levels are completed or Player loses all lives.

6. System record Player’s score to High Score List, if Player score is higher than the lowest score in high score list and displays the High Score List.

7. System returns to Main Menu.

**Alternative Flows**:

A) Player collects the bonuses during game:

1. Cursor collides with bonuses.

2. Player tries to collect bonuses by using his cursor.

3. System removes the bonus, whether user collects it or bonus falls from screen.

4. If user collects the bonus System makes necessary changes to apply the feature of bonus and system updates score of the Player according to bonus.

### **2.5.4. Change Settings**

**Use Case Name**: Change Settings

**Primary Actor**: Player

**Event Flow**:

-Player desires to change game settings: disabling game sounds or enabling game sound, changing the background, changing the cursor type.

-System updates the settings which are changed by player.

**Pre-condition**: For first running, game settings will be set as default. If Player changes game settings, adjusted settings will be saved and used by system.

**Post-condition**: Game settings are updated.

**Entry Condition**: Player selects “Change Settings” button from Main Menu.

**Exit Condition**: Player selects “Back” to return to Main Menu.

**Success Scenario Event Flow**:

1. Player presses “Change Settings” button to make changes about game settings.

2. Game settings are displayed to Player in “Change Settings” screen by System.

3. Player adjusts settings according to his desire.

4. System updates game settings successfully.

**Alternative Flows**:

A. If Player desires to use default settings at any time:

1. Player selects “Default Settings” button from “Change Settings” screen.

2. Default game settings are updated by System.

B. If Player requests to return previous menu at any time:

1. Player selects “Back” button from “Change Settings” screen.

2. Game settings are updated by System.

3. Player returns to Main Menu.

### **2.5.5. View Help**

**Use Case Name**: View Help

**Primary Actor**: Player

**Event Flow**:

- Player wants to learn purpose of game and how to play it.

- System shows a text document explaining purpose of game, player controls, bonuses and penalty object types.

**Pre-conditions**: Player should be in Main Menu.

**Post-condition**: -

**Entry Condition**: Player selects “View Help” from Main Menu.

**Exit Condition**: Player selects “Back” to return to Main Menu.

**Success Scenario**:

1. Player selects “View Help” from Main Menu.

2. System displays help document giving instructions about main purpose of game, game controls and bonuses, baits, bombs and rocks.

**Alternative Flows**:

A. If Player requests to return previous menu at any time:

1. Player selects “Back” button from “View Help” screen.

2. Player returns to Main Menu.

### 2.5.6 View Credits

**Use Case Name**: View Credits

**Primary Actor**: Player

**Event Flow**:

- Player wants to learn names of people and their contact information created Survivor.

- System displays contact information of software developers with their names.

**Pre-conditions**: Player should be in the Main Menu window.

**Post-condition**: -

**Entry Condition**: Player selects “View Credits” from Main Menu.

**Exit Condition**: Player selects “Back” to return to Main Menu.

**Success Scenario Event Flow**:

1. System displays contact information and names of people who developed Survivor.

**Alternative Flows**:

- If player desires to return to Main Menu at any time:

1) Player selects “Return to Main Menu” button to return to Main Menu.

2) System displays Main Menu.

## 2.6 Use-Case Models

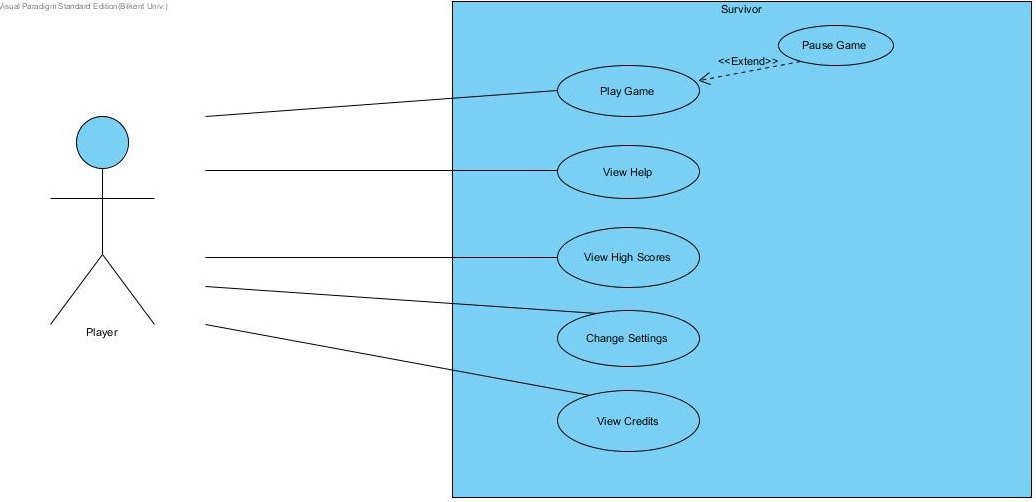
This section provides information about the main use case model of Survivor game, detailed use case explanations are included below.

Figure 14: Illustrates the use case model of Crazy Ball

## 2.7 User Interface

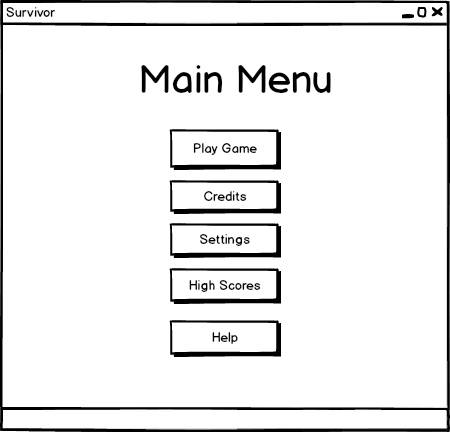


Figure 15: Main Menu Window



Figure 16: In game window

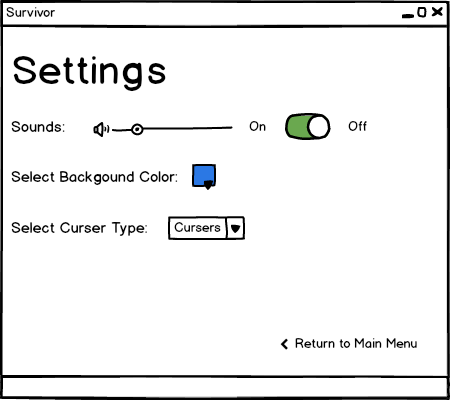


Figure 17: Settings window

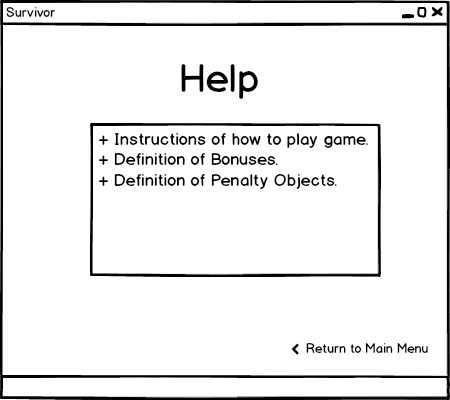


Figure 18: Help Window

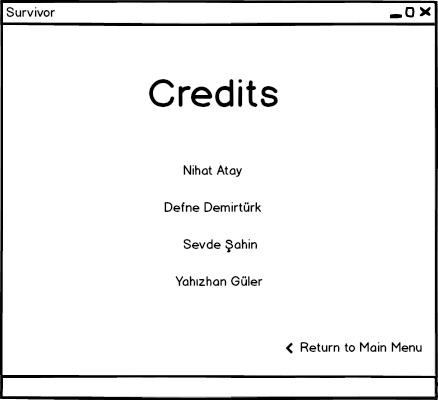
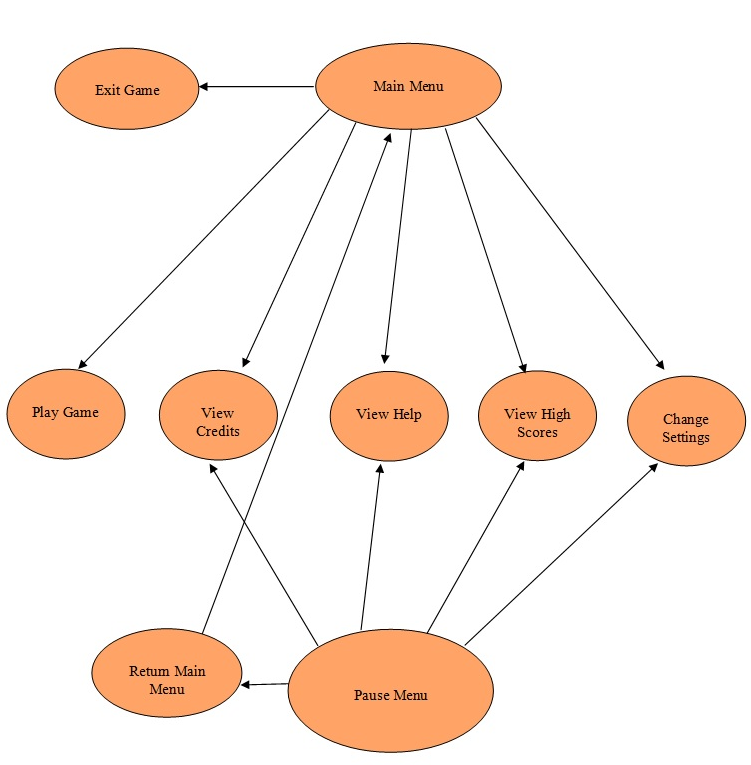


Figure 19: Credits Window

### 2.7.1 Navigational Path



### 

Figure 20: Navigational Path

### 2.7.2 User-Friendly Interface

The game will consist of a user-friendly interface through which the user will be able to play the game easily. This feature of the game is important to make the user feel comfortable while playing the game.

# 3. Analysis

## 3.1 Object Model

### 3.1.1 Domain Lexicon

### 3.1.2 Class Diagrams

The game Survivor consists of 16 classes.

**GameEngine** class is the class where the main management of the game is handled. It manages the lifecycle of the game by cooperating with other manager classes. It stores the score, high score and time features of the game.

**CollisionManager** class reports the collision type that occurs in the current state of the game and also it updates game objects to the GameEngine when needed.

**InputManager** class carries the management of inputs which are basically the interruptions that are caused by buttons.

**MainMenu** class handles all options of the game by contacting with the related classes. It also represents the features of our user interface.

**FileManager** class handles the connection between the files associated with the game. It handles mostly recording the scores of players.

Furthermore, GameObject class consists of 5 different types of objects. Namely, Bait, Cursor, Ray, Bonus and Penalty classes. We have Penalty class and it has 2 child classes, named Rock and Bomb. These are the objects which have negative effect on the gameplay. Moreover, we have 3 different types of bonuses named, ExtraSpeed, ExtraPoint and LifeSpan. These are the objects which have positive effect on the gameplay.

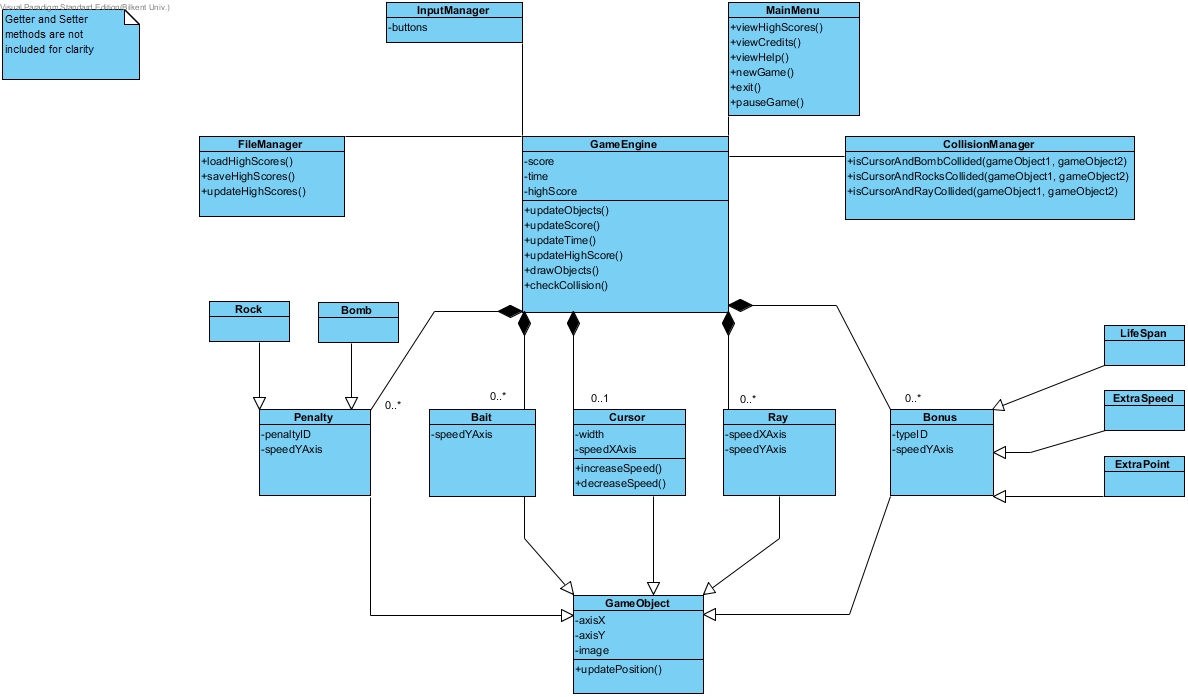


Figure 21: Class Diagram of Survivor Game

## 3.2 Dynamic Models

This part provides detailed information about Survivor game by showing some important scenarios of the game in sequence diagrams and state diagrams. Since the GameEngine, FileManager, Player and InputManager objects are the important objects of Survivor, behavioral scenarios of these particular objects are described in sequence diagrams. Moreover, the Balt, Cursor, Player and Ray objects are also important objects so their states are examined in the state chart diagrams.

### 3.2.1 State Chart

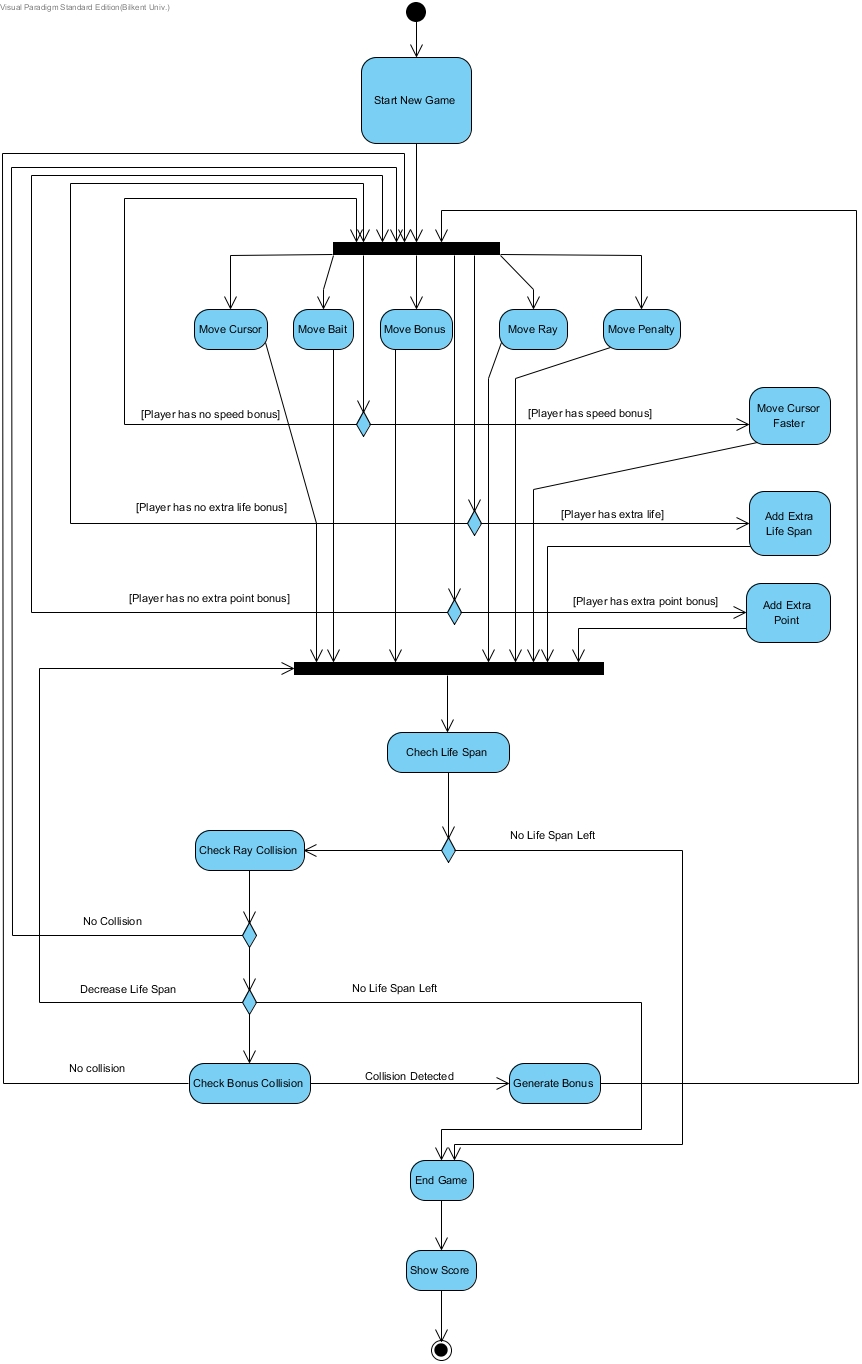


Figure 22: State Chart Diagram

When the user selects Play Game option from the Main Menu, the system initializes the game features, objects and crates the game scenario. When the preparation part is done the system waits for the user to interact with the mouse in order to start the cursor moving. When the user starts to move the cursor the system checks the game area to understand whether there are any baits or other objects falling from the top.

From this point, the system acts according to different scenarios until user’s lifespan is completed. When a collision is detected, the system decides next operation according to type of collision. For instance if the cursor collides with a bait, the players points increase and the game continues. However if the cursor collides with a bomb the player loses one life. Moreover if all of the lives of the player is lost the game stops and unfortunately the player loses the game. Then the system shows the score of player, finally.

If the detected collision is between the cursor and bonuses system applies bonus features to game status. In other cases system simultaneously updates game objects and plays collision sounds according to colliding game objects. Moreover, if player still has life and there is at least one life left, system maintains game loop and look for next collision. If the player still has life and he/she has collected at least 10,000 points, system provides access to the next level.

### 3.2.2 Sequence Diagrams

#### 3.2.2.1 Start Game

**Scenario:** The Player requests the System to start game by pressing the icon on desktop. After that System initializes a full screen window and opens the main menu page that has buttons for activities such as starting the game. Then the player clicks the start game button. Now the System creates the game engine to perform the show. After that game engine calls the file manager and it starts setting the game by getting the settings, getting images, getting scores and also the determining the level of the game. System then loads current scene by reading its object distribution from file manager, which just consists cursor, ray, player, bait and penalty objects (rock and bombs). Lastly, system enters the game loop in order to continuously update the game.

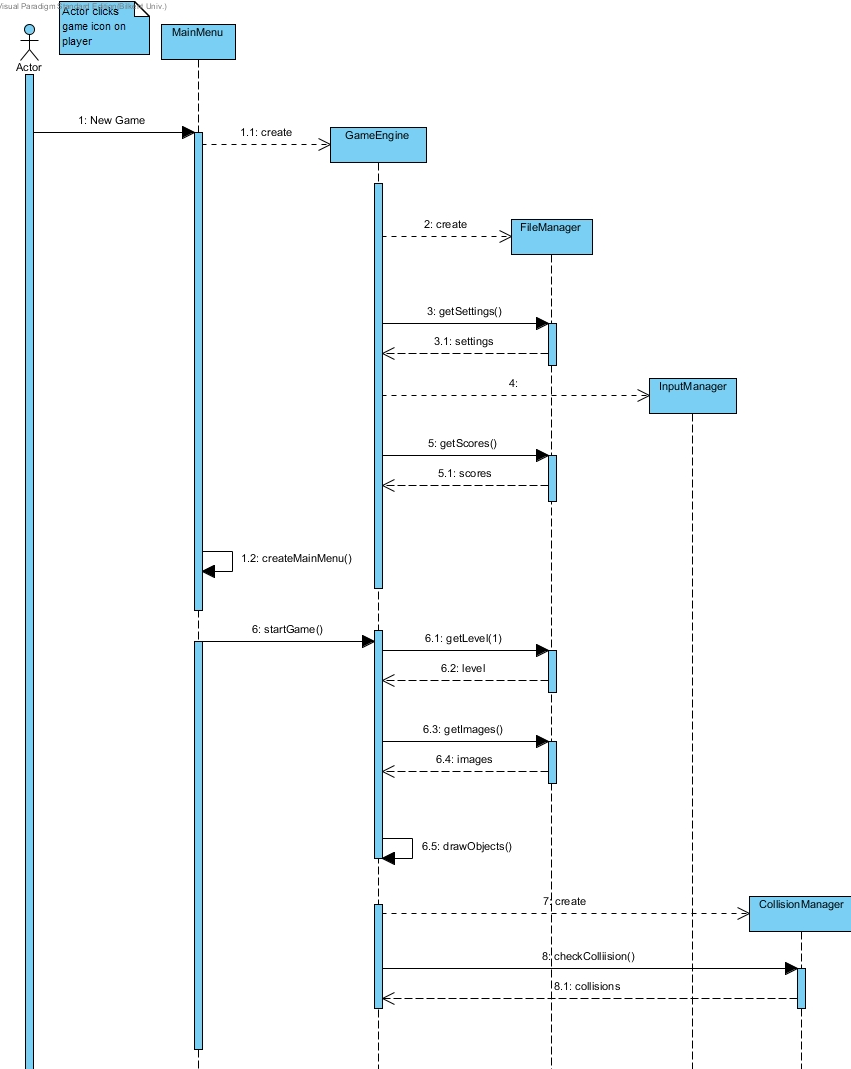
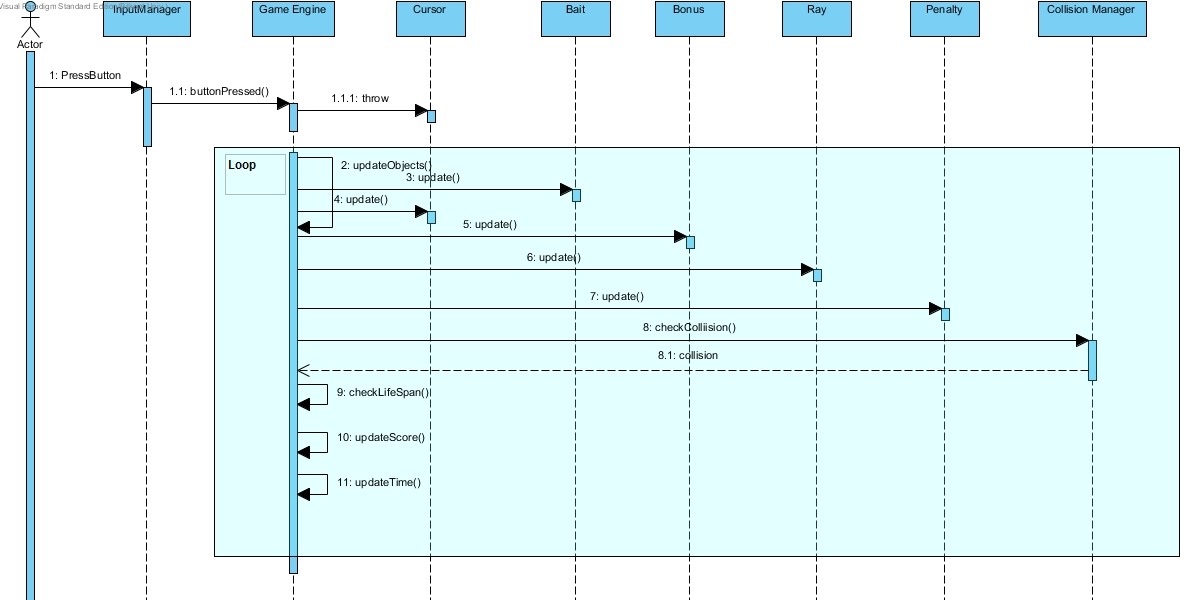


Figure 23: Sequence diagram which explains start game scenario

Here MainMenu is a boundary object by which user can perform some activities like starting game, viewing credits, changing settings. The locations of the game objects are stored in FileManager class, and it is responsible for the organization of the game screen.

#### 3.2.2.2 Play Game

Figure 24: Play game sequence diagram

**Scenario:** When user starts the game by selecting the play game choice the system initializes the game by instantiating game engine first. Then the GameEngine calls the FileManeger and they together set the settings for the game. When this preparation process is finished system waits user to interact with mouse to start the game. Firstly user starts the game by moving the cursor right and left to catch the baits. However, there could also be penalty objects falling from the above. If there is rocks or bombs that collide with the cursor, system checks the Player’s number of lives to make a decision between two alternatives which are continuing game and ending game. If Player has no life, game is over. If Player has life, the GameEngine contacts with the CollisionManager to check whether there is a collision. Then if there is collision the GameEngine updates the life span of the player and it may decrease the time left for the player to finish the level. When a collision is detected, the system decides next operation according to type of collision. The time and life span damage that is caused by bombs are much more effective than that of rocks.

Moreover, if the cursor collides with a bait the user gains points and that affects the possibility of him to pass the level. Other than the baits, the user can hit some bonuses which could produce alternative scenarios as having extra speed, gaining extra points or even increasing the life span. When any of these options happen GameEngine updates all of the objects and also contacts with the CollisionManager and produces the new settings according to the collision type that the CollisionManager decides.

#### 3.2.2.2 Change Settings

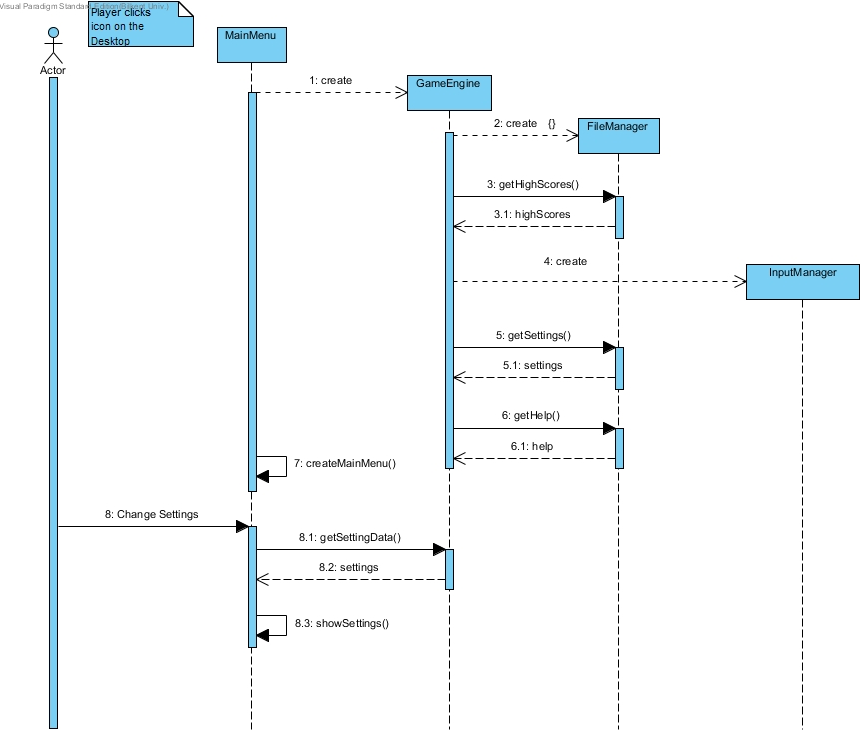


Figure 25: Change Settings Sequence Diagram

**Scenario:** Player decided to change the game settings. He presses change settings button from the MainMenu object, the System displays settings, and the Player changes the settings regarding his desires. After that, when Player presses back button on the settings menu, the GameEngine updates the settings of the game to the new ones.

Here, the Player, as usual, contacts the System via the MainMenu object, and sends the command, change settings. Then, the MainMenu object contacts with the GameEngine object so that the command is done. After that, the GameEngine works with the FileManger simultaneously to create the changes in the settings as the Player goes back to the playing the game. Moreover, in order to make some of the changes the GameEngine could also work with the InputManager. After this process is done the Main menu requests the GameEngine to return the settings and the GameEngine provides the settings that are changed. Finally the MainMenu shows the changed settings to the Player.

# 4. Conclusion

In conclusion, in this analysis report, we aimed to create an action game called Survivor. Our report has two main parts named, requirements specification and System Model.

In requirements specification, we examined mostly all of the possible requirements which any player could have performed. Moreover, we have indicated our functional and nonfunctional requirements regarding these performances.

In system model, our report consists of the following sections:

1.Use case model

2. Dynamic models

3. Class model

4. User interface

Firstly, we have decided our use case scenarios. Then, in the second part, System Design, we have made our sequence diagrams and activity diagrams. In our sequence diagrams we tried to demonstrate the possible actions that the player could take. Activity diagram indicates mainly our game play. It represents the actions of our game components, cursor, baits, bonuses and penalty objects. Moreover, our class diagram represents our implementation.

To sum up, Survivor is a game that we have decided to develop from the idea of DXBall. However, the game’s visuals and features will be unique to its attributes.