
Bilkent University

Department of Computer Engineering

**CS319-3E-DE Project**

*The Defenders: Cinematic Edition*

Analysis Report

Daniyal Khalil, Leyla Ismayilova, Emil Alizada, Leyla Hashimli, Zulfugar Verdiyev

Instructor: Eray Tuzun
Teaching Assistant(s): Baris Ardic, Hasan Balci, Alperen Cetin

Progress Report
October 18, 2019

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Project of CS319.

# Contents

# Analysis Report

*Project short-name: Defenders*

## 1 Introduction

This is CS 319 analysis report for project called Defenders. The report is divided into multiple sections. In the Proposed System section we discuss the overall essence of the game. We also give brief information regarding the functionality of the game. Proposed System has also subsections where we discuss the analysis of our game in detail. In the Glossary section we define all the technical terms and provide meaningful definitions for non-experts to understand. In the references part we cite the sources that we used while making this report.

The aim of the Project is to recreate a previously made arcade game called the Defenders made in 1980. The recreation is supposed to be a modern day implementation of the previous with new modifications and features to make the game more interesting and exciting for the user. The previous defenders obviously had a lower level GUI and features, and this model will exceed that game in all aspects.

Figure 1: Original game

# 2 Proposed System

## 2.1 Overview

Defenders is a shooting-staying alive arcade game created by William Electronics in 1980. The player goes around. This is where you provide the details of the results of your analysis work.d a long, mountainous landscape and shoots the targets that can cause a potential damage to the player. The field is circular, so moving in one direction will eventually bring you back where you started. The controllers help you navigate the player, to both move around to escape the attacks and attack back. It is a continuous game, so the goal is to score the highest point and break a record, as in most of the arcade games. The player is armed with a laser-like weapon which can shoot rapid strikes in one horizontal direction. Changing direction also causes small movement along the intended direction. The Game also has multiple modes. Choosing each mode gives you a different perspective, while the general intention and functionality remains the same. Each player can also create a profile to keep their scores.

### 2.1.1 Mode and Characters

In our project, we decided to change the spaceship and aliens theme completely. Instead of this theme, this version will have 3 modes, which are well-known universes from comics and books. As characters, heroes and villains of these universes will be used. To illustrate, the game will start with Marvel Universe as default. Player will use heroes of Marvel Universe to save the world from attackers. The same will also apply for other universes. Also, since there are lots of heroes in these universes, player can buy other characters in the store. The power hierarchy of both heroes and villains will also maintain in the game. According to their power place in hierarchy, their health will change. Basically, powerful characters will not die easily.

## 2.2 Functional requirements

The first screen that player encounters is the “Profile” screen when they enter a game. In this screen player either choses from the options below:

- Create New Profile
- Load Profile
- Settings

If the player opts to create new profile game leads him to the screen with fields to fill in such as:

- Username

After successful creation of the profile or log in operation player is proceeded to the “Main” screen where he is able to select game mode out of two available, which are:

- Avengers

- Justice League

The “Main” screen also includes functionalities below,

- Start Game
- Mode
- Store
- Settings

### 2.3 Non-functional Requirements

A bulleted list is below:

- JavaFX will be used for smooth animations and overall performance.
- Game will have the ability to save different profiles for different players.
- The game is coming as self containing package which will not require installation.
- We are limiting our game to 30 FPS in order to get maximum quality/performance ratio.

### 2.5 System Models

#### 2.4.1 Scenarios

a. Creating a new Profile
    i. Start the application
    ii. Go to Create New Profile
    iii. Enter a unique username

b. Creating a profile with already defined username
    i. Start the application
    ii. Go to Create New Profile
    iii. Enter already existing username
    iv. Change username to make it unique

c. Loading a Profile
    i. Start the application
    ii. Go to Load Profile
    iii. Choose a profile from the list

d. Increasing/Decreasing the Sound Effects/Audio Volume
    i. Start the application
    ii. Go to Settings

iii. Set the Volume

e. Buying a character/weapon
    i. Start the application
    ii. Load/Create new profile
    iii. Go to Store
    iv. Buy the character/weapon (if enough coins)

f. Selecting a character/weapon
    i. Start the application
    ii. Load/Create new profile
    iii. Go to store
    iv. Select from ‘Owned’ character/weapons

g. Select Mode
    i. Open the application
    ii. Load/Create Profile
    iii. Go To Select Mode
    iv. Select Mode

h. Play Game
    i. Open the application
    ii. Load/Create Profile
    iii. Go to Start Game

i. Pause Game
    i. Open the Application
    ii. Load/Create Profile
    iii. Go to Start Game
    iv. Press Pause

j. Change Volume while Play Game
    i. Open the Application
    ii. Load/Create Profile
    iii. Go to Start Game
    iv. Press Pause
    v. Set Volumes in the Pause Menu

k. Exit Game

i. Press cancel on the top right corner anytime.

l. Unpause Game
   i. Start Application
   ii. Load/Create Profile
   iii. Go to Start Game
   iv. Press Pause
   v. Press Pause again to Unpause

### 2.4.2 Use-case Model

Use-case model below represents user's all interactions with the system in Defenders game. User is only actor in this system.

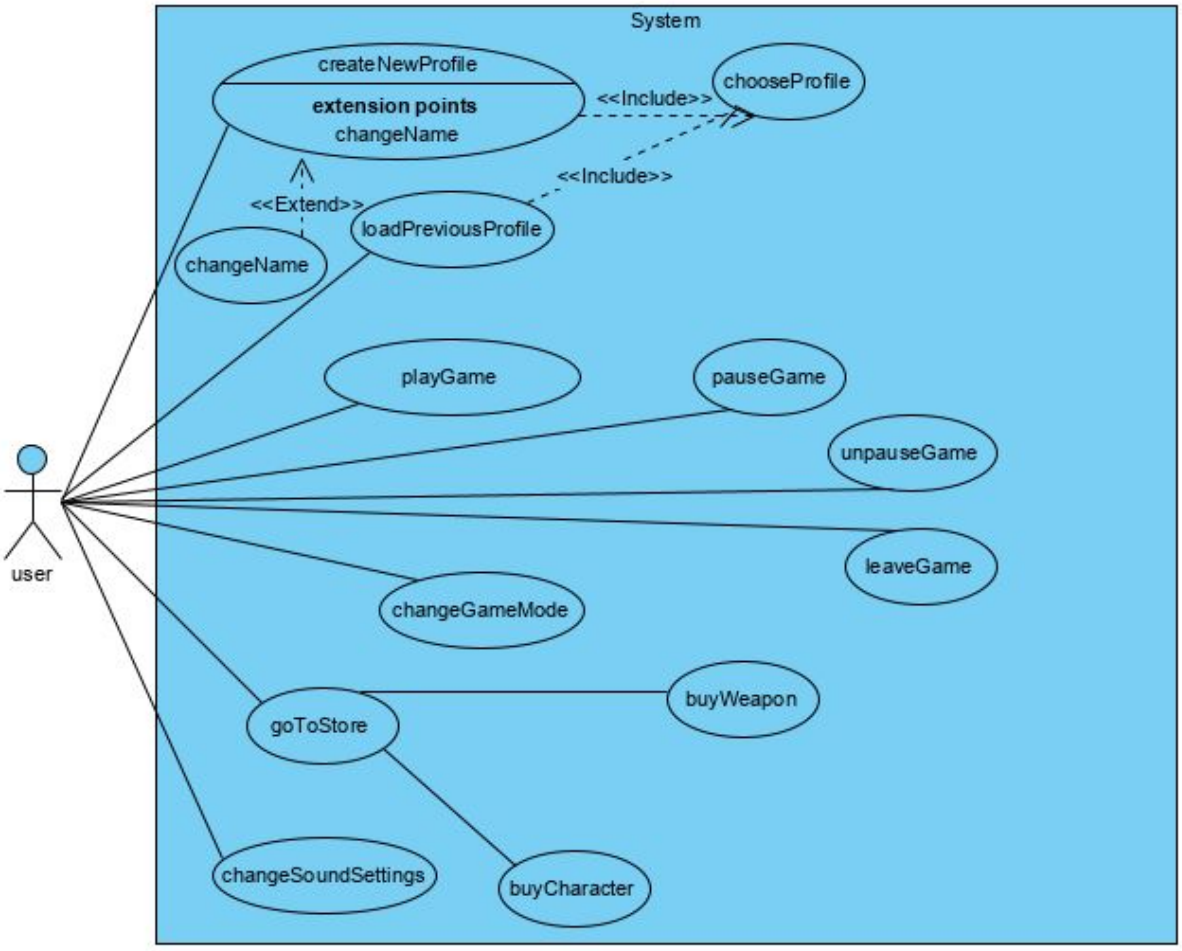



Figure 2: User Case Diagram of Defenders game

- **createNewProfile**
    - Participating Actor: User
    - Entry Conditions: None
    - Exit Condition: Create Profile / Close Pop up
    - Flow of Events:
        - System: Dialogue Pops Up
        - User: Enters a username
    - Special Requirement: None

- **changeName:**
    - Participating Actor: User
    - Entry Conditions: The User is in createProfile
    - Exit Condition: Create Profile
    - Flow of Events:
        - System: Dialogue Pops Up
        - User: Enters a username
        - System: If username is unique, createProfile. If not ask the user again.
        - User: Enters username again if not Unique
    - Special Requirement: Needs to enter an already used username in createProfile

- **loadPreviousProfile:**
    - Participating Actor: User
    - Entry Conditions: None
    - Exit Condition: Load Profile / Close List
    - Flow of Events:
        - System: A list of users pops up
    - Special Requirement: None

- **chooseProfile:**
    - Participating Actor: User
    - Entry Conditions: Needs to LoadProfile
    - Exit Condition: Loads Profile
    - Flow of Events:
        - User: Choose a profile to load
    - Special Requirement: None
- **playGame:**
    - Participating Actor: User
    - Entry Conditions: Load/Create Profile
    - Exit Conditions: Exit Game
    - Flow Events:
        - User: Moves Character and shoot weapons
        - System: Creates Monsters
        - User: Dies/Pause/Exit Game
    - Special Requirements: None

- **changeGameMode:**
    - Participating Actor: User
    - Entry Conditions: Load/Create Profile
    - Exit Condition: Select Mode
    - Flow of Events:
        - System: Mode window pops up
        - User: Chooses a mode
        - System: Changes the mode.
    - Special Requirement: None

- **goToStore:**
    - Participating Actor: User

    - Entry Condition: Load/Create Profile
    - Exit Condition: Close Store
    - Flow of Events:
        - User: Chooses to Store
        - System: Shows Store
    - Special Requirement: None

- **buyWeapon:**
    - Participating Actor: User
    - Entry Conditions: In Store
    - Exit Condition: Buy Weapon
    - Flow of Events:
        - User: Selects on the weapon he wants
        - System: If User has enough money, weapon selected, if not return alert user.
    - Special Requirement: Needs to have enough money.
- **buyCharacter:**
    - Participating Actor: User
    - Entry Conditions: In Store
    - Exit Condition: Buy Character
    - Flow of Events:
        - User: Selects on the character he wants
        - System: If User has enough money, character selected, if not return alert user.
    - Special Requirement: Needs to have enough money.

- **changeSoundEffects:**
    - Participating Actor: User
    - Entry Conditions: None

- Exit Condition: Close Pop up
- Flow of Events:
    - System: Dialogue Pops Up
    - User: Alters the Volumes
- Special Requirement: None

### 2.4.3 Object Class Model

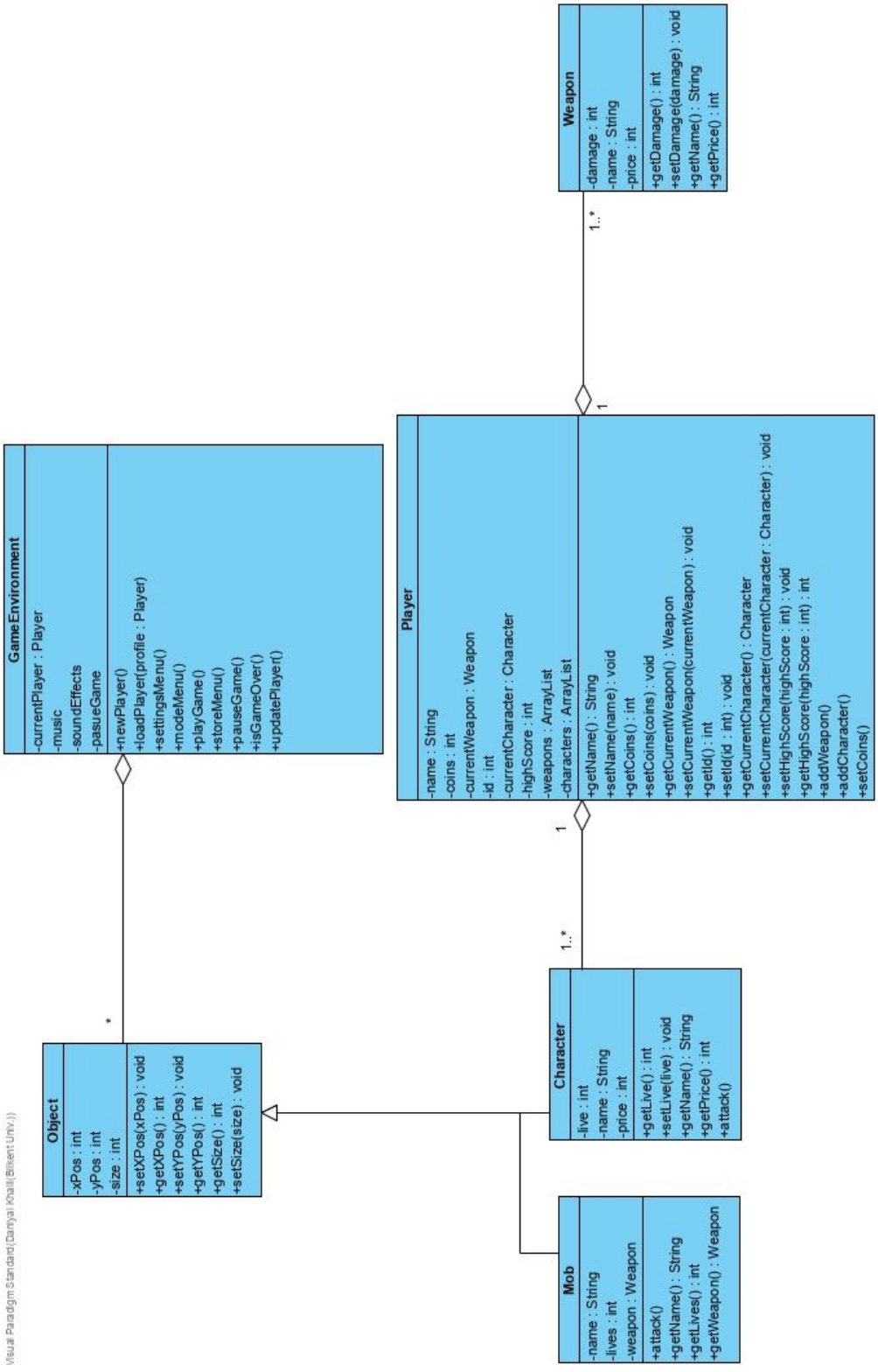



Figure 3: Class diagram for Defenders game

1. Player Class:
    - Attributes:
        i. string name: The unique username of the player
        ii. int coins: The current number of coins that the player has
        iii. weapon currentWeapon: The weapon user is using currently from the weapon store
        iv. character currentCharacter: The current character the user is using. (for e.g Thor)
        v. int highScore: The highest score this user has ever achieved
        vi. Arraylist<weapon> weapons: The list of weapons the user owns
        vii. Arraylist<character> characters: the list of characters the user owns.
2. Weapon.
    - Attributes
        i. string name: The unique name of the weapon.
        ii. int damage: The amount of damage the weapon can do.
        iii. int price: The price it is in the store for.
3. Character
    - Attributes
        i. string name: The unique name of the character.
        ii. int lives: The number of lives the character has.
        iii. int price: The price it is in the store for.
4. Mob (Computer Controlled NPCs)
    - Attributes
        i. string name: The unique name of the enemy.
        ii. int lives: The number of lives the character has.
        iii. Weapon weapon: The only weapon an enemy can use.
5. Object (Super class for all the game objects stated above inherit)
    - Attributes
        i. int xPos: x-coordinate of the object on the screen

ii. int yPos: y-coordinate of the object on the screen
iii. int size: The size of the object on the screen.

6. GameEnvironment (The class for the interaction of all the objects in the game, this class will interact with the GUI)
    - Attributes
        i. Player currentPlayer: The current user that is logged in the game.
        ii. int music: The int number for the volume of music in the game.
        iii. intSoundEffects: The int number for the volume of SoundFx.
        iv. boolean pauseGame: The boolean for whether the game is paused or not.

### 2.4.4 Dynamic Models

a. Activity Diagram:

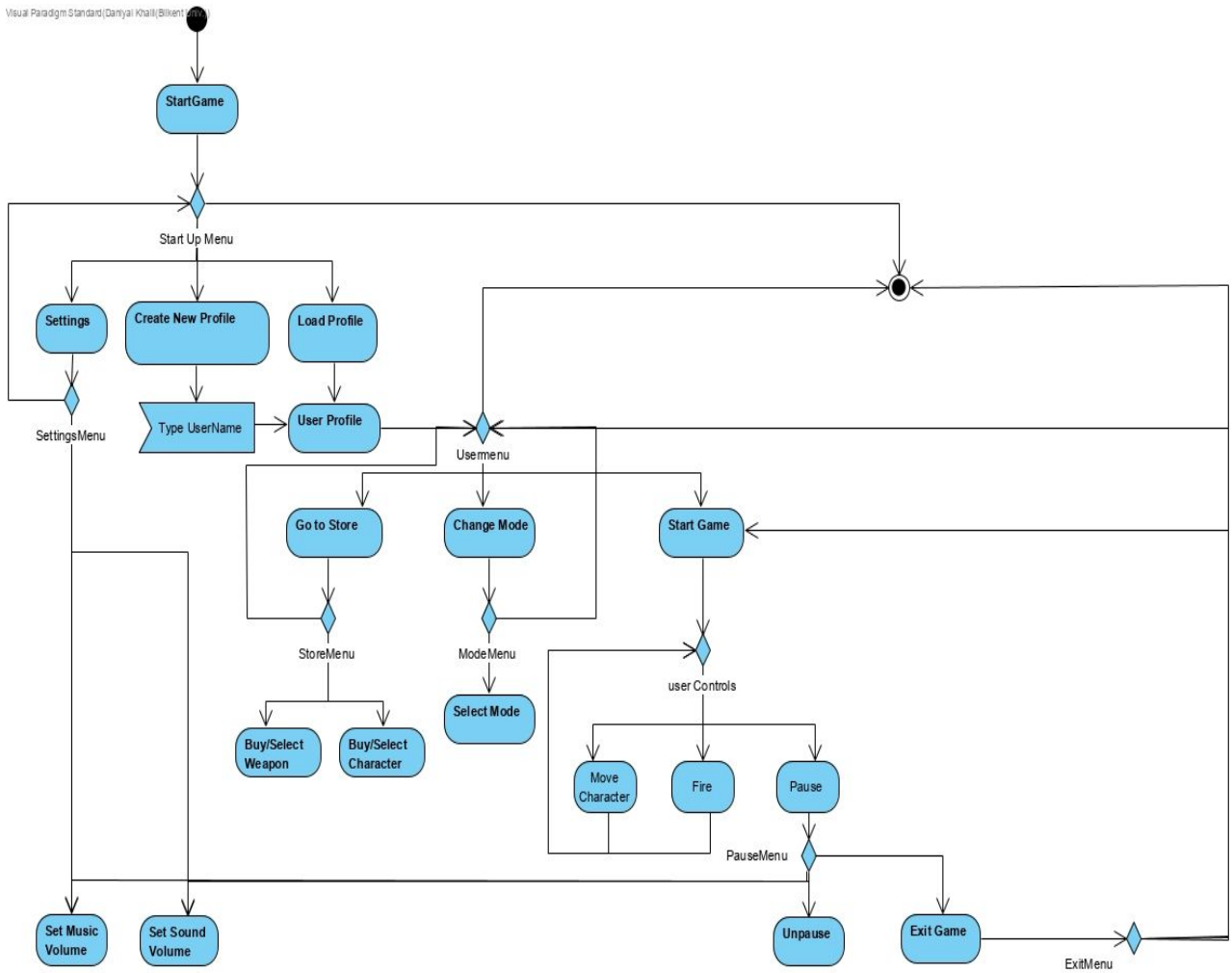



Figure 4: Activity Diagram of Defenders game

b. Sequence Diagrams:

The first diagram explains user-system interaction regarding its way to the gameplay. Firstly, user can choose a new profile. Then he/she will have to type name to create the profile. Two things can happen - one that the name is already taken, and two the name is not there so you can create your profile. If the name is already taken, then profile sends you a message stating that the name is taken, so you can choose another unique name. Afterwards, user continues with the game. When he/she reaches the gameplay, since our game is non-stop, eventually he/she will lose and system will notify that the game is over.

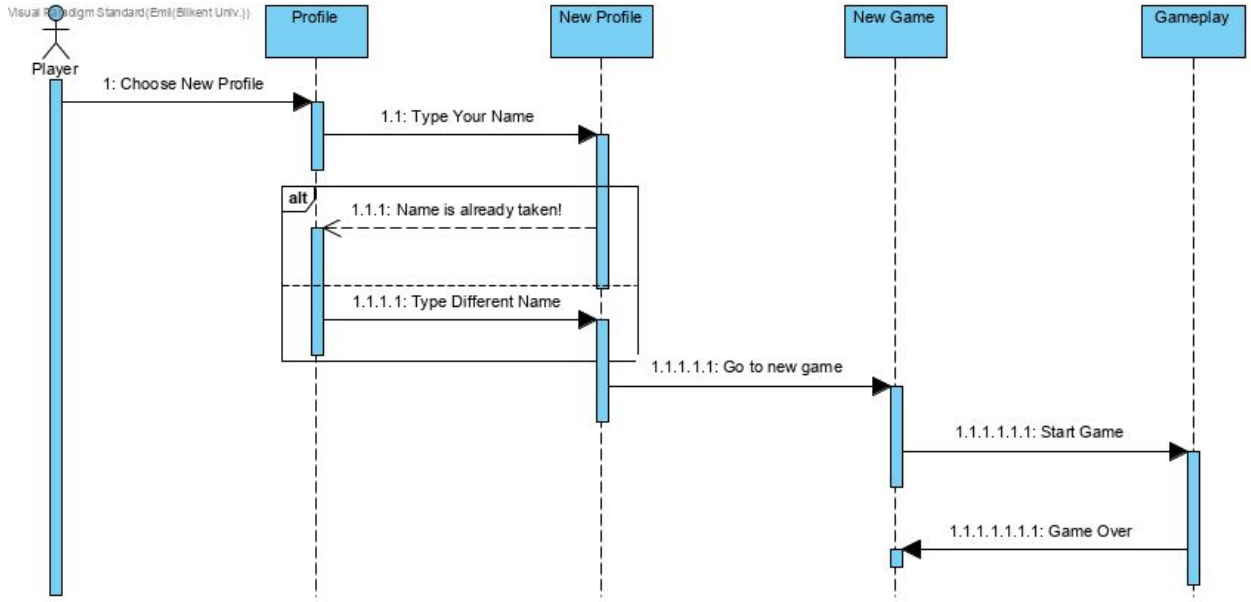



Figure 5: Sequence diagram for playing a game

This is the sequence diagram that displays the interaction between user and its way to the store. It is basically an alternative option to the previous sequence diagram. User can either choose to buy something or return without buying.

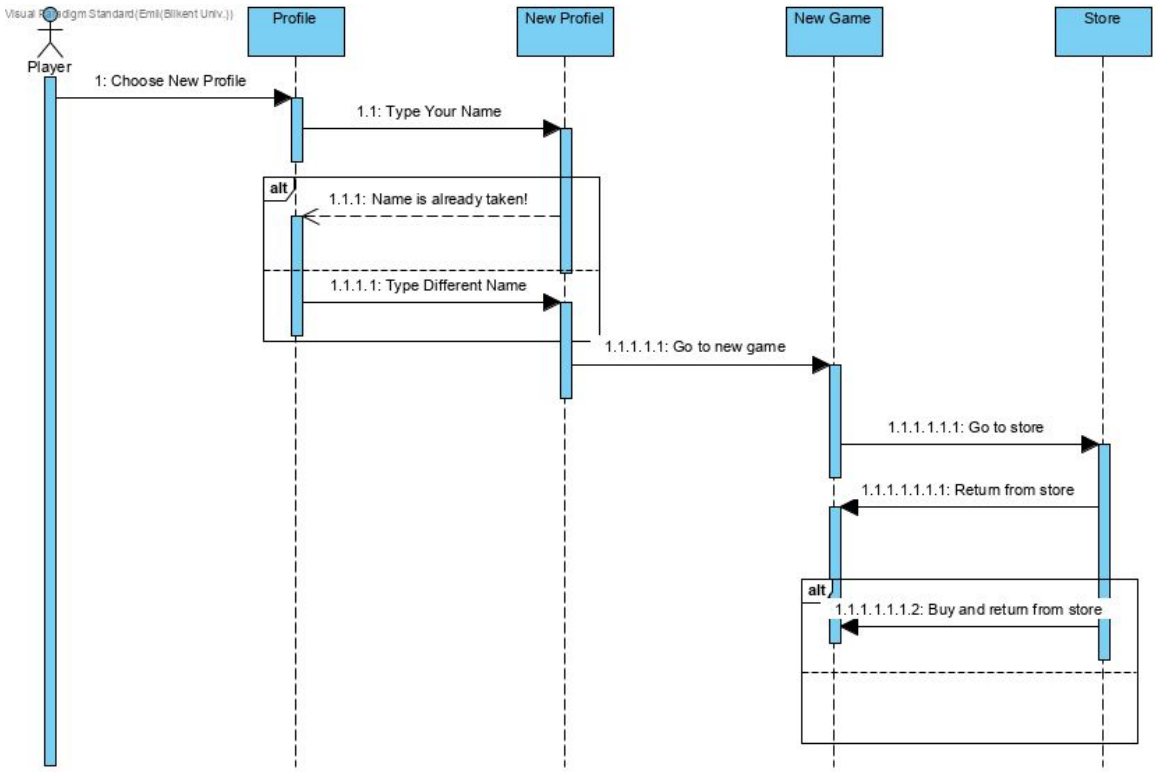



Figure 6: Sequence diagram for buying an item in store

This is the sequence diagram depicting user-mode interaction. User will choose the mode, once he/she chose the new game. The player definitil

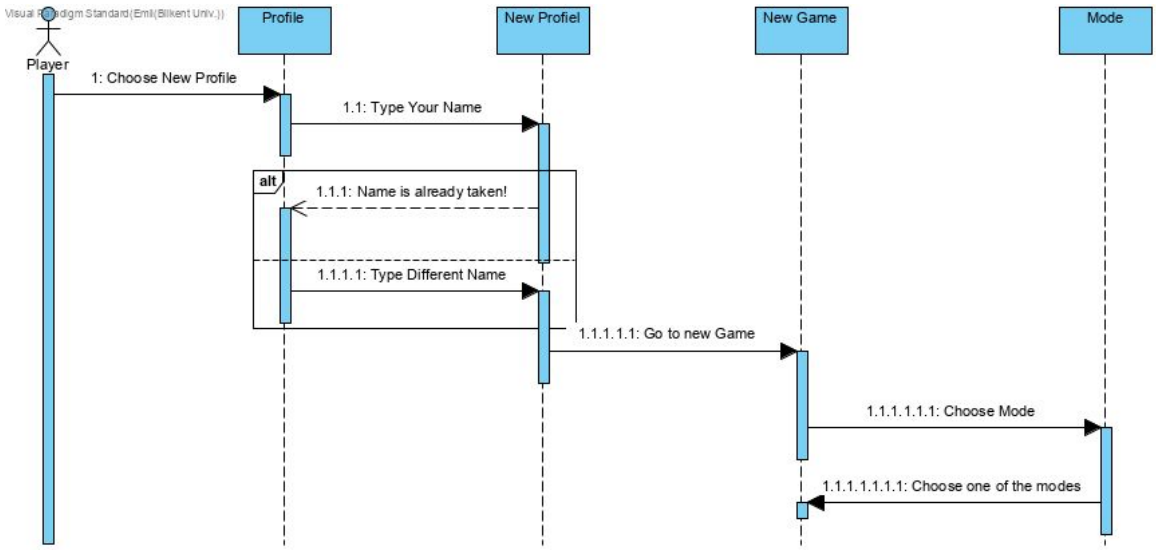



Figure 7: Sequence diagram for changing the mode of game

c. State Diagrams:

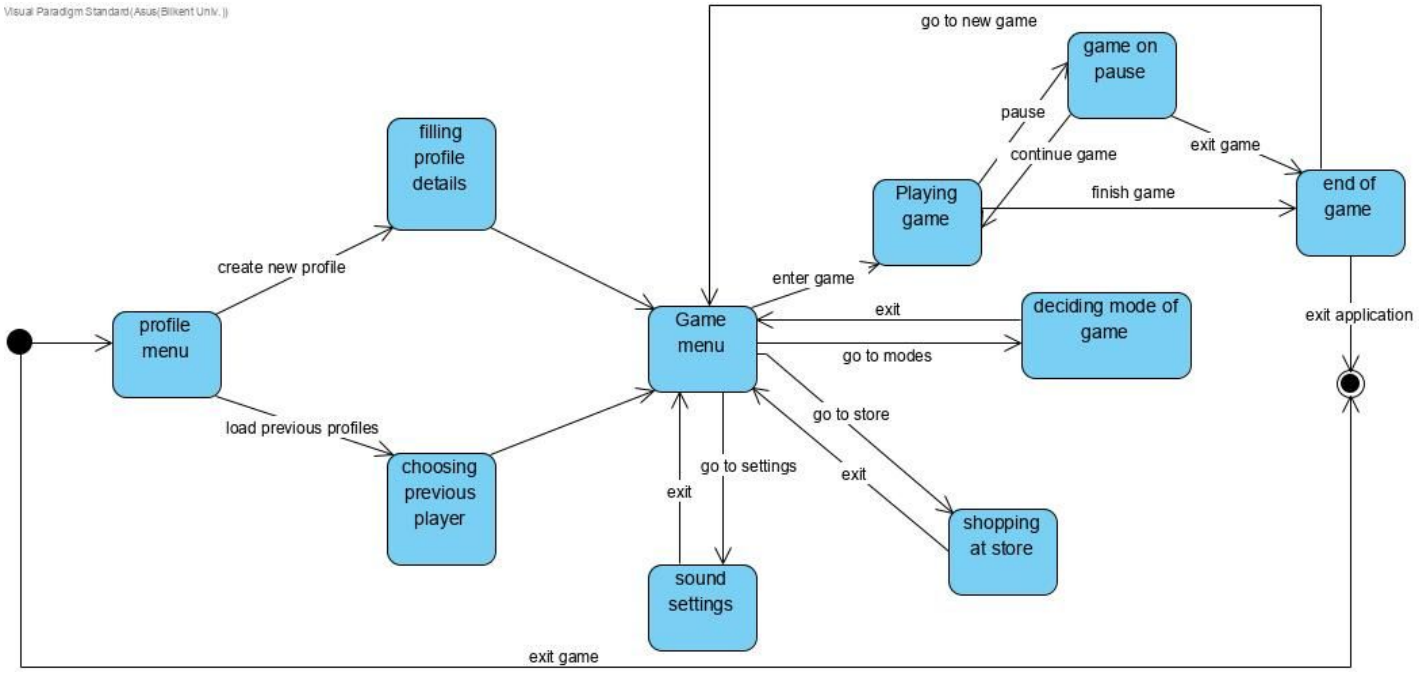



Figure 8: State diagram of Defenders game

### 2.4.5 User Interface

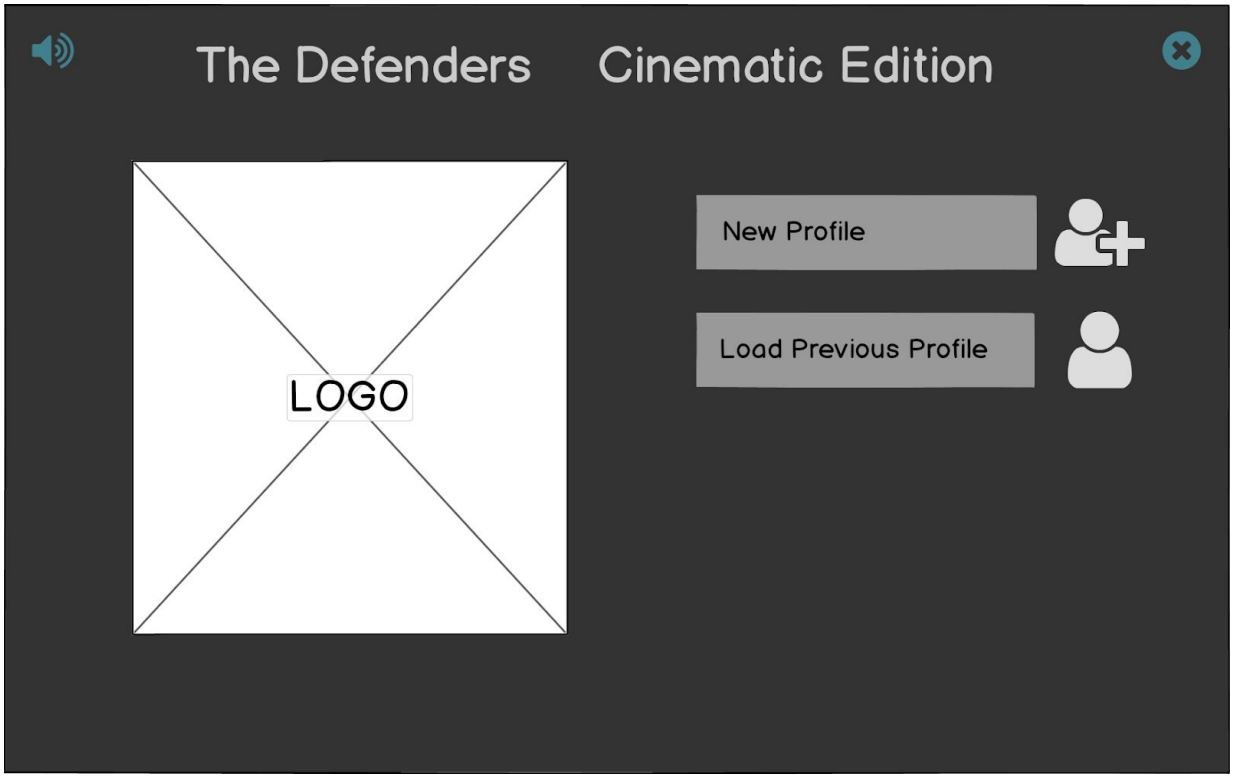



Figure 9: Opening screen of Defenders game

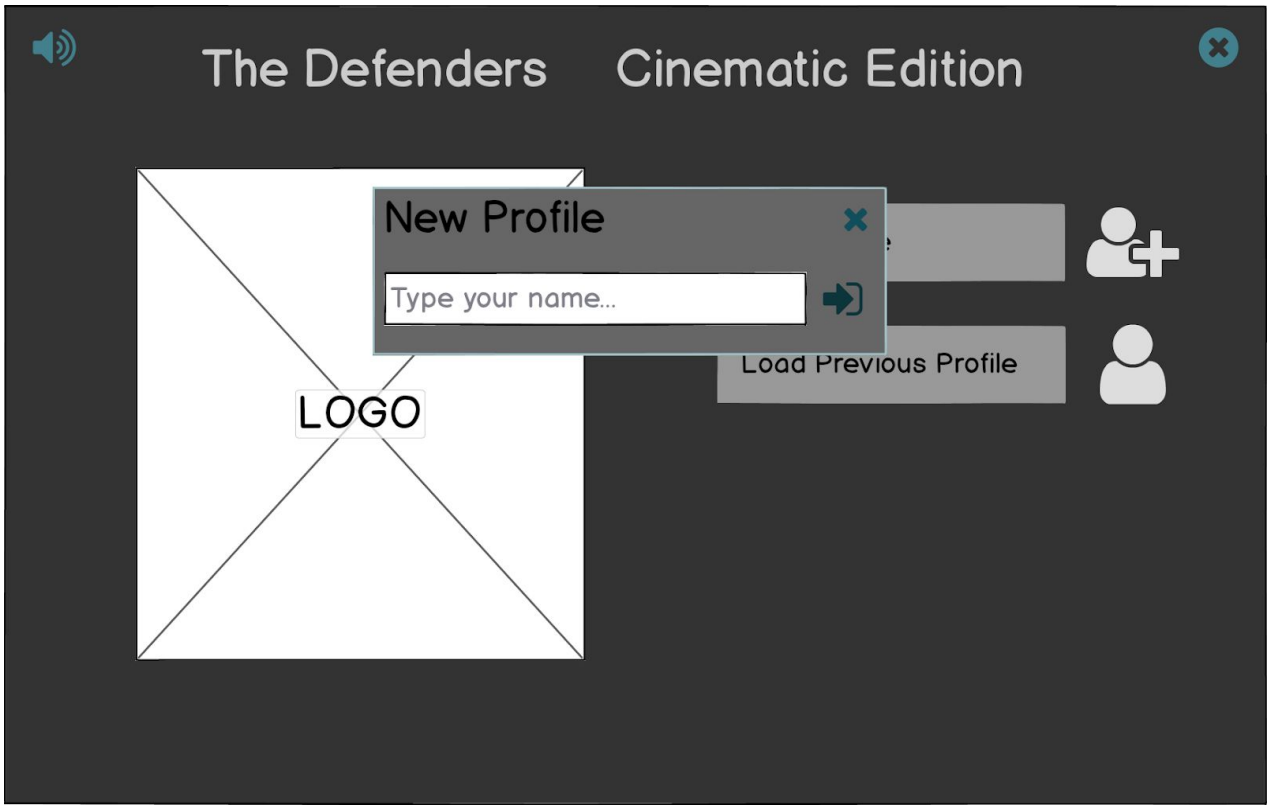



Figure 10: Creating new profile


Figure 11: Choosing previous profile

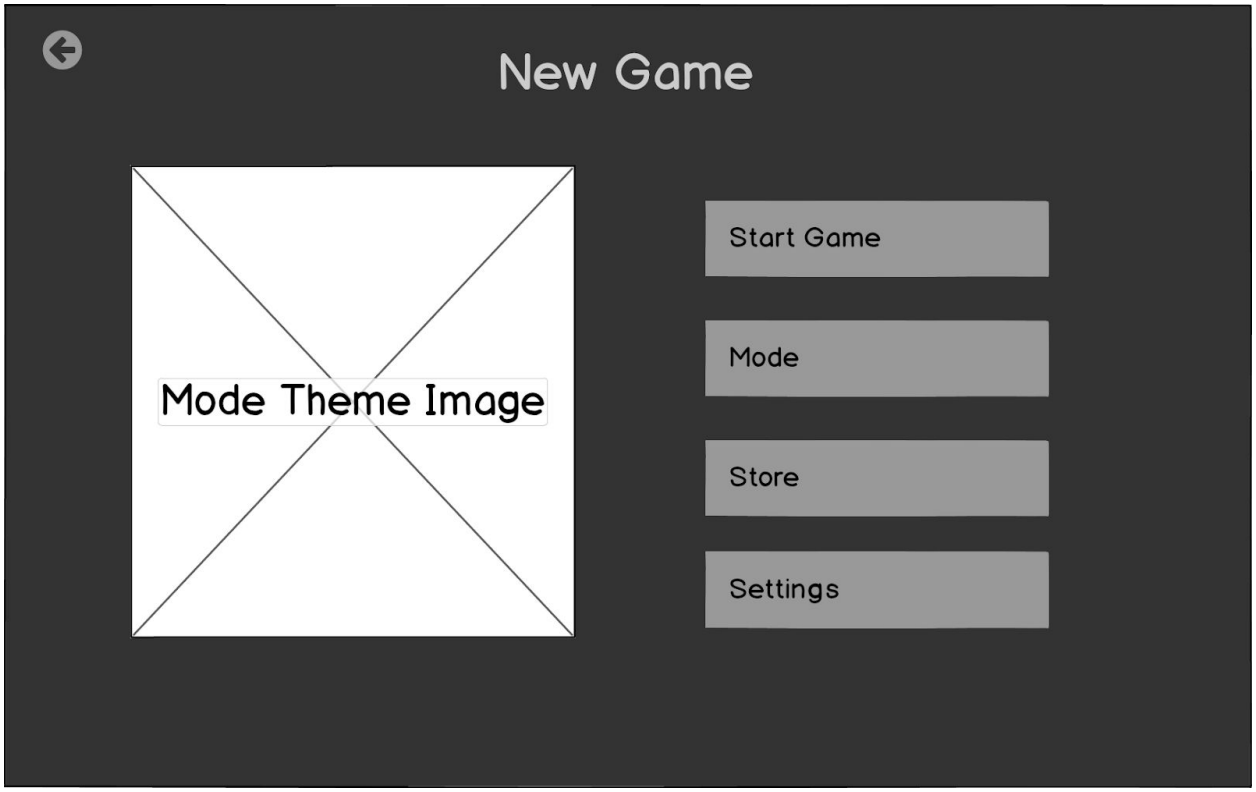



Figure 12: new game page

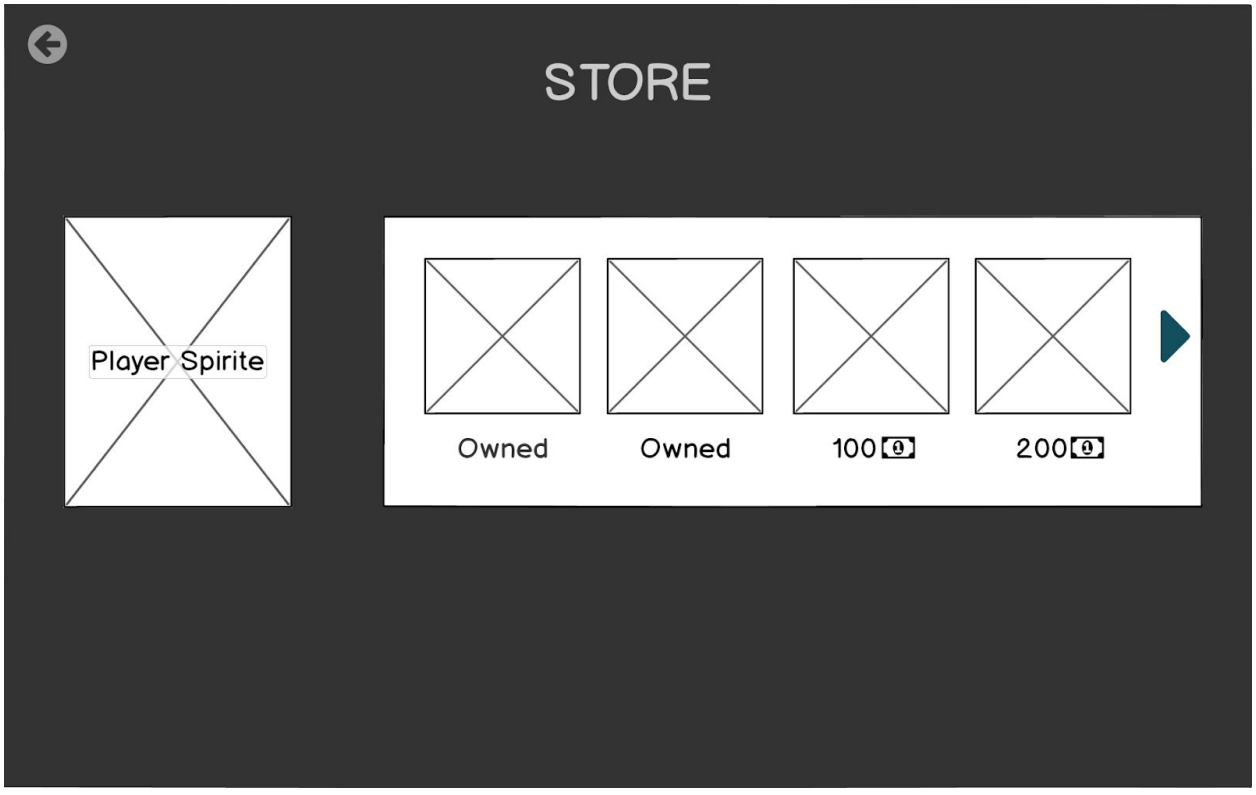



Figure 13: Shopping at the store

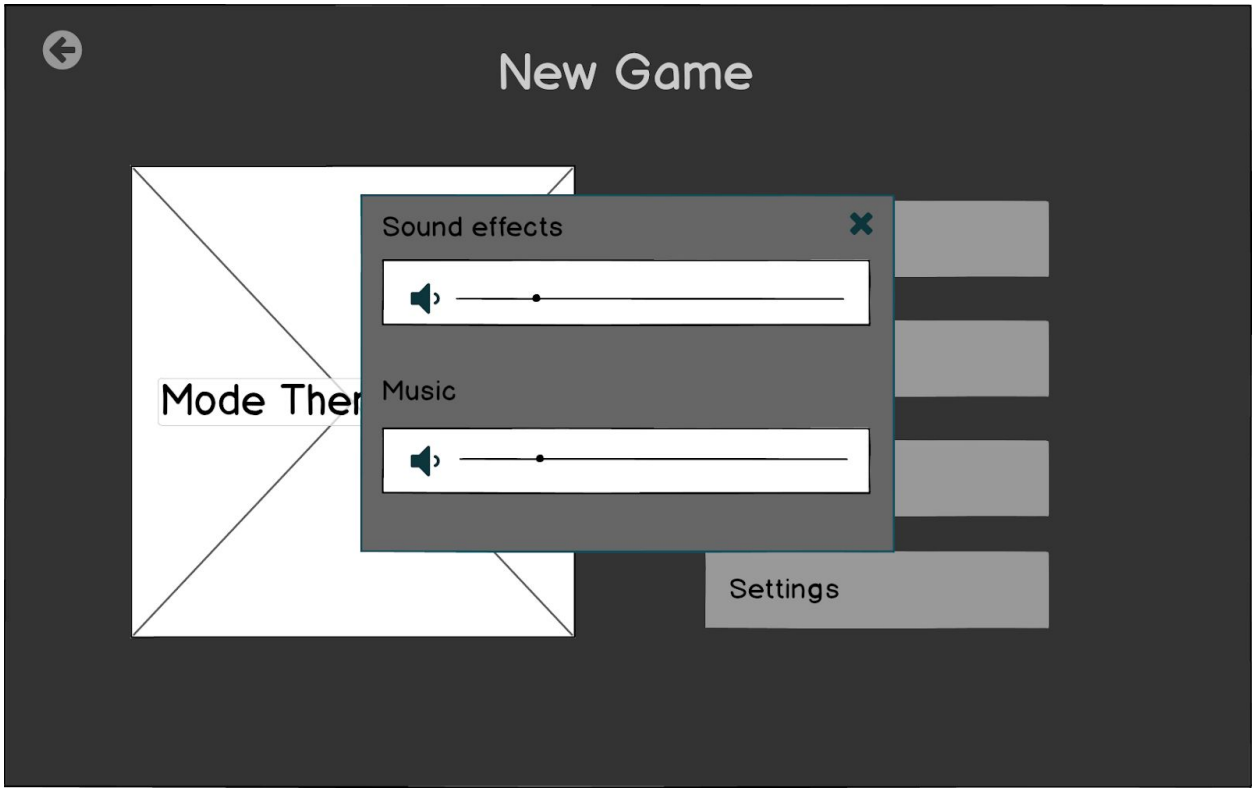



Figure 14: Sound settings

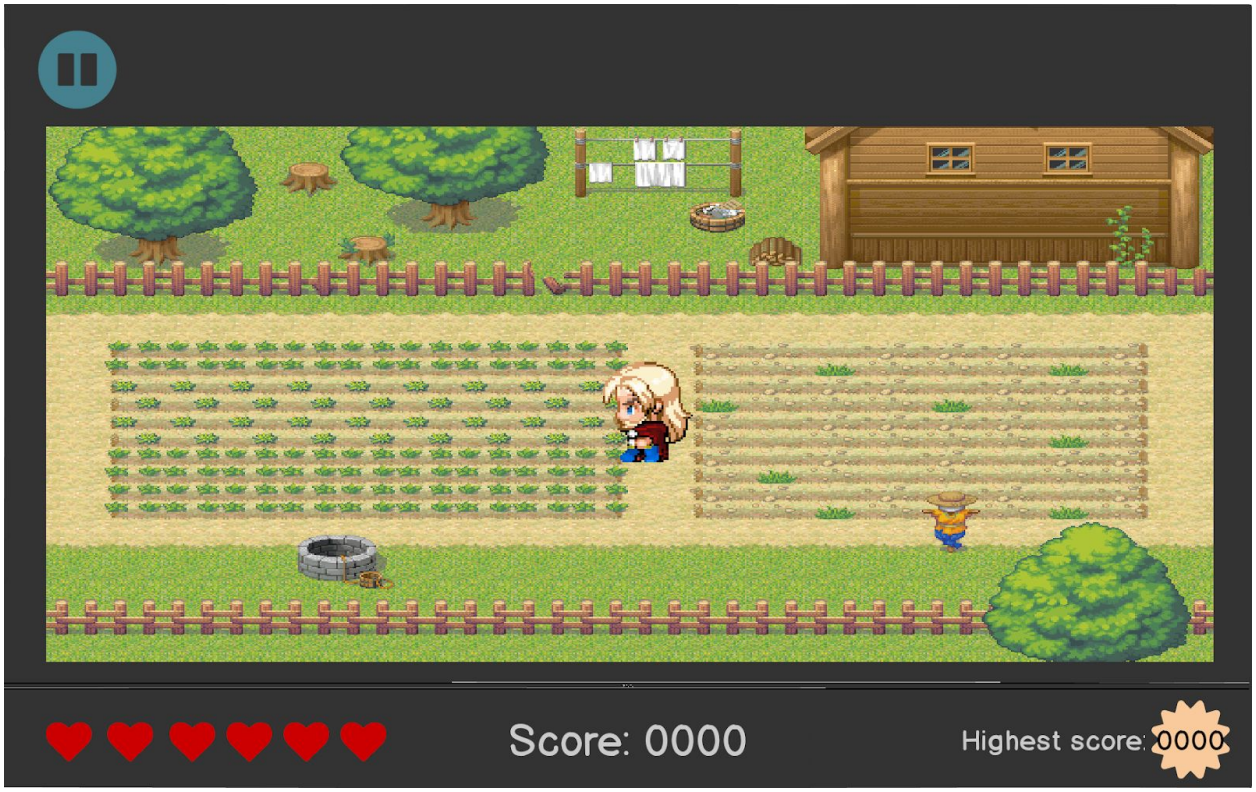



Figure 15: Game screen

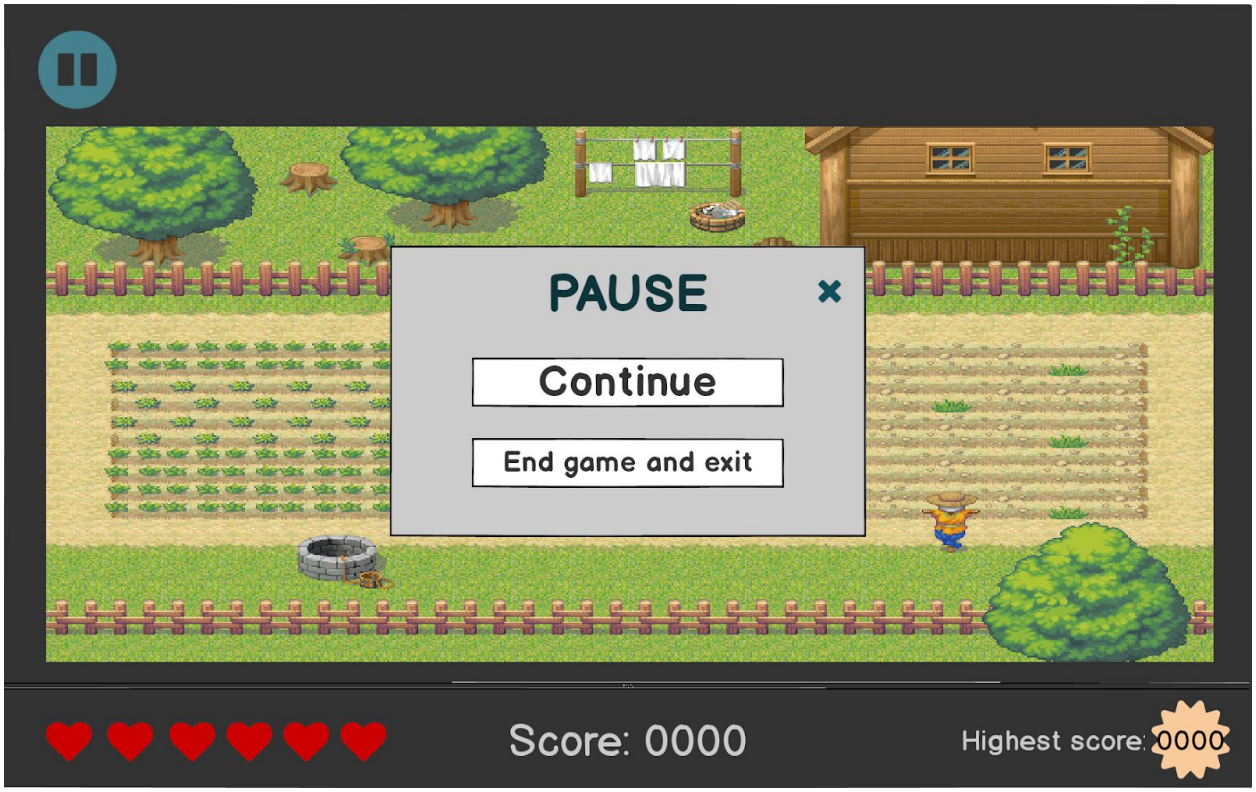



Figure 16: Pause on game

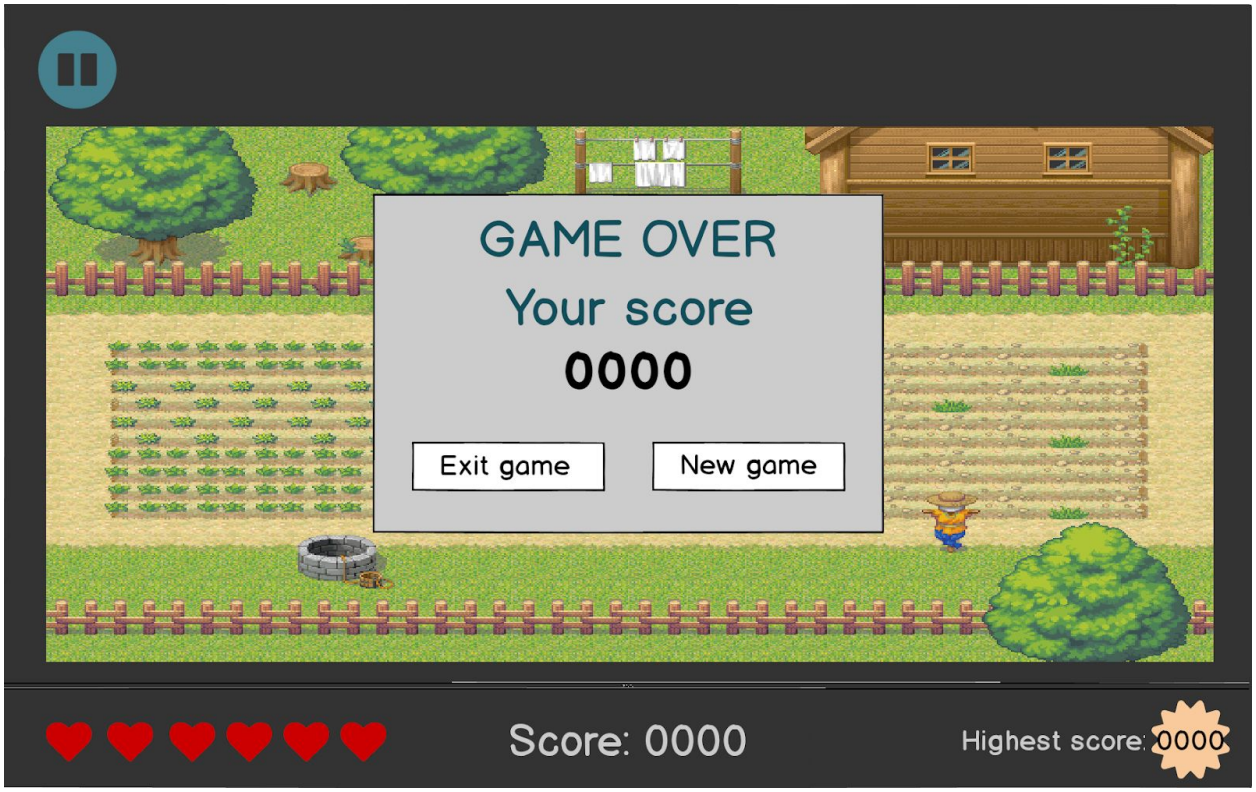
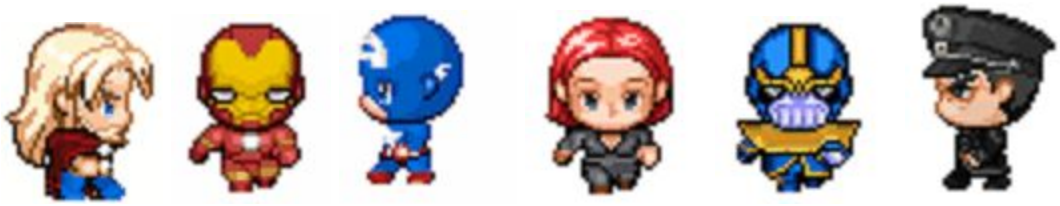



Figure 17: Game over screen



[3]

Figure 18: Animated characters of Defenders game

## 3 References


[1] Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.

[2] https://www.arcade-museum.com/game_detail.php?game_id=7547

[3] https://forums.rpgmakerweb.com/index.php?threads/marvel-characters-sets-sv-battlers-aveng
ers-spider-man-x-men-more.101244/