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
December 1, 2019

Contents

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 [1]. 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.

2. Current System

The basis that we are using for our project is Defender II designed in 1981. The game came out as an arcade game. The game is a single player game, even though it does offer 2 players option in the menu it still does not provide a multiplayer functionality. The 2 player is just turn based play between 2 users. The game works in such a way, that the user has a ship that can be moved, either left, right,up or down but the user can only shoot in either left or right direction. The screen follows the user whenever the user moves left or right. The game pane loops the user back to the other end after the user reaches one end of the screen. The enemies come from outbounds of the screen and don't just spawn randomly in the screen of the user. Enemies can shoot as well but they do not always shoot in the same direction as user. The user can kill enemies by shooting only. The user dies if it collides with the enemy ships or gets hit by their bullets. For every enemy you kill you get score points. There are humans in the game as well, in the bottom of the pane, the enemy ships can pick up the humans from the bottom of the screen there is no penalty if the human gets taken away. The user needs to kill the enemies to defend the humans from them. This is nearly the overall gameplay of the Defenders II.

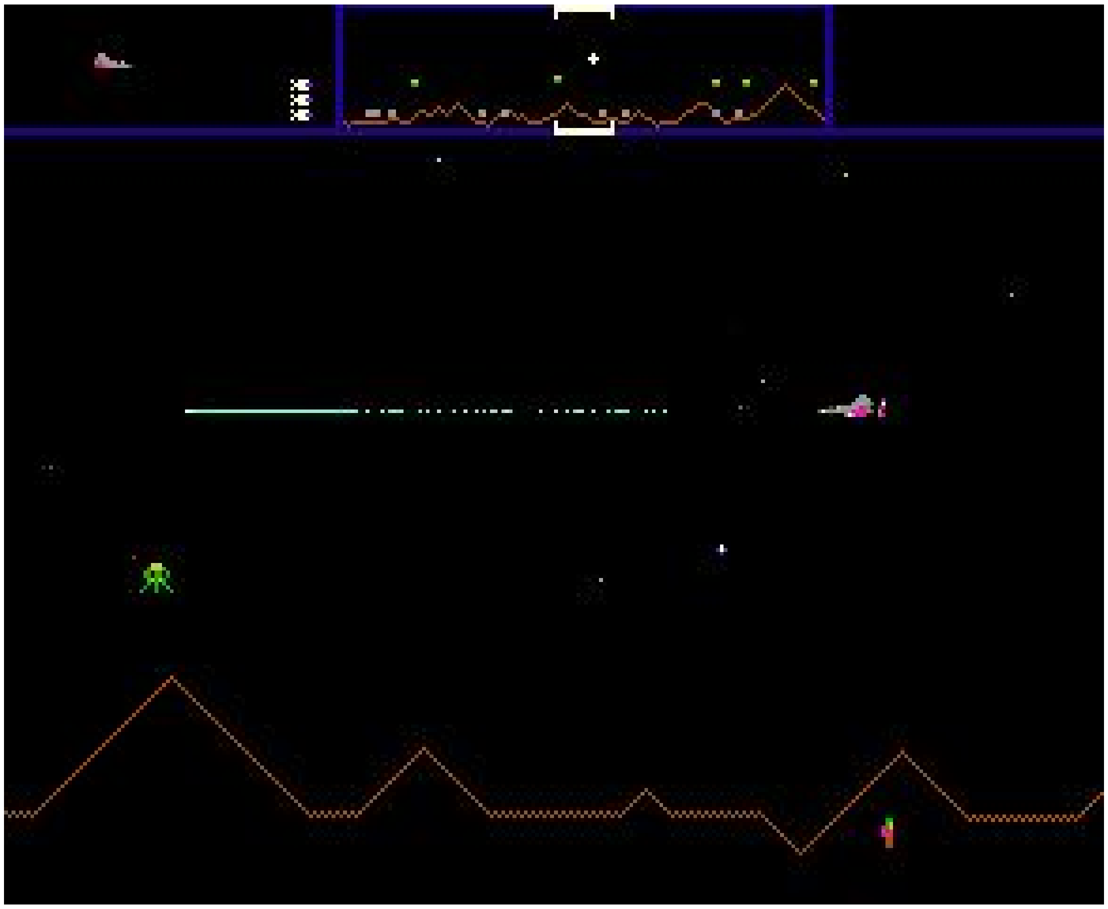


Figure 1: Original game [2]

3. Proposed System

3.1. Overview

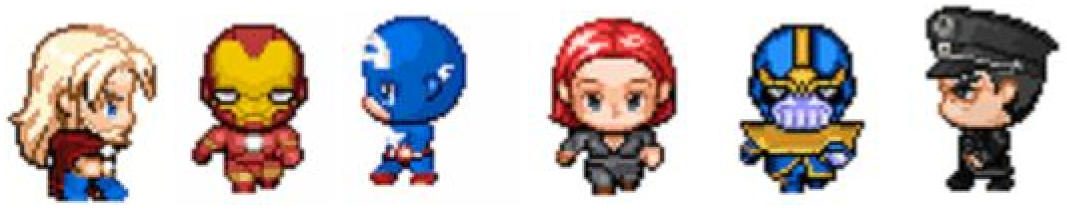
Defenders is a shooting-staying alive arcade game created. The player goes around a long, mountainous landscape and shoots the targets that can cause a potential damage to the player. The field is circular, so moving in one direction will eventually bring you back where you started. The controllers help 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.

3.1.1. Mode

In our project, we decided to change the spaceship and aliens theme completely. Instead of this theme, this version will have 2 modes, Marvel Universe and Justice League, which are well-known universes from comics and books. As characters, heroes and villains of these universes will be used. If user does not change mode, the game will start with Marvel Universe mode as default. It is also possible to change game mode to Justice League as an alternative game mode.

3.1.2. Characters and Enemies

The game consists of the player, enemies and civilians. In default mode, the player will use heroes of Marvel Universe to save the world from attackers. The same will also apply for Justice League universes. The power hierarchy of both heroes and villains will maintain in the game. According to their power place in hierarchy, their health rate will change. Basically, powerful characters and enemies will not die easily. Also, since there are lots of heroes in these universes, player can buy other characters in the store. However, mixing the characters of the two different modes are not allowed. Civilians will just be stationary and get attacked by the enemies.


[2]

3.1.3. Weapons

In our game, both enemies and player has special weapon. The characters use the weapons of the chosen mode. To illustrate, in the game Captain America, the hero of the Marvel Universe, uses the iconic shield. Moreover, all weapons has damage property. Basically, some weapons are stronger than others in killing. Also, the user can buy additional and stronger weapons then default one in the store.

3.2. Functional Requirements

The first screen that player encounters is the “Login” screen when they enter a game. In this screen player should choose from the options below:

- Create New Profile : Choosing this will direct the user to a new screen in which the user will set a unique username for their game, if the name is not unique and the profile already exists the user will be asked to select a new username. The user can go back to the Login screen with the back button.

- Load Profile: This will show the user a list of all the profile that have been in the game, and the user can select from any of the profiles to be loaded. You can go back to login screen from this.

After the user Creates/Loads a profile, they will be directed to the Menu Screen. Which contains the following options.

- Play Game : This will allow the user to start the game.

- Mode : This will allow the user to go into 'Modes Screen', In modes the user will have the option to choose from 2 Modes. Currently the game only offers a choice between either Marvel/DC Universe. Choosing the Mode will automatically change the Characters, Weapons, Enemies from that Mode. For example choosing Marvel will give you Characters such as Captain American, and choosing DC will give you Batman.

- Store : This will allow the user to access 'Store', the store contains weapons and characters, that the user will be able to buy with coins, which they have. The items and characters with have a different amount of lives and damage respectively. For example, Thor will have more lives than Captain American because he is a Demi-God.

- Settings : This will allow the user to access settings, the only settings that can be changed in the game are Sound and Audio Effects, the user can either decrease or increase volume with the help of the slider bar.

After the Play Game, the user will be shown a screen with the character and enemies, coming from out of bounds of the screen, the user can move using the controls, and shoot down the enemies. Killing the enemies will provide the user with score points and coins, which can be used in the Store. The idea is to buy all Characters and Weapons. The Enemies will get stronger after a certain score is reached, making the game more challenging and encouraging users to invest more in the game.

3.3. Non Functional Requirements

- Usability : The user does not require any expertise to play this game, it has a very simple gameplay and design.
- Performance : The game frames will be limited to 30 FPS, because we are setting up 2D Rpg type set up, this would make the game run smoothly even in 30 FPS as well.
- Deployment: The game will be deployed in an executable file, hence making it easily executable even if the system does not have the libraries being used in the game installed in their.
- Safety : The game will automatically right all user data for their purchases in the text file, hence the user does not need to save every time.
- System Constraints: The game can work on systems with low memory and capacities, the RAM can go as low as 256 MB.
- Portability: Since the game comes in an executable file, it is very portable and can be transferred using any data traveller.
- Stability: The game has a very stable design, due to being developed while using the Object Oriented Paradigm Design.
- Security : The user data saved will be encrypted and not tangible by the user, hence allowing minimum level security breaching.
- Scalability: The game's features such as Modes, Characters, and Weapons can be extended for more features.

3.4. System Models

3.4.1. Scenarios

- Creating a new Profile
    - Start the application
    - Go to Create New Profile
    - Enter a unique username
- Creating a profile with already defined username
    - Start the application
    - Go to Create New Profile
    - Enter already existing username
    - Change username to make it unique
- Playing the Game
    - Start the application
    - Load/Create Profile
    - PlayGame
    - Move Character
    - Shoot Enemies
    - Get killed by enemy/bullets
- Loading a Profile
    - Start the application
    - Go to Load Profile
    - Choose a profile from the list
- Increasing/Decreasing the Sound Effects/Audio Volume
    - Start the application
    - Go to Settings
    - Set the Volume
- Buying a character/weapon
    - Start the application

    - Load/Create new profile
    - Go to Store
    - Buy the character/weapon (if enough coins)

- Selecting a character/weapon
    - Start the application
    - Load/Create new profile
    - Go to store
    - Select from ‘Owned’ character/weapons

- Select Mode
    - Open the application
    - Load/Create Profile
    - Go To Select Mode
    - Select Mode

- Play Game
    - Open the application
    - Load/Create Profile
    - Go to Start Game

- Pause Game
    - Open the Application
    - Load/Create Profile
    - Go to Start Game
    - Press Pause

- Change Volume while Play Game
    - Open the Application
    - Load/Create Profile
    - Go to Start Game
    - Press Pause
    - Set Volumes in the Pause Menu

- Exit Game
    - Press cancel on the top right corner anytime.

- Unpause Game

- Start Application
- Load/Create Profile
- Go to Start Game
- Press Pause
- Press Pause again to Unpause

3.4.2. Use Case

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

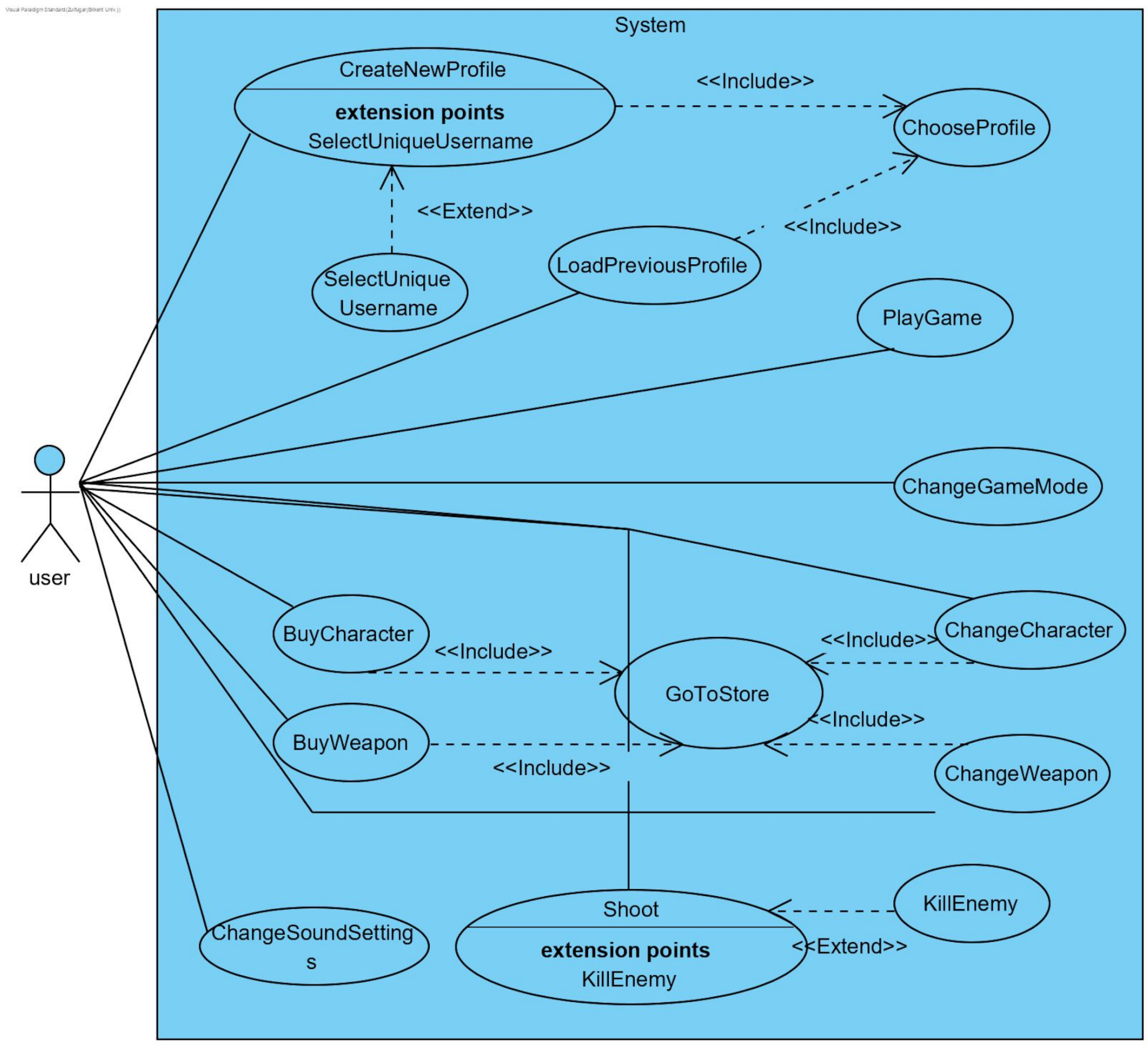


Figure 3: User Case Diagram of Defenders game

- 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
- SelectUniqueUsername:
    - 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.

- ChangeCharacter:

    - Participating Actor: User
    - Entry Conditions: In Store
    - Exit Condition: Change Character
    - Flow of Events:
        - User: Selects on the character he wants
        - System: If User has purchased character before, character selected, if not return alert user.
    - Special Requirement: Selected character must be purchased before.

ChangeWeapon:

- Participating Actor: User
- Entry Conditions: In Store
- Exit Condition: Change Weapon
- Flow of Events:
    - User: Selects weapon he wants
    - System: If User has purchased weapon before, character selected, if not return alert user.
- Special Requirement: Selected weapon must be purchased before.

- changeSoundSettings:

    - 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

- Shoot:

    - Participating Actor: User
    - Entry Conditions: Playing Game
    - Exit Conditions: Release the Shooting Button
    - Flow of Events:
        - System: Creates Bullet Object

    - User: Press Button that was binded to shoot
  - Special requirements: None

- Kill

  - Participating Actor: User
  - Entry Conditions: Shoot
  - Exit Conditions: None
  - Flow of Events:
    - System: Removes object that were killed
  - Special Requirements: None

3.4.3. Object Class Model

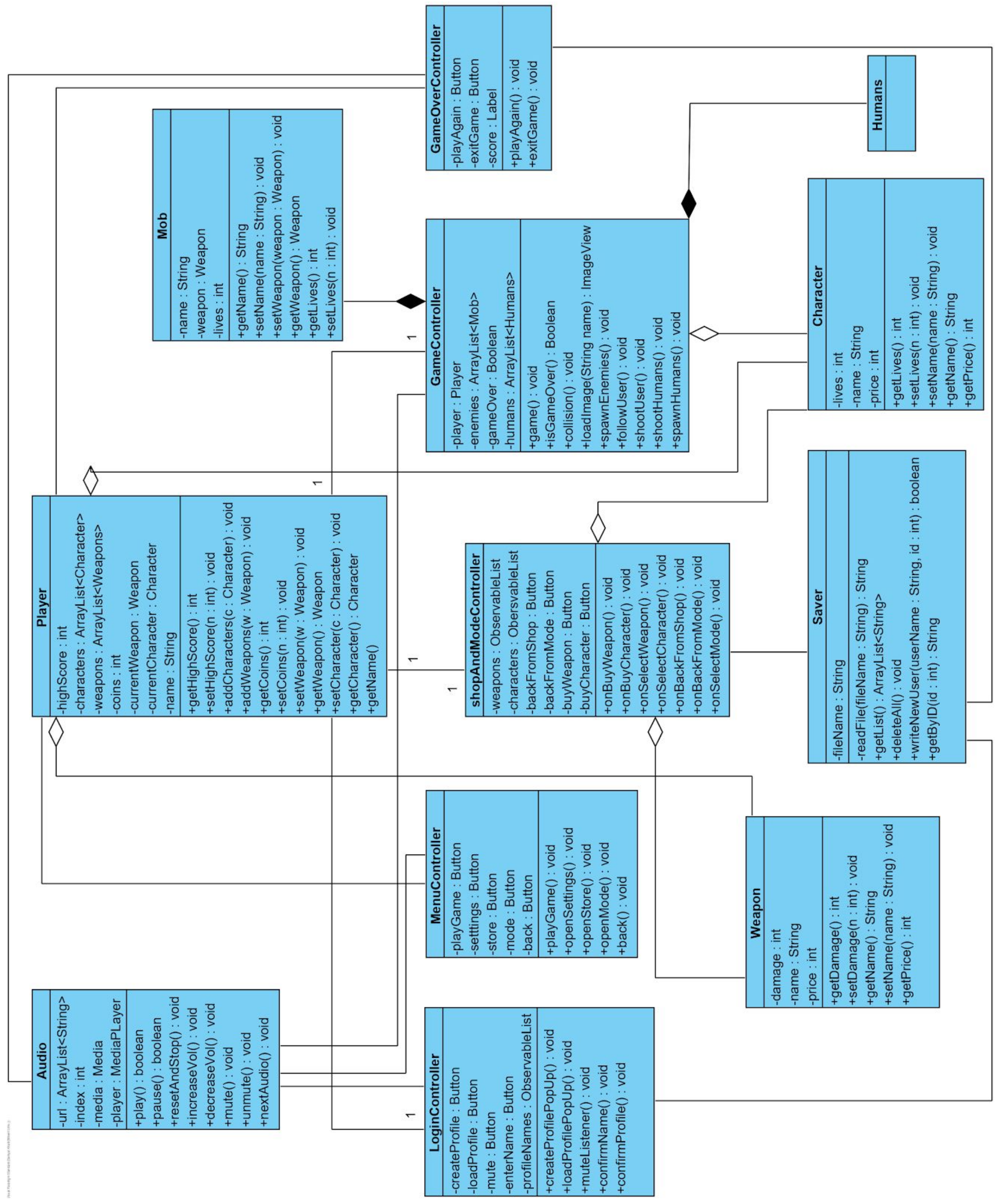
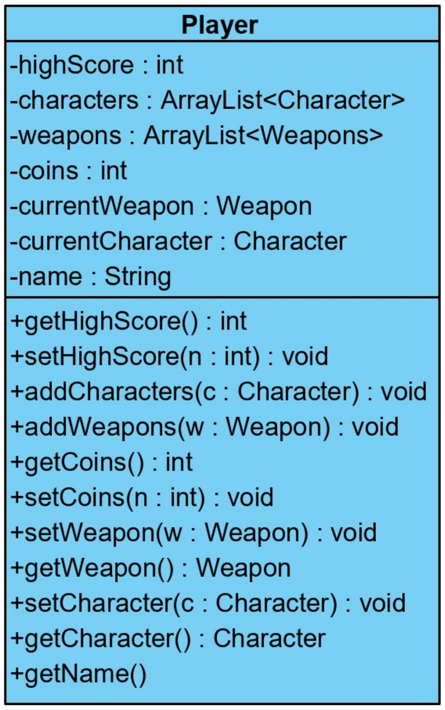


Figure 4: Class diagram for Defenders game

1. Player Class:


Player
-highScore : int
-characters : ArrayList<Character>
-weapons : ArrayList<Weapons>
-coins : int
-currentWeapon : Weapon
-currentCharacter : Character
-name : String
+getHighScore() : int
+setHighScore(n : int) : void
+addCharacters(c : Character) : void
+addWeapons(w : Weapon) : void
+getCoins() : int
+setCoins(n : int) : void
+setWeapon(w : Weapon) : void
+getWeapon() : Weapon
+setCharacter(c : Character) : void
+getCharacter() : Character
+getName()

- Attributes:

  i. int highScore: The highest score this user has ever achieved

  ii. Arraylist<character> characters: the list of characters the user owns.

  iii. Arraylist<weapon> weapons: The list of weapons the user owns

  iv. int coins: The current number of coins that the player has

  v. weapon currentWeapon: The weapon user is using currently from the weapon store

  vi. character currentCharacter: The current character the user is using. (for e.g Thor)

  vii. String name: The unique username of the player

- Operations:

  i. getHighScore() : int :- Returns the highScore attribute as int.

ii. setHighScore(n: int) : void :- Sets parameter as highscore

iii. addCharacters (c : Characters) : void :- This is used to add characters in the arraylist of the owned characters that the user has.

iv. addWeapons (w : Weapons) : void :- This is used to add weapons in the arraylist of the owned weapons that the user has.

v. getCoins(): int :-Returns the coins of the player

vi. setCoins(n: int): void:- sets coins as parameter

vii. setWeapon(w: Weapon): void:- sets Weapon as default Weapon

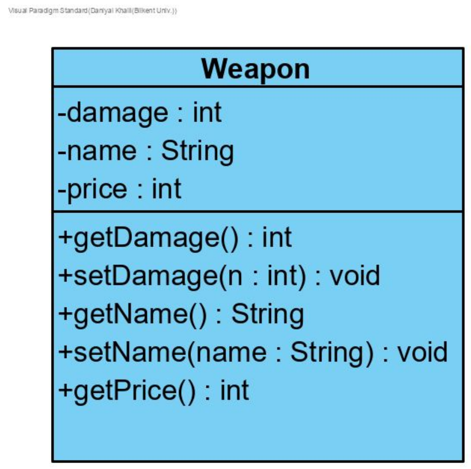
viii. getWeapon():Weapon :- returns default weapon

ix. setCharacter(c: Character): void :- sets character for player

x. getCharacter():Character :- returns the default character

xi. getName() void :-returns the name of the player

2. Weapon.


Visual Paradigm Standard(Daniyal Khalil(Bilkent Univ.))
Weapon
-damage : int
-name : String
-price : int
+getDamage() : int
+setDamage(n : int) : void
+getName() : String
+setName(name : String) : void
+getPrice() : int

- Attributes:

i. int damage: The amount of damage the weapon can do.

ii. string name: The unique name of the weapon.

iii. int price: The price it is in the store for.

- Operations:

    i. getDamage(): int :- returns the damage attribute

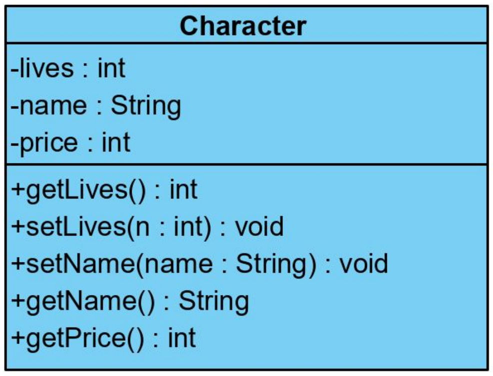
    ii. setDamage(n: int):void :- sets the damage attribute

    iii. getName(): String :- returns the name of the weapon

    iv. setName(name: String): void:- sets the name of the weapon

    v. getPrice(): int :- returns price of the weapon in the store

3. Character



- Attributes:

    i. int lives: The number of lives the character has.

    ii. string name: The unique name of the character.

    iii. int price: The price it is in the store for.

- Operations:

    i. getLives():int:-returns live of the character

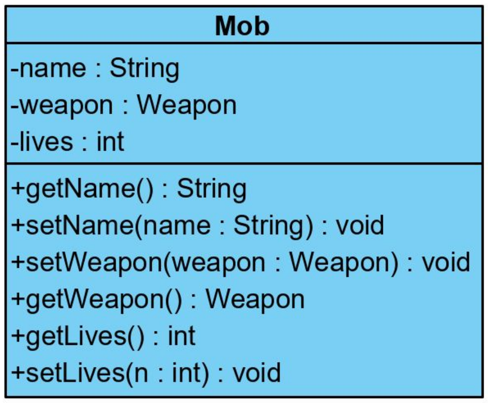
    ii. setLives(n: int):void:- sets the live to the character

    iii. setName(name: String):void:- sets name to the character

    iv. getName():String:- returns the name of the character

    v. getPrice():int :- returns the price of the character

4. Mob (Computer Controlled NPCs)

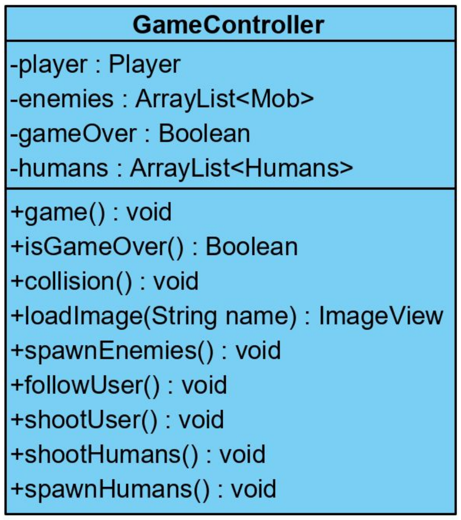


- Attributes:
    i. string name: The unique name of the enemy.
    ii. Weapon weapon: The only weapon an enemy can use.
    iii. int lives: The number of lives the character has.
- Operations:
    i. getName(): String:- returns the name of the mob
    ii. setName(name: String):void :- set the name to the mob
    iii. setWeapon(weapon: Weapon): void:- sets the weapon to the Mob
    iv. getWeapon():Weapon:- returns the Weapon of the mob
    v. getLives(): int :- returns the lives of the mob
    vi. setLives(n: int):void:- sets the live o the mob

5. Humans

This is a class that simple inherits the imageview class from javafx library and uses its functions, since humans only stay stationary and get killed.

6. GameController



GameController

-player : Player
-enemies : ArrayList<Mob>
-gameOver : Boolean
-humans : ArrayList<Humans>

+game() : void
+isGameOver() : Boolean
+collision() : void
+loadImage(String name) : ImageView
+spawnEnemies() : void
+followUser() : void
+shootUser() : void
+shootHumans() : void
+spawnHumans() : void

- Attributes

  i. Player currentPlayer: The current user that is logged in the game.

  ii. ArrayList<Mob> enemies:

  iii. Boolean gameOver:

  iv. ArrayList<Humans> humans:

- Operations:

  i. game(): void:- Makes the game frame and then sets all the components of the game in the frame and starts the timer function of javafx.

  ii. isGameOver(): Boolean :- Returns whether the game is over or not by checking the number of user's lives.

  iii. collision(): void:- Checks for the collision between the user and the enemies and their bullets and decreases lives accordingly or kills enemies

  iv. loadImage(String name): ImageView:- An auxiliary method to load the animation for the given string from the root folder

  v. spawnEnemies():void:- Creates new enemies in the screen, but they are Always out of bounds.

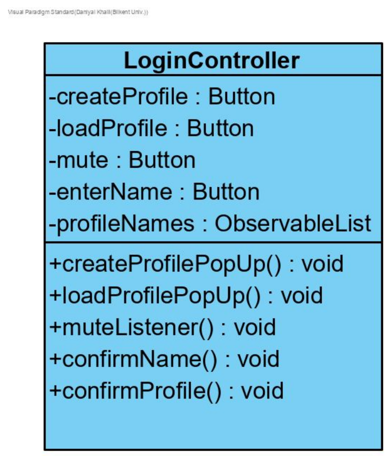
vi. followUser():void:- This makes the enemies follow the user location.

vii. shootUser():void:- This makes the enemies shoot their projectiles. The Chances of their bullets being in the same direction as the user is random

viii. shootHumans():void:- The probability of when an enemy will kill a human,
if a human is killed, the score is reduced.

ix. spawnHumans():void:- This is used to make new humans after the Previous ones are killed by the enemies.

7. LoginController



- Attributes:

  i. Button createProfile: Button that is used to create new profile

  ii. Button loadProfile: Button that is used to go to list of previous profiles

  iii. Button mute: Button that is used to mute the background music

  iv. Button enterName: Button that creates to profile after name is entered

  v. ObservableList profileNames: The list that will show the list view of all the save profiles

- Operations:

i. createProfilePopUp():void:- opens pop up to create new profile

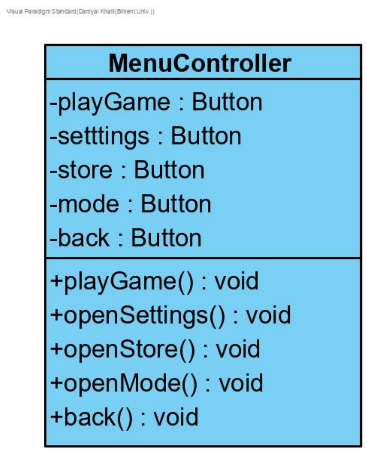
ii. loadProfilePopUp():void:- opens pop up to load one of previous profiles

iii. muteListener():void:- mutes the background music

iv. confirmName():void:- The Button for entering the name typed by the user

v. confirmProfile():void:- Loads the profile after selection

8. MenuController



- Attributes:

i. Button playGame: Button that starts new game

ii. Button settings: Button that is used to open settings

iii. Button store: Button that is used to go to store

iv. Button mode: Button that goes to mode screen

v. Button back: the button that is used to go to login screen

- Operations:

i. playGame():void:- starts to play new game

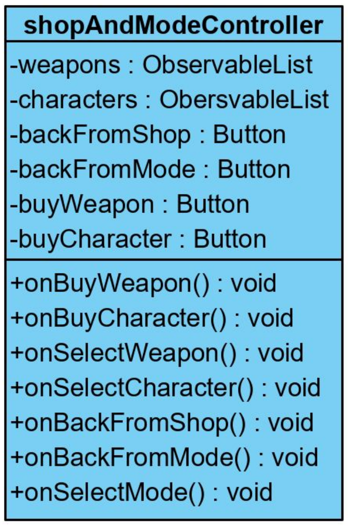
ii. openSettings():void:- opens settings screen

iii. openStore():void:- opens store screen

iv. openMode():void:- opens mode screen

v. back():void:- goes back to login screen

9. ShopandModeController:



- Attributes:

i. ObservableList weapons: list of weapons to use in game

ii. ObservableList characters: list of characters that user choose as spirit

iii. Button backFromShop: the button used to go back to game menu from shop

iv. Button backFromMode: the button used to go back to game menu from mode

v. Button buyWeapon: the button adds the chosen button to belongings

vi. Button buyCharacter: the button adds the chosen character to belongings

- Operations:

i. onBuyWeapon():void:- adds the weapon to the belongings

ii. onBuyCharacter():void:- adds the character to the belongings

iii. onSelectWeapon():void:- specifies the weapon that used in the game

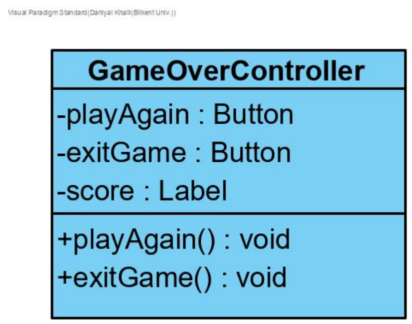
iv. onSelectCharecter():void:- specifies the character that used in the game

v. onBackFromShop():void:- passes the user back to game menu from shop

vi. onBackFromMode():void:- passes the user back to game menu from mode

vii. onSelectMode():void:- listener for when the radio button for the mode is selected

10. GameOverController



- Attributes:

i. Button playAgain: Button on gameOver pop up that is used to start the game again

ii. Button exitGame: Button on gameOver pop up that is used to exit the game

iii. Label score: label that shows the score

- Operations:

i. playAgain():void:- method that starts the game again

ii. exitGame():void:- method that finishes and exits the game

3.4.4. Dynamic Models

3.4.4.1. Activity Diagram

In the Activity Diagram it is displayed how player is interacting with the UI of the game is started. User can either **Create Profile** or **Load Profile** previously created ones. If user chose to **Create Profile** s/he is asked to provide unique username. After successful login to profile user is headed to main menu screen, where s/he is provided one of the options to choose from:

- Settings
- Mode
- Store
- PlayGame

In case of selection **Settings** menu, user is able to change the level of music and sound effects of the game. If **Mode** menu option was selected user is given opportunity to choose from two available game modes (Marvel and DC). Selecting **Store** option will result in user being transferred to in game market in order to either to choose from available special weapons and characters or to purchase new ones for in game currency. Finally, if **Play Game** was selected user is able to play Defenders: Cinematic Edition. (Figure 5)

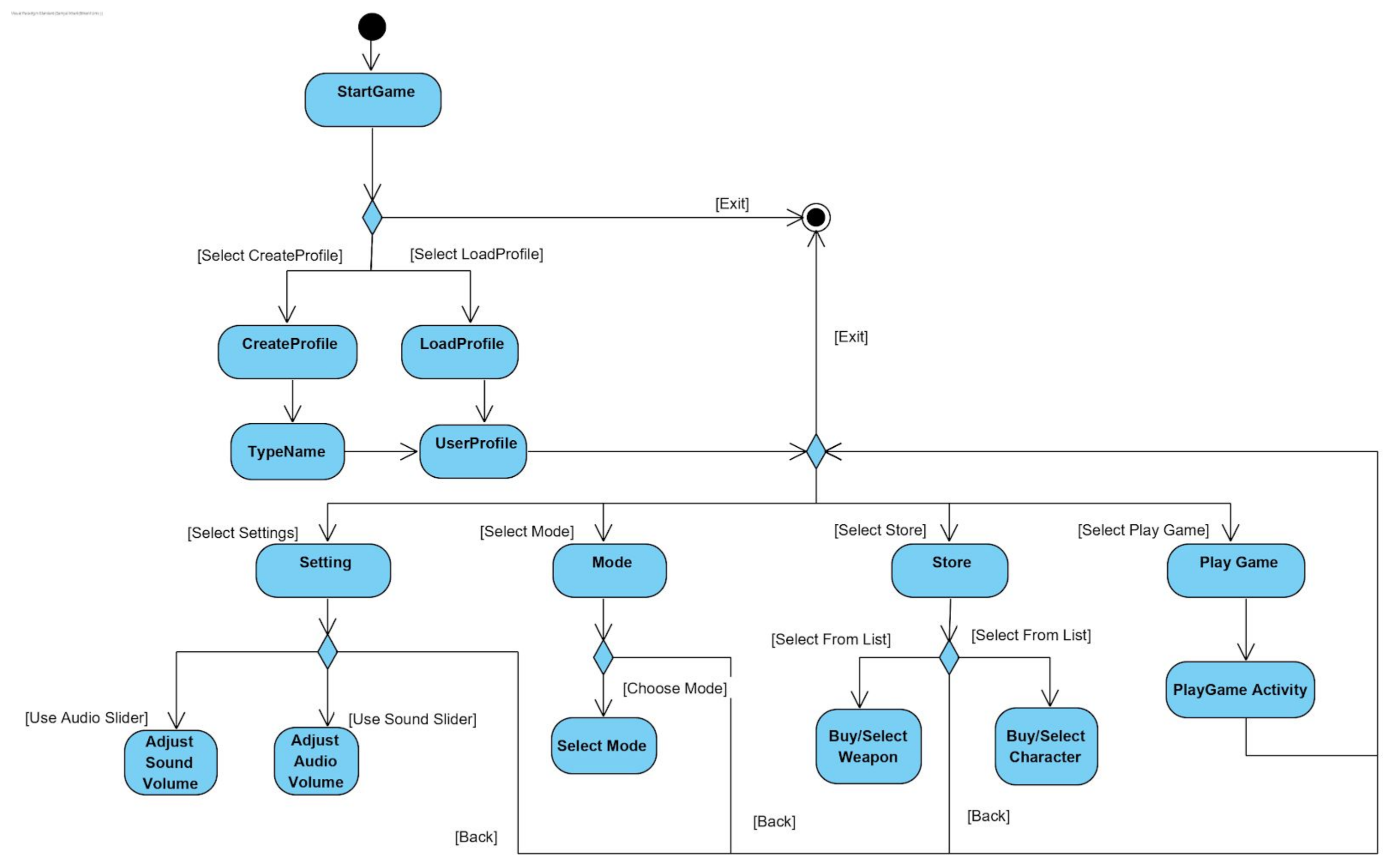


Figure 5: Activity Diagram of Defenders game

Once player started to play the game his main goal is to kill enemies by moving, aiming and shooting them. This can be done using keyboard input to go in either of four directions or to shoot. Gameplay can be paused using same input type. After the game was paused user will either exit the game or continue playing by pressing buttons with appropriate names. If game was continued after being paused cycle is continued.(Figure 6)

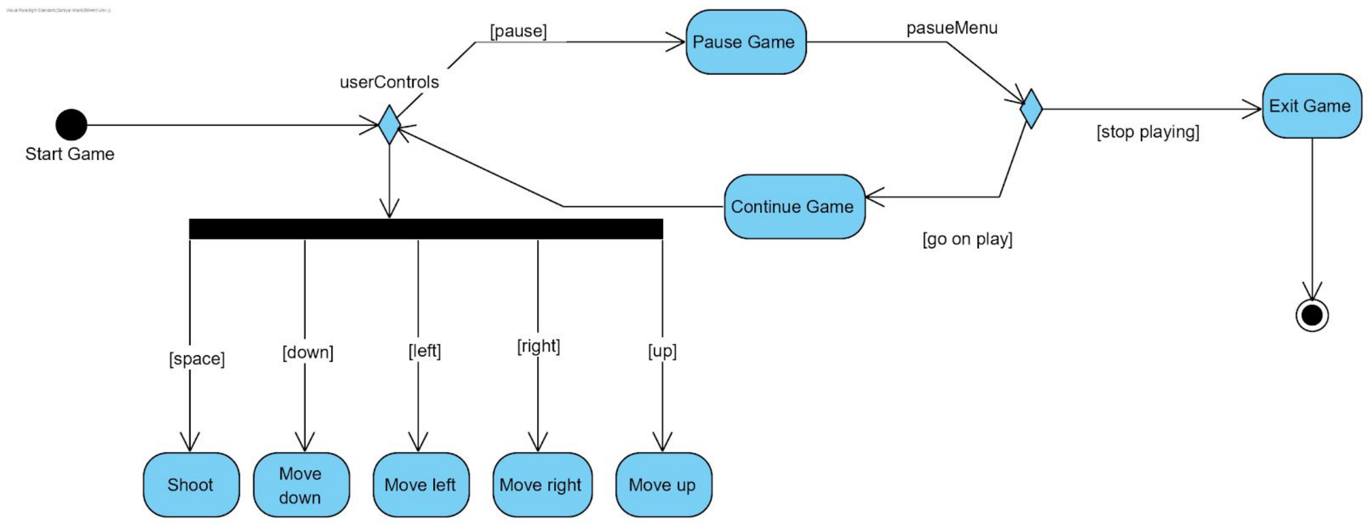
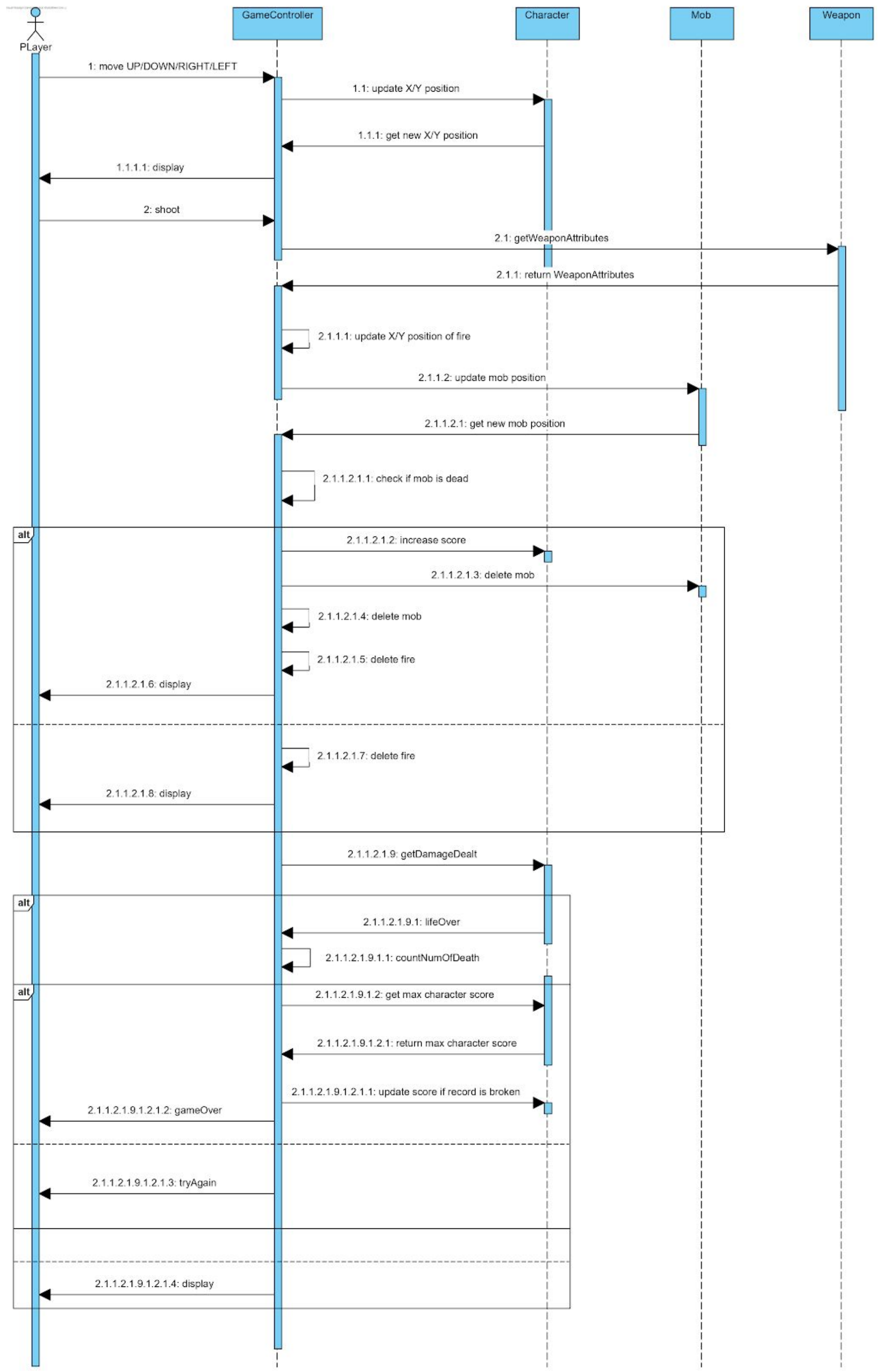


Figure 6: GamePlay Activity

3.4.4.2. Sequence Diagram

This is the sequence diagram depicting game-play. First user chooses to go to one of four directions. Then GameController class will update the position of the character, and will increase or decrease its current x/y positions. When player shoots, Weapon class will give the GameController class the damage level of the currently used weapon, and the player will subsequently damage the mob in a given level. GameController class also checks for the damage given by the enemy, and thus if your life level has decreased to zero or below after the damage, you immediately die. If your death count is 3 then the game is over. Otherwise you get another chance.


PLayer
GameController
Character
Mob
Weapon
1: move UP/DOWN/RIGHT/LEFT
1.1: update X/Y position
1.1.1: get new X/Y position
1.1.1.1: display
2: shoot
2.1: getWeaponAttributes
2.1.1: return WeaponAttributes
2.1.1.1: update X/Y position of fire
2.1.1.2: update mob position
2.1.1.2.1: get new mob position
2.1.1.2.1.1: check if mob is dead
alt
2.1.1.2.1.2: increase score
2.1.1.2.1.3: delete mob
2.1.1.2.1.4: delete mob
2.1.1.2.1.5: delete fire
2.1.1.2.1.6: display
2.1.1.2.1.7: delete fire
2.1.1.2.1.8: display
2.1.1.2.1.9: getDamageDealt
alt
2.1.1.2.1.9.1: lifeOver
2.1.1.2.1.9.1.1: countNumOfDeath
alt
2.1.1.2.1.9.1.2: get max character score
2.1.1.2.1.9.1.2.1: return max character score
2.1.1.2.1.9.1.2.1.1: update score if record is broken
2.1.1.2.1.9.1.2.1.2: gameOver
2.1.1.2.1.9.1.2.1.3: tryAgain
2.1.1.2.1.9.1.2.1.4: display
Figure 6: Sequence diagram for playing a game

This diagram explains user-system interaction regarding its way to the gameplay. Firstly, user can create a new profile. Then he/she will have to type username 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 LoginController sends you a message stating that the name is taken, so you can choose another unique name. Afterwards, user continues with the game.

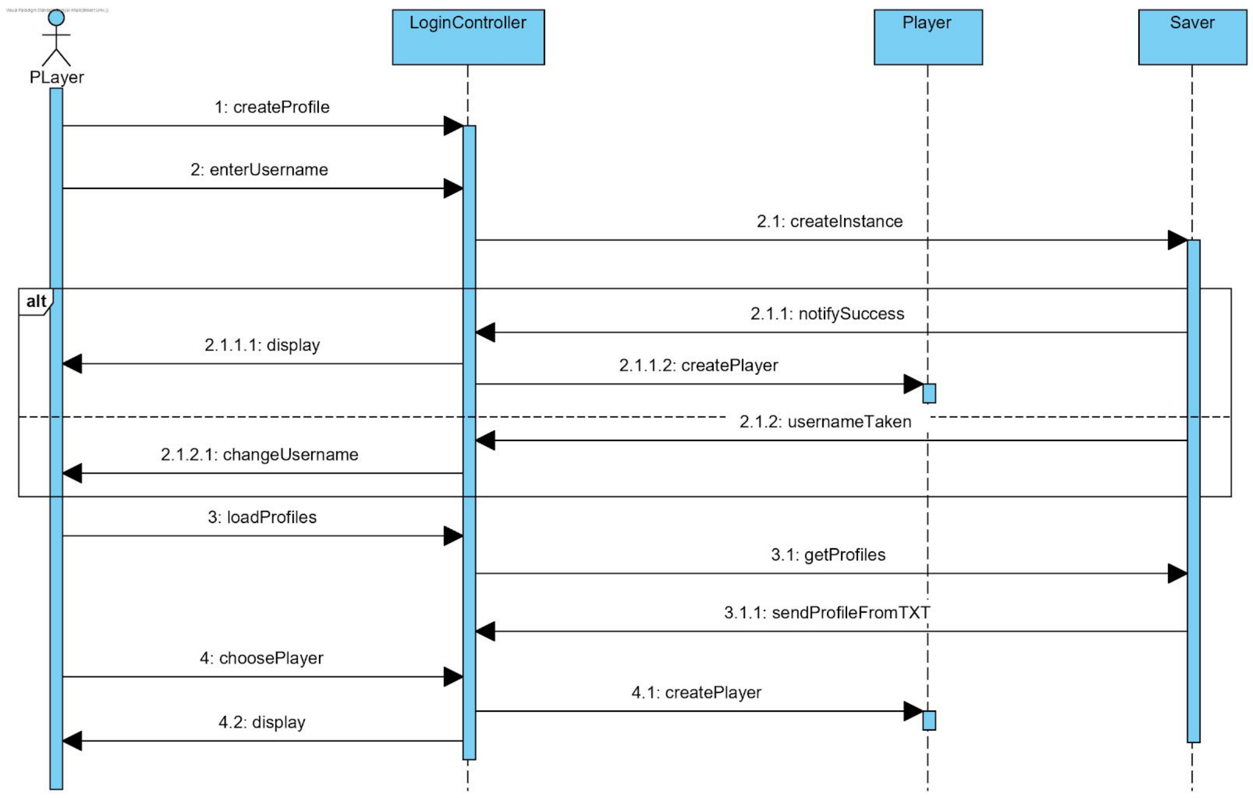


Figure 7: Sequence diagram for loading the user profile

3.4.4.3. State Diagram

This state diagram explains user interface interaction. Firstly, the user is profile menu state. The user has to decide whether he/she wants to create a new profile or to load previous profile. After deciding profile, user is in game menu state. From that point, the user can go back to profile menu by using the back button. Moreover, the user can go to mode state to choose mode, to shopping at store state to buy or choose weapon or to sound settings to increase or decrease the level of sound effects and background music in the game. The user is able to come back to the game menu state by using the back button. Entering to gameplay is supplied by game menu state. In playing game state, the user is able to go pause state by using the pause button. Then on pause state, user decides whether to continue game and go back to playing game state or end the game.

Ending game sends the user to end of game state where user can choose new game and go to game menu state or exit application.

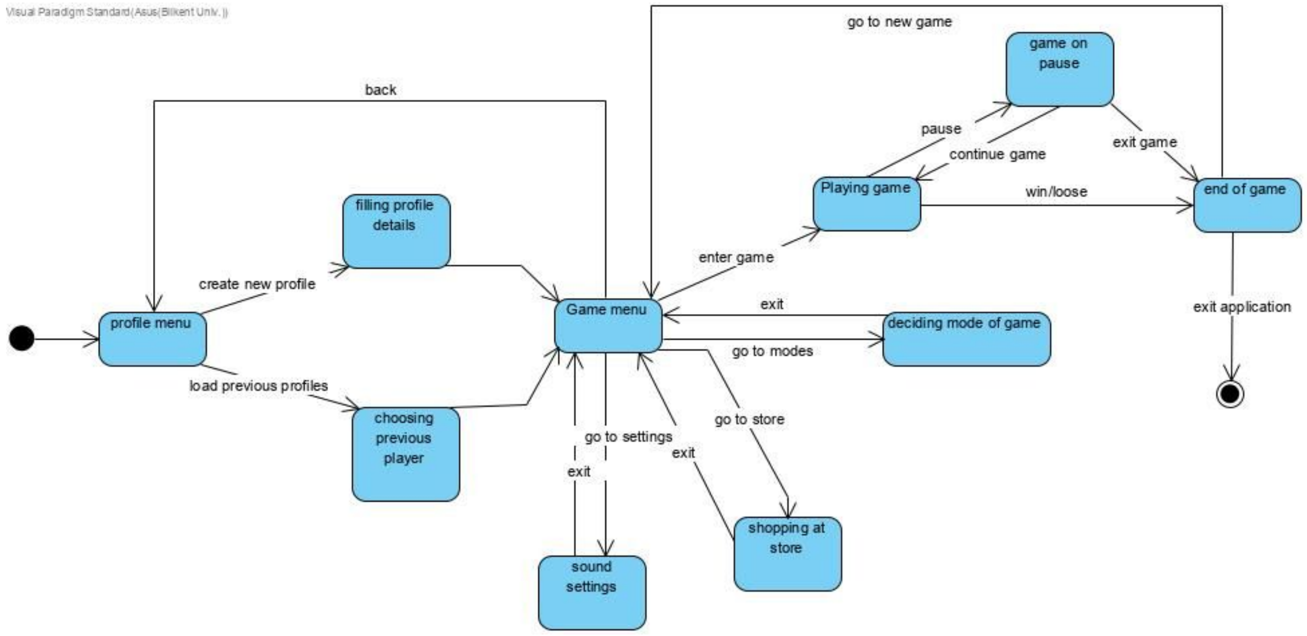


Figure 8: State diagram of Defenders game UI

This state diagram displays the gameplay states of the spirit. The spirit can stand or either moving in the gameplay. It can pass to moving state from standing state with up, down, right and left buttons. After any collision, spirit is dead. Then if there are any lives left, the game starts again. The spirit has 3 lives in every play. When there are no lives left, game overs and spirit comes to final state.

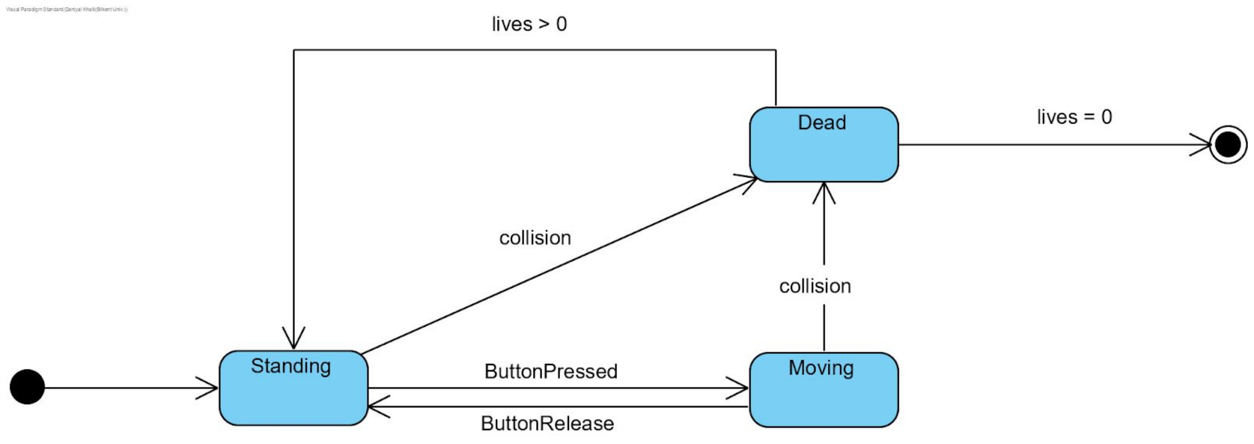


Figure 8: State diagram of Game Play

3.4.5. User Interface

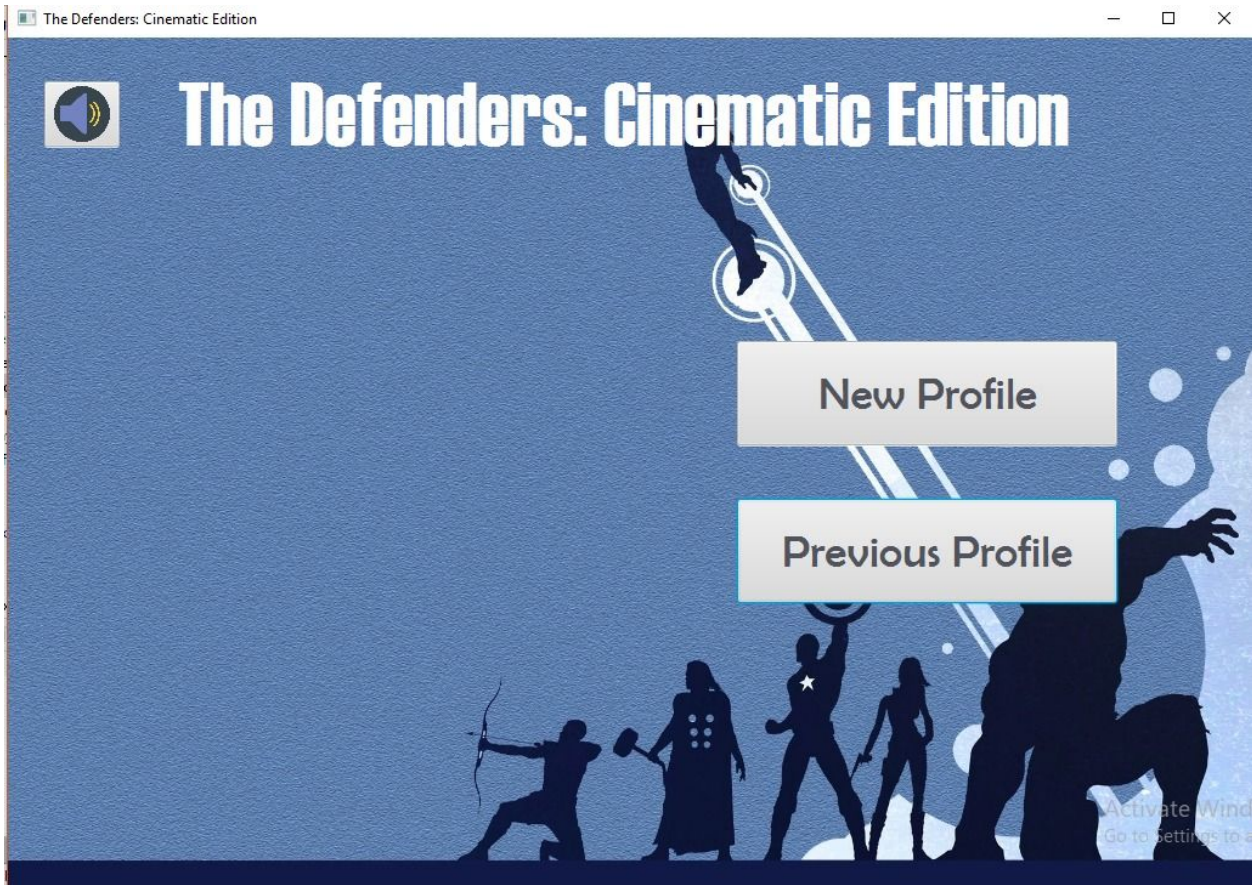

The Defenders: Cinematic Edition
The Defenders: Cinematic Edition
New Profile
Previous Profile

Figure 9: Opening screen of Defenders game

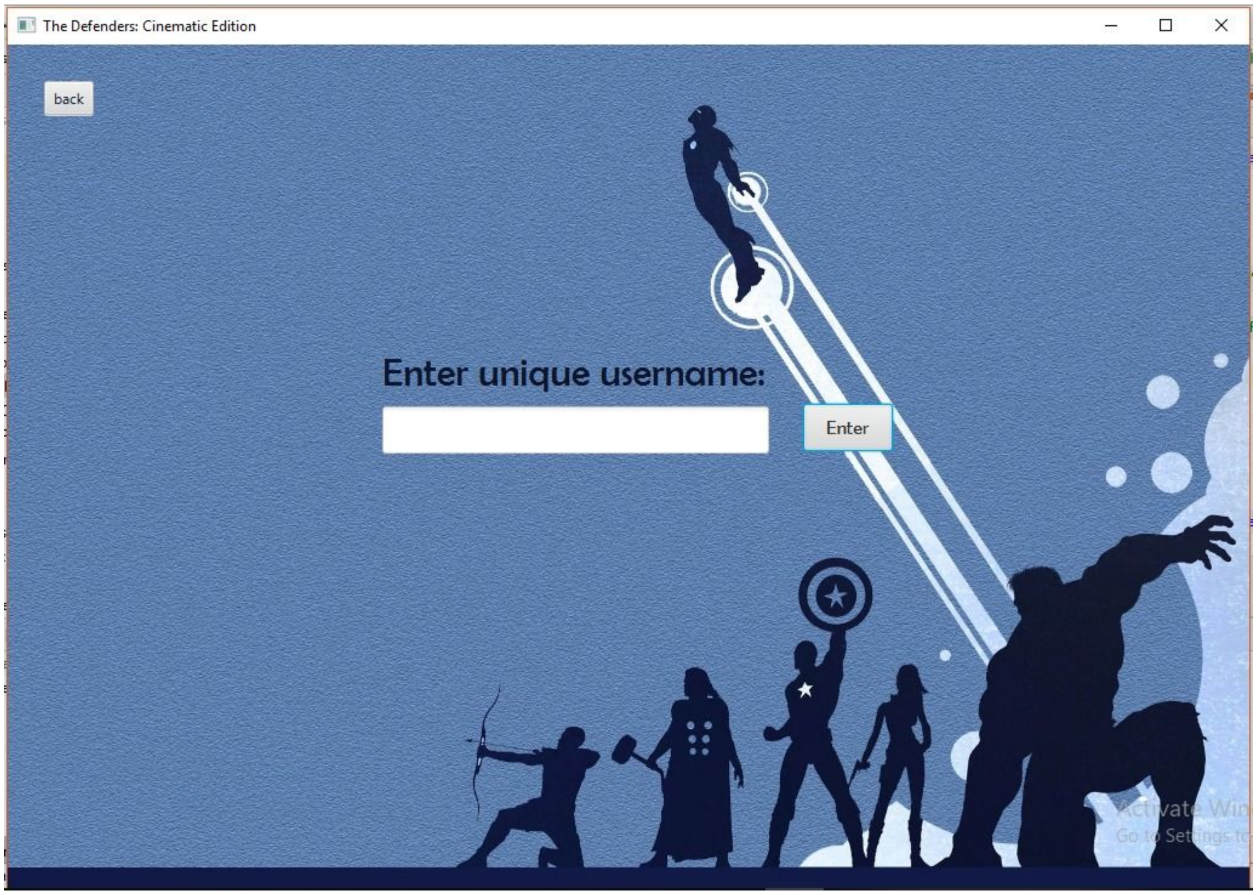


Figure 10: Creating new profile

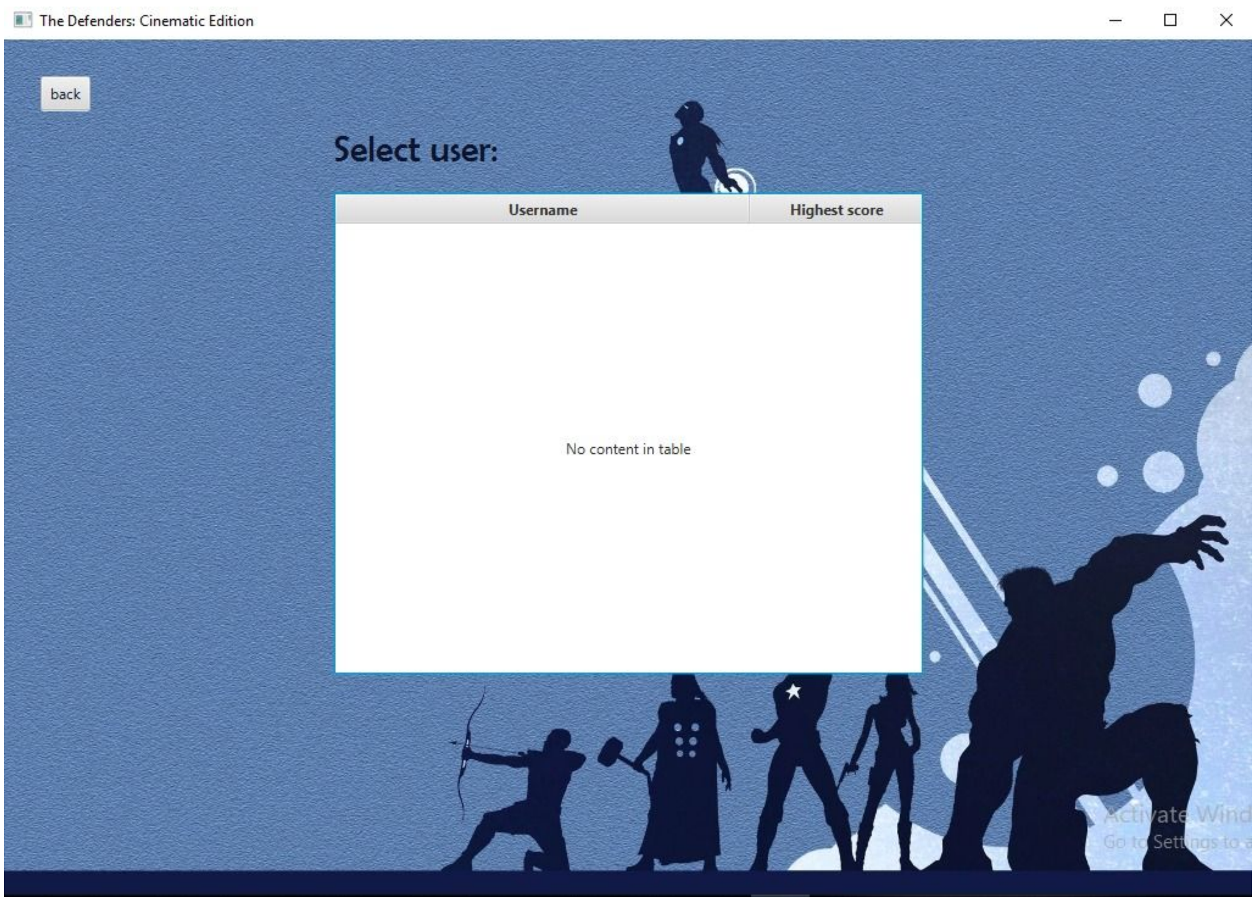


Figure 11: Choosing previous profile

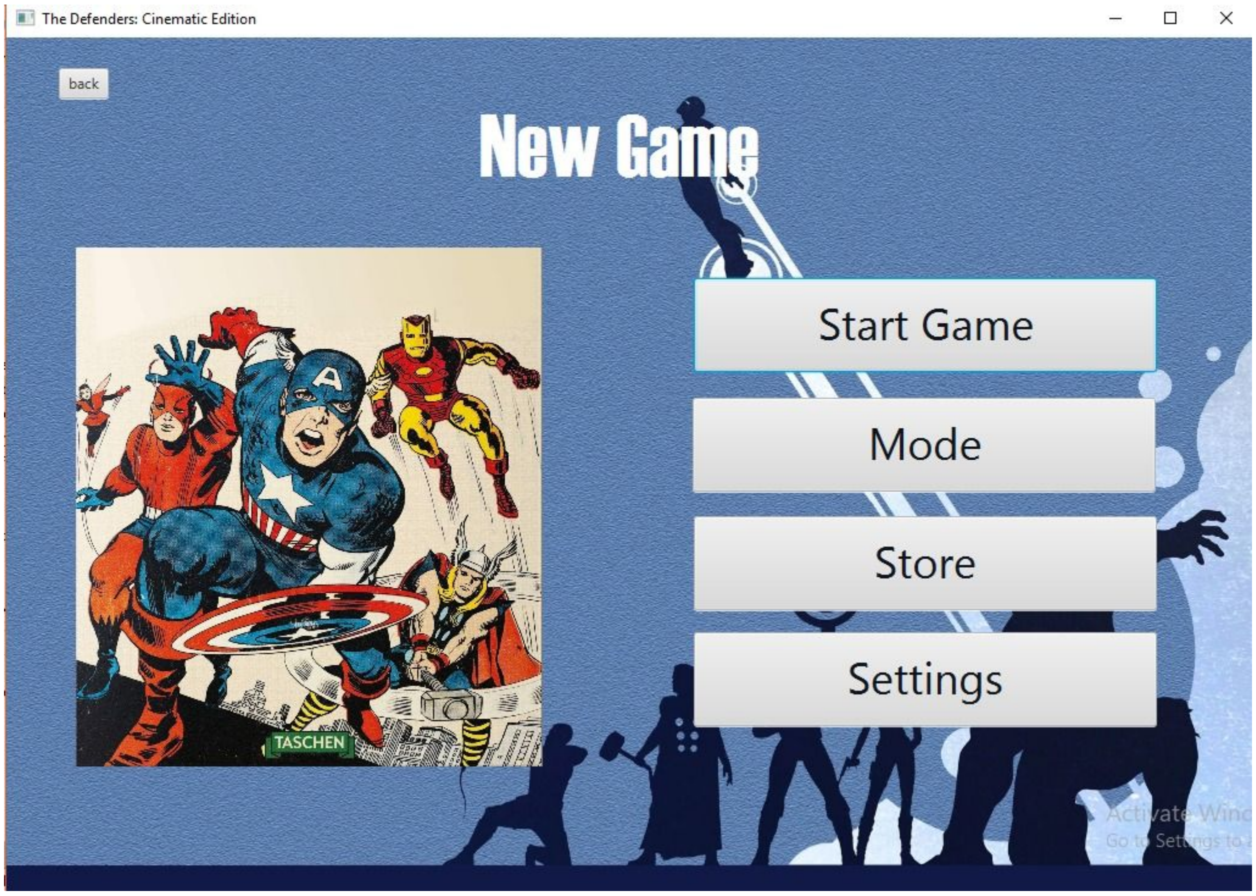


Figure 12: new game page

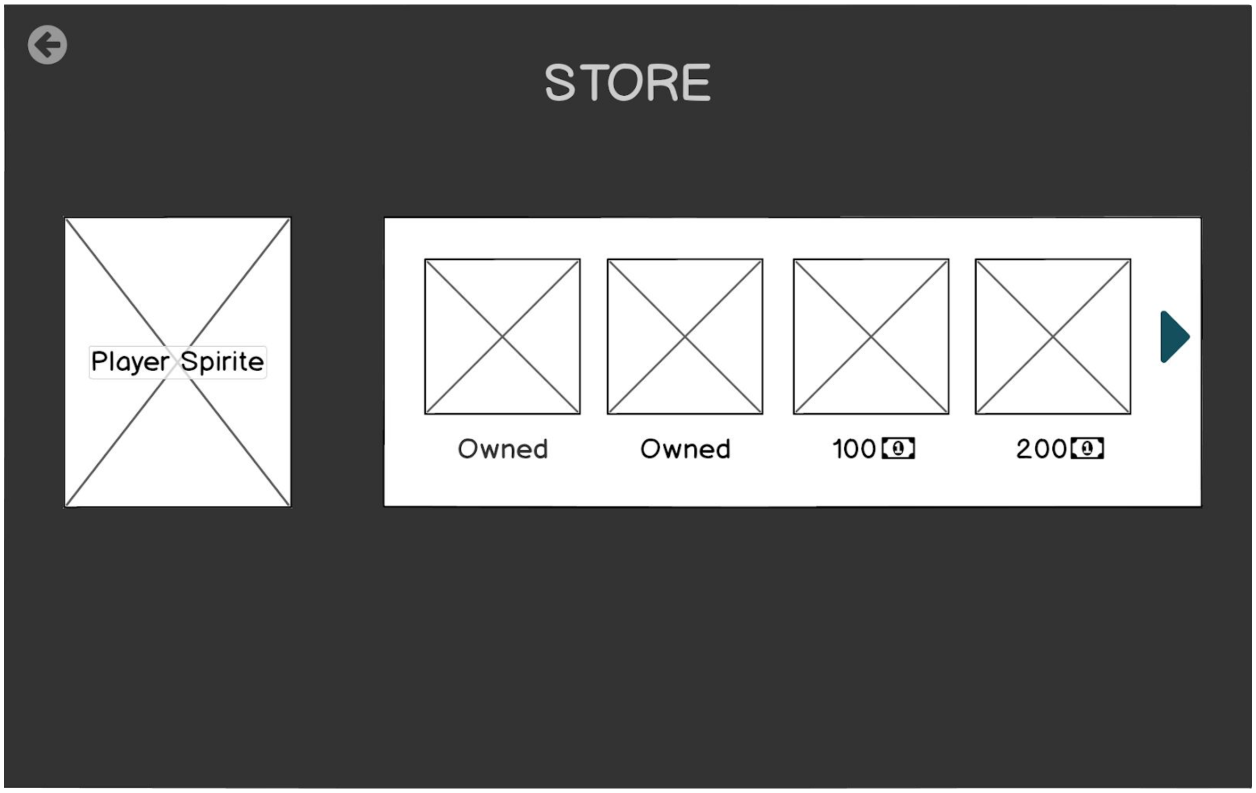


Figure 13: Shopping at the store

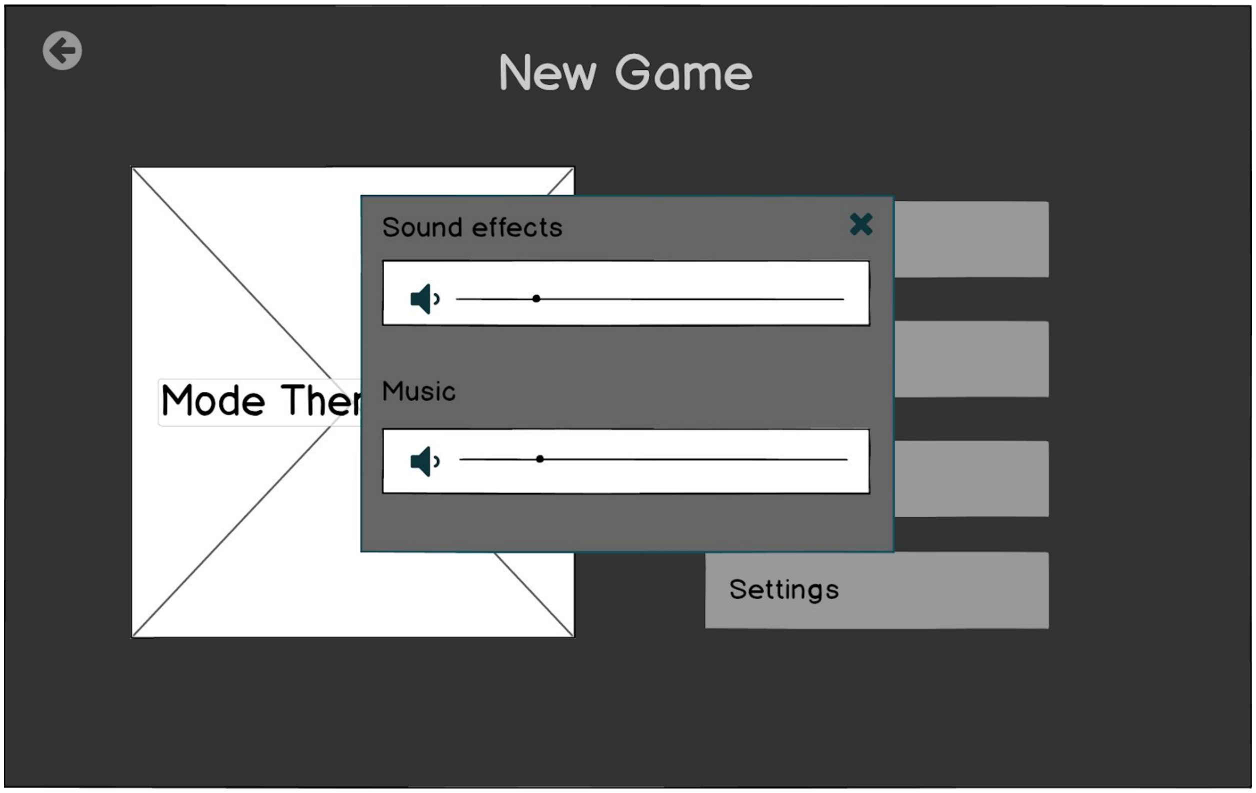


Figure 14: Sound settings

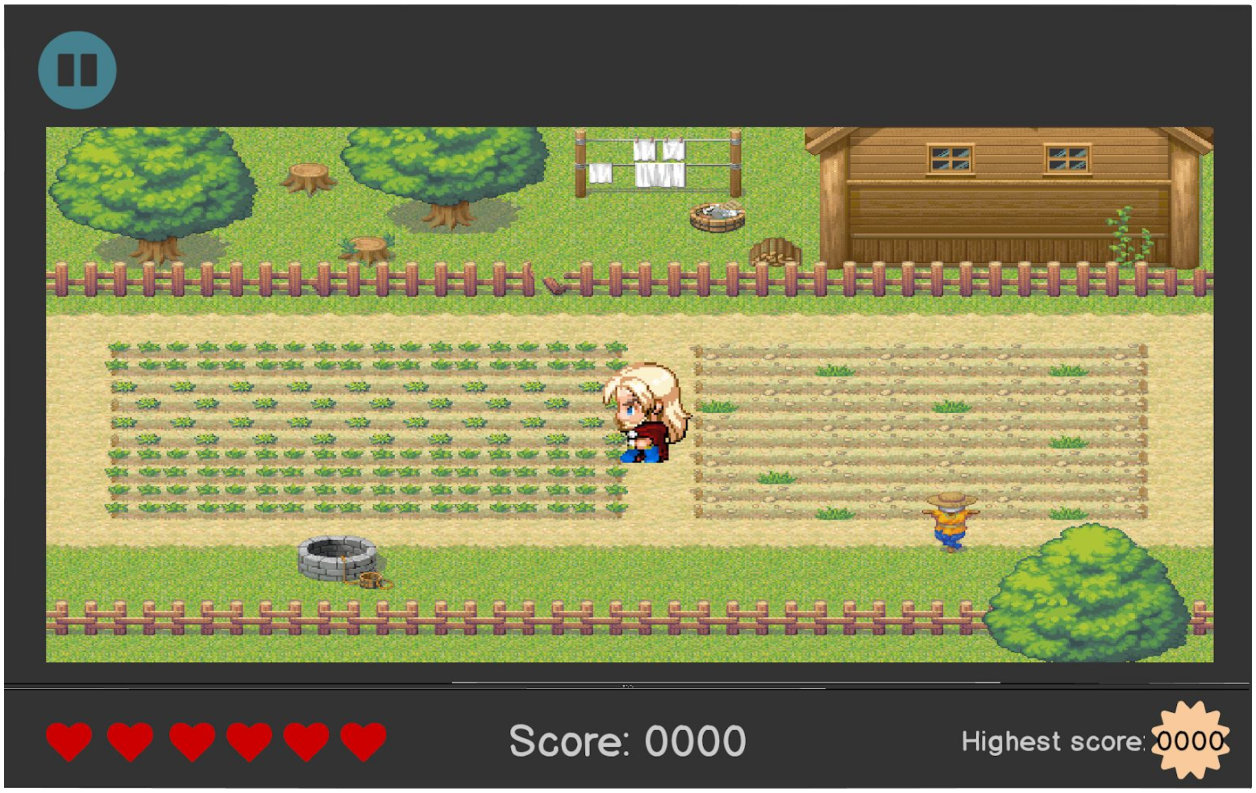


Figure 15: Game screen

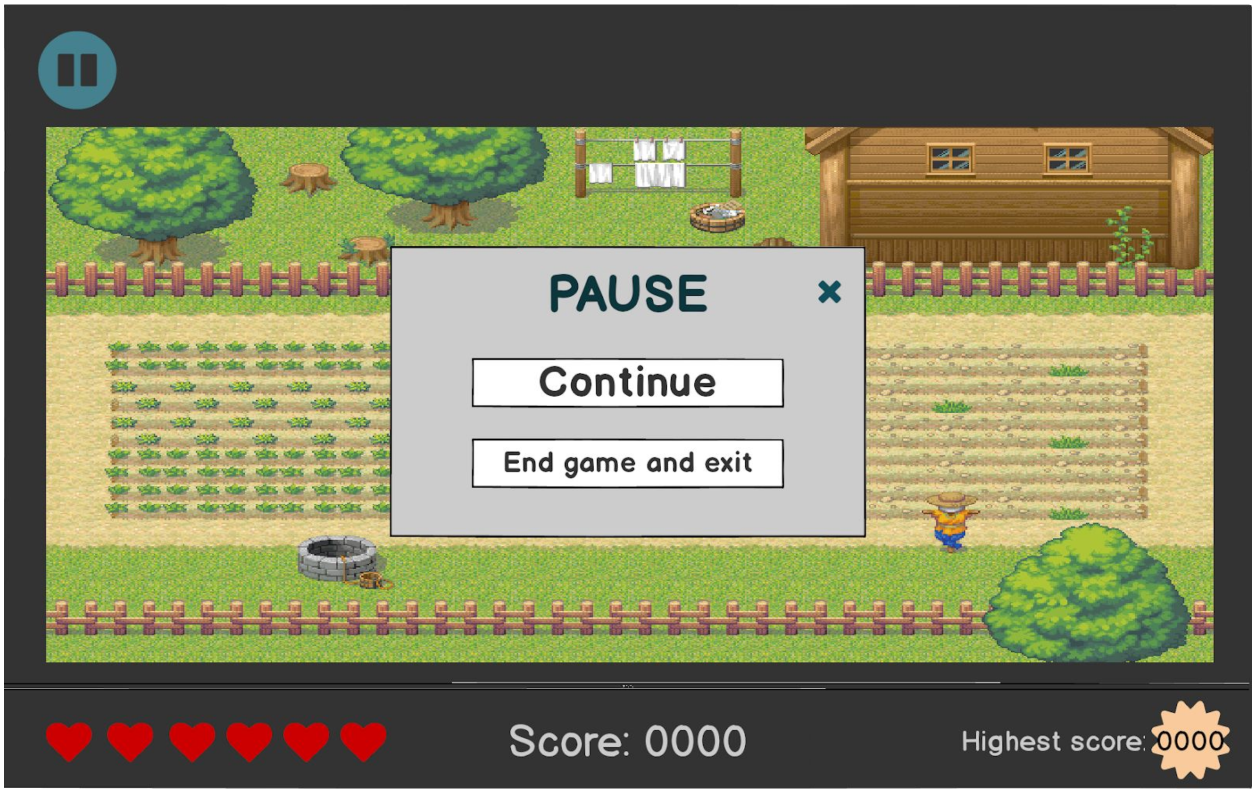


Figure 16: Pause on game

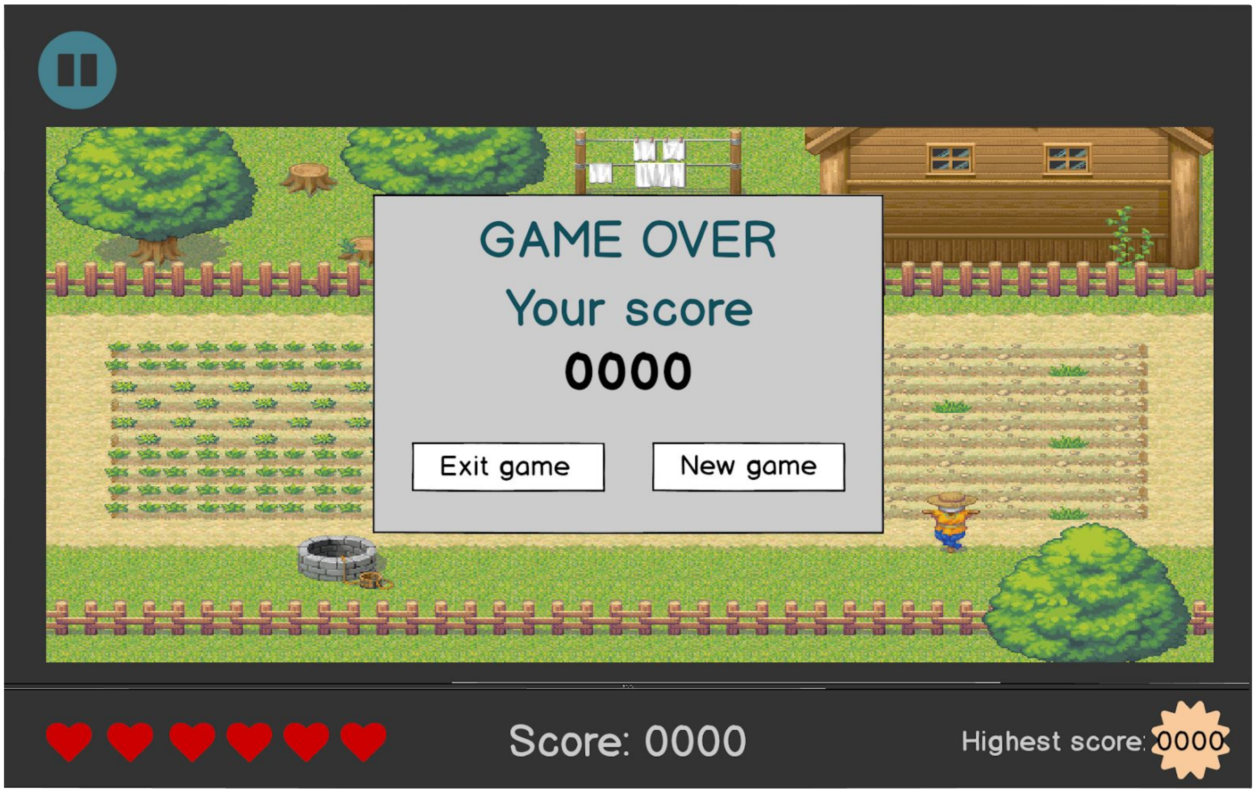


Figure 17: Game over screen

4. References

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

[2] http://www.free80sarcade.com/defender.php