CS 319 - Object-Oriented Software Engineering

Analysis Report

RUNNING JON

Group

ERKAN ÖNAL

NİHAT ATAY

BARIŞ ARDIÇ

MERT KARA

Table of Contents

Table of Figures

1.Introduction

Running Jon is a desktop application action game. Its logic is similar to the games like Space Intruders. However, its scenario is different than space intruders. Our game's scenario is based on the famous book and also TV series; Game of Thrones. We are planning to improve our games by increasing the quality of the graphics and game effects.

In Running Jon, there is a main character whose name is Jon Snow and the main purpose of the game is to save the Lady Sansa from the enemies. Jon Snow will have the capabilities of throwing blades to the enemies. He will be able to move left and right. He could improve his blade's strong by dropping different items from the enemies that he killed. He got totally three lives. On the enemy side, there are different types of solider like ordinary soldiers, white walkers and king's guardian. They also throw blades. When they hit you or if blades of them hit you, you lose one life point. They have different capabilities. Ordinary soldiers are weak compared to others. King's guardian got huge armor and good attack power. White walkers attacks are decreasing your speed along with your lives. If you kill them, you get extra power-ups like increasing your attack power, throwing multiple blades to different directions and increasing your speed to move left or right. The span of the bonuses will be definitely limited, they will be between 8-12 seconds.

When we come to the point of map, it will not be like a tile map or smilar to this. However; it will be a dynamic background to the gamepanel, that is to say, while game procedes, the background will also move. We will do it asa two picture one above another but they will appear as one picture. While timer procedes, we will start to get the image's x and y coordinates from the beginning of the picture and when it comes to the end, we

will set x and y coordinates to the start point of the picture again and we will do it as loop so we will have a movable map.

If we come to the sound point, we will add different sound for different actions. For example, when user start the game, there will be a menu sound and in game, there will be a different background sound. In collisions, there will be a hit sound and after taking bonuses, there will be cool sounds. We will also add specific sounds from the charachters of game of thrones. For example, in game, while facing with bosses, we will play these sounds.

The movements of the Jon Snow will be controlled by keyboard arrows and when user hits space button. Character will throw blades to enemies.

2. Requirement Analysis

2.1 Overview

The game is composed of three levels. In each level, the difficulty increases. After end of each level, there will be strong boss having fast moves and fast attacks. Main purpose here is that completing the game without losing all of your life points and saving Lady Sansa. To do that, user will be careful about running away from the blades of enemy and he should not collide with them. In addition, to increase the chance of saving Lady Sansa, user should collect the power-ups falling down from enemies because when game goes along, to take on strong enemies, character needs to be more powerful.

In the game, any time that user wants to pause, he can pause and he can continue later on.

In the game, there will be also high score side. The more enemies user kills the more points he will get and the more power-ups he collects the higher his point will be. Strong enemies' points will be higher than the weak ones. Bosses have the highest points.

The backgrounds of each level is different than other one and the sounds of different enemies and bosses are different.

2.1.1 List of Characters

Figure 1: Jon Snow

The main subject of the game.

Figure 2: Solider One

Weak Solider.

Figure 3: Solider Two

This solider’s attacks are stronger than the solider one.

Figure 4: Boss number 1

This is the boss of level 1, Cercei Lannister.

Figure 5: Boss number 2

This is the boss of level 2, Tywin Lannister.

Figure 6: Final Boss

This is the final Boss, Ramsey Bolton.

2.1.2 List of Weapons

Figure 7: Black Sword

This is the weak sword that user throws.

Figure 8: Blue Sword

This is the average strong sword that user throws.

Figure 9: Purple Sword

This is the strongest sword that user throws.

2.1.3 List of Bonuses

Figure 10: Speed Up

This is speed increase item that can drop from enemies.

Figure 11: Life

This is +1 extra life item that can drop from enemies.

Figure 12: x2

This is double attack item that can drop from enemies.

Figure 13: x3

This is triple attack item that can drop from enemies.

2.2 Functional Requirements

- User will be able to control Jon Snow by using keyboard arrows.
- User will able to throw blades to enemies to eliminate them.
- User will be able to access the help menu that consists of information about how to play the game, items and useful tips.
- The game shall have three different levels. The game is more challenging at higher levels.
- User can see the high scores of top 10 players
- The user shall be allowed to pause the game and then resume
- The player character can be buffed with power ups that drop from dead enemies.
- The program will end the game session if the player character is out of health points.
- Enemies will shoot the player character with projectiles when these projectiles collide with player character health points will be removed from the character.
- The game sound can be turned of or on by the user.

2.3 Non-functional Requirements

2.3.1 Usability

- The game should be user-friendly interface and easy to use
- The game should have Main Menu button to access Main Menu in each page
- The player should be able to access Options page from Pause Menu
- Concepts of the levels in the game are easy to understand and react to.
- The game shall offer numerous power-ups.
- The game music shall be atmospheric.
- A level shall be harder than previous levels.

2.3.2 Performance

- Control mechanism of the game shall have short response time that allows the player to play with minimal delay.
- Jon Snow object shall respond at most 1 millisecond.

2.3.3 Supportability

- The system should be open to development

2.4 Constraints

- The game will be implemented in Java.
- The game will be played with 60 FPS.
- Draws in the game will be smooth.
- The game will run all operating systems that support Java.

2.5 Scenarios

2.5.1. Play Game

Use Case Name: Play Game

Primary Actor: Player

Entry Condition: Player hit the "Play Game" button in the Main Menu.

Exit Condition: Player selects "Return To Main Menu" from the "Pause Menu", OR

Player has completed all three levels successfully, OR

Player has lost all three lives before the end of the game.

Event Flow:

- Player starts the game.
- System creates game environment for player.
- Player completes all three levels successfully.
- If player has better score than players that are in high scores table, system asks player's name to save high scores table.
- System records player's name and his/her score to high score table.
- Player returns to Main Menu.

Alternative Event Flow:

- Player lots his/her 3 lives and game ends. If player has better score than players that are in high scores table, system asks player's name to save high scores table. Player returns to Main Menu.
- Player chooses to exit game before it finishes or before s/he loses all three lives.

2.5.2 Pause Game

Use Case Name: Pause Game

Primary Actor: Player

Entry Condition: Player hits to “Esc” button.

Exit Condition: Player continues to play the game, OR

Player chooses to return to the main menu, OR

Player exits from game.

Event Flow:

- Player hits to “Esc” button during the game.
- System displays “Pause Menu”
- Player goes back to play game again.

Alternative Event Flows:

- Player chooses to return main menu. If player has better score than players that are in high scores table, system asks player’s name to save high scores table. Player returns to Main Menu.
- Player chooses to exit from whole game.

2.5.3 View High Scores

Use Case Name: View High Scores

Primary Actor: Player

Entry Condition: Player hits "View High Scores" button in the Main Menu.

Exit Condition: Player hits "Back to Main Menu" button in the View High Scores window.

Event Flow:

- Player selects the View High Scores in the Main Menu.
- System reads highest scores from file and displays these scores on the screen.
- Player sees highest scores in the game, then s/he returns to the Main Menu by pressing "Back to Main Menu" button in the View High Scores window.

2.5.4 View Credits

Use Case Name: View Credits

Primary Actor: Player

Entry Condition: Player hits "View Credits" button in the Main Menu.

Exit Condition: Player hits "Back to Main Menu" button in the View Credits window.

Event Flow:

- Player selects "View Credits" in the Main Menu.
- System displays names of contributors.
- After seeing contributors' names, player returns to the Main Menu.

2.5.5 View Help

Use Case Name: View Help

Primary Actor: Player

Entry Condition: Player selects “View Help” in the Main Menu, OR

Player selects “View Help” in the Pause Game.

Exit Condition: Player returns back to Main Menu, OR

Player returns back to play game.

Event Flow:

- Player chooses to display help menu.
- System displays information about game and some instructions about how to play game.
- Player returns back to Main Menu.

2.5.6 In Game (General Scenario)

Use Case Name: In Game

Primary Actor: Player

Entry Condition: Player hit the “Play Game” button in the Main Menu.

Exit Condition: Player selects “Return To Main Menu” from the “Pause Menu”, OR

Player has completed all three levels successfully, OR

Player has lost all three lives before the end of the game.

Event Flow:

- Player starts the game.
- Player moves right and left, uses space button to fire bullet and destroy the enemies.
- Player kills the soldiers by firing bullet and bonuses are dropped from the enemy
- Player takes the bonus, depending on the type of the bonus, he can increase its speed, double or triple its attack type or he can get health
- After cleaning all enemies in map, where number of enemies are decided early in level classes, he faces the boss and kills.
- He procedes to the next level.
- In the next level, he loses one life point because he is hit by bullet(sword) of the enemy, he loses one lifepoint
- He kills the final boss and congratulations screen appear or he loses all life points and game over screen appear.
- User is returned to the main menu

Alternative Event Flow:

- Player lots his/her 3 lives and game ends. If player has better score than players that are in high scores table, system asks player's name to save high scores table. Player returns to Main Menu.

Player chooses to exit game before it finishes or before s/he loses all three lives.

2.6 Use – Case Model

This section provides information about the main use case model of Running Jon.

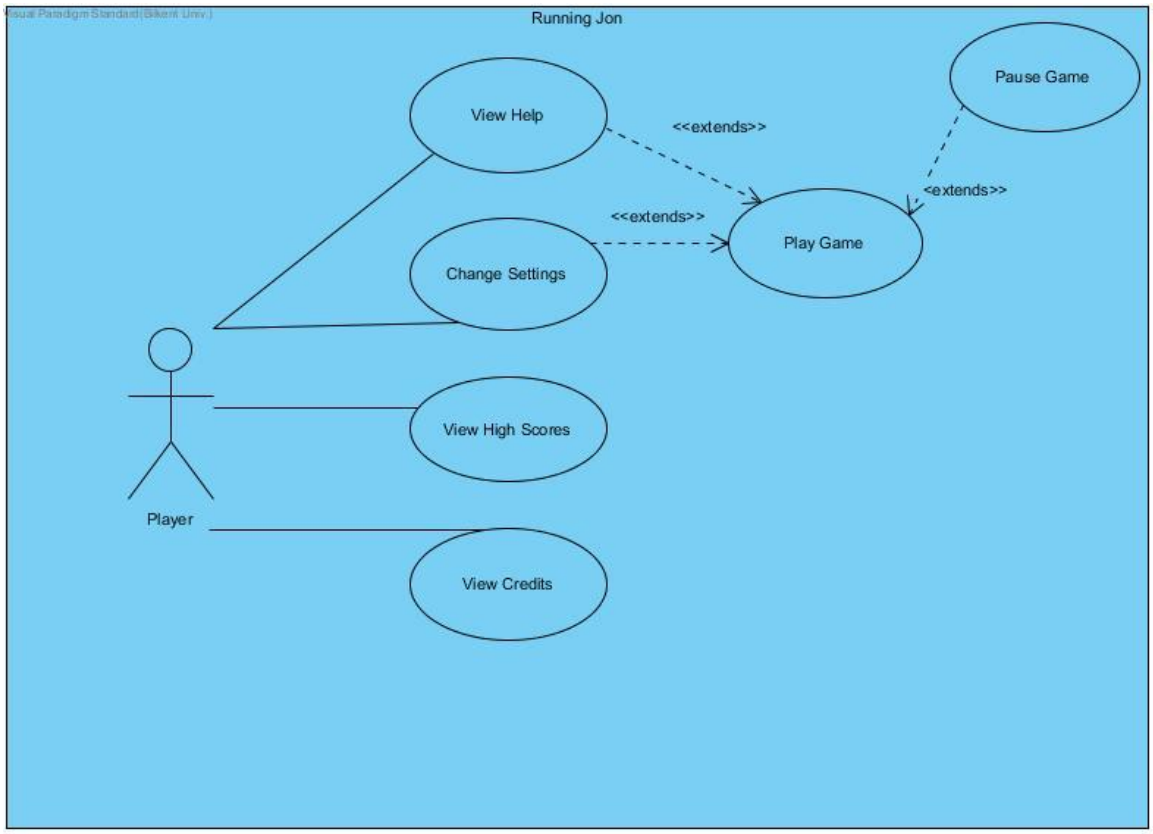

Figure 14: Use-case model of Running Jon

2.7 User Interface

2.7.1 Navigational Path

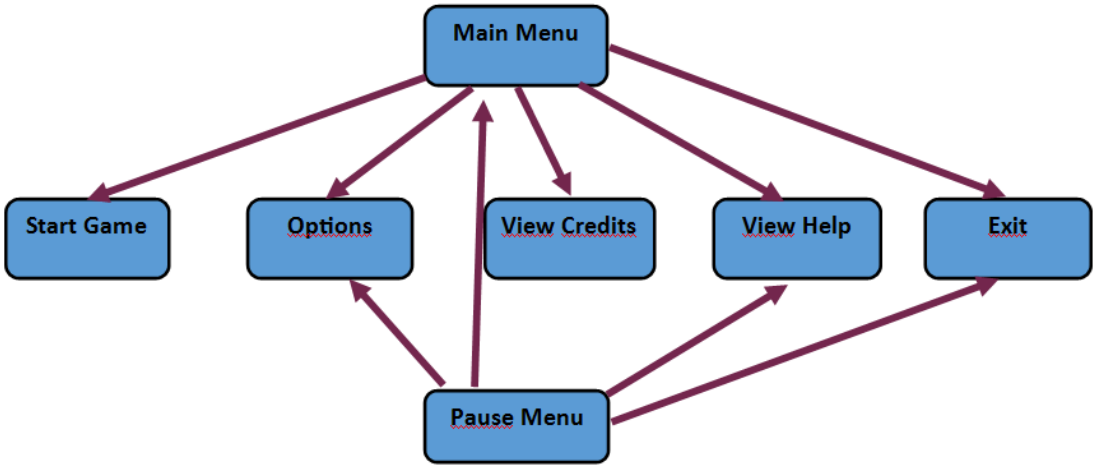

Figure 15: Navigational Path

2.7.2 GUI Mockups

2.7.2.1 Main Menu

Main menu is the first screen user sees when the game is executed. Main menu leeds user to game panel, help window, game options, high scores table, credits and the exit button.

Figure 16: Mockup for Main Menu

2.7.2.2 Help Window

Help window consists of a block of text that gives information on game controls and various power ups that user can collect throughout the game.

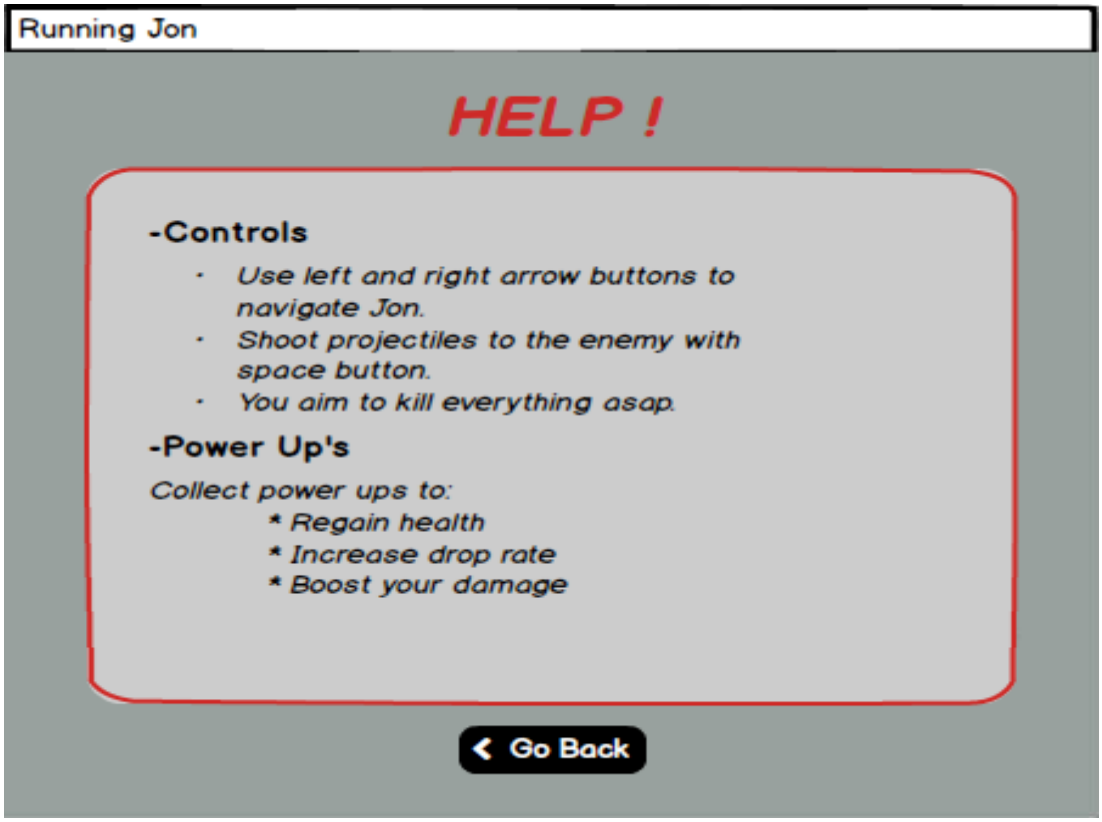

Figure 17: Mockup for Help Menu

2.7.2.3 Options Window

User can change game settings from this window such as gameplay difficulty, background soundtrack or in game sound.

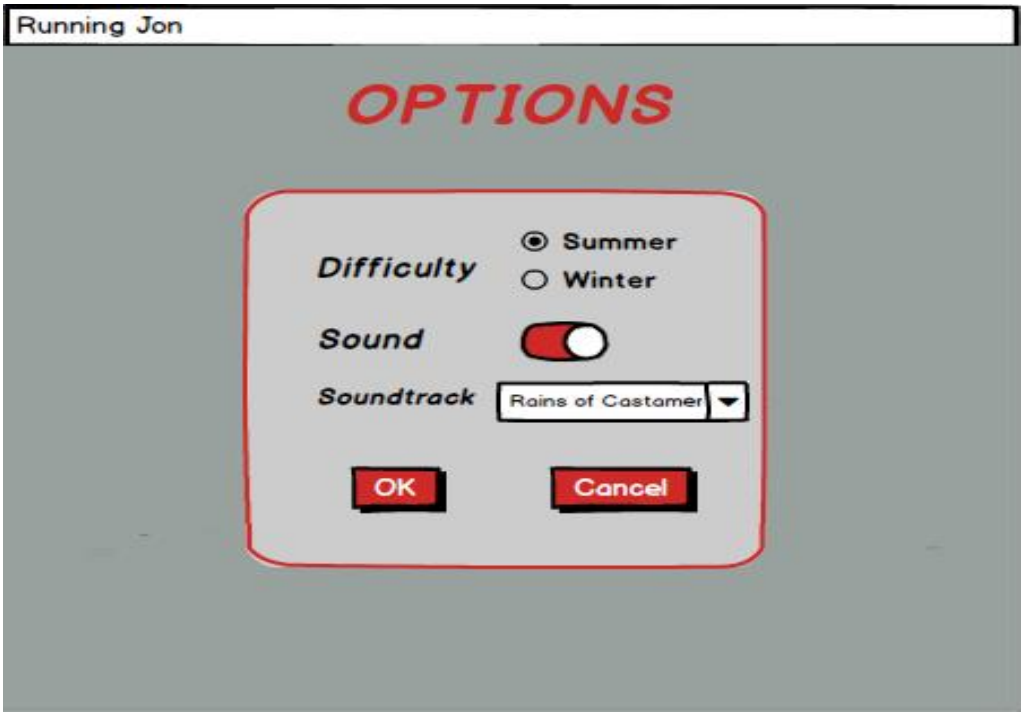

Figure 18: Mockup for Option Menu

2.7.2.4 High Scores Window

This window consists of a table that shows reached lvl and the score of a gameplay session if the users score is qualified to in the top ten among all players. Username is asked at the end of the game session if the score is high enough to be in top ten.

Figure 19: Mockup for High Scores Menu

2.7.2.5 Credits

This window shows the developers of this project.

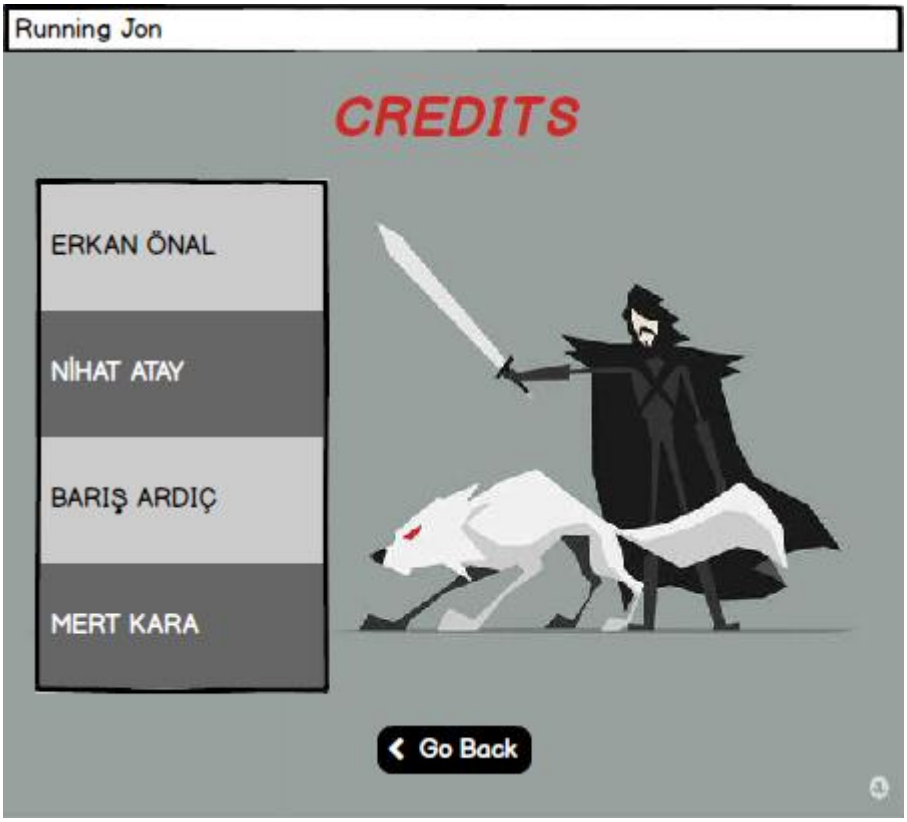

Figure 20: Mockup for Credits Menu

2.7.2.6 Game Paused

This is a pop up window which appears when user pauses the game. User can continue to play or exit the program.

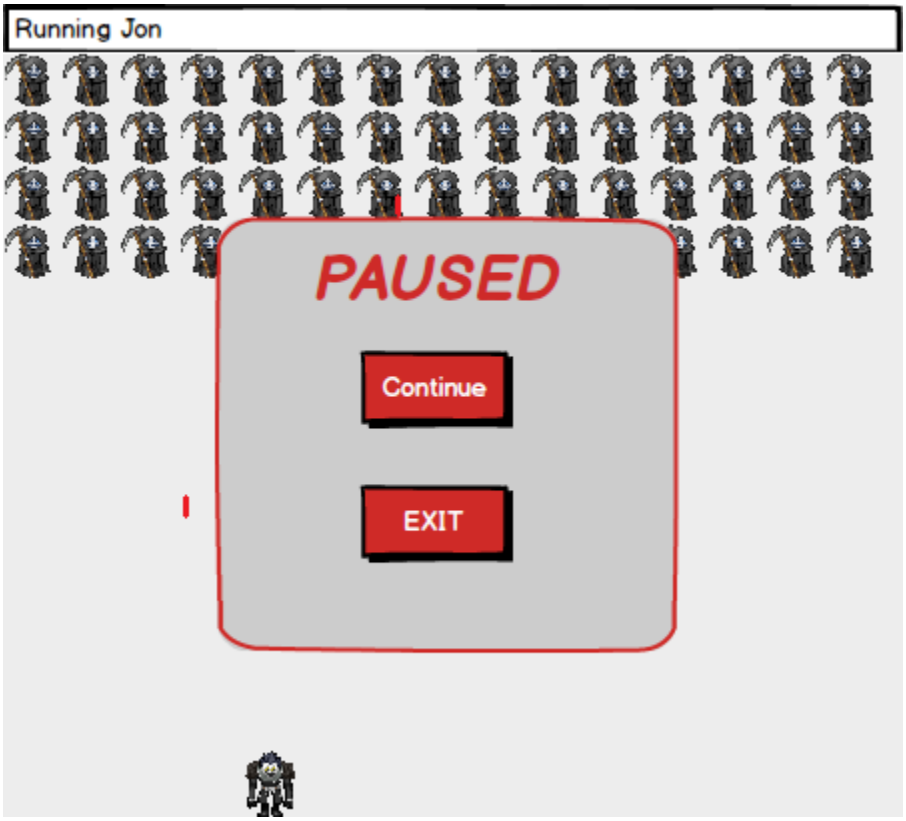

Figure 21: Mockup for Pause Menu

2.7.2.7 Level Is Completed

This is a popup window activated when user successfully finished a level. This popup displays the score and user options are to continue with the next level or to exit the program.

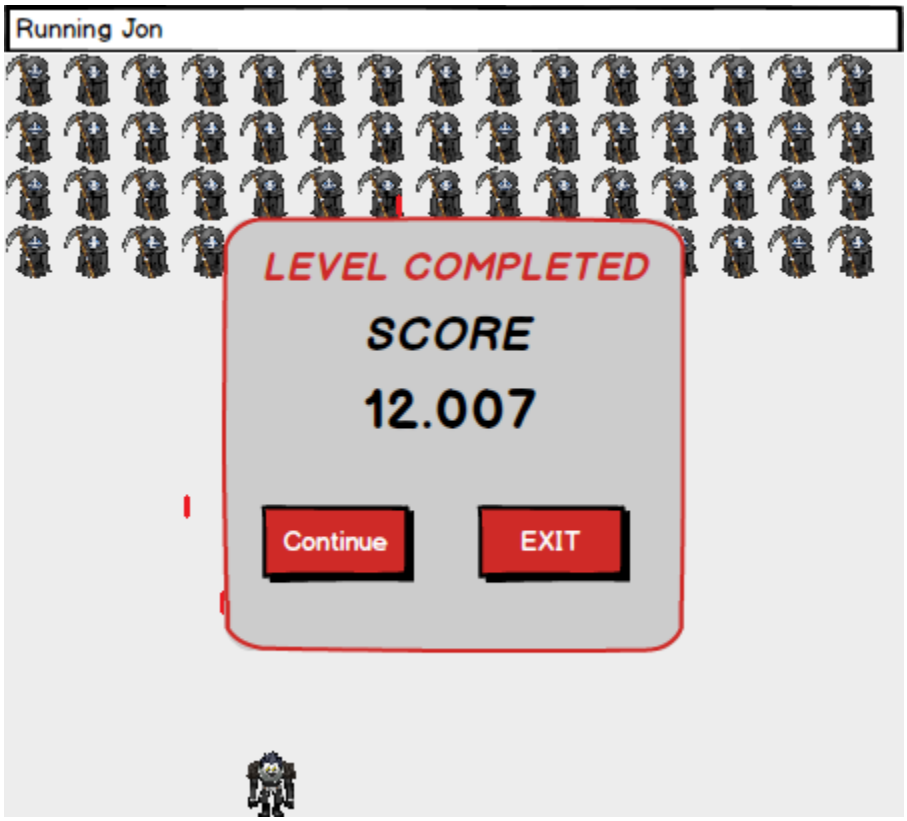

Figure 22: Mockup for Level completed

2.7.2.8 In Game Action

This is a prototype in game picture from the game. Characters and many things, surely, will be different but we added this to give you an idea about the game.

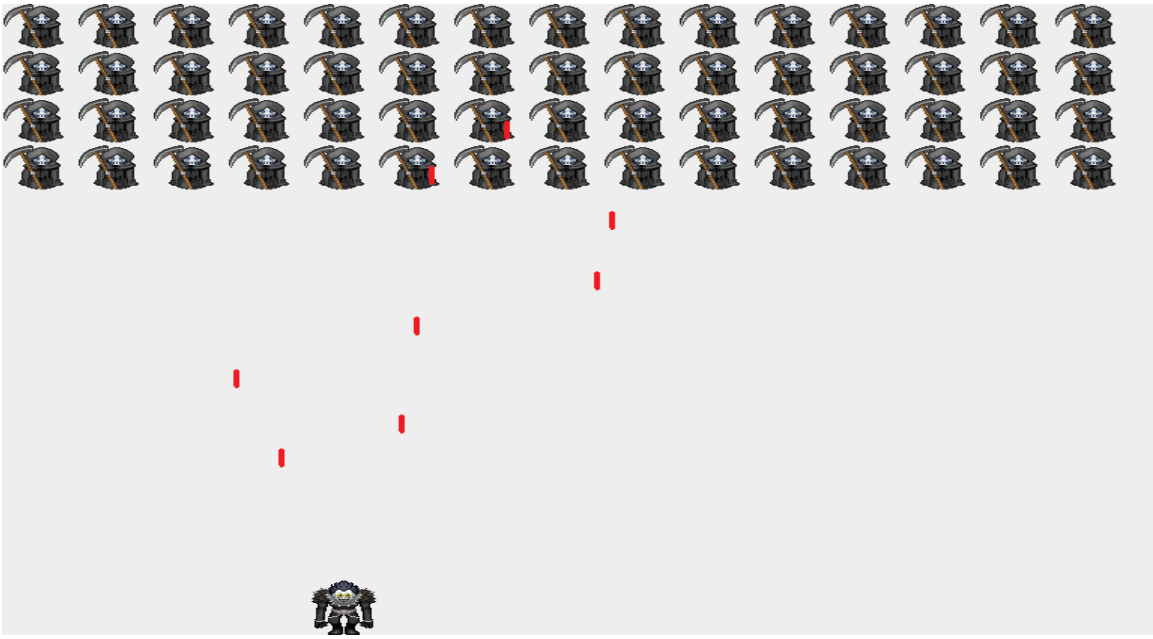

Figure 23: Mockup for In Game View

3. Dynamic Models

3.1 Object Model

3.1.1 Class Diagram

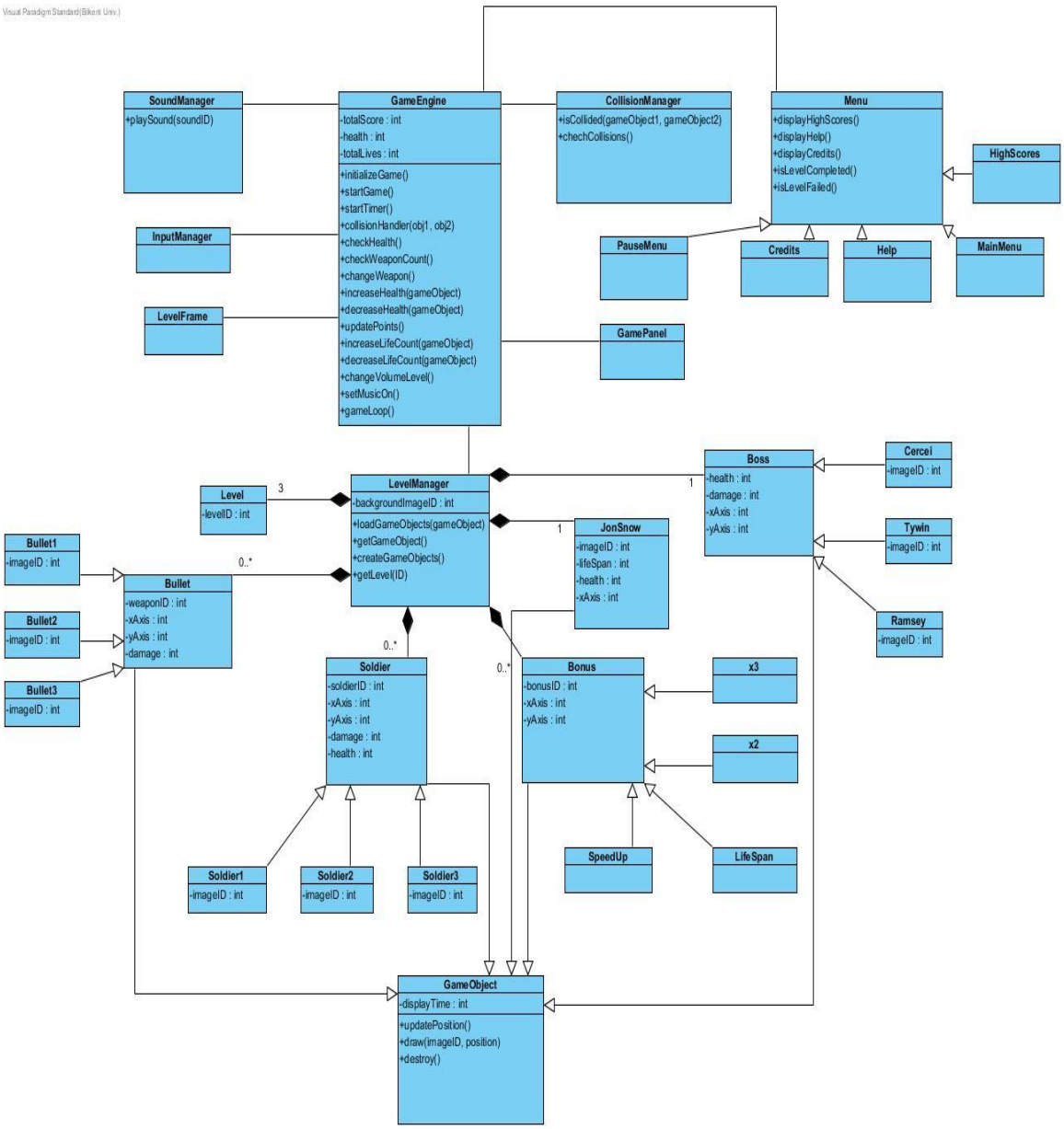

Figure 24: Class Diagram of Running Jon

3.2 Dynamic Models

3.2.1 State Chart Diagram

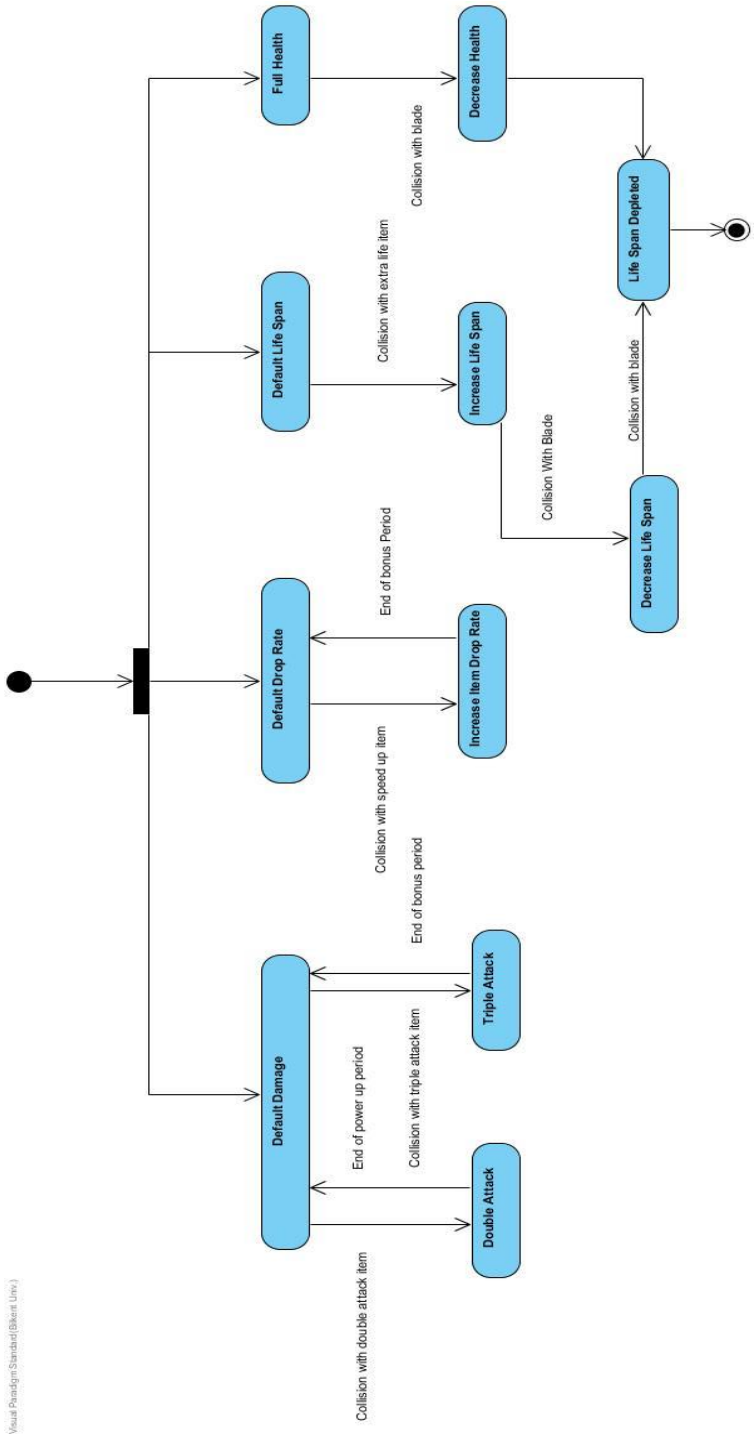

Figure 26: State chart diagram of Running Jon

3.2.2 Activity Diagrams

This diagram shows how main menu activities are carried out.

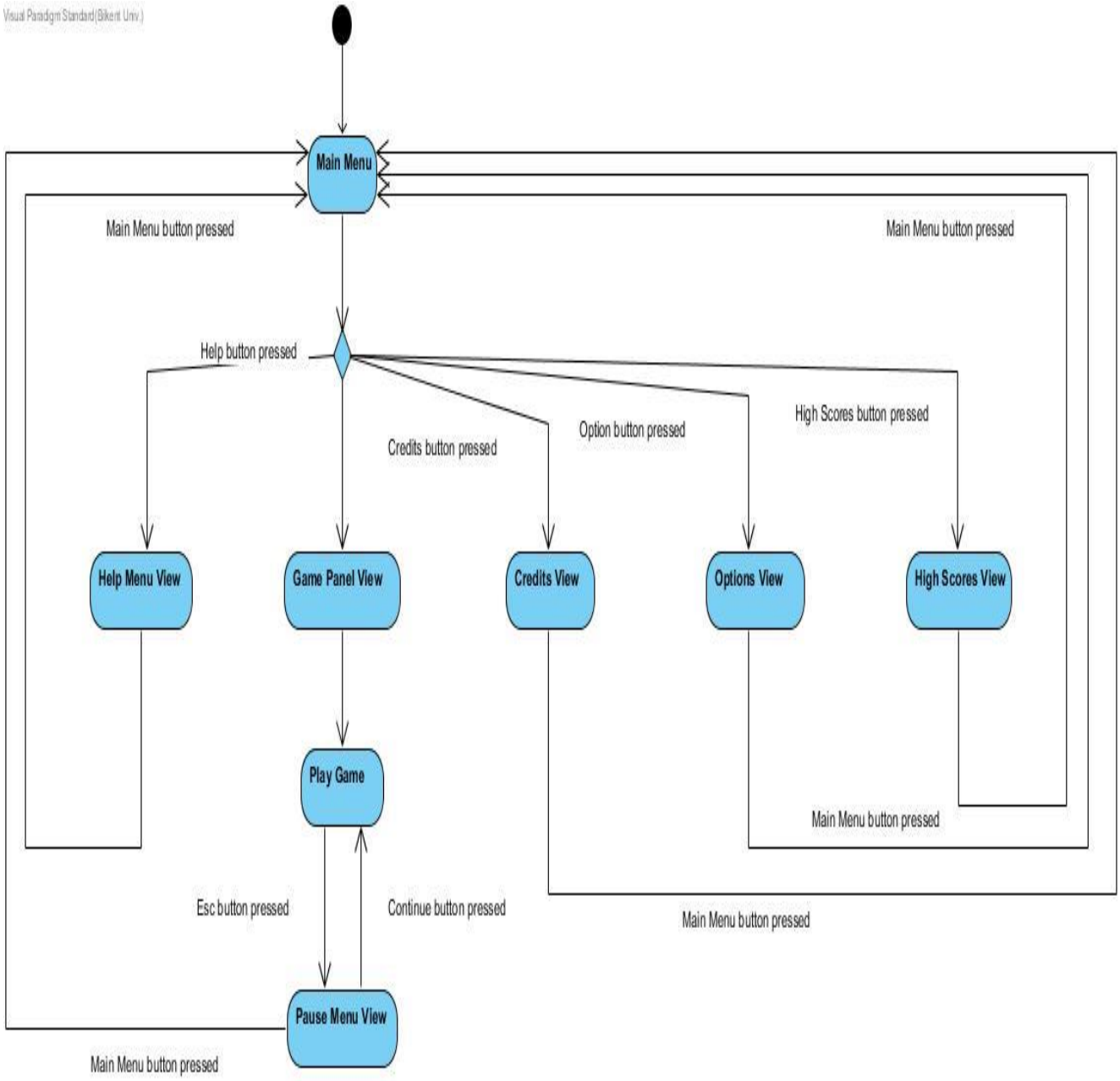

Figure 27: Activity Diagram of Overall Game Play

This diagram shows how game is maintained.

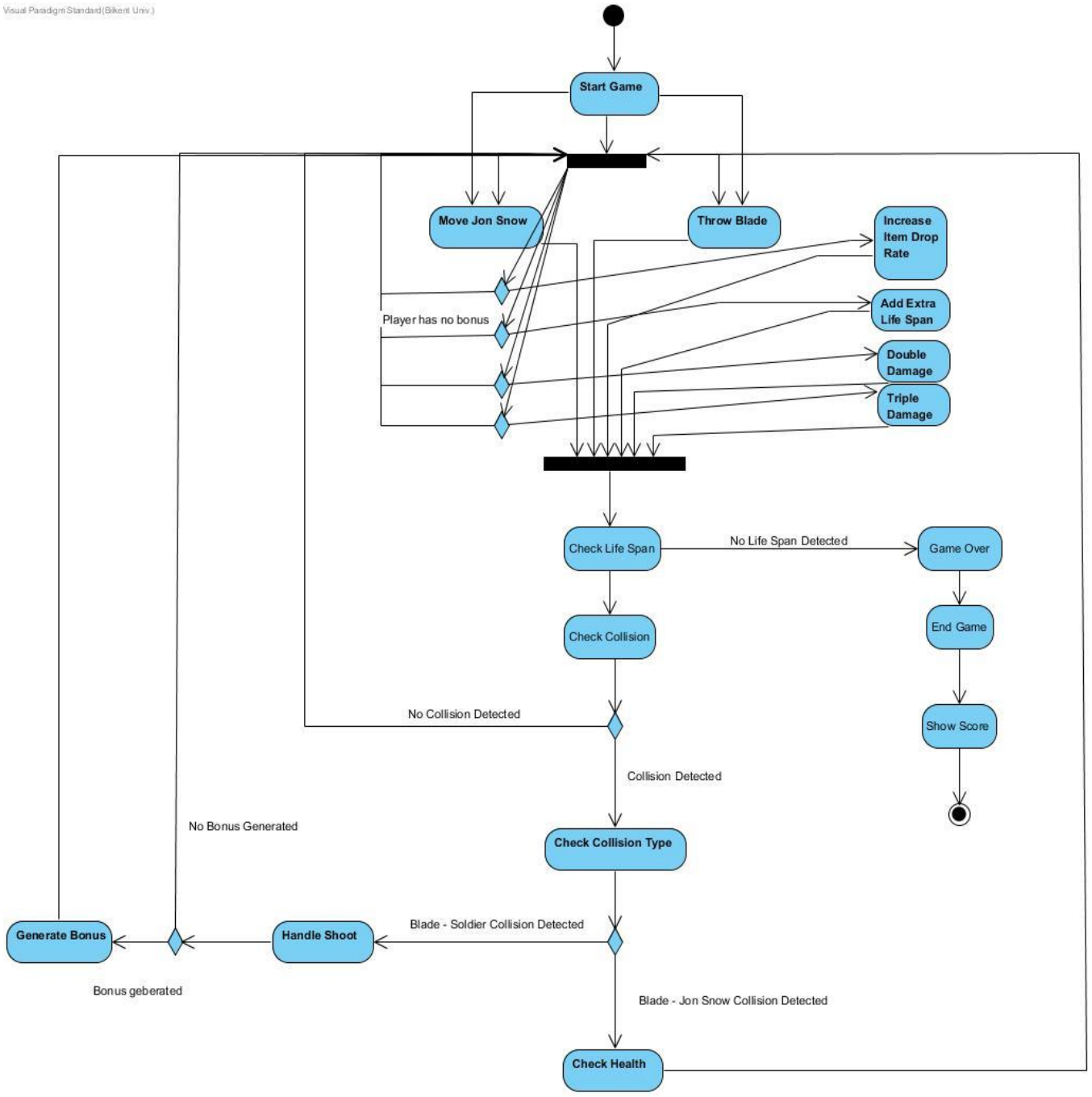

Figure 28: Activity Diagram of Play Game

3.2.3 Sequence Diagrams

3.2.3.1 Starting a new game

Scenario:

Player first requests to start game by pressing start game button from Main Menu. GameEngine initializes the game by creating the level frame, and level manager creates the necessary game objects which are Jon Snow and the soldier for the beginning. As initial objects are ready, the game starts and, system enters the game loop in order to continuously update the game.

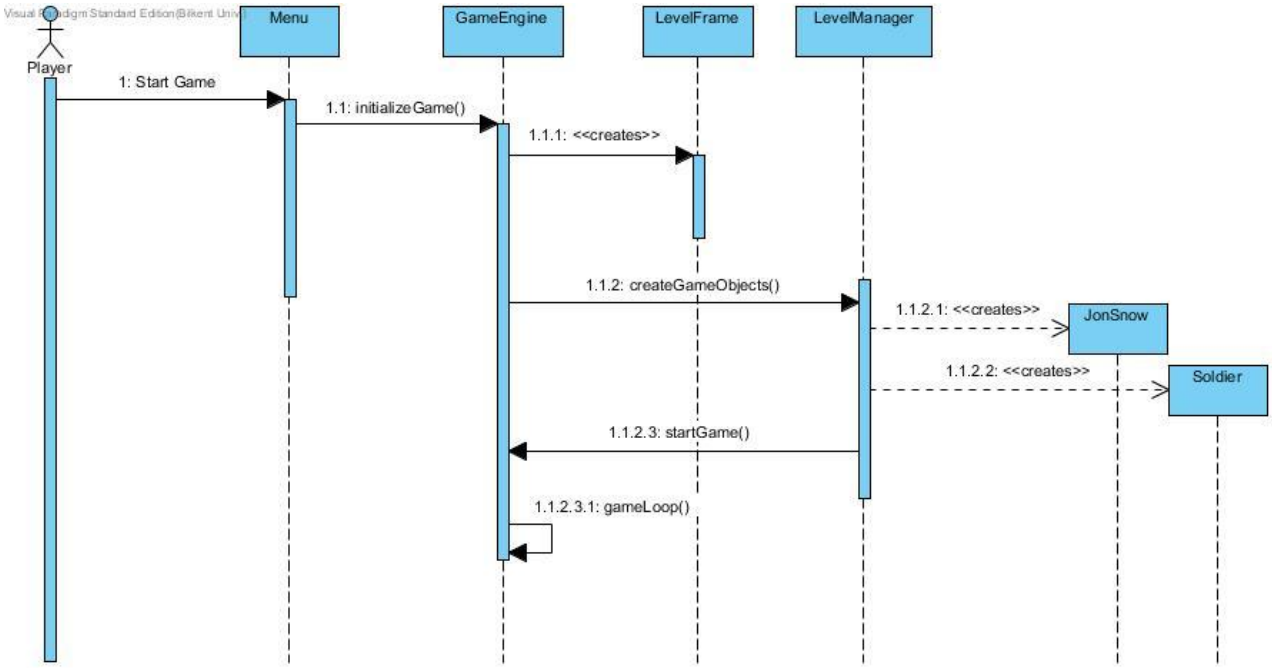

Figure 29: Sequence Diagram of New Game

3.2.3.2 Fire Bullet(Sword)

Scenario:

In this scenario, user first hits space button and from the input manager this input is taken, then GameEngine creates the bullet and adds it to map and sound for this action is also created in GameEngine then lastly, the changes are repainted to GamePanel. Only the creation of the bullet covered here.

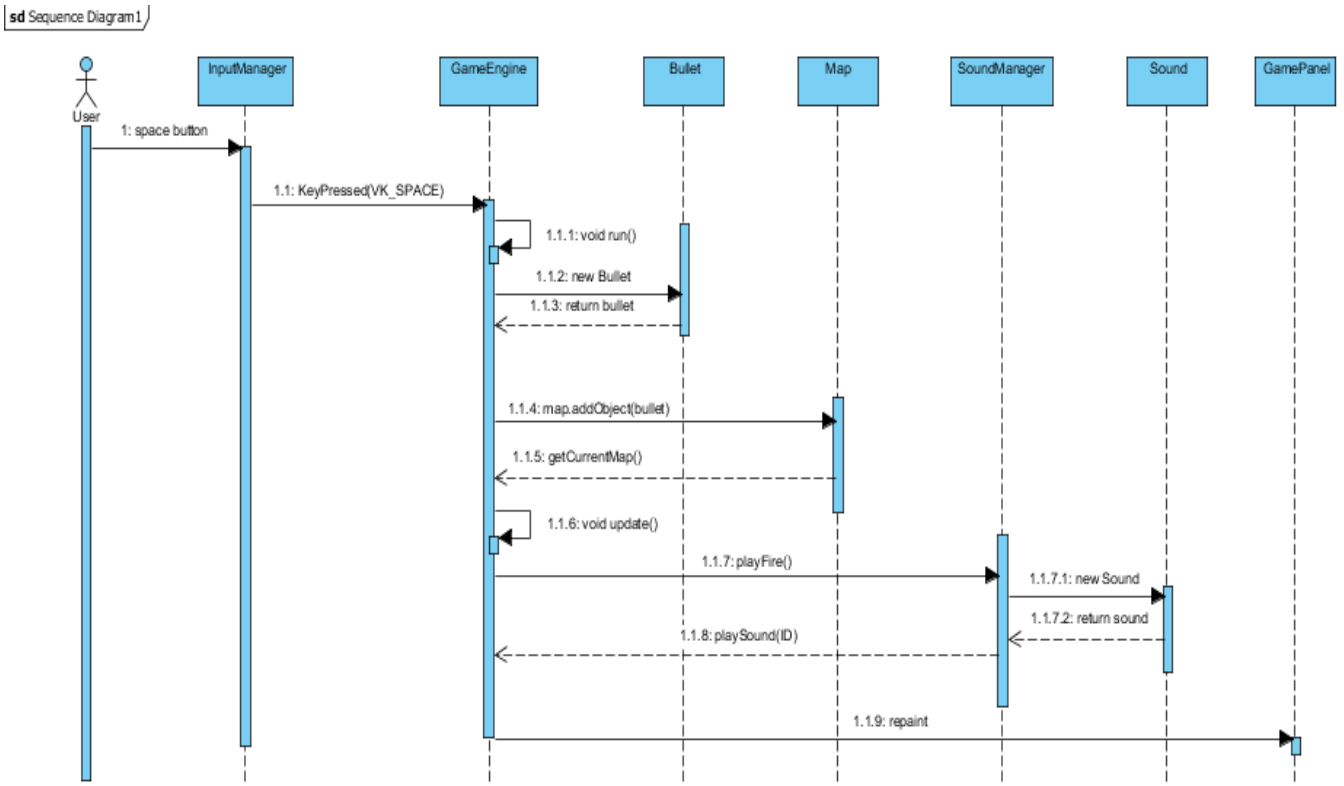

Figure 30: Sequence Diagram of Fire Bullet

3.2.3.3 Collision between bullet and soldier

Scenario:

In this scenario, the bullet thrown by user collides with enemy soldier and destroys it. The moment that bullet hits the soldier, GameEngine plays the sound and with the instance of LevelManager object, it removes the soldier, as arrays of soliders in one level is created in LevelManager, it removes it from there and lastly, it draws it on GamePanel.

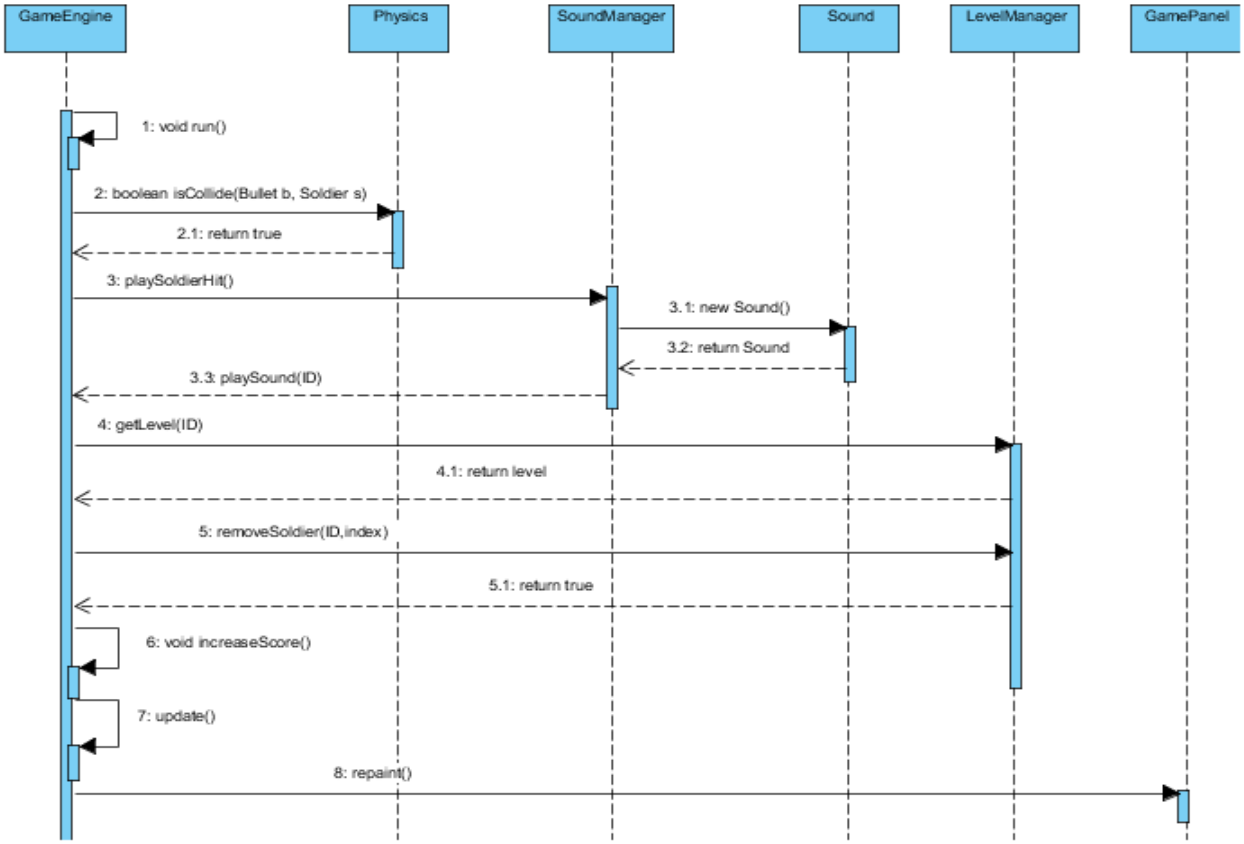

Figure 31: Sequence Diagram of Collision Between Soldier and Bullet

3.2.3.4 Collision between player and soldier

Scenario:

In this scenario, while moving the player to a location, the location user moved intersects with a soldier. The collision is detected by the method of Physics class(isCollide) and damage of the soldier is calculated, in this case, it is one heart point.

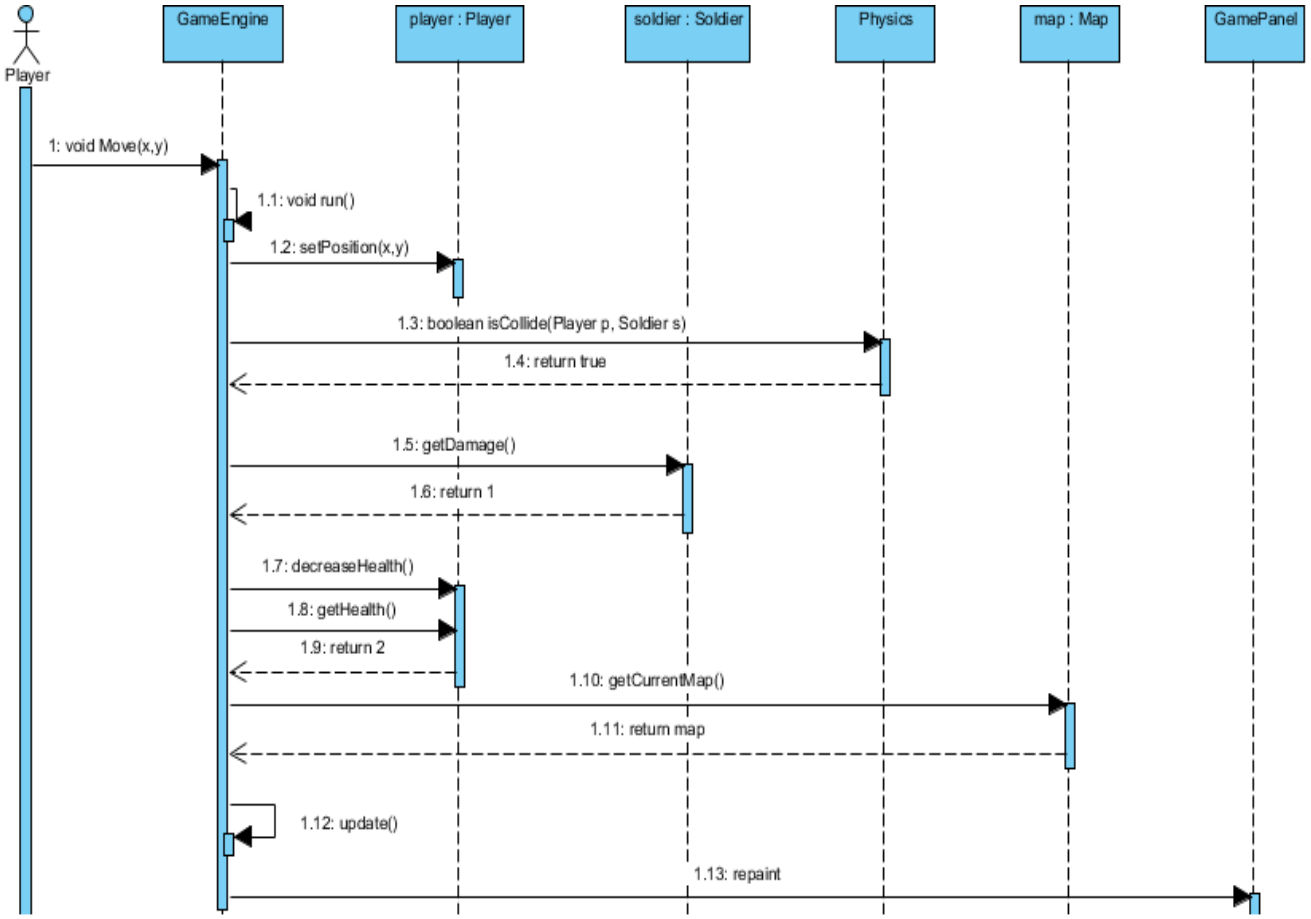

Figure 32: Sequence Diagram of Collision Between Soldier and Player

3.2.3.5 Getting the bonus from destroyed soldier

Scenario:

In this scenario, getting a bonus is shown. The sound is being played and depending on the type of the bonus whether it is a health bonus, speed bonus or x2 attack bonus, players' these qualities are upgraded according to span of the bonus.

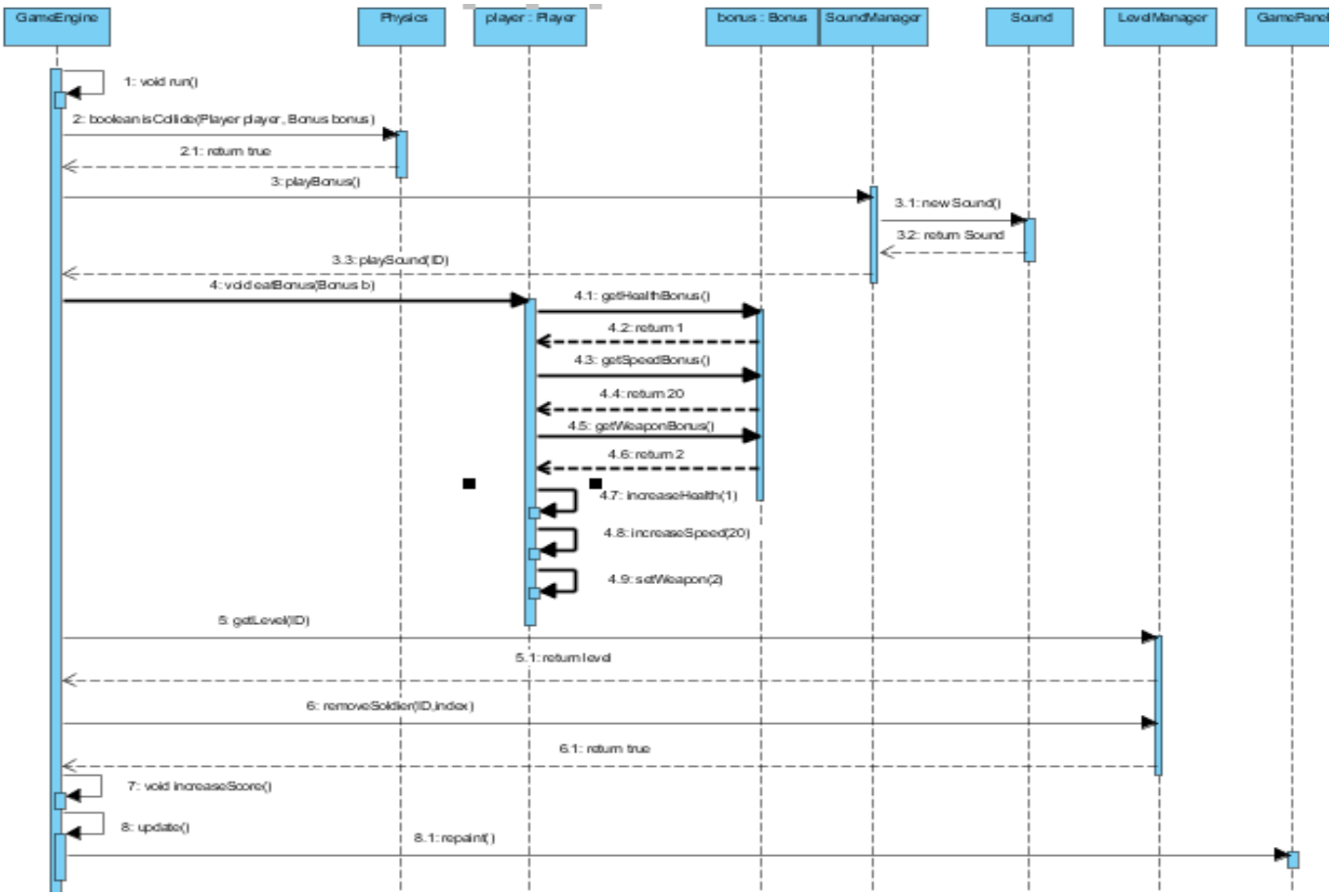

Figure 33: Sequence Diagram of Getting Bonus from Destroyed Soldier

3.2.3.6 Update the level

Scenario:

In this scenario, how to pass to level 2 is shown. After killing the boos of level one, in this case, Cercei, sound of passing a level is played and level 2 is taken from the level manager and it drawn into GamePanel.

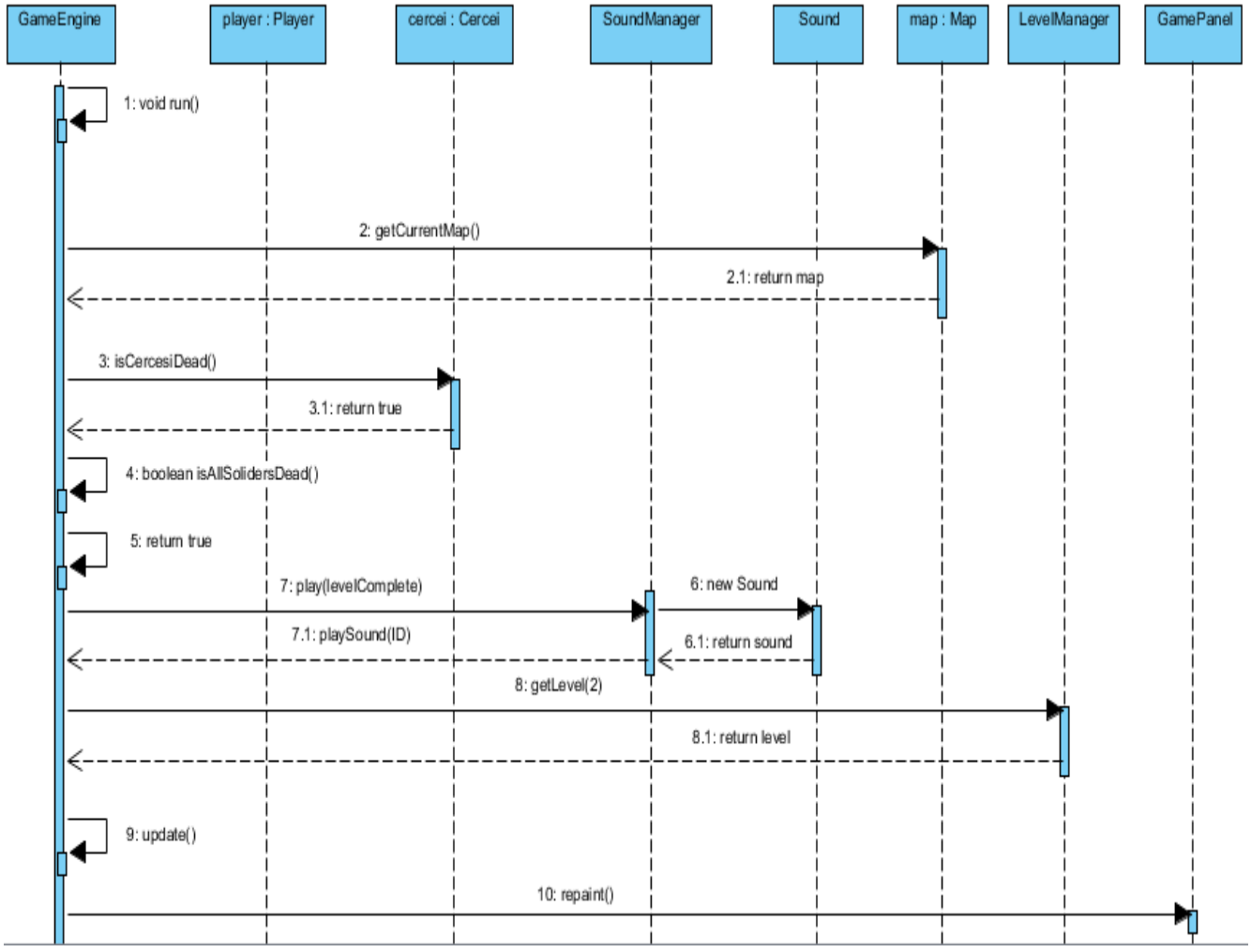

Figure 34: Sequence Diagram of Update Level

3.2.3.6 Change settings

Scenario:

If user wants to change the settings, he presses the settings button from the menu and settings panel is shown. There are options of sound off-on, change background. He makes the sounds off and he changes background.

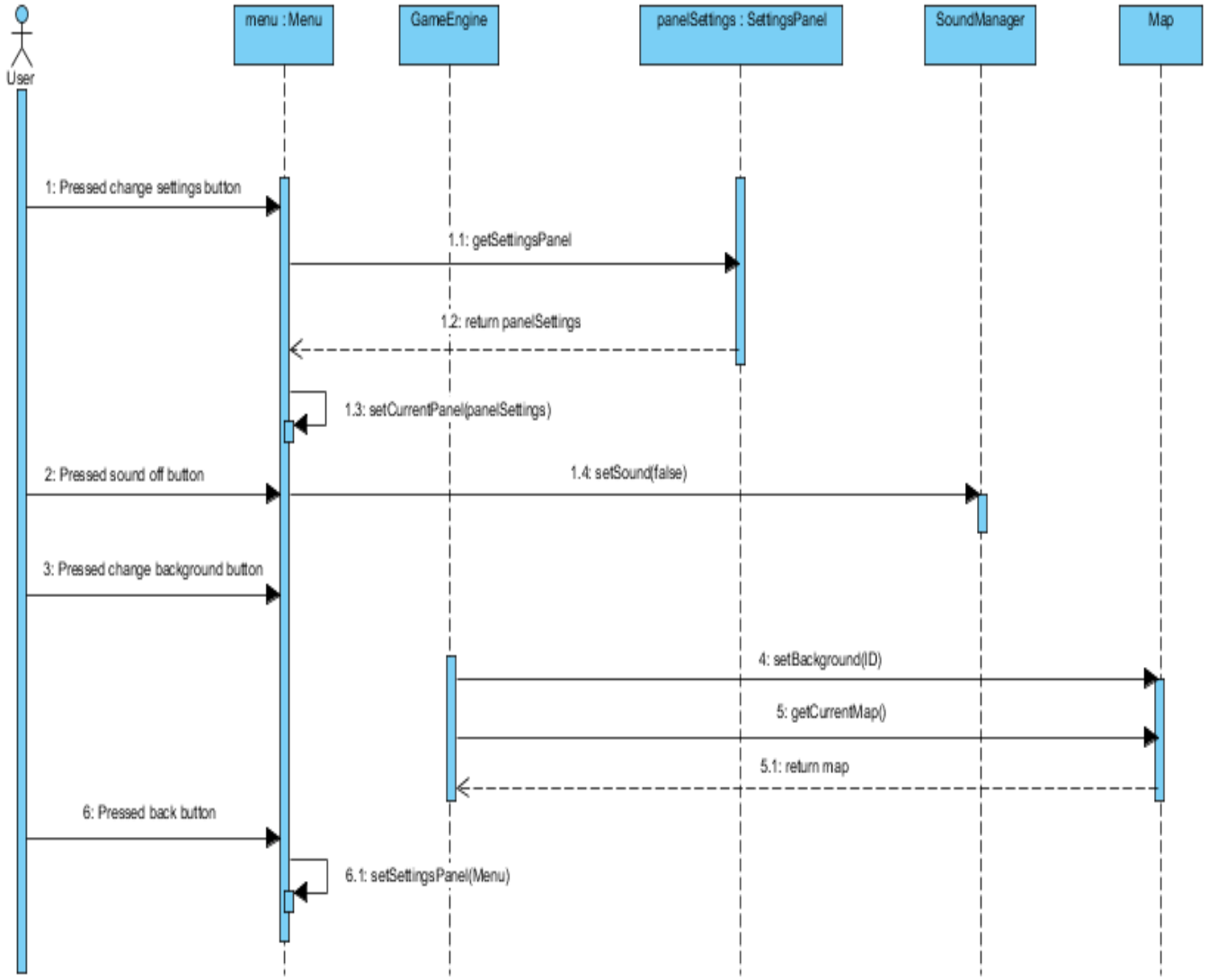

Figure 35: Sequence Diagram of Change Settings

4.Conclusion

In the analysis report, we tried to give the clear picture of what project does and how we shaped the class structure according to requirement of this game. We tried to include possible scenarios that help you undertand the concept of the game more. We think that the most important part of the project is to create a good object-class diagram as it will be the skeleton of the project. We tried to do it in a efficient way. We tried to apply fundamental object oriented concepts like inheritance and polymorphism.

While writing this report, visual paradigm was a great tool for us because it is really easy to use and very good at creating good diagrams. Therefore; we used it while creating every diagram and for mock-ups, we used balsamiq. It was also very good tool for us.

Along the process of creating the project, we will try to stick to this object plan and use cases.