Bilkent University

**CS 319 - Object-Oriented Software Engineering**

**Analysis Report**

**Man of War**

**Group 1-D**

* Berat Tuna Karlı
* Ceren Uysal
* Serdar Erkal
* Selim Fırat Yılmaz

**Table of Contents**

1.Introduction 5

[2.Proposed System](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.3znysh7) 6

[2.1 Overview](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.2et92p0) 6

[2.2 Gameplay](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.tyjcwt) 6

2.3 Character 7

[2.4. Enemies](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.1t3h5sf)  7

[2.5. Power](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.4d34og8) 8

[2.5. Map & Levels](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.2s8eyo1) 9

[3. Requirements](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.17dp8vu)  9

[3.1. Functional Requirements](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.3rdcrjn) 9

[3.1.1 Tutorial](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.26in1rg)  9

[3.1.2 Save-Load game](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.lnxbz9)  9

[3.1.3 Pause menu](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.35nkun2)  10

[3.1.4 Settings](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.1ksv4uv)  10

[3.1.5 Additional Requirements](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.44sinio)  10

3.1.5.1 Credits 10

3.1.5.2 Full Screen Option 10

3.1.5.3 Attack System 10

3.1.5.3 Change Difficulty 10

[3.2 Non-functional Requirements](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.2jxsxqh)  11

3.2.1 Graphical Smoothness 11

[3.2.2 Easy Gameplay](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.3j2qqm3)  11

3.2.3 Additional Non-functional Requirements 11

3.2.3.1 Maintability 11

4 System Models 12

4.1 Use Case Model 12

4.1.1 Main Menu System 12

4.1.1.1 Use Case Description 12

4.1.1.1.1 Use Case of Change Settings, Change Input Controls, Change Volume 12

4.1.1.1.2 Use Case of Play Game 12

### 4.1.1.1.3 Use Case of Select Difficulty, Set Easy/Normal/Hard 13

### 4.1.1.1.4 Use Case of Load Game, Select File Location

### 4.1.1.1.5 Use Case of Exit Game 14

### 4.1.1.1.6 Use Case of View Help 14

### 4.1.1.2 Pause Menu System 14

### 4.1.1.2.1 Use Case of Continue Game 14

### 4.1.1.2.2 Use Case of Save Game 14

### 4.1.1.2.3 Use Case of Select Save Location 15

### 4.1.1.2.4 Use Case of View Help 15

### 4.1.1.2.5 Use Case of Back to Main Menu 15

[4.2 Dynamic Models](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.2xcytpi)  16

[4.2.1 Object and Class Model 1](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.1ci93xb)6

[4.2.2 Sequence Diagram 1](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.3whwml4)7

4.2.3 Activity Diagram 22

4.2.3.1 Explanation of Activity Diagram 23

4.2 User Interface – Navigational paths and screen mock-ups 25

4.2.1  Navigational Path 25

[4.2.2 Main Menu 2](https://docs.google.com/document/d/1dN0N2xBvKTlE8WIDNTlasDYi8QI_ZIOulUSCUh_6Yq0/edit#heading=h.3as4poj)6

4.2.5 Game Play 27

5. Glossary 31

6. References 31

# 1.Introduction

CS 319 Project is inspired by Pokemon FireRed1. This project is a 2d quick time event based adventure game. This project will be based on the dialogs and battles. This role playing game's story will mainly include the adventures of unknown guy which will be less-unknown progressively during the player plays the game. As the player explores the world of our game, new quests will appear. Exploring will be through the dialogs and quests mainly consist of the battles. As the player completes the quests, he/she will earn powers. These powers the character could collect will help the player to fight better in the battles.

We choose this project because we think that this idea fits well in the OOP structure and this is a great opportunity for the usage of object-oriented programming. Moreover, in our childhood, we all had played 2d adventure games. This game will be nostalgia for all of us and for the users who play this game. In addition to this, we would like to study about the process behind creating a game. We will develop this project in Java because we all have experience in Java language. IntelliJ IDE will be used as a development platform. Final project will be deployed to .JAR through Maven2.

# 2.Proposed System

## 2.1.Overview

The logic behind Pokemon FireRed and our game is almost same but there are several differences like the dialog system, battle system, graphics, collectable item system, types of enemies and so forth.

Our dialog system would be less complicated with respect to Pokemon FireRed. Our dialogs will be based on the given basic information.

Battle system would have vast differences. In Pokemon FireRed, there is turn based battle system but we planned to have quick time event based battle system. We have searched QTE3 and some QTE based games. We thought that this is more challenging and more interesting to the player, therefore we chose QTE.

In our game there will be two types of enemies, one of them is the basic mobs and the other is bosses. The bosses are more powerful than the mobs.

## 2.2.Gameplay

The player can only use the keyboard. In order to move the main character, the player will use the arrow keys. Player can collect collectable items, just by passing through the item. As we mentioned, our battle system will be based on quick time event system\* and the player can beat the enemies with certain damage in every time.

## 2.3.Character

https://lh4.googleusercontent.com/DFnkR_VicbAPHgGfKAgucBINrUKhZrXda9DLW0JYc_0Fi3mMfKUkwgNR4ENcNUe3EVP-N-ji6nRld63vuo-1Sx0lZDubEJzdQxNZbVGlppl5c6I5KZXEK8j3BOJcbycXLXxSyAo_

* Character can move in 4 ways.
* Character is able to collect the items from the mobs, bosses and the collectable items.
* Character will have the attack point and the health bar.
* Character can fight with the different types of the enemies.

## 2.4. Enemies

There are 2 main types of enemies in this game.

Mobs:

https://lh6.googleusercontent.com/hrnY7v9FrSPi6YU5R7ki7EpMYCwQpcIeIzI4POVMFuz4f6BZtxtXl8OplkPqF3xQXduHtKqozvRSvAYWNxBZCcqokeYJHevjDymQ4-QHzvkmROCyVSxXRBR1EMYqY4IrtTw-_Vdi

* Have attack point and the health bar.
* Have power for the player if player defeat them.
* Have less power than the bosses.

Bosses:

https://lh3.googleusercontent.com/i-GdcLgeeljOttJTTND5yLDeoIY-09EGzK6TlpgDZ_KRLFAVYhugvadWVx8Iw80Bh2VQwlq15UBIncGRvDl4eGiYaHlJ8lV1yWeVxmusCqj6kudDJz96WShZ62CHSA0JbEYHM_2b    https://lh5.googleusercontent.com/-ELu-gYPMsnNzJZDVgP_k3cuuZZzzj9h4t5DqZIQoKOAcu3PuTF8uddLyLQq_YYCXElWYUs-tqjW7hg9i-3OTEFOqKH3O7wx17GRBmlRkRKRGDDP37sDLIftVBTHeWS5FFK-sy9T     https://lh6.googleusercontent.com/BplKbeQFDizZQHJlzgUrqKF-EE9HRYTFv3vxsMedpAZLp12ZgC_r6meiJrt3uBFlNiGxZw0gOQnKy06DwTIucgfl4XeJrvwOj70VcBuGYiIKOi0-BWbXZyFX7UiVFVvFQKJ5xidO

Various Bosses

* There are two types of bosses red and blue.
* Player attack his/her specific attack power(blue or red) to blue or red boss.
* Have attack point and health bar.
* Have power for the player if player defeat them.
* Have more power than the mobs.

## 2.5. Power

***https://lh4.googleusercontent.com/oU2EXP5lZihZgKQl3B2meRYHadLtllxQZ5hLbDCjnjFwAqWf9YxszP_CvubPP7wQIh_Gr5hoxbYo0gcuARmS3LDG_Pan04xYht4keDXZgYWG6T9_blco7nloNWWLQH67A0Z0eRExhttps://lh3.googleusercontent.com/j38HCO9K22fCx0XTXUWVKLEgUa2hPm5kPxoC4IhmdSZ5wIg3htLvwNAAzRZG4_8hDM3ZOE-W7mVkbrZ0IpAhPKlxfBQev8kRPAhIsGNLHaYdcTez7bI9GXKOQOksJBNHZocvGpJD***

* These powers will be character’s attack power.
* Player can collect powers in game through mobs and bosses.
* The bosses have more powers to collect.
* There are red and blue powers.

## 2.6. Map & Levels

* Three different levels (one bonus level might be included).
* On the screen only the current level will appear.
* Every level has at least one gate which opens to the next level.
* Collectable items and mobs will be randomly distributed according to the difficulties of the levels.
* Bosses will be located in front of the gates. If player does not kill the boss he/she will not be able to go through the gate.
* If the player dies, the levels reset and if the player passes to the next level, his/ her health bar will be reloaded.

# 3. Requirements

## 3.1. Functional Requirements

### 3.1.1 Tutorial

This help option will provide player general information about how to play the game. This is an optional feature in the main screen. There will be at most three pictures which clearly show how to play the game.

### 3.1.2 Save-Load game

Player can save the last level of the game and his/ her stats. He/she can load the previous game and then he/she can continue. In the main screen, there will be an option for the load.

### 3.1.3 Pause menu

If player clicks pause during game, this stops the game and opens up the pause menu. In the pause menu, there will be options for saving game, loading game and settings.

### 3.1.4 Settings

Player can turn down or turn off the background music or the game sounds.

### 3.1.5 Additional Requirements

### 3.1.5.1 Credits

* Information about the designer of the game.
* Links to GitHub and the project website.

### 3.1.5.2 Full Screen Option

* The player will be able to choose playing the game whether in full screen or not with respect to his or her preference.

### 3.1.5.3 Attack System

* The player is able to fire with originally space key but in the future the player will change the space key with respect to his/her preference.

### 3.1.5.4 Change Difficulty

* The player will be able to choose the difficulty of the game before s/he starts playing the game. This option will make the game more interesting and more challenging for the player.

## 3.2 Non-functional Requirements

**3.2.1 Graphical Smoothness**

Our game will have pixel graphics since nowadays these graphics are becoming very popular. Even though we choose the pixel graphics, there will be animations and our game will be smooth looking.

**3.2.2 Easy Gameplay**

Man of war has an easy gameplay. This game is designed without considering the age or gender of the player. This is why everyone is able to play our game easily.

**3.2.3 Additional Non-functional Requirements**

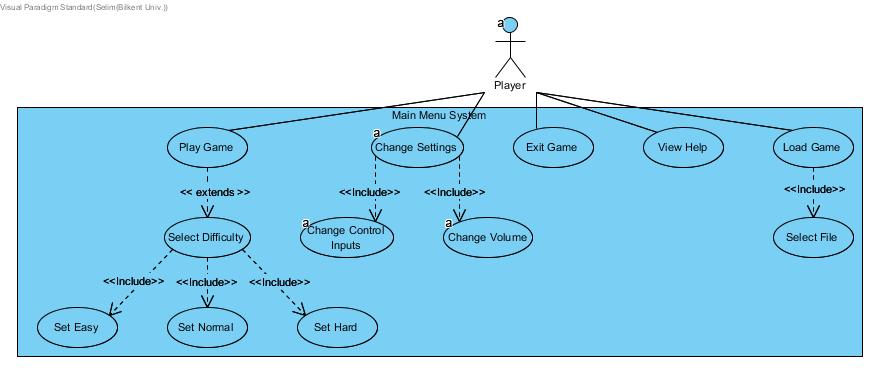
**3.2.3.1 Maintability**

Man of War has two separated “presentation” and “logic” parts in the context of codes. This two separated parts allow us to work more easily and independently. Also the errors which might occur will be solved quickly and easily in the manner of time and difficulty.

# 4. System Models

## 4.1 Use Case Models

### 4.1.1 Main Menu System



**4.1.1.1 Use Case Description**

### 4.1.1.1.1 Use Case of Change Settings, Change Input Controls, Change Volume

Primary actor is the “Player”, if player wants to change settings of the game with respect to his/her preference, (s)he clicks the “Settings” Button. Pre-condition is users must be on the main screen. Post-condition is that user have to submit the changes which are made by himself/herself. Entry condition is clicking the “Settings” Button on the main screen. Exit condition is clicking “Back to Menu” Button. Event flow is that the frame is open, options for settings are displayed.

### 4.1.1.1.2 Use Case of Play Game

Primary actor is the “Player”, if player wants to start playing the game, (s)he clicks the “Play The Game” Button. Pre-condition is player must be on the main frame. Post-condition is not required for this use case. Entry condition is clicking the “Play The Game” button on the main screen. Exit condition is pausing the game and clicking “Back to Menu” button or “Back” button on select difficulty screen. Event flow is that “Select Difficulty” screen is open.

### 4.1.1.1.3 Use Case of Select Difficulty, Set Easy/Normal/Hard

Primary actor is the “Player”. Thus, player can decide to play at easy/normal/hard difficulty. Pre-condition is that player must be clicked to the “Play Game”. Post-condition is not required for this use case. Entry condition is clicking “Set Easy/Normal/Hard” buttons on the select difficulty screen. Exit Condition is selecting different difficulty again on “Select Difficulty” screen. Event flow is that the game screen is open, player starts to play game with selected difficulty.

### 4.1.1.1.4 Use Case of Load Game, Select File Location

Primary actor is the “Player”, if player wants to start playing the game from saved game, (s)he clicks the “Load Game” Button. Pre-condition are users must be on the main frame and there must be a saved game from the previous play. Post-condition is not required for this use case. Entry condition is clicking the “Load” Button on the main screen. Exit condition is clicking “Load” Button. Event flow is “Load” Button is clicked and game screen is open, player starts to play game from where he has left.

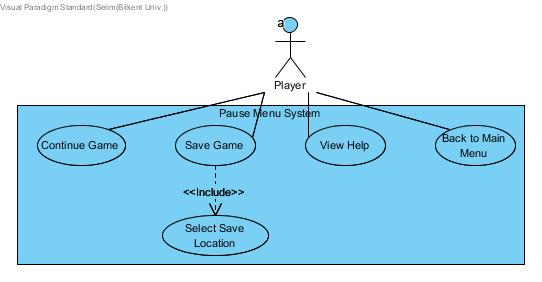
### 4.1.1.1.5 Use Case of Exit Game

Primary actor is the “Player”, if player wants to stop playing the game, (s)he clicks the. Pre-condition is the users must be on the main frame and there must be an ongoing game play. Post-condition is not required for this use case. Entry condition is pressing the “ESC” key. Exit condition is the disappearing of the game screen. Event flow is pressing the ESC key and exiting the game.

### 4.1.1.1.6 Use Case of View Help

Primary actor is the “Player”, if player wants to view help about the game, (s)he clicks the “View Help” Button. Pre-condition is the users must be on the main screen. Post-condition is not required for this use case. Entry condition is clicking “View Help” button on the main screen. Exit condition is having the information screen appeared. Event flow is clicking the “View Help” button and the information screen will appear.

### 4.1.1.2 Pause Menu System



### 4.1.1.2.1 Use Case of Continue Game

Primary actor is  the “Player”, if player wants to continue the game, (s)he clicks the “Continue Game” Button. Pre-condition is the users must be on the “Pause Menu”. Post-condition is not required for this case. Entry condition is clicking the “Continue Game” button on the Pause Menu. Exit condition is restarting to play the game. Event flow is the Pause Menu screen disappears and the game screen is opened and the player restarts to play the game.

### 4.1.1.2.2 Use Case of Save Game

Primary actor is the “Player”, if player wants to save the game, (s)he clicks the “Save Game” Button. Pre-condition is the users must be on the “Pause Menu”. Post-condition is not required for this use case. Entry condition is clicking the “Save” Button on the Pause Menu. Exit condition is pausing the game and clicking “Save” Button. Event flow is the game screen is opened, the player starts to play game.

### 4.1.1.2.3 Use Case of Select Save Location

Primary actor is the “Player”, if player wants to save current game state to a spesific location, (s)he selects the location to save. Pre-condition is the users must be on the “Pause Menu”. Post-condition is not required for this use case. Entry condition is clicking the “Save” Button on the Pause Menu. Exit condition is clicking “OK” Button. Event flow is the game screen is opened, the player starts to play game.

### 4.1.1.2.4 Use Case of View Help

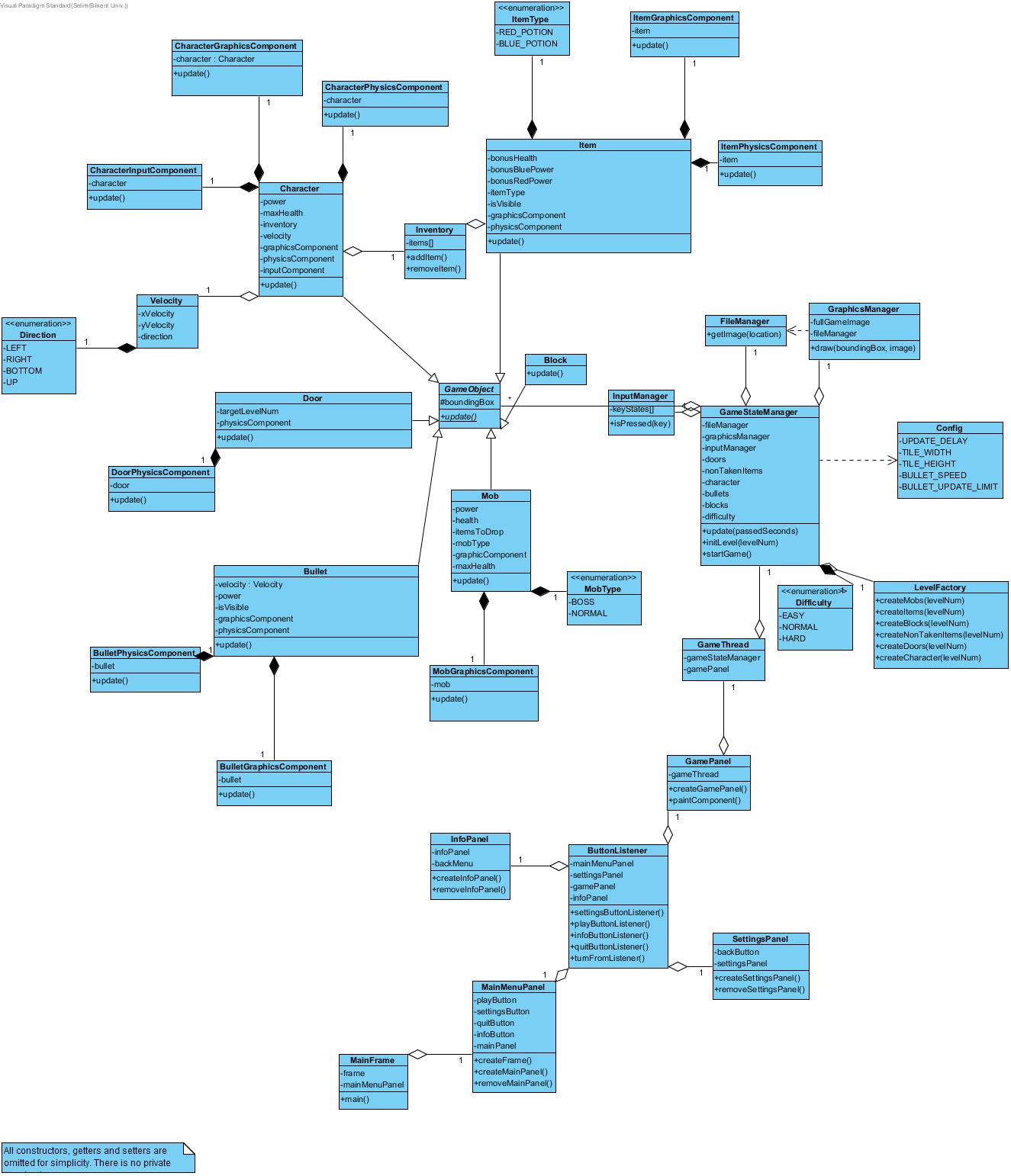
Primary actor is  the “Player”, if the player wants to learn from the View Help. Pre-condition is the users must be on the “Pause Menu” . Post-condition is not required for this use case. Entry condition is clicking “View Help” button on the main screen. Exit condition is having the information screen appeared. Event flow is clicking the “View Help” button and the information screen will appear.

### 4.1.1.2.5 Use Case of Back to Main Menu

Primary actor is  the “Player”, if the player wants to go back to main menu from the pause menu. Pre-condition is the users must be on the “Pause Menu” . Post-condition is not required for this use case. Entry condition is clicking  the “Back to Main Menu” button in the Pause Menu. Exit condition is clicking “Resume” button to continue the game. Event flow is clicking “Back to Menu” button and the main menu will appear.

***4.2 Dynamic Models***

**4.2.1 Object and Class Model**



In our design, We aimed to divide our code into some parts in order to ease collaboration in our team. “MainFrame” is the first class constructed by the main method. Each panel in GUI divided into classes like “SettingsPanel”, and “MainMenuPanel”. “GamePanel” class initializes the “GameStateManager” and “GameThread”. “GameThread” class is the responsible of the game loop. Executes update methods of the game objects through “GameStateManager” and repaints the “GamePanel”. “GameStateManager” is to control the game state and contains all its game objects, manages coherence between utility managers and game objects, contains all managers and game objects but interferes with them as little as possible. GameObject class is a main template of the objects in the game. Common properties of objects in the game are represented in this class. Each concrete GameObject specialization class has some components like “CharacterPhysicsComponent”, and “CharacterGraphicsComponent”. These components are composited in the related concrete GameObject specialization class so that this class can manage different responsibilities in different parts like drawing character to screen in “CharacterGraphicsComponent” and checking collisions in “CharacterPhysicsComponent”.

**4.2.2 Sequence Diagram**

**Scenario:** Player Cengiz wants to play the game

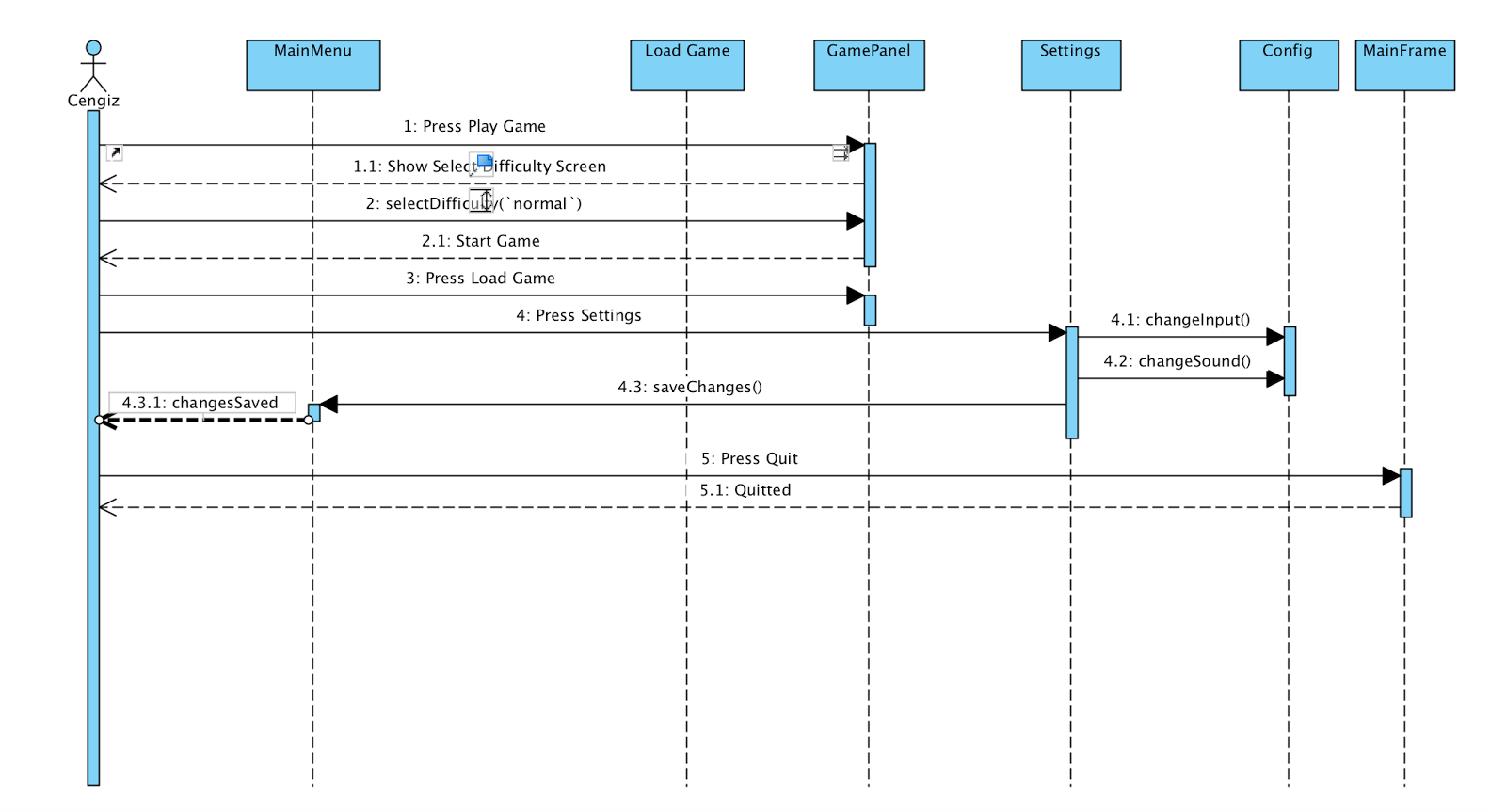
Player Cengiz clicks the desktop icon of the game and starts the game. There is the main menu which is shown, Cengiz chooses to play a new game by clicking “Play Game” option. The difficulty of the game is also selected. Then, Cengiz starts to play the game.

**Scenario:** Player Cengiz wants to change the settings

Player Cengiz wants to change the settings of the game. He clicks on the Settings button on the menu screen. In the Settings Menu, there are 2 different options such as Change Sound and Change Input. Change Sound is responsible for changing the sounds of the game such as up or down. Change Input is responsible changing the keys which are required for playing the game. In order to complete this process, clicking the “Settings” button and doing the operations with respect to Cengiz’s preference should be done.

**Scenario:** Player Cengiz wants to quit

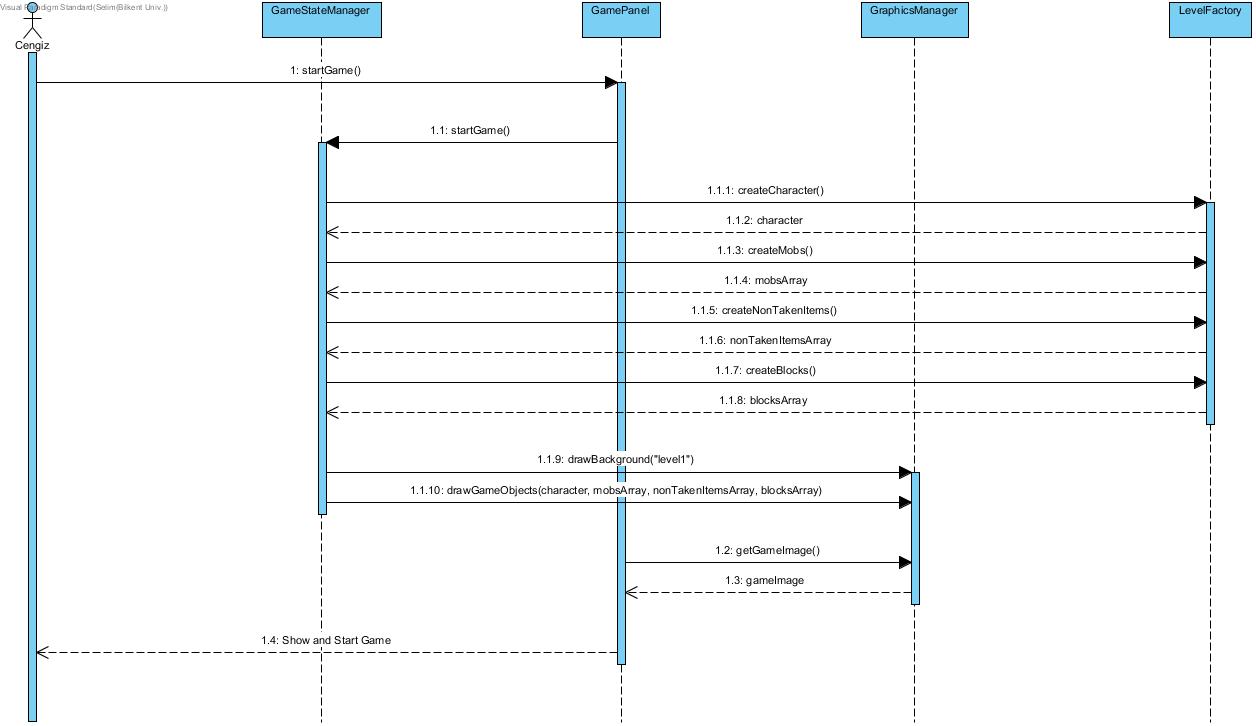
If Cengiz wants to quit the game, he should click the “Quit” button on the main frame. In the game, the player is able to quit directly without even starting to play the game. After clicking the “Quit” button, Cengiz closes the main frame and quits from the game.



**Scenario:** PlayerCengiz starts playing the game

Cengiz chooses to play a new game by clicking “Play Game” option. The difficulty of the game is selected, the game is loaded and the game screen appears. By loading the level state; the character, the mobs, the boss and the items are also automatically loaded. This process is done by “Level Factory”. The background of the level and the game objects are drawn with the help of “Graphics Manager”. The game is displayed by “Game Panel”.

Finally, Cengiz starts playing the game.

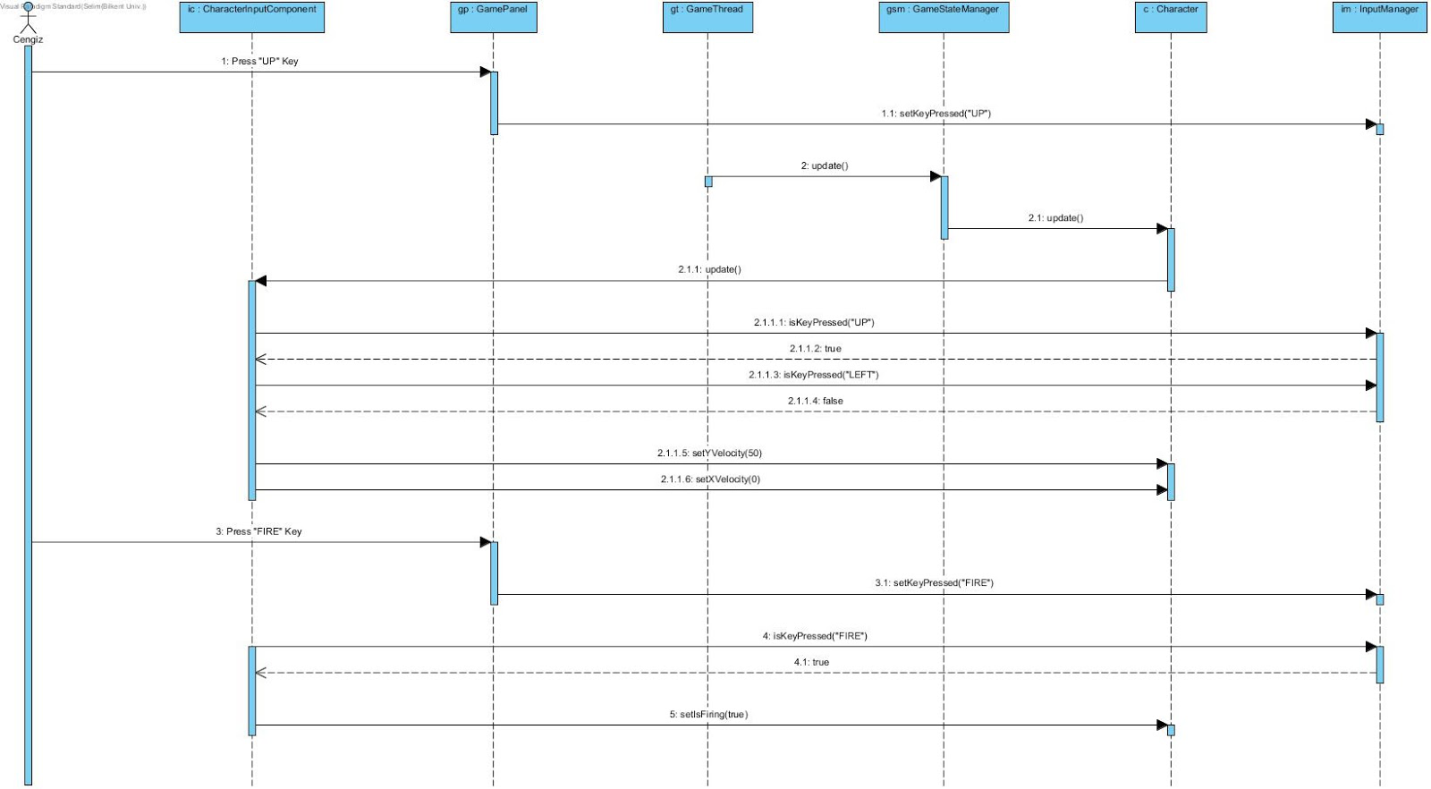


**Scenario:** Player Cengiz wants to move his character “UP”

If Cengiz wants to move his character in the direction of Up, he should press the “UP” key. “Game Panel” gets this movement desire and the “Input Manager” initialises the pressed key as “UP”. “Game Thread” updates and checks the “Game State Manager” and it also updates and checks “Character” in every 50 ms. The “Character Input Component” gets the required information from the “Input Manager” and sets the velocities with respect to given input and then updates the “Character”

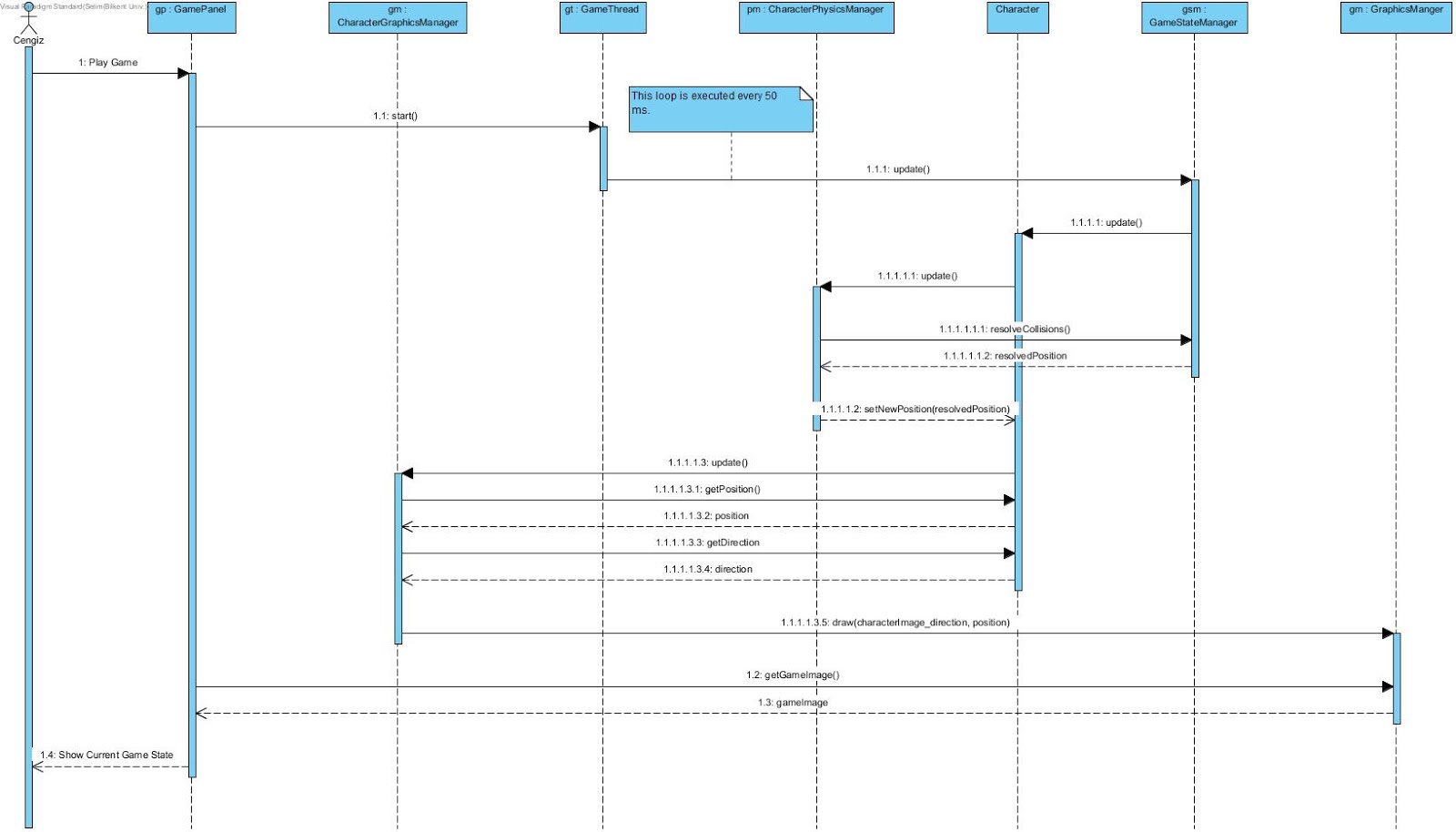
**Scenario:** Player Cengiz wants to fire

If Cengiz wants to fire, he should press the “FIRE” key. After pressing the “FIRE” key, “Game Panel” gets this fire desire and notifies the “Input Manager”. After the feedback from the “Input Manager”, “Character” of Cengiz starts to fire.



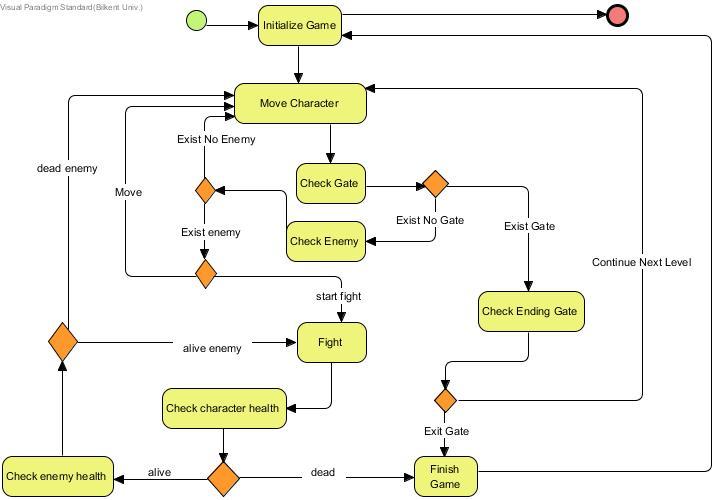
**Scenario:** Player Cengiz wants to see the current game state

If Cengiz wants to learn his current game state, he should have started playing the game. The “Game Panel” updates the “Game Thread” as started. “Game State Manager” is updated by “Game Thread” in every 50 ms. The process of updates from “Game State Manager” to “Character” and “Character” to “Character Physics Manager” are done. The positions of the “Character” is resolved and set. After that, the “Character Graphics Manager” is updated. The position and direction is getted. Finally, the game image and the current state of the game is displayed.

****

**4.2.3 Activity Diagram**

The activity diagram which is shown below is responsible for presenting running the mechanism of the game.



**4.2.3.1 Explanation of Activity Diagram**

Firstly, the player should have downloaded the game and in order to start playing it (s)he must have clicked the icon of the game and have chosen “Play The Game” option. In the beginning, the game is initialized. The player starts to move the main character. In every move, the system checks whether there is any gate or not.

* If there is no gate

The character keeps moving on and checking for the enemies.

If there is any enemy, the character has two options

1. First one is running away from the enemy without fighting with it.
2. Second one is the character may start to attack the enemy.

After the fight the health of the character is checked.

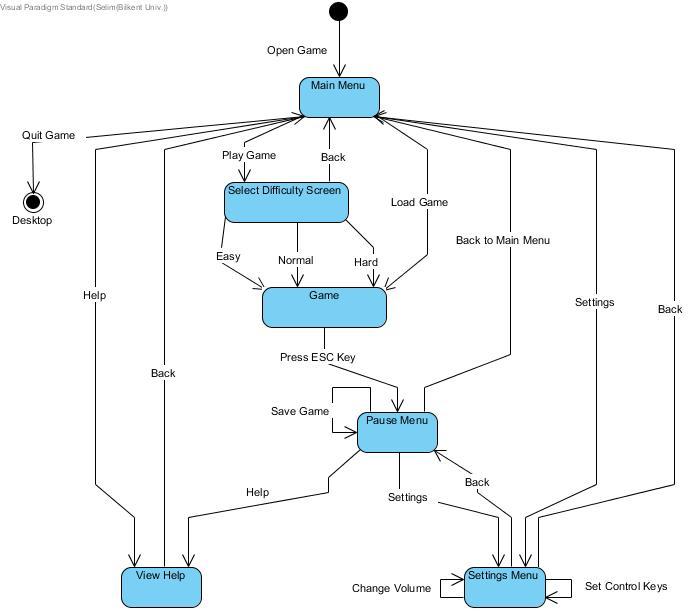
* If the health of the character is positive, the character keeps fighting or running away from thee enemy.
* If the health of the character is 0 or negative, the game is finished and the player goes back to the beginning of the game and starts to move on initializing game or exit.
* If there is a gate in front of the character

System checks whether the gate is the last gate or not.

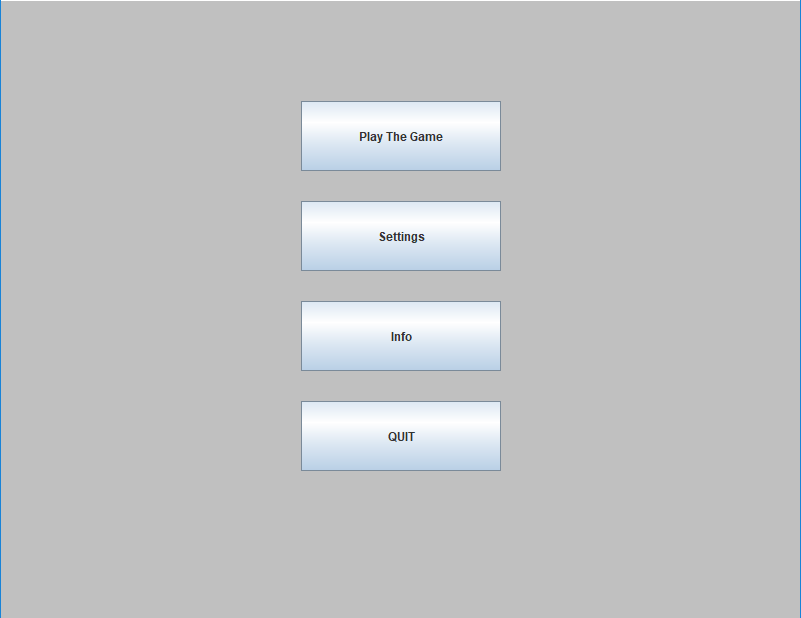
1. If the gate is not the last gate, the character passes through the next level and starts to move on.
2. If the gate is the last gate, the game is finished and the player goes back to the beginning of the game and starts to move on initializing the game or exit.

**4.2 User Interface – Navigational paths and screen mock-ups**

***4.2.1  Navigational Path***



***4.2.2 Main Menu***

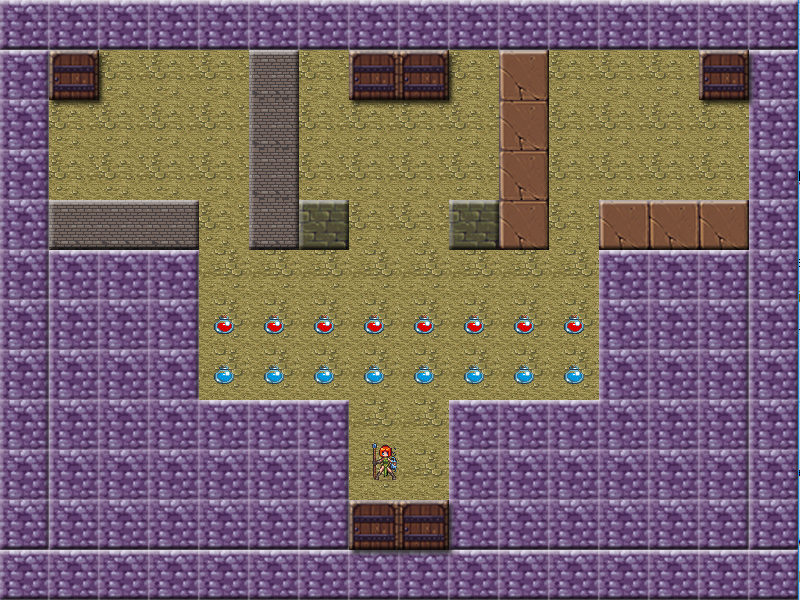


(Main Menu) [Design is low Priority]

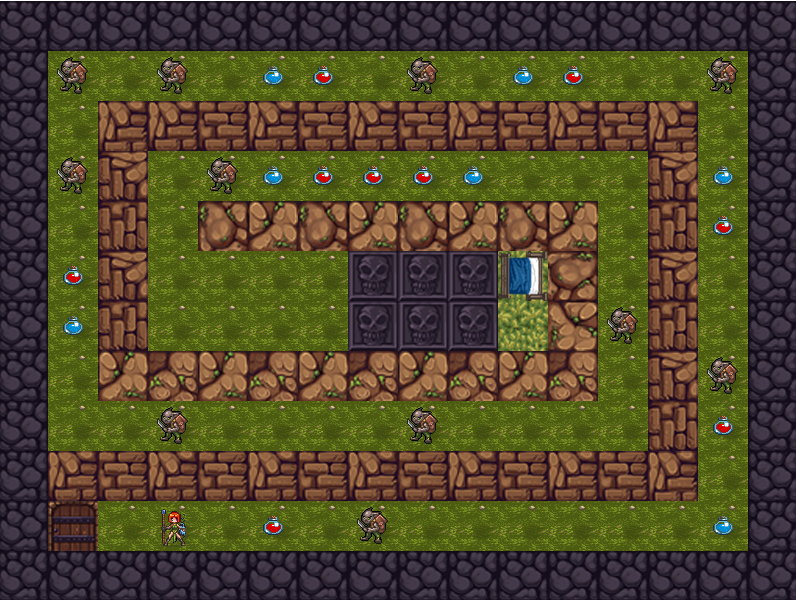
***4.2.3 Game Play***



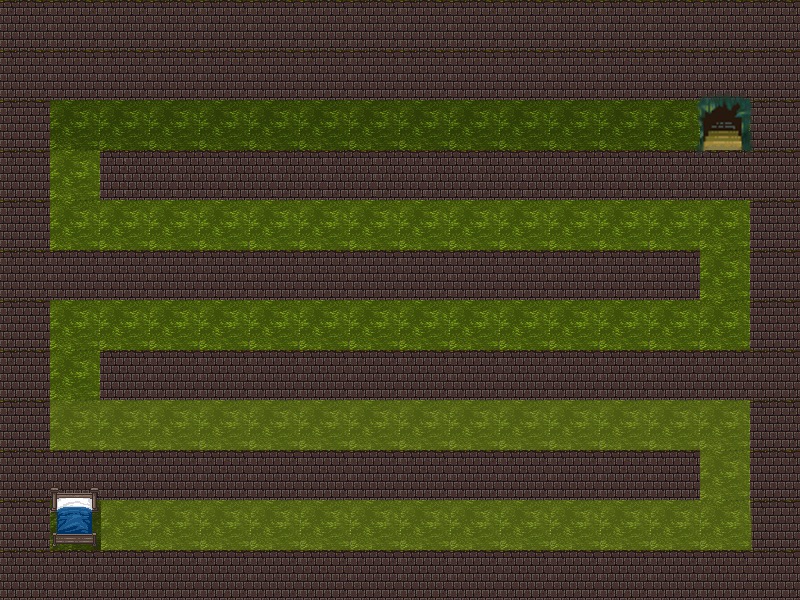
(Level 1) [High Priority]



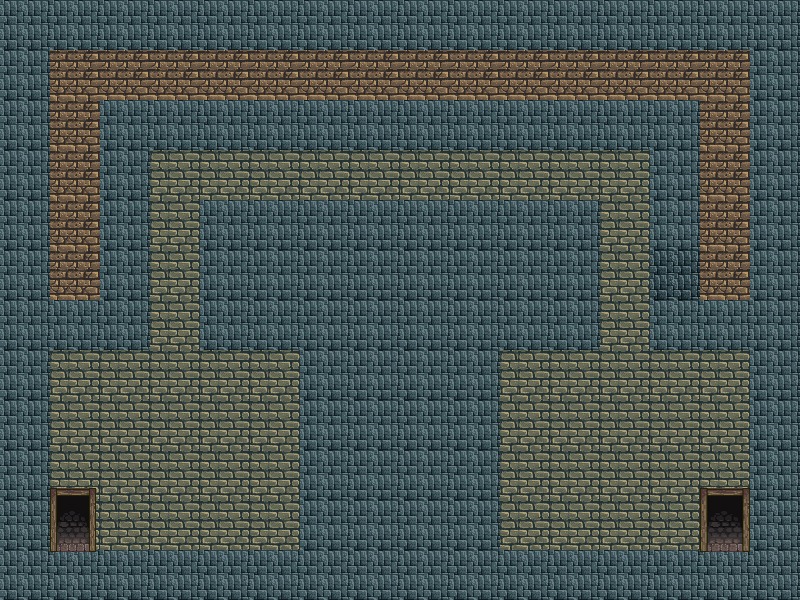
(Level 2) [High Priority]

****

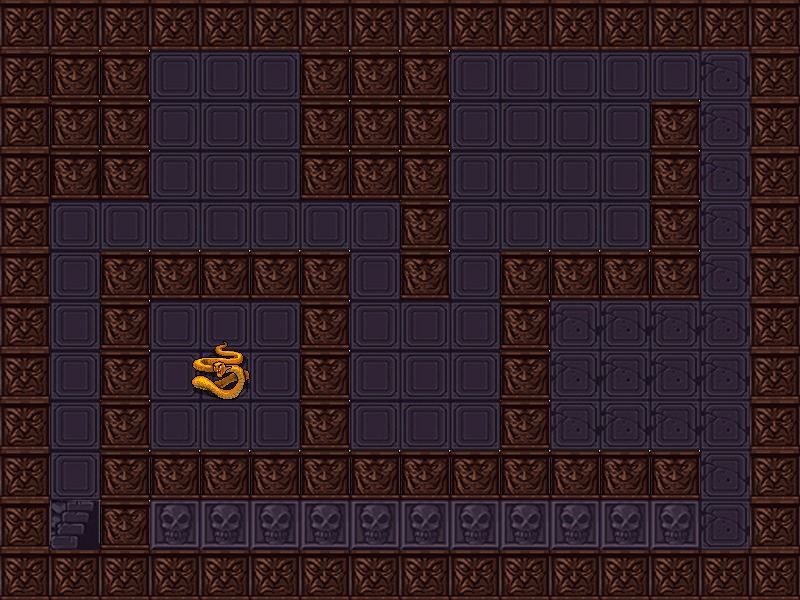
(Level 3) [High Priority]

****

(Level 4) [Medium Priority] (Mockup)

****

(Level 5) [Low Priority] (Mockup)

****

(Level 6) [High Priority] (Mockup)

# 5. Improvement Summary

* Every object have an update method. This method updates the graphic component in every 50ms so that this structure provides more smooth graphics.
* The attack system of the game is changed and improved. Previously game used to have quick time event based attack system but the gameplay wasn’t fluent enough. That lack of fluency made the game less interesting and challenging. New attack system allows the game become more smooth and playable.
* File system and file management totally changed. Previously the game had more complicated file system. New file system only responsible for getting the inputs such as txt files, png files, jpeg files and so forth.This provided less coupling and more coherence.
* In first iteration the game hadn’t save and load options. As it’s planned save and load system has been added.

Due to changes in the attack system, graphic update system, file system the use case, sequence, class and activity diagrams  and their descriptions have been changed and improved.

**6. Glossary**

**Pokemon FireRed**

A 2D roleplay game (RPG) released in 2004.

**Maven**

A software project management and comprehension tool. Based on the concept of a project object model (POM).

**QTE** (**quick time event**)

A method of [context-sensitive](https://en.0wikipedia.org/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvQ29udGV4dC1zZW5zaXRpdmVfdXNlcl9pbnRlcmZhY2U) [gameplay](https://en.0wikipedia.org/index.php?q=aHR0cHM6Ly9lbi53aWtpcGVkaWEub3JnL3dpa2kvR2FtZXBsYXk) in which the player performs actions on the control device shortly after the appearance of an on-screen instruction.

**7. References**

[1] “Pokemon FireRed research and video example of the game flow”

<https://www.pokemon.com/us/pokemon-video-games/pokemon-firered-version-and-pokemon-leafgreen-version/>

<https://www.youtube.com/watch?v=yUS1IcC5CBY>

[2] “Research about Maven”

<https://maven.apache.org/>

[3] “Quick Time Event research and one game project example from MIT”

<https://www.giantbomb.com/quick-time-event/3015-6/>

<https://scratch.mit.edu/projects/88806129/>