 Bilkent University

Department of Computer Engineering

Object Oriented Software Engineering Project

CS 319 Project: Civilizational Wars

Analysis Report

Project members:

• Fuad Aghazada \_ 21503691

• Seyfullah Yamanoğlu \_ 21400697

• Bayram Muradov \_ 21503664

• Berk Erzin \_ 21201516

Supervisor: Uğur Doğrusöz

Progress : November17, 2017

Contents

1. Introduction 1

2. Proposed System 1

… 1 Overview 1

1.1 Controls 2

1.2 Character 2

1.3 Enemies 3

… 1.3.1 Bosses 3

1.4 Bonuses 3

1.5 Maps 4

1.6 Game states 4

… 2.Functional Requirements 4

…. 3.Non-functional Requirements 5

. . 4.Pseudo Requirements 5

3. System Models 5

3.1 Use case diagram 6

3.2 Use case descriptions 6

3.3 Class diagram 15

3.4 Dynamic diagram 15

…. 3.4.1 Sequence diagrams 15

.... 3.4.2 Activity diagram 21

3.5 User Interface 21

4. What is new? 26

4. Glossary 26

5. References 26

Analysis Report

CS 319 Project: Civilizational Wars

# Introduction

This project will be a 2D platform game. There will be some main characters, with which the player should be able to pass different levels of the game by controlling them. Different levels will have distinctive maps, where the player will encounter with some different enemies and fight them in order to achieve the winning state of each level.

By writing this report as a group we aimed to convey some general and detailed information about the analysis of this game, specifically, gameplay, game elements, functional, non-functional requirements, and system model diagram.

# Proposed System

## Overview

The game will consist of 3 distinctive maps, each of which will represent different periods of timeline. The aim of playing this game is to finish all the levels, without losing all lives given at the beginning of the game. While playing the game according to his/her gameplay, in order to motivate the player there will be some surprise boxes that contain different functional equipment like shield, health, weapon and etc. However, it will not be easy to finish the level, even if the player has come to the end of the map, because there will be a boss enemy, which will have considerably much powerful techniques of fighting and a more health level than the typical enemies encountered during the level.

* 1. **Controls**

In order to supply interaction between the player and the game, mainly keyboard will be used as an input device. (in some cases, mouse could also be used) The controls will be denoted on the help section of the game, which will be appear on the menu screen. Player will be also able to change the default section to his/her own accustomed controls on the keyboard, which will be done on the settings section of the game.

* 1. **Character**

Like most 2D Platform games, in this project there will be characters to control.

* Character will be able to move in 2 directions: left and right. Physics of characters will also include jumping, fighting and etc.
* There will 3 different character for different levels: fighter from medieval age, soldier from modern age, and robotic soldier from post-modern age.
* Each character will have different fighting style and weapons - lance, rifle (carbine) and laser gun, each of which will have different damage on the enemies.
* Characters will have health limit: at the beginning, current character will be given default number of health bars. After losing all of the health bars, game state of losing will come true. Nevertheless, health bars can be found from the surprise boxes, which will be analyzed in the next sub-sections of the report.
  1. **Enemies**

Enemies will be the game objects, which will try to prevent the main character to finish the maps.

* There will be distinct types of enemies in each level, like soldiers, tanks spaceships, respectively for the levels of the game.
* Like the characters, enemies will also have health level, which will depend on the type of the enemy. To exemplify, a tank will considerably much health level than an enemy soldier.
* Enemies will also have different size of damage on the player. Each of enemies will have different damage interval so that the given damage to the character will be a random number from this interval.

**1.3.1 Bosses**

In order to finish the level, the player will also have to beat the boss of the level. Bosses will be also enemies, but with some additional features:

* Bosses will have considerable higher health level than other enemies so that it will take time and strategy to defeat them.
* Bosses will be able to attack the character in higher damage sizes.
  1. **Bonuses**

Every player would like to get bonuses while playing the games. Therefore, there will be surprise boxes:

* Contain random items like health bar, more damage level for weapons and shield (non-permanent).
* Will appear in random times and random locations on the maps.
  1. **Maps**
* The game will contain 3 different maps, each of which will denote distinctive period of timeline: medieval, modern and post-modern ages.
* Timelines are going to include ancient, modern, and postmodern ages.
* In the game, there will be characters with the matching specifications to each timeline as mentioned before.
  1. **Game states**
* The first game state, the user will observe, will be the menu state, in which there will be options like play, load game, settings and so on (analyzed in the next section of the report)
* Player will lose, in case he/she loses all the lives given and obtained during the game play.
* Player will win (level), if he/she can defeat boss at the end of each level.
* Pause of the game will also be added as a state for the game.

## Functional Requirements

* The player will be able to control the game character using some hotkeys from the keyboard.
* In the settings of the game:

- the control hotkeys can be modified for providing comfort for the player

- difficulty of the level can be changed

- music of the game can be turned on or off

* The game can be paused using the pause button on the game screen.
* The levels can be saved so that the player will be able to continue from the level he/she left the game by loading it from the game menu.
* By choosing the help section on the menu, the player can get information about how to play the game - default controls of the game.
* In the about part, there will be references and the names of the group members who participated in the project.
* There will be moving map objects, which will mix some more adventure to the game. (moving tile blocks) (new)
* By choosing the multiplayer option, two players will be able to play from the same device together. (new)

## Non-functional Requirements

* The gameplay will be easy to adapt.
* The source code of the project and documentations will be sufficiently readable and easy to learn.
* The graphics will be smooth as possible.
* The game itself will be sufficiently responsive with small delays.
* Playing the game, the player could memorize the his/her childhood games.
* AI for the enemy objects will be improved. (new)

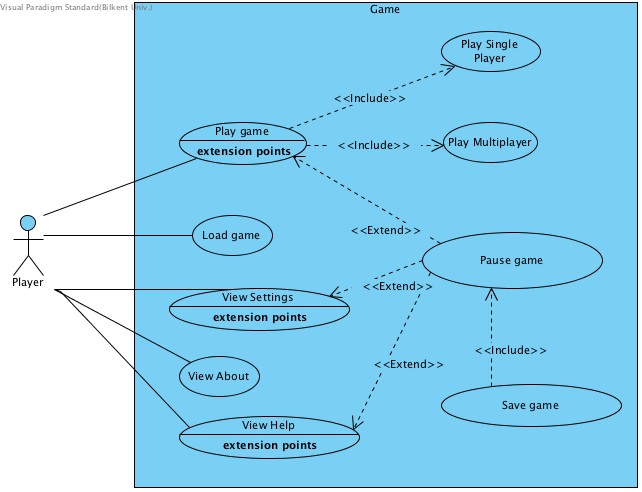
## Pseudo Requirements

1. The game will be implemented using Java swing library.
2. Some textures will be created by Adobe Photoshop and other design programs

## 

## 3. System Models

1. Use-case model



Picture 1. Use case diagram of the game

### Use-case scenarios

*Use-case 1*

**Use-case name:** Play single player

**Participating actors:** Player

**Entry condition:**

* Player clicks the button “Play Game” on UI

**Exit condition:**

* Player managed to finish the level.
* Player has failed the level.
* Player on purposely exit during the game.

**Main flow of events:**

1. Application starts.
2. Menu of the game appears.
3. Player clicks the button “Play Game”
4. The game starts.

**Alternative flow of event:**

**#1:**

1. Player wants to change settings of the game
2. Makes required changes in settings panel
3. Returns back to the main menu and starts the game.

**#2:**

1. Player wants to view the help page to get informed about how to play and other details.
2. Returns back to the main menu and starts the game.

*Use-case 2*

**Use-case name:** Pause game

**Participating actors:** Player

**Entry condition:**

* The game has already started and player continues playing

**Exit condition:**

* Player quit the game.
* Player changed the settings and returned back to game
* Player viewed the help page and returned back to game
* Player just paused to take a break and after a while returned back to game

**Main flow of events:**

1. Application starts.
2. Main menu appears.
3. Player clicks the “Play Game” button.
4. During the game player wants to pause the game so that s/he can take a break, change the settings, view the help page or save the game.

**Alternative flow of event:**

1. Player finished the level without any pause.

*Use-case 3*

**Use-case name:** Save game

**Participating actors:** Player

**Entry condition:**

* Player started the game and paused at any point

**Exit condition:**

* Player saves the game at that point of pause

**Main flow of events:**

1. Application starts.
2. Main menu appears.
3. Player starts the game.
4. Plays for a while and pauses the level.
5. Clicks the “Save game” button in the pause menu to save the level at that point of pause, regarding player’s position, weapon, enemy’s positions, game time, etc.

**Alternative flow of event:**

1. Player doesn’t save the game during the whole gameplay.

*Use-case 4*

**Use-case name:** Load game

**Participating actors:** Player

**Entry condition:**

* Player launches the game and waits on the menu

**Exit condition:**

* Player loads the level he passed before and starts the game.

**Main flow of events:**

* + - 1. Game launches.
      2. Menu of the game appears with the options.
      3. Player presses the “Load game” button.
      4. Loaded maps of the game appears on a new panel.
      5. Player selects the level he left.
      6. Selected level is constructed.
      7. Player starts the game.

**Alternative flow of event:**

Player has not passes any level

Since “Load game” button is inactive, Player cannot choose this option.

*Use-case 5*

**Use-case name:** Settings

**Participating actors:** Player

**Entry condition:**

* Player starts the game and waits on the game menu screen.

**Exit condition:**

* Player did the needed configurations and returns back to the main menu by applying the changes, OR.
* Player did no configurations and returns back to the menu.

**Main flow of events:**

* + - 1. Game launches.
      2. Menu screen appears.
      3. Player chooses the button with the label of “Settings”
      4. Player did the necessary changes on the settings.
      5. The changes are applied to the game system, so that it will not be lost.
      6. Player returns back to the main menu.

**Alternative flow of event:**

* The default settings are remained as the same.

*Use-case 6*

**Use-case name:** View Help

**Participating actors:** Player

**Entry condition:**

* Player starts the game and waits on the game menu screen.

**Exit condition:**

* Player did observe the instructions for the game in the Help panel, and returned back to the main menu.

**Main flow of events:**

1. Game launches.

2. Menu screen appears.

3. Player chooses the button with the label of “Helps”

4. Player learns the game instructions.

5. Player returns back to the main menu.

**Alternative flow of event:**

* Player plays the game without getting help from this section.

*Use-case 7*

**Use-case name:** About

**Participating actors:** Player

**Entry condition:**

* Player starts the game and waits on the game menu screen.

**Exit condition:**

* Player looks through the additional information about the game and returns back to the menu.

**Main flow of events:**

1. Game launches.

2. Menu screen appears.

3. Player chooses the button with the label of “About”

4. Player looks through the info in the entered section.

5. Player returns back to the main menu.

**Alternative flow of event:**

* The default settings are remained as the same.

*Use-case 8*

**Use-case name:** Play multiplayer

**Participating actors:** Player1, Player2

**Entry condition:**

* Player1 or Player2 clicks the button “Play Game” on UI
* In the next panel, one of them clicks “Multiplayer” button

**Exit condition:**

* At least one of the players managed to finish the level.
* Both of the players have failed during the level.
* One of the players on purposely exits during the game.

**Main flow of events:**

1. Application starts.

2. Menu of the game appears.

3. One of the players clicks the button “Play Game”

4. One of the players clicks the button “Multiplayer”

5. The game starts.

**Alternative flow of event:**

**#1:**

1. One of the players want to change the settings of the game

2. Make required changes in settings panel

3. Return back to the main menu and start the game.

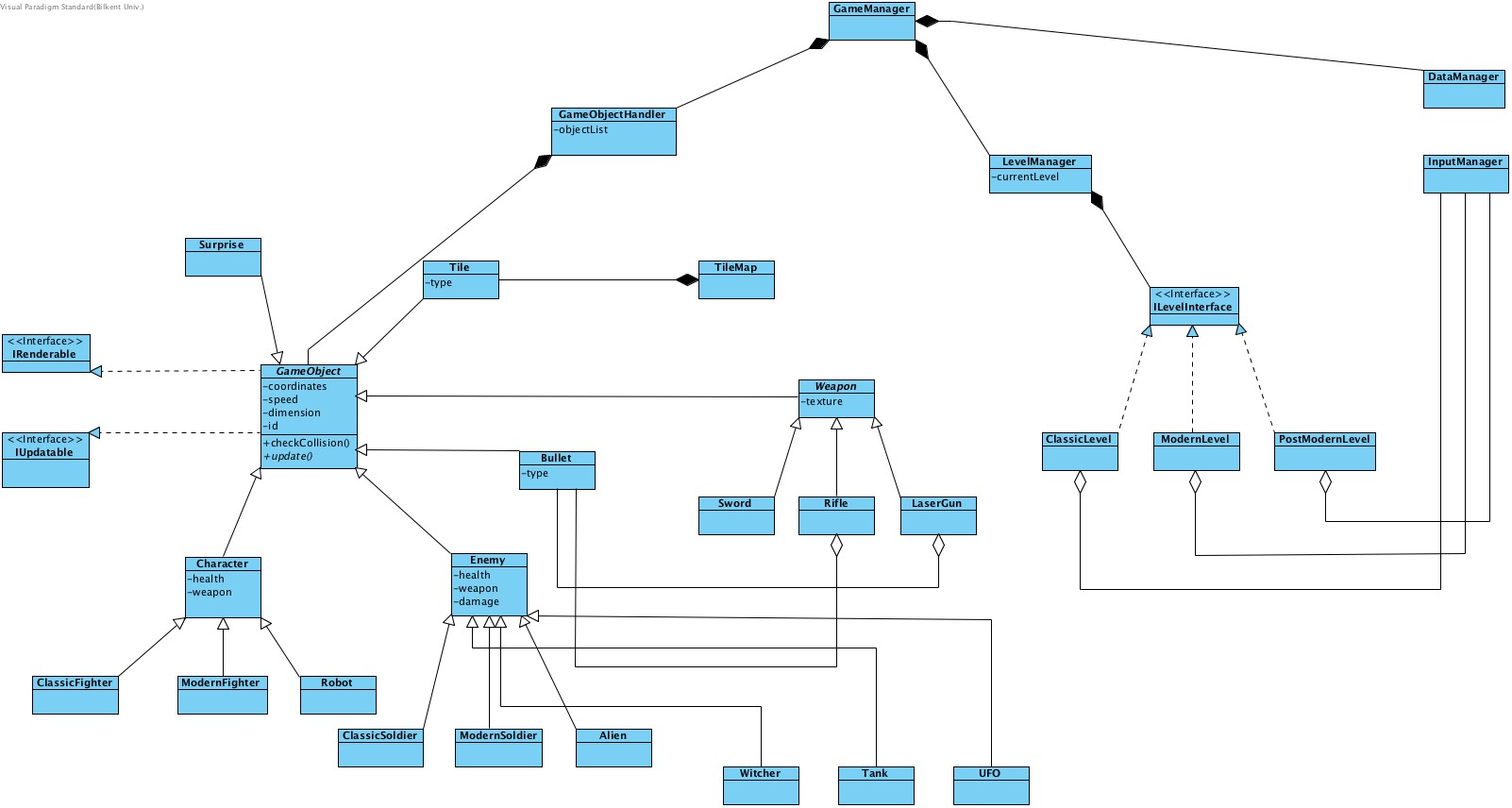
**#2:**

1. Players want to view the help page to get informed about how to play and other details.

2. Return back to the main menu and starts the game.

### Object and Class Model

* This is the rough object model of the project. [2]



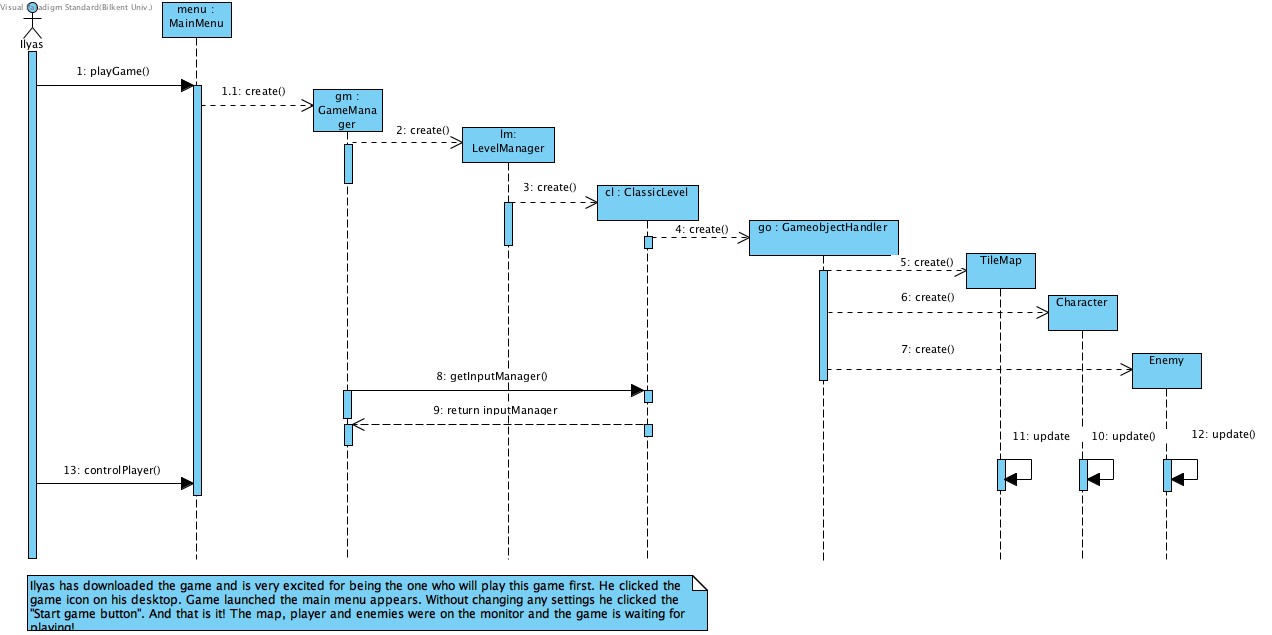
Picture 2. Class diagram of the game

### Dynamic Models

* 1. Sequence diagrams

**1. Scenario name:** Starting the game

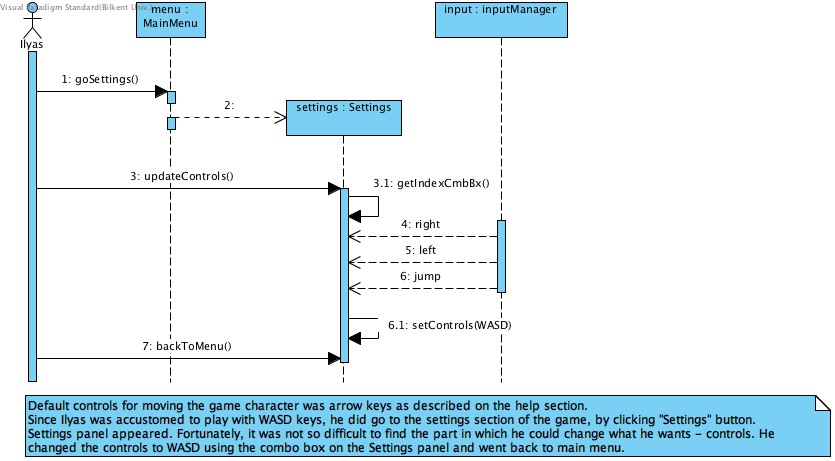
**Scenario description:** Player - Ilyas has downloaded the game and is very excited for being the one who will play this game first. He clicked the game icon on the desktop. Game launched and the main menu appeared. Without changing any settings, he clicked “Start game” button. And that is it! The map, player and enemies were on the monitor and the game is waiting for being played. (Picture 3)



Picture 3. Sequence diagram for staring the game.

**2. Scenario name:** Changing default moving controls

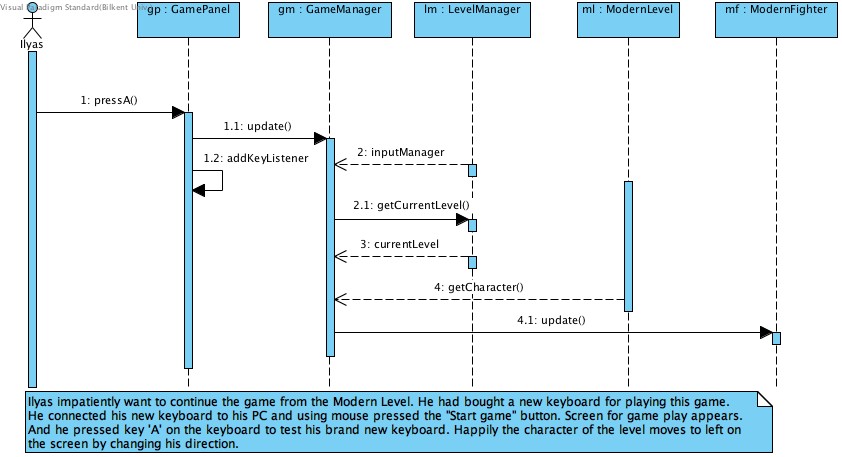
**Scenario description:** Default controls for moving the game character was arrow keys as described on the help section. Since Ilyas was accustomed to play with WASD keys, he did go to the settings section of the game by clicking “Settings” button. Settings panel appeared. Fortunately, it was not so difficult to find the part in which he could change what he wants – controls. He changed the controls to WASD using the combo box on the Settings panel and want back to main menu. (Picture 4)



Picture 4. Sequence diagram for changing controls scenario

**3. Scenario name:** Moving the character (to left)

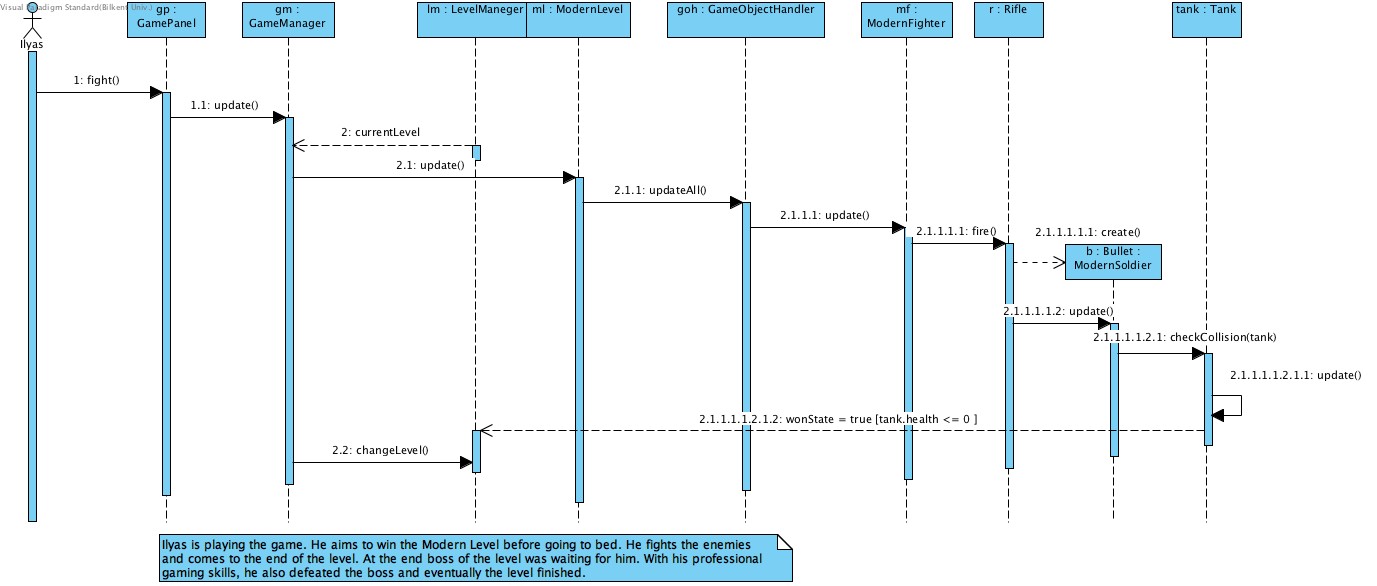
**Scenario description:** Player – Ilyas impatiently want to continue the game from the Modern Level. He had bought a new keyboard for playing this game. He connected the new keyboard to this PC and using mouse clicked the “Start game” button on the main menu. Screen for game play appears. And he pressed key “A” on the keyboard to test his brand-new keyboard. Happily, the character of the level moves to his left on the screen by changing his default direction. (as default character’ direction is to right) (Picture 5)



Picture 5. Sequence diagram for moving the character

**4.Scenario name:** Winning the level

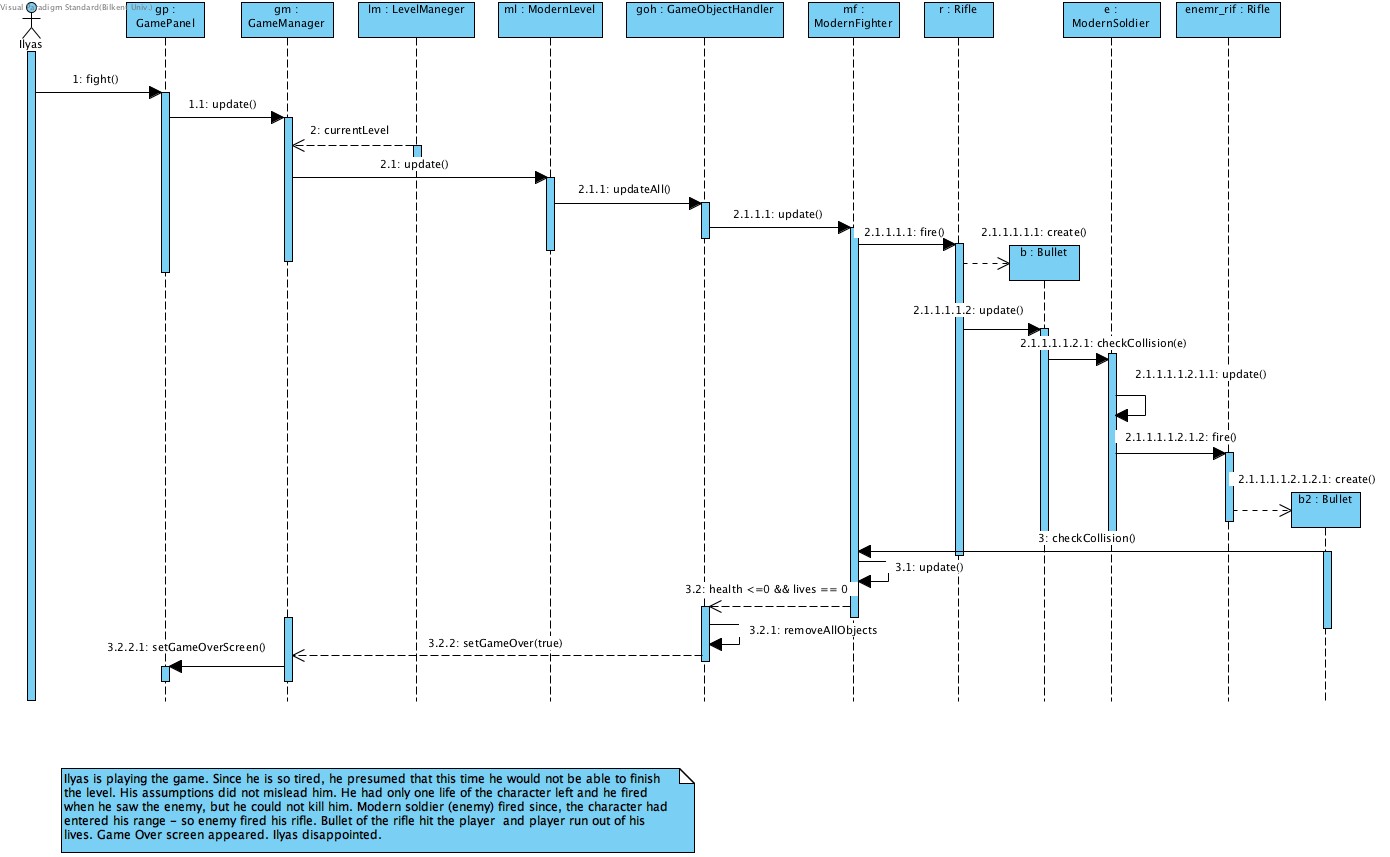
**Scenario description:** Ilyas is playing the game. He aims to win the Modern Level before going to bed. He fights the enemies and comes to the end of the level. At the end boss of the level was waiting for him. With his professional gaming skills, he also defeated the boss and eventually the level finished. (Picture 6)



Picture 6. Sequence diagram for winning the level scenario

**5.Scenario name:** Losing the game.

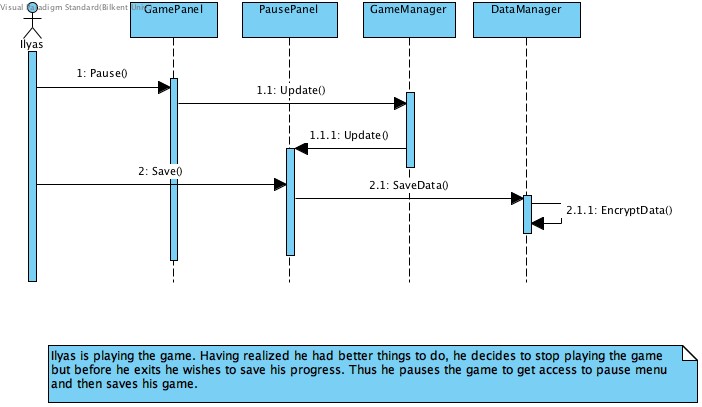
**Scenario description:** Ilyas is playing the game. Since he is so tired, he presumed that this time he would not be able to finish the level. His assumptions did not mislead him. He had only one life of the character left and he fired when he saw the enemy, but he could not kill him. Modern soldier (enemy) fired since, the character had entered his fire range – so enemy fired his rifle. Bullet of his rifle hit the player and player run out of his lives. Game Over screen appeared. Ilyas disappointed. (Picture 7)



Picture 7. Sequence diagram for losing the game scenario

**6.Scenario name:** Saving the game.

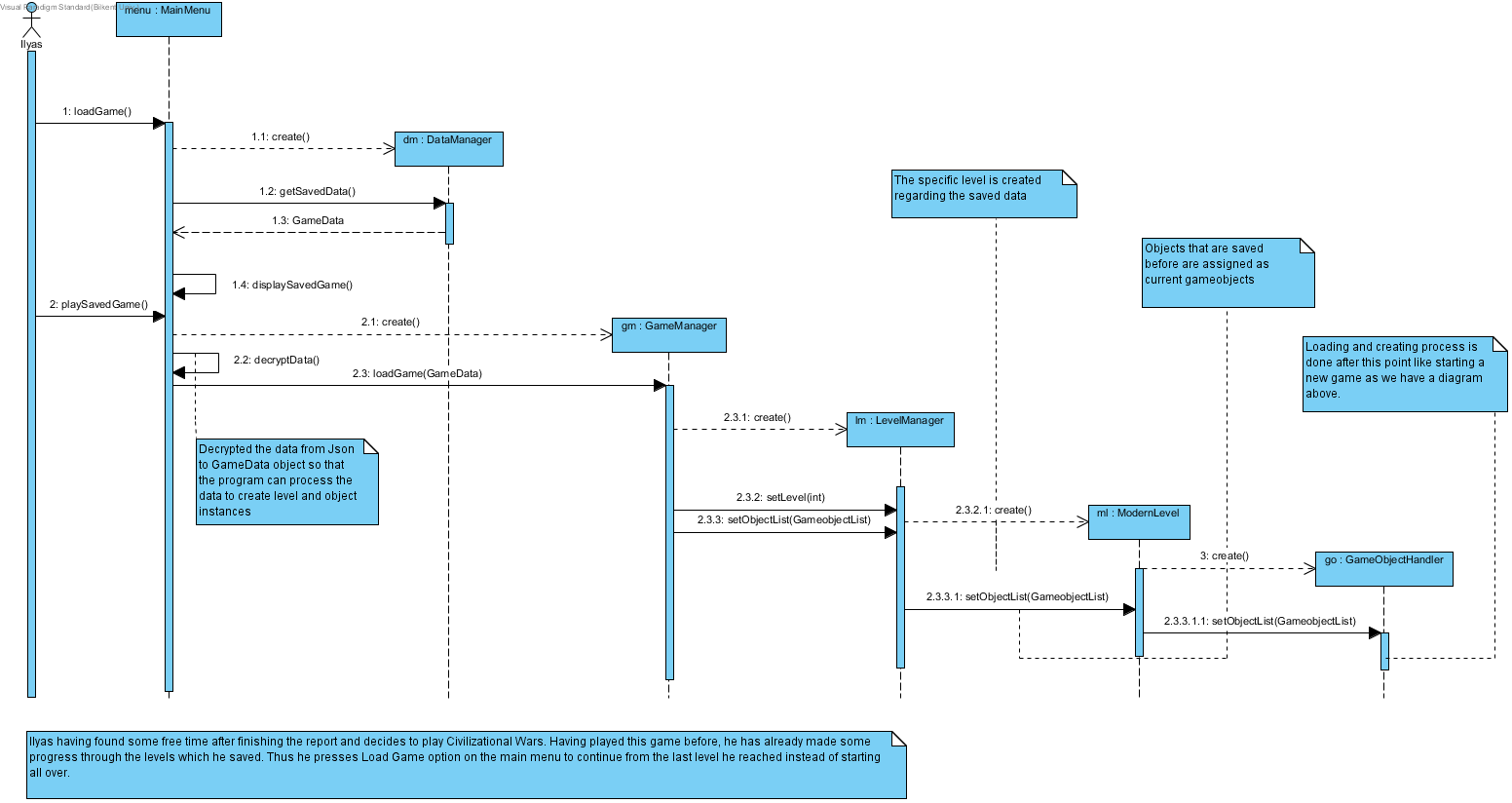
**Scenario description:** Ilyas is playing the game. Since he is so tired, he will go to bed soon, but before that he wants to save the game, because he did not want to start the part he had already played. While the game play, he pressed the pause button and pause screen appeared. In this screen, there are some options to select. (resume, save game, settings) He clicked the save game option and the game was saved. (Picture 8)



Picture 8. Sequence diagram for saving game scenario

**7.Scenario name:** Loading the game.

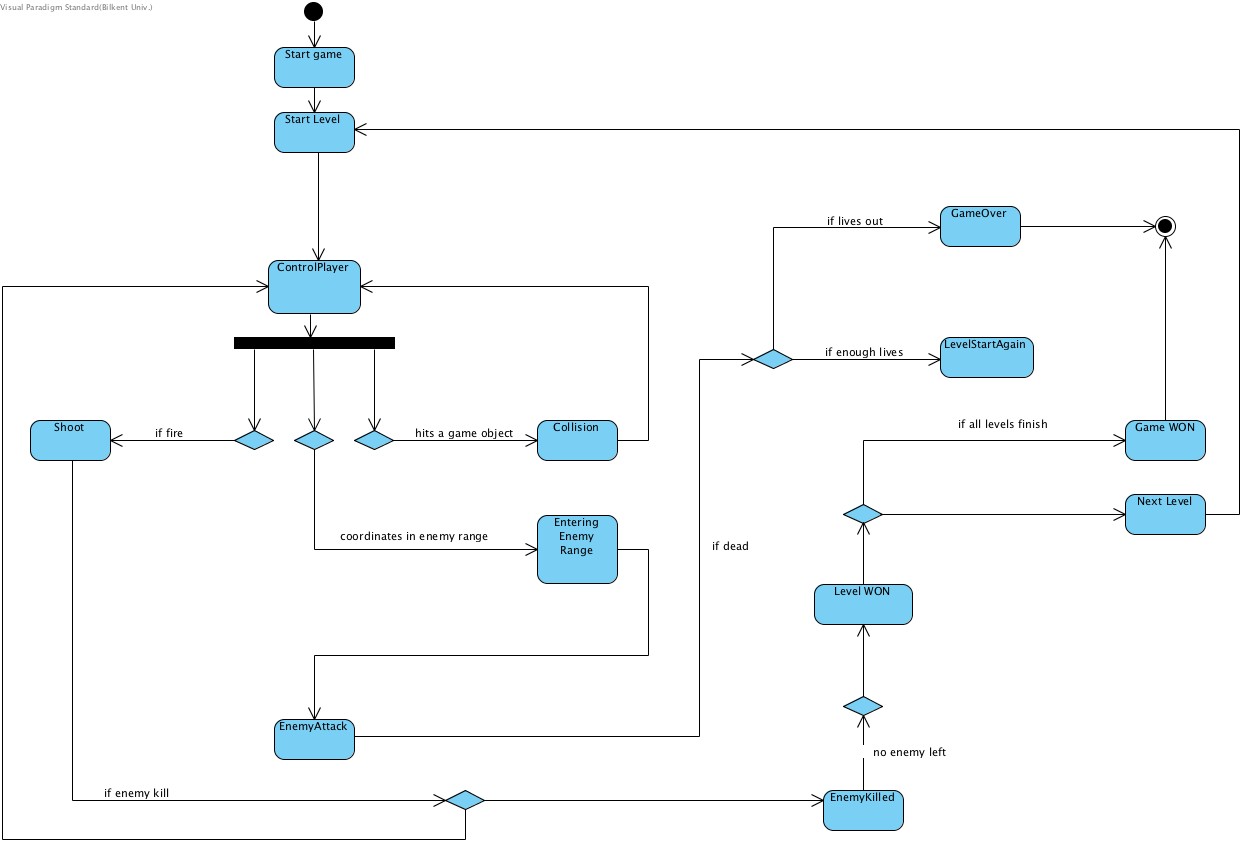
**Scenario description:** Ilyas had found some free time after finishing the report and decided to play Civilizational-Wars. Having played this game before, he has already made some progress through the levels which he saved. Thus he pressed “Load game” button on the main menu to continue from the last level he reached instead of starting all over. (Picture 9)



Picture 9. Sequence diagram for loading the level scenario

* 1. Activity diagram

This diagram shows the basic activities during the game play, such as figting with the enemies, got killed by them, winning the game or level, losing the game. (game over state)

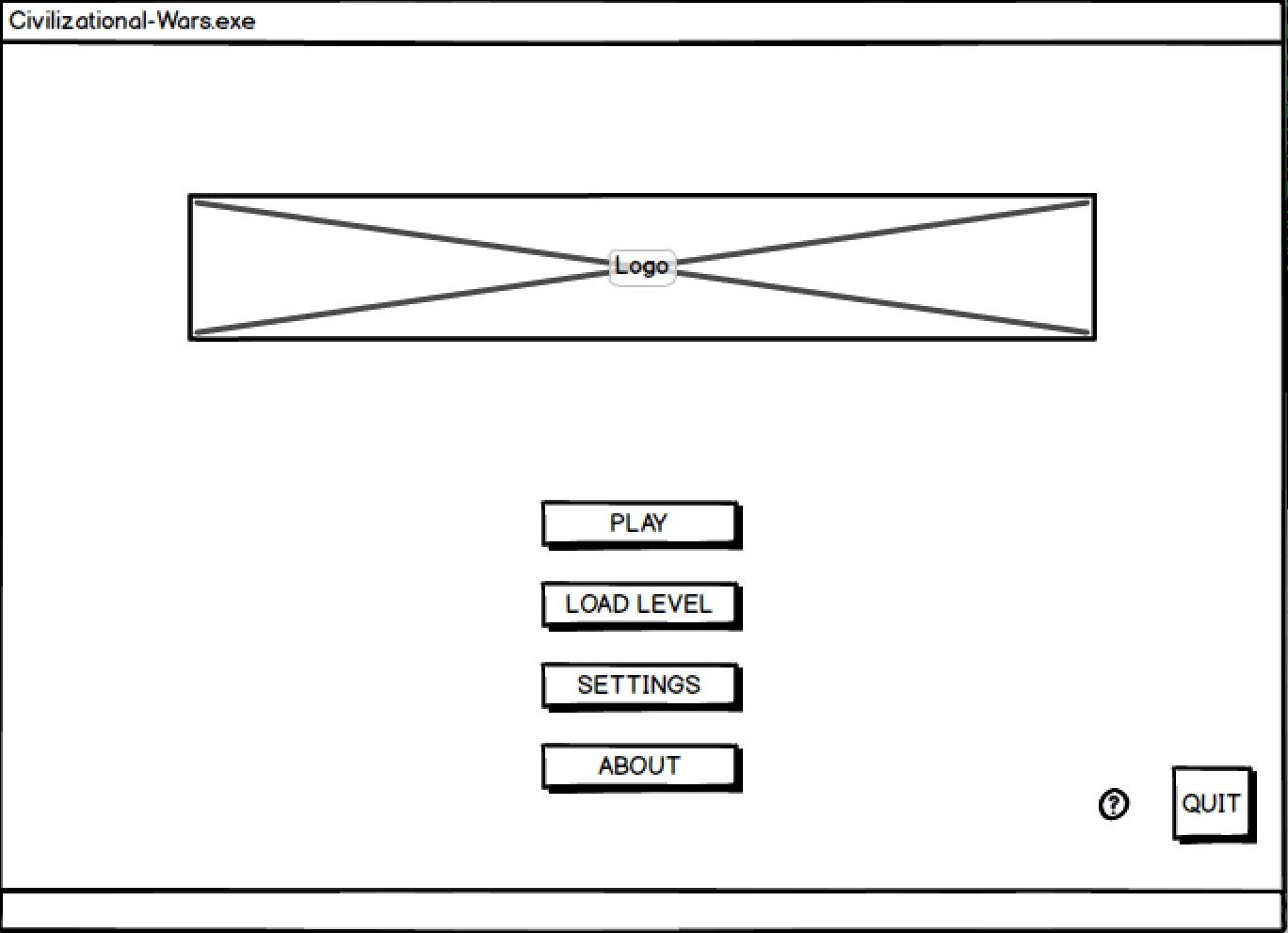


Picture 10. Activity diagram for the basic game activities

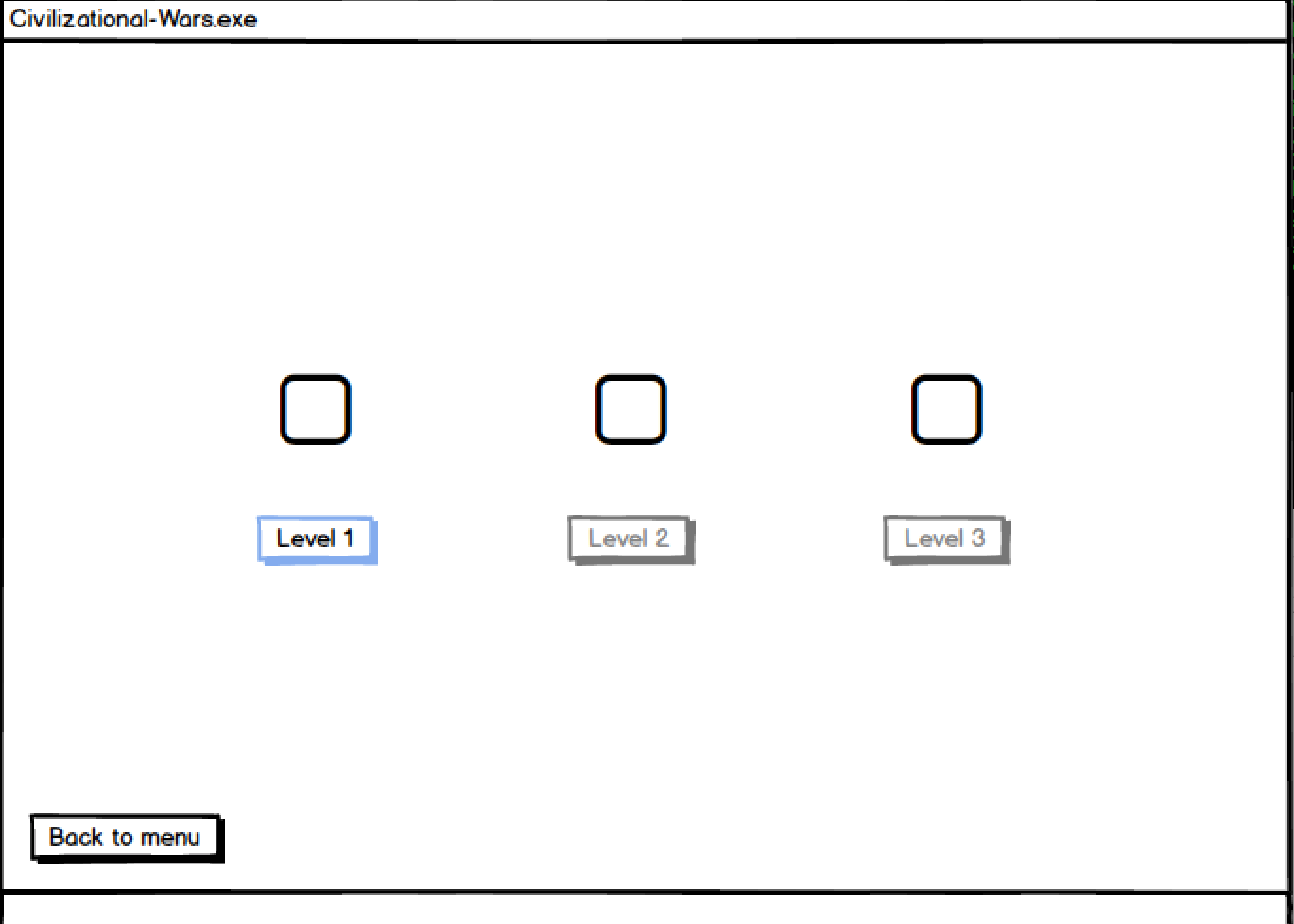
### User Interface

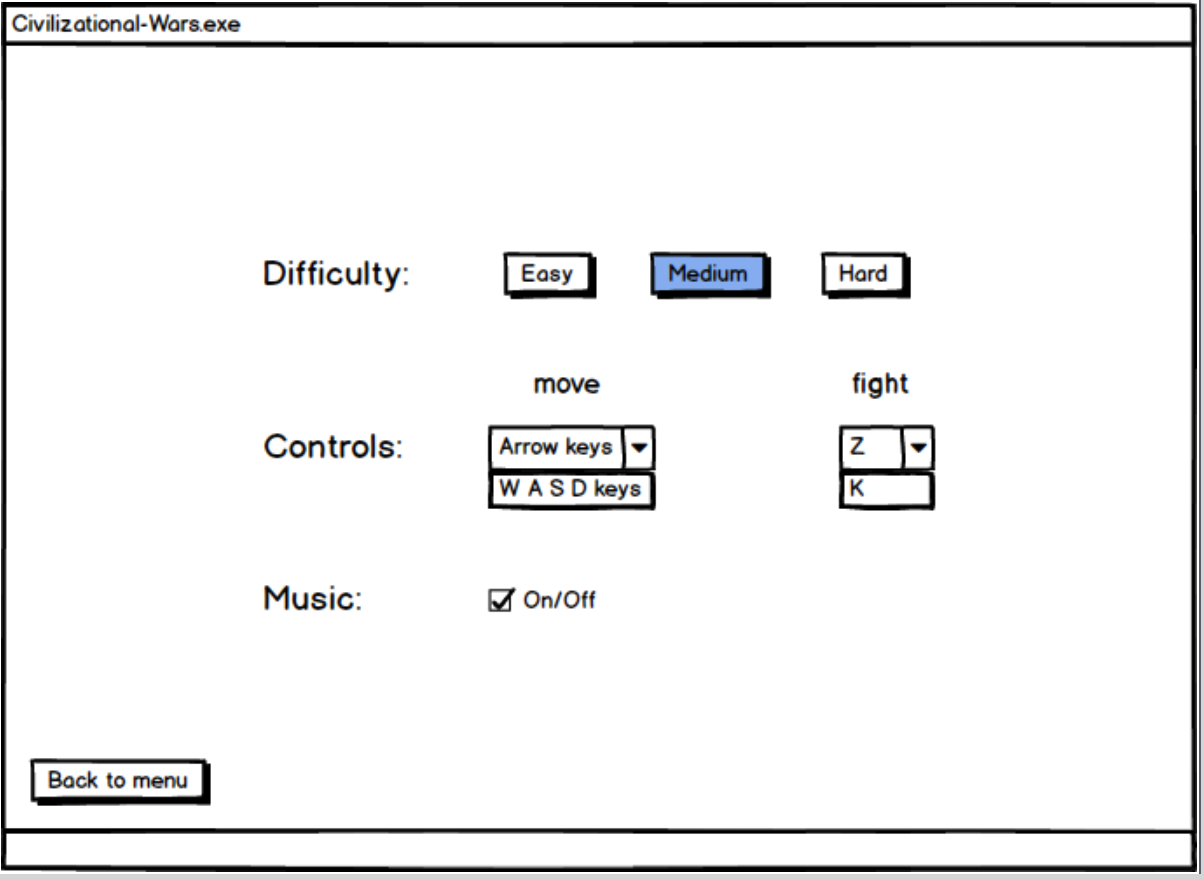
* These are some mock-up screens of the different states of the game. [3]

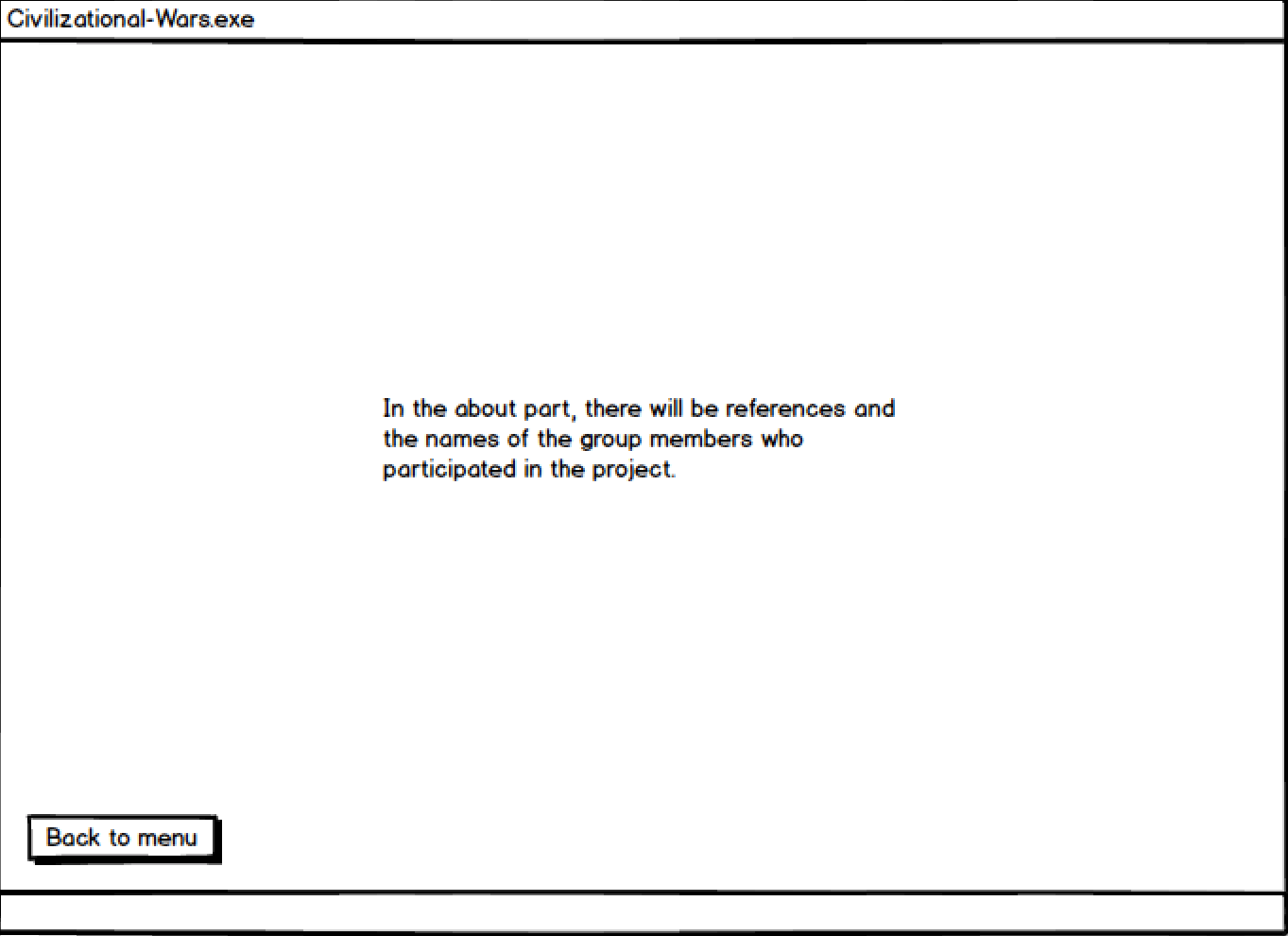
**NOTE**: These are just rough mock-up designs for the game. As the project goes on the progress, some textures for each part of the game screens will be added.

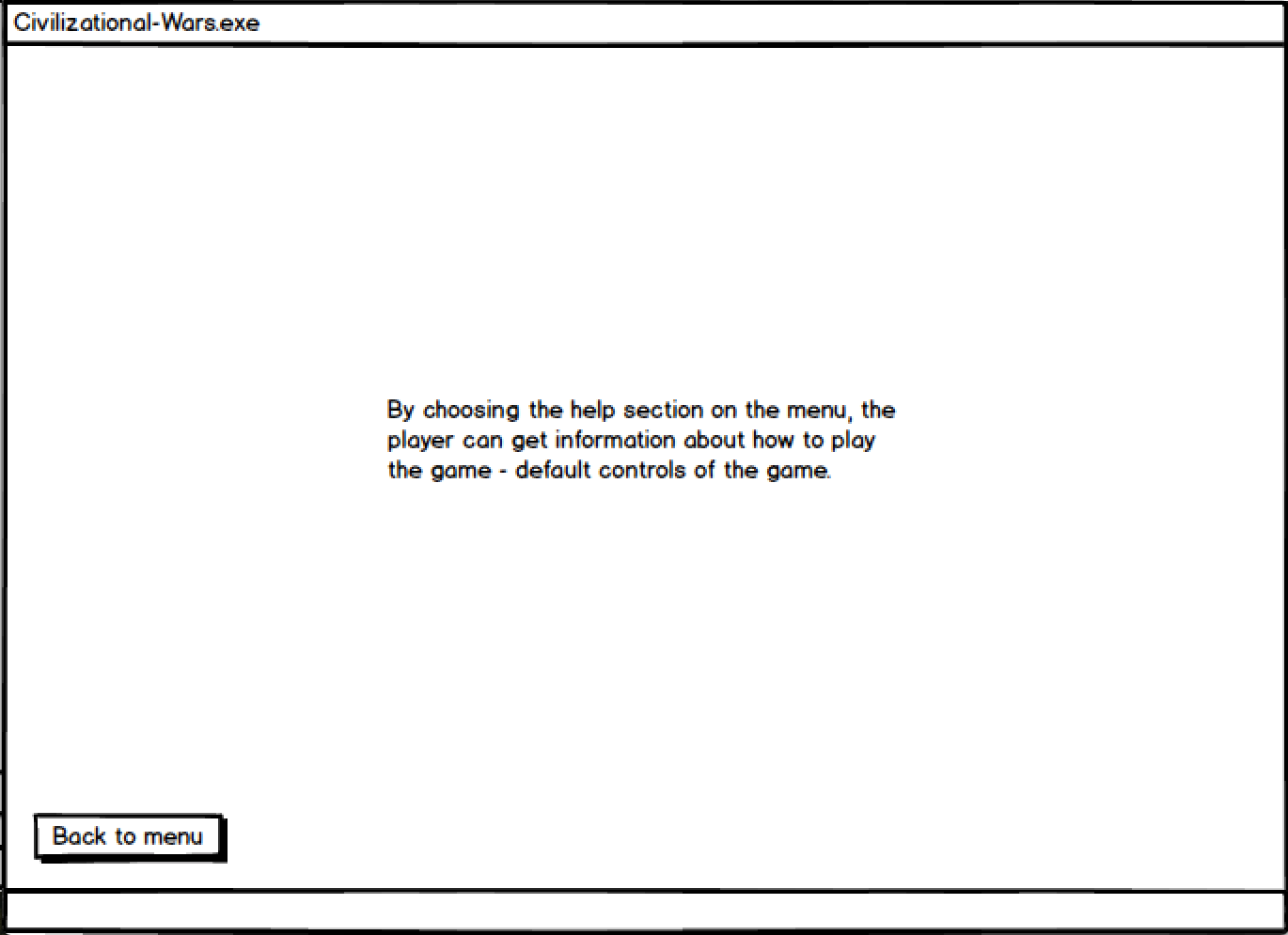
**Main Menu: **

**Play:**

**Load Level: **

**Settings:**

**About: **

**Help:**

# What is new?

Functional requirements

* Player can now play with his friends by choosing the multiplayer option of the game. (Picture 1 + Use case description 8)
* Moving map object will be added.

Non-functional requirements

* Improving the readability of the documentations and the source code.
* Improvements on the enemy AI.

Class diagram

* Added more classes and improved the interactions among them. (Picture 2)

Dynamic diagrams

* Added 5 more sequence diagrams for the scenarios – moving the character, changing the moving controls from the settings, saving the game, loading the game, starting the game. (Picture 3-9)
* Added an activity diagram for basic activities of the game system. (Picture 10)

# Glossary

* Mockup - a model or replica of a machine or structure, used for instructional or experimental purposes.

# References

1.Reference game: <http://www.maxgames.com/game/age-of-war-2.html>

2.Visual Paradigm – for diagrams

3.Balsamiq – for mock ups