 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: October 9, 2017

Contents

1. Introduction 1

2. Proposed System 1

1. Overview 1

2. Functional Requirements 4

3. Non-functional Requirements 5

4. Pseudo Requirements 5

System Models 5

2. Use-case scenarios 6

3. Object and Class Model 13

4. Dynamic Models 13

5. User Interface 14

4. Glossary 17

5. References 17

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.

## Non-functional Requirements

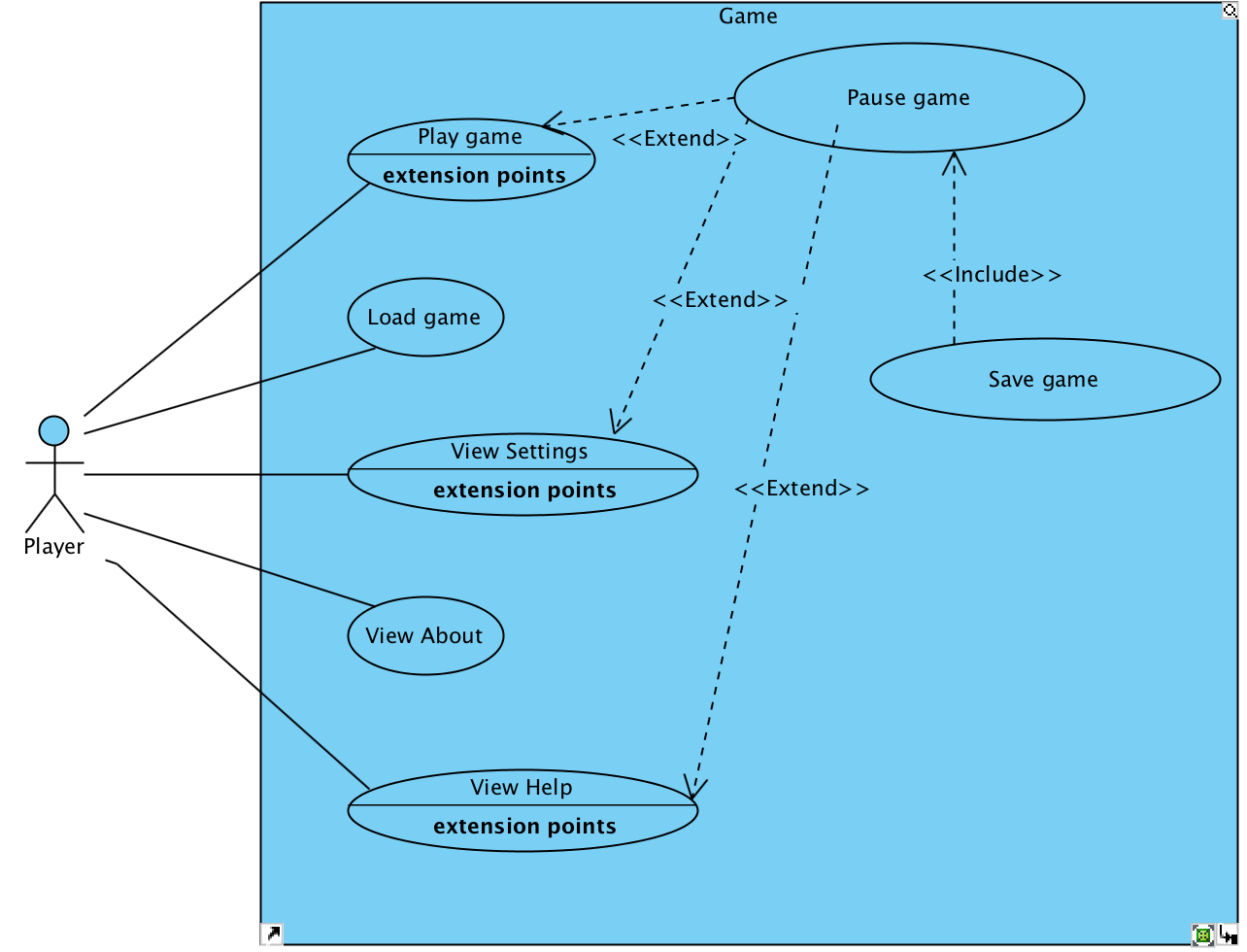
* The gameplay will be easy to adapt
* The graphics will be smooth as possible
* The game itself will be sufficiently responsive with small delays.

## Pseudo Requirements

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

## System Models

1. Use-case model



### Use-case scenarios

*Use-case 1*

**Use-case name:** Play game

**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.

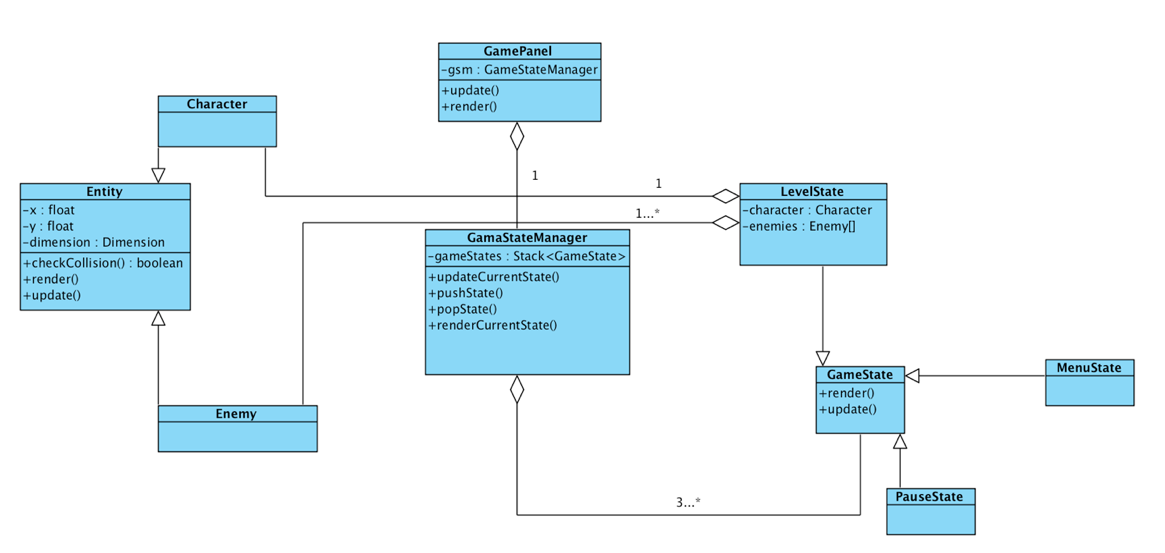
5. Player returns back to the main menu.

**Alternative flow of event:**

* The default settings are remained as the same.

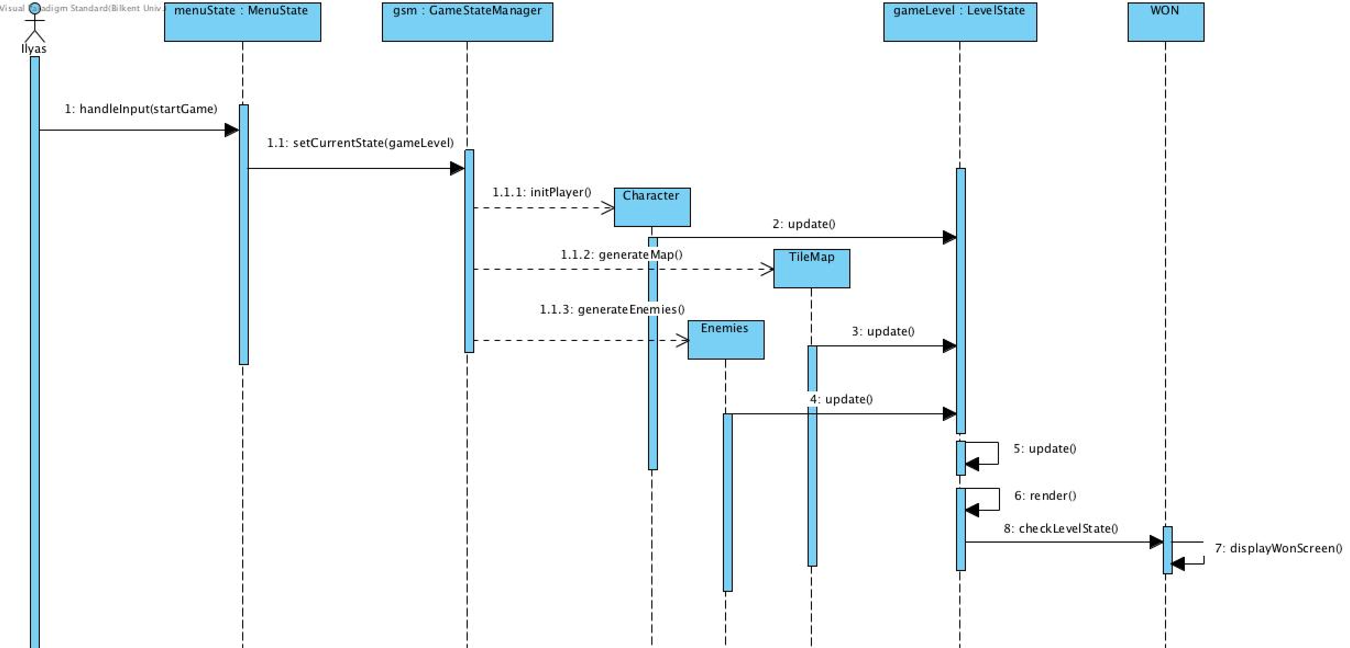
### Object and Class Model

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

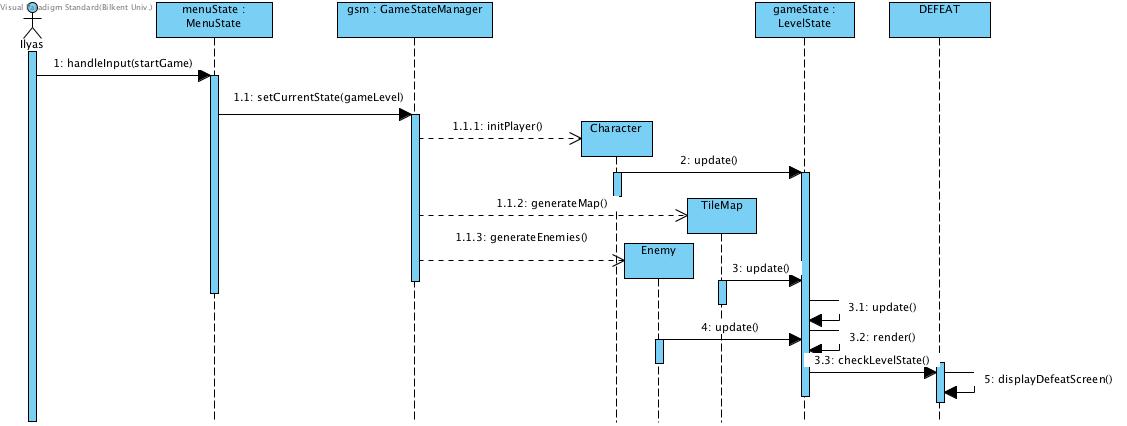


### Dynamic Models

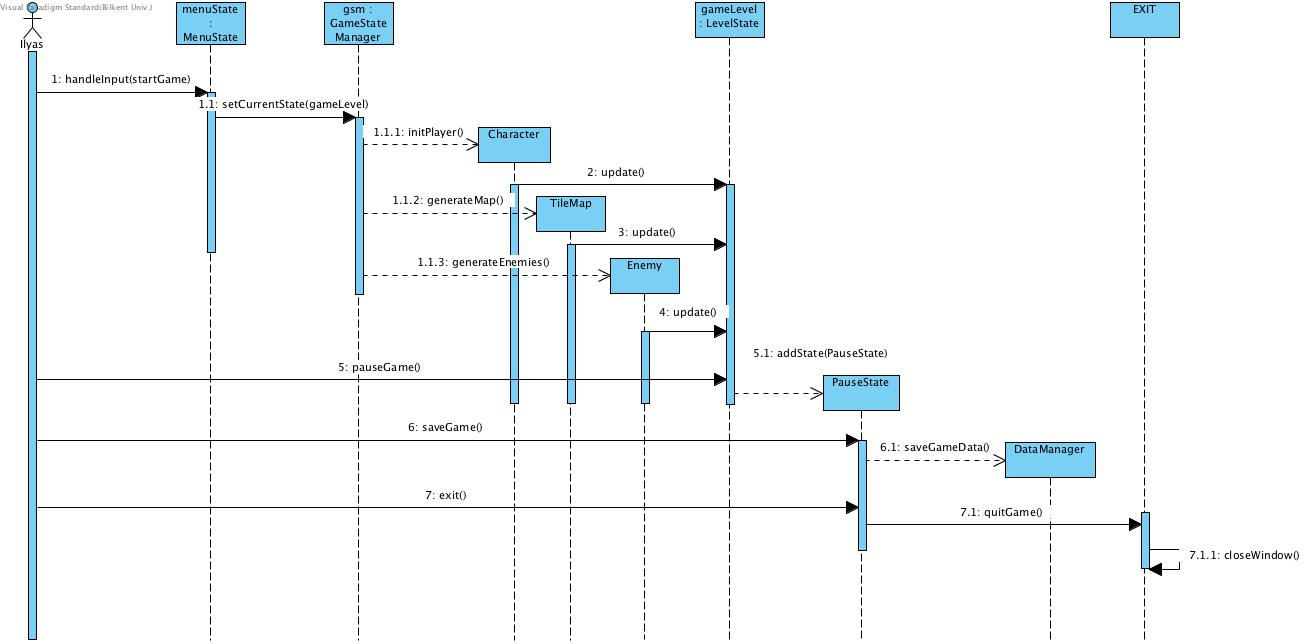
* 1. Sequence diagrams

1. Player - Ilyas starts the game and plays the level of the game. Game elements appear, he plays the level and won the level. 

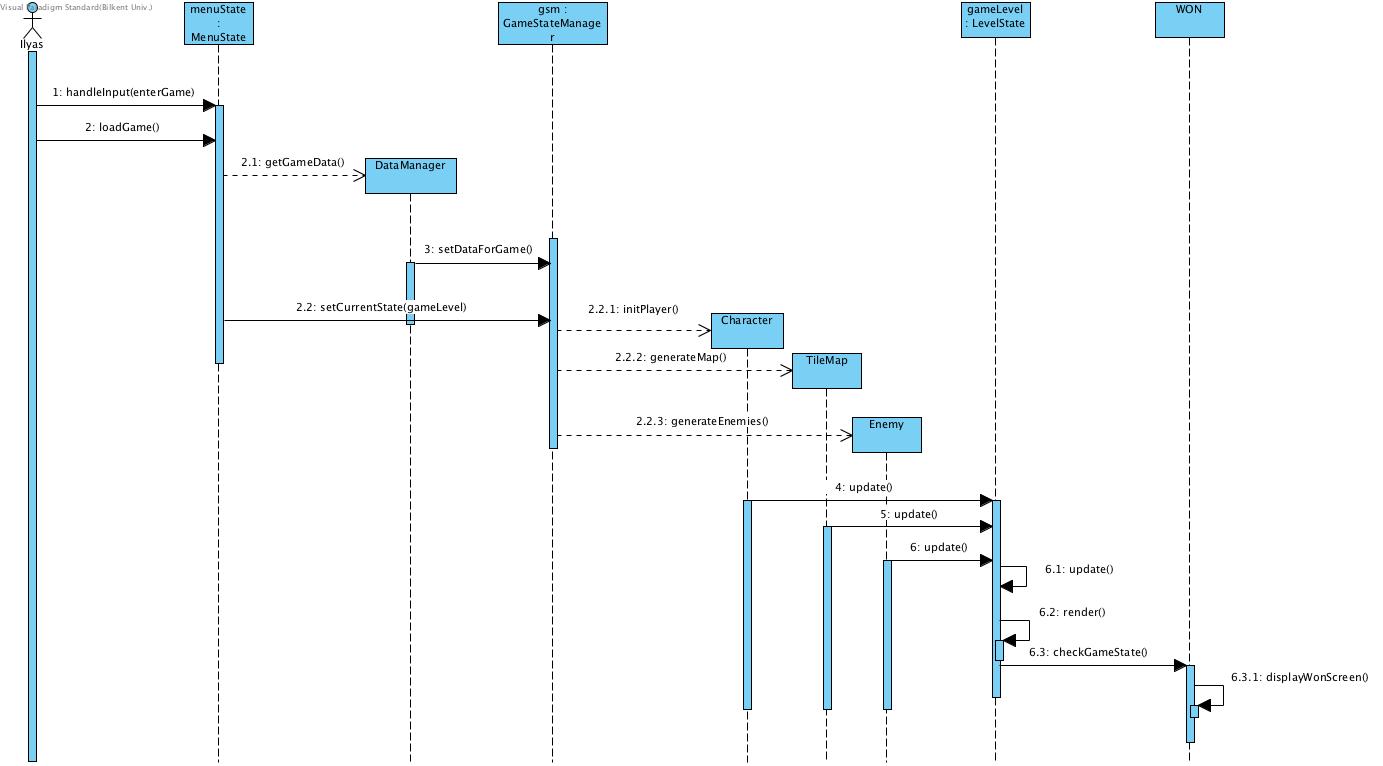
2. Player – Ilyas starts the game and plays the level of the game. Game elements appear, he plays the level but, since he lost all his lives given (obtained), he lost the game.



1. Player – Ilyas starts the game and plays the level of the game. During the game play, suddenly he pauses the game, because his mom calls him for eating. Small menu appears on the center of the screen and using the “Save Level” option he saves the game and quits the game.



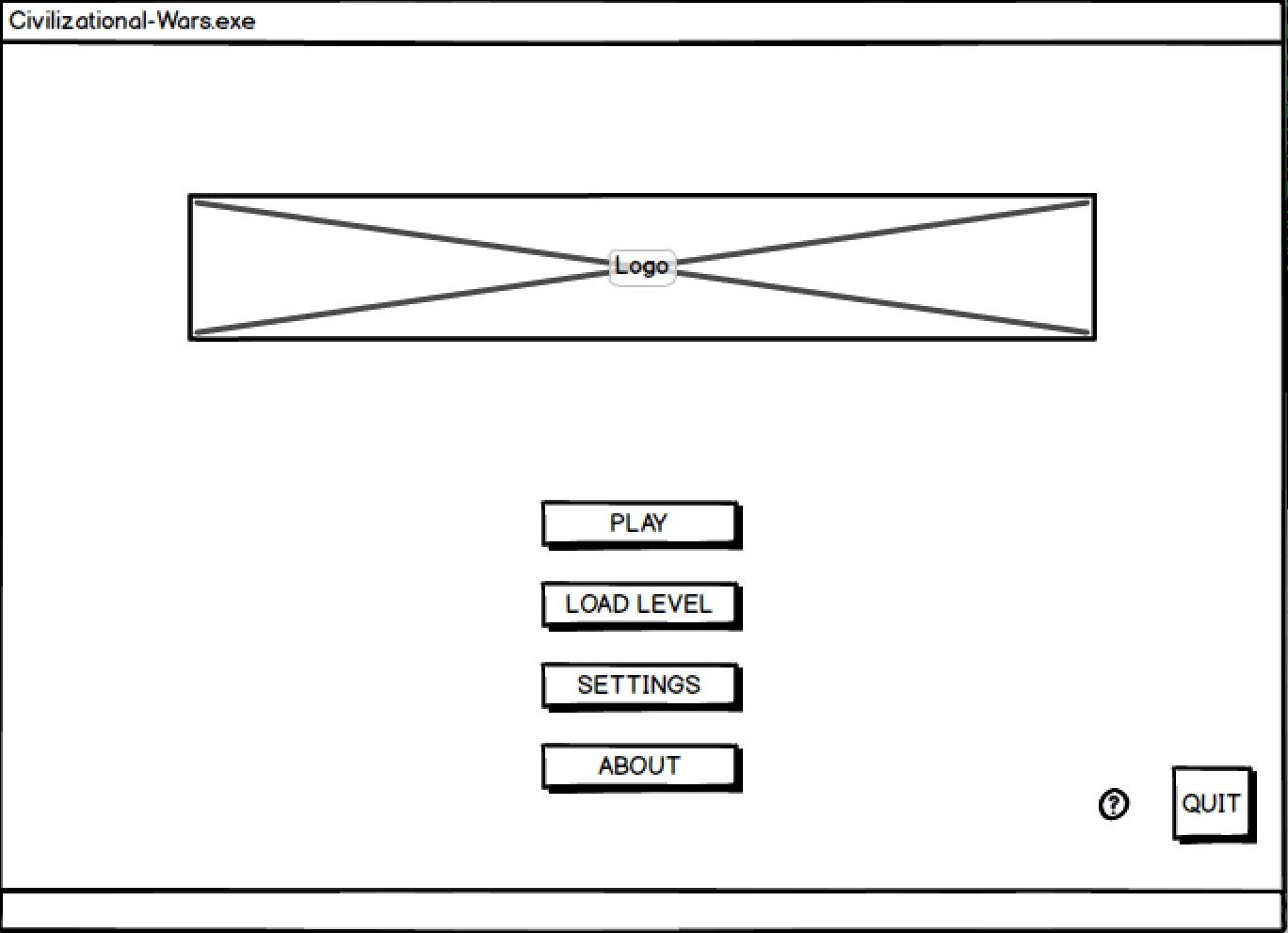
4. After eating Player – Ilyas returns back to play the game again. He wants to continue where he left the game, so he opens the game and on the menu state he selects “Load Level” option. The level he wants to continue is constructed and he plays the level and won it.



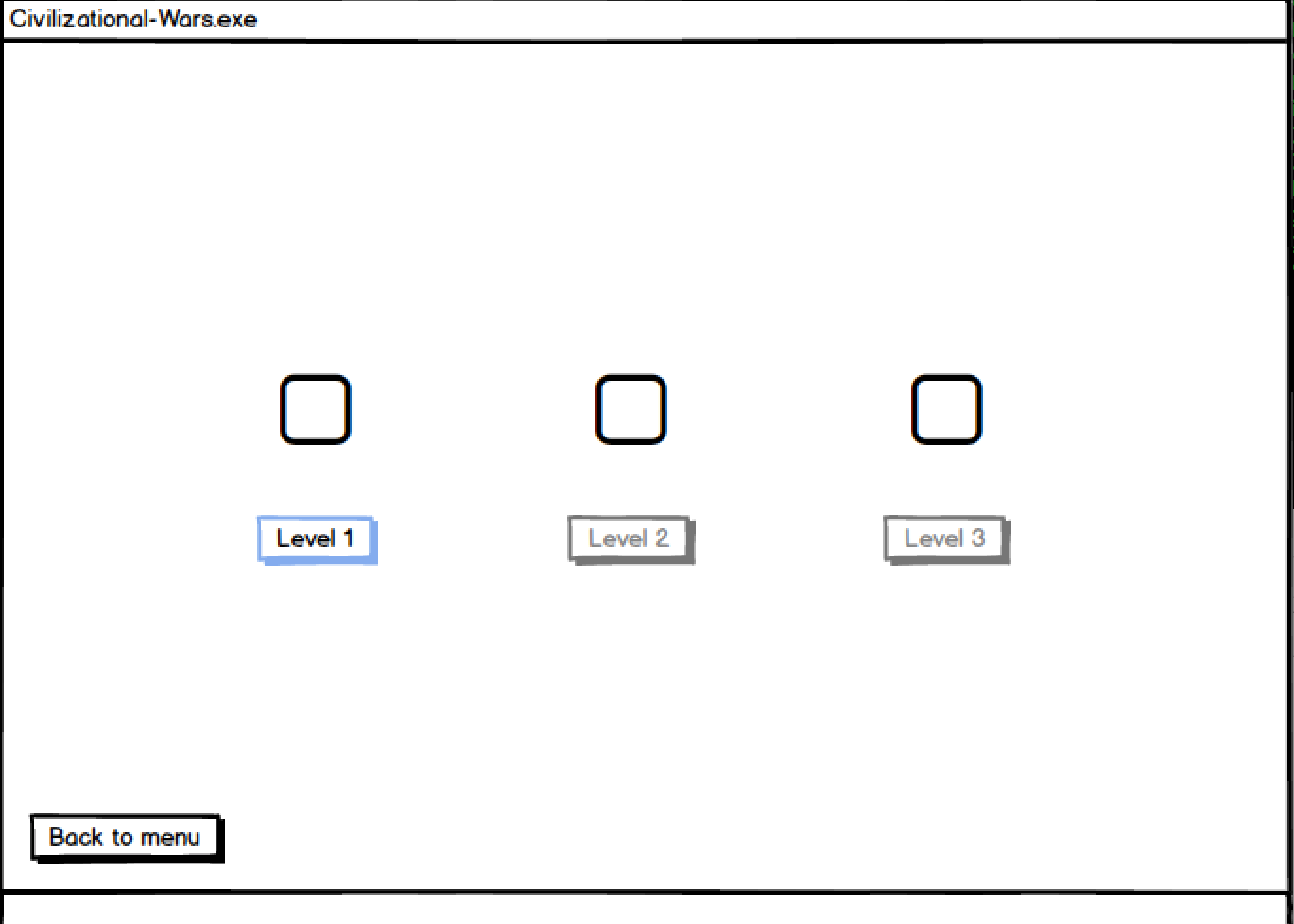
### User Interface

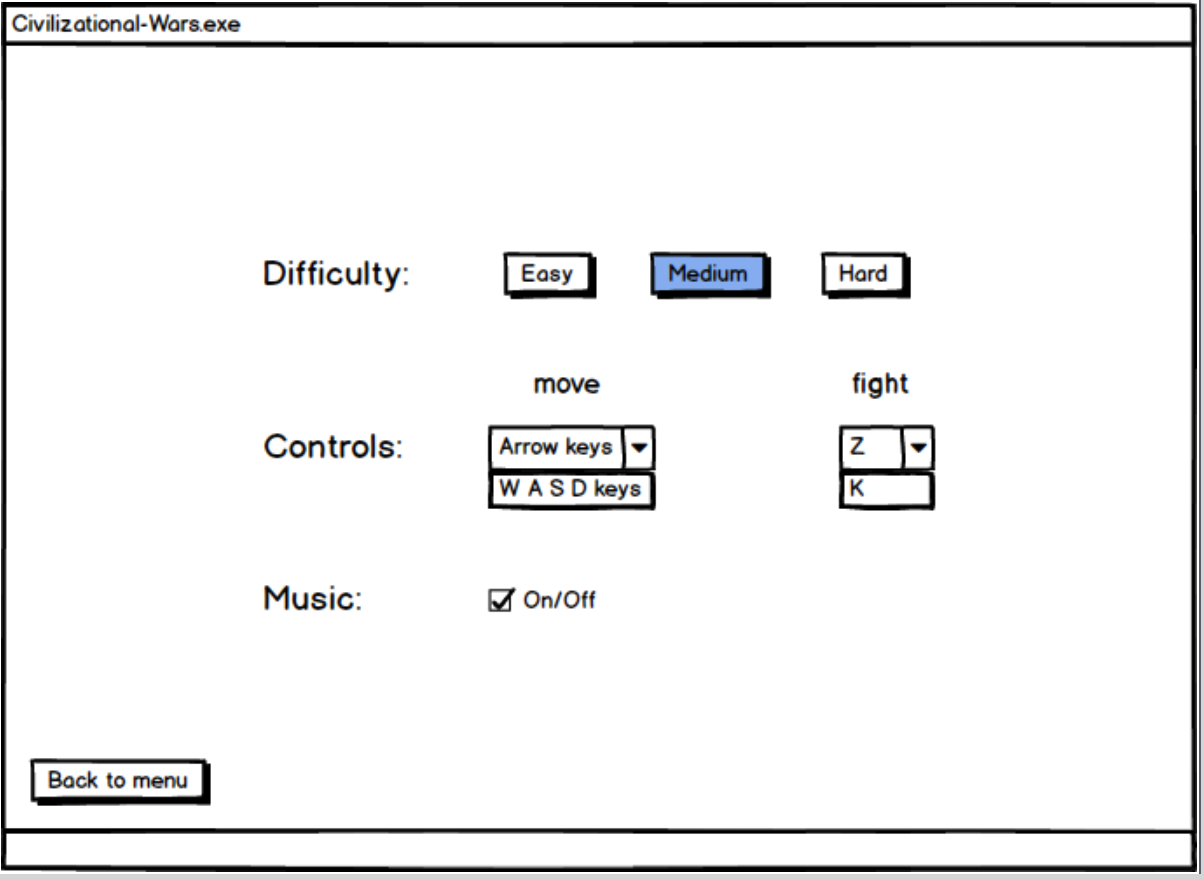
* These are some mock-up screens of the different states of the game. [3]

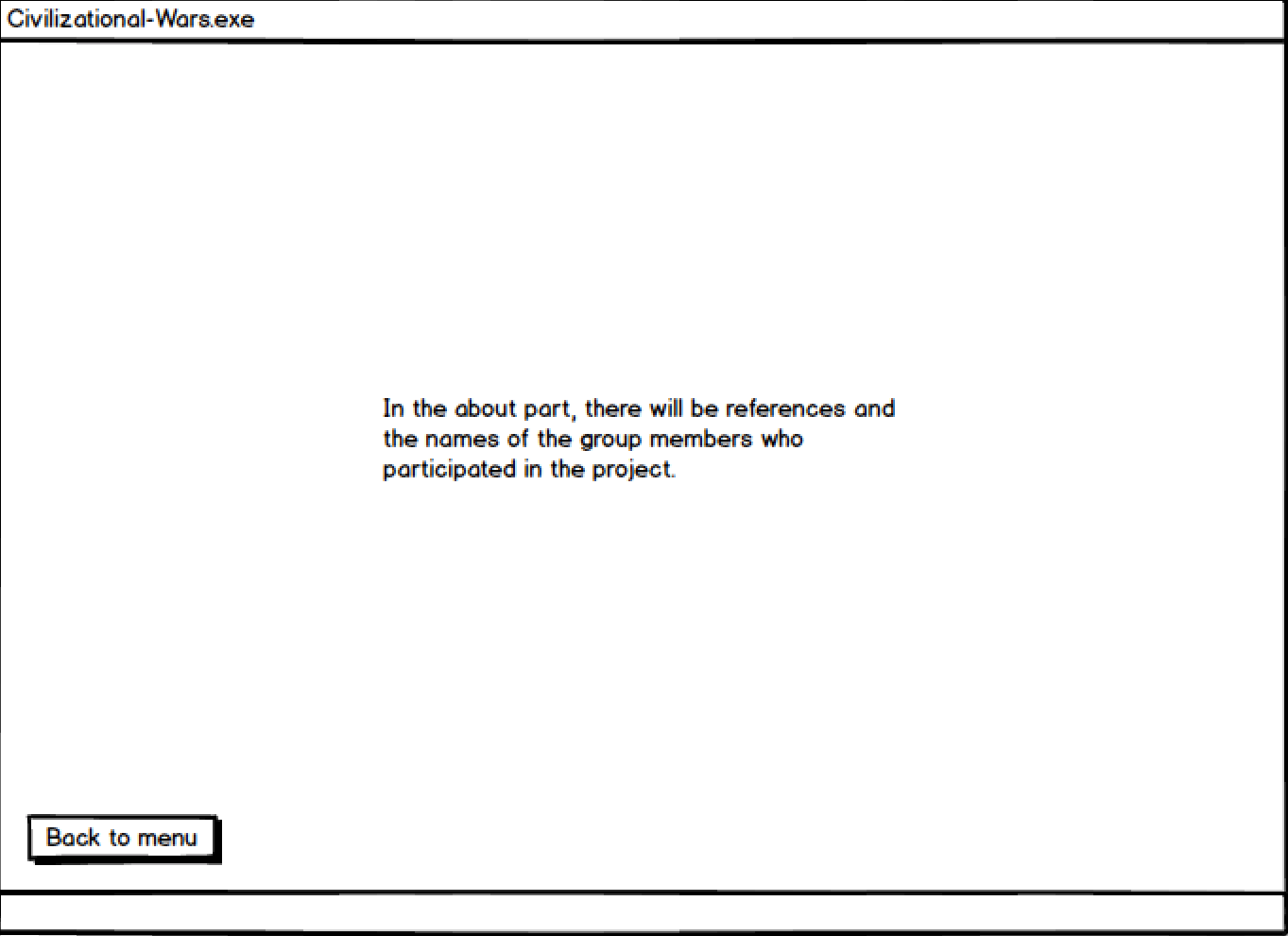
**NOTE**: These are just rough mock-up designs for the game. As the project goes on the progress, some textures for each part of the game screens will be added.

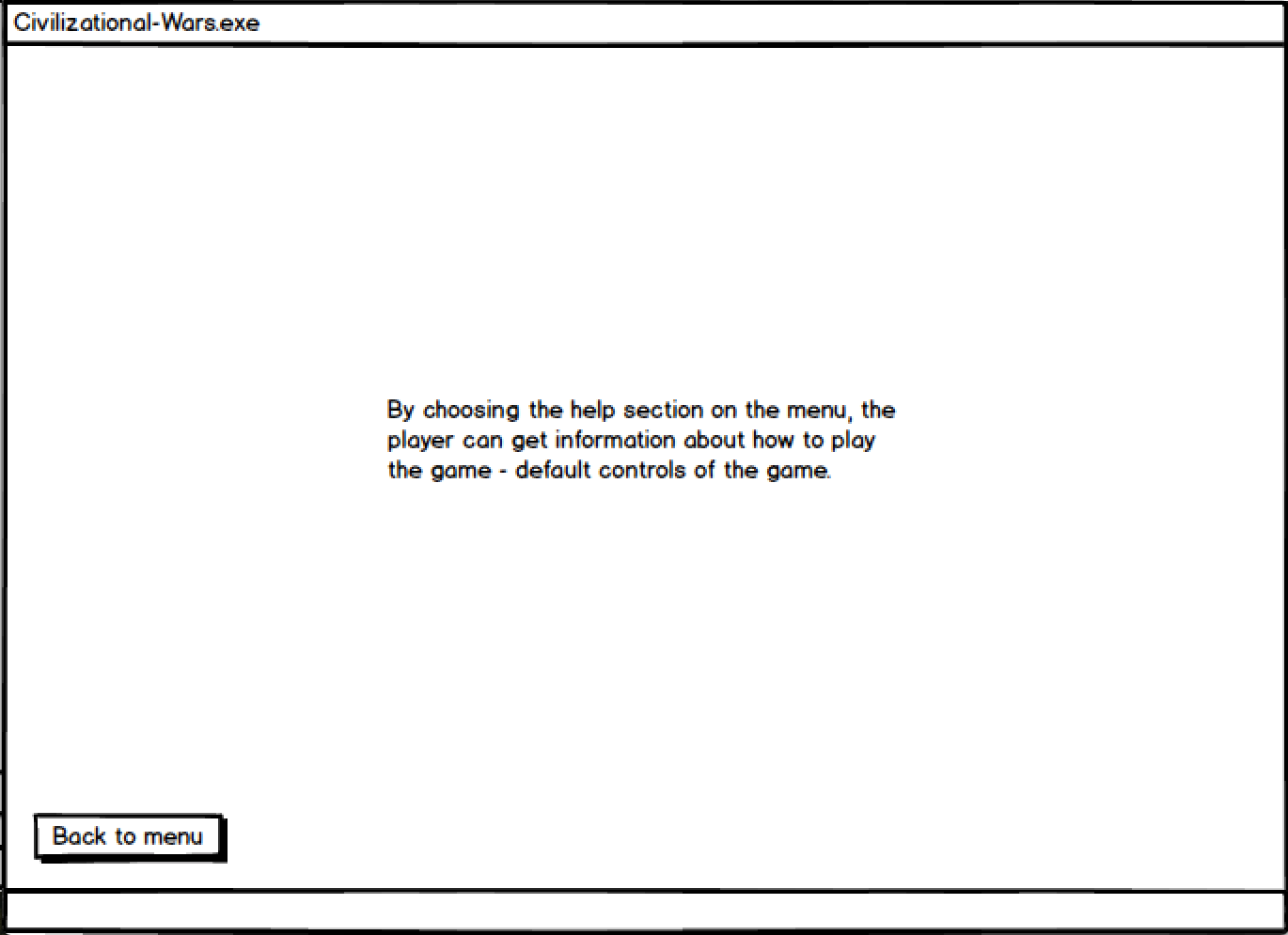
**Main Menu: **

**Play:**

**Load Level: **

**Settings:**

**About: **

**Help:**

# 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