 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 3, 2017

Contents

1. Introduction 1
2. Proposed System 1
3. Overview 1
4. Functional Requirements 2
5. Non-functional Requirements 2
6. Pseudo Requirements 2
7. System Models 2
8. Scenarios 2
9. Use-Case Model 2
10. Object and Class Model 3
11. Dynamic Models 3
12. User Interface 3
13. Glossary 3
14. References 3

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.

By writing this report as a group we aimed to convey some general and detailed information about the analysis of this game, specifically, 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.

## 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 …

## System Models

1. Use-case model

### Use-case scenarios

*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

### Dynamic Models

### User Interface

# Glossary

Glossary for any domain-specific terms you use in your report.

# References

1. Object-Oriented Software Engineering, Using UML, Patterns, and Java, 2nd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice-Hall, 2004, ISBN: 0-13-047110-0.