

Bilkent University

Department of Computer Engineering

Object-Oriented Software Engineering

EmojiStrike

Analysis Report

Project Group Ali Altaf Salemwala, Bora Bardük, Eren Çalık, Kıvanç Gümüş

Supervisor: Bora Güngören

Analysis/Draft Report

February 25, 2017

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Object-Oriented Software Engineering course CS319/1.

Contents

[1 Introduction 1](#_Toc475802750)

[1.1 Purpose 2](#_Toc475802758)

[1.2 Scope 2](#_Toc475802759)

[1.3 Definitions 2](#_Toc475802760)

[1.4 References 2](#_Toc475802761)

[1.5 Overview 2](#_Toc475802762)

[2 Overall Description 1](#_Toc475802751)

[2.1 Product perspective 2](#_Toc475802758)

[2.2 Product Functions 2](#_Toc475802759)

[2.3 User Characteristicsl 2](#_Toc475802760)

[2.4 Constraints 2](#_Toc475802761)

[2.5 Assumptions and dependencies 2](#_Toc475802762)

[3 Specific Requirements 1](#_Toc475802752)

[3.1 External Interface Requirements 1](#_Toc475802753)

[3.2 Functional Requirements 1](#_Toc475802754)

[3.3 Performance Requirements 1](#_Toc475802755)

[3.4 Design Constraints 2](#_Toc475802756)

[3.5 Software System Attributes 2](#_Toc475802757)

[3.6 Other Requirements 2](#_Toc475802757)

[4 Appendixes 2](#_Toc475802763)

[5 Index 2](#_Toc475802764)

Analysis Report

EmojiStrike

# Introduction

## Purpose

This document will present a detailed description of the EmojiStrike game. It will introduce the purpose and scope of this game, followed by a detailing of the features, requirements, and constraints of the product. The document will also contain cases which explain how the game system will react to stimuli from the players. The intention of this Software Requirements Specification document is to serve as a future reference point for the developers and the clients, as well as a map for the future development of the project.

## Scope of Project

This game is a multiplayer turn-based game, shown to the user as a 2-D map on the monitor screen. The game consists of avatars belonging to the players dropped in random places on a pre-defined map, followed by players then attacking each other with ranged and un-ranged weapons, with the last remaining player being the victor. Power-ups designed to enhance gameplay (for example, by making a player temporarily invincible, or increasing the potency of their weapons, or increasing a wounded player’s health) will be added to the map at certain time periods during the game.

The system will be designed to maintain player interest and interactivity in the game, as it is intended as an object of leisure. It will attempt to maintain player participation until the end of the game and will provide challenges to the player while also maintaining a record of previous victors, to foster competition.

## Glossary

Table 1 Glossary used for EmojiStrike

|  |  |
| --- | --- |
| Term | Definition |
| Power Up | Objects which boost the abilities of the player that captures them. |
| Avatar | The image used by the player to represent them. |

## References

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

## Overview of Document

The Overall Description section of this document gives an overview of the functionality of the game. It describes the informal requirements and is used to establish a context for the technical requirements specification section that follows.

The Requirements Specification section of this document is intended primarily for the developers, and so describes in technical terms the details of the functionality of the product. Both sections of the document describe the same game, but since they are intended for different audiences, different language has been used.

# Overall Description

## Product Perspective

Our product is self-contained, and standalone in terms of functionality. There are no external data servers required since it will be a turn-based game on the same machine. Our main actor is the players who are going to play the game.

## System Interfaces

The user will use the keyboard to pass in inputs. While selecting an Emoji avatar for each player, and the map of the game, players will use the four arrow keys to navigate the menu, the Enter key to select, and the Backspace key to move back. The user will use the letter and number keys to type in their character's name. During gameplay, the W, A, S, D keys will be used to move the character, and the four arrow keys will be used to aim the weapon, while the Space Bar will be used to fire.

The “P” button will be used to move the game in and out of the Pause mode. The “R” button will be used to restart the game and bring it back to the main menu, and the “Q” button will be used to quit the game. “Shift”+ “Q” saves the game first and then quits. A dialog box will open asking the user to type in a name for the saved game.

The outputs will be passed using the computer monitor and speakers. The user interface display will be visible on the monitor, and game sounds will be audible through the speakers.

The game will be saved in a sorted order according to the date. The file which will be used will be in the root directory of the game and will be a CSV file to keep track of different data easily. The data will consist of the game state, date, and desired name to each save file.

## User Interface

The game will be displayed in the maximum windowed resolution. In the top-left corner will be a menu with options to pause, quit, quit + save, and restart the game. Next to the commands will be the letters which should be pressed to bring the user into the desired state. The rest of the available space will be occupied by the game.

During each move, a small circular timer will display the amount of time remaining for the player currently doing their move.

When beginning the game, the users will select the number of players. Following this, they will select the Emoji and set the name for each player in sequence, and then finally select the map they will play on, after which they game will start.

The users will select Emojis from a list, and type in the name for each player. They will also select the number of players from a list of pre-approved numbers. A table of maps with small previews will allow them to select the map.

If a player chooses to load the game, there will appear a list of files which the user may load the game from. While saving, there will be a text input box where the user types in the saved game’s name and press Enter to quit.

## Hardware Interfaces

Our program will use Java’s automated HID USB device recognition to connect our application to hardware. The required keyboard, sound, and display interface will be detected automatically without the manual utilization of physical ports. Java will interpret the signals automatically and convert them to Java Virtual Machine’s inputs. The supported devices matter on the port availability between a computer and its physical port interfaces. It can be USB-C, USB-A, HDMI, etc.

## Software Interface

The game will be written in Java language and will interact with an operating system of any platform using the Java Virtual Machine. The Java specifications are:

* + - Java Version 8 Update 121 from www.java.com
    - Communications Interface
    - N/A

## Communications Interface

N/A

## Memory Constraints

In the Java Virtual Machine, -Xmx is 2 gigabytes, and the -Xms is 256 megabytes. The program will use the default settings.

## Operations

There are four modes of operation. There is the “Loading” mode, the “Main Menu” mode, the “Pause” mode, the “Game” mode.

During the “Loading” mode, no keys can be used. There are no user operations in this mode since the program is loading the game.

Throughout all the following modes, the users can use the “Q” key to quit the game, and the “R” key to bring them back to the main menu and delete any progress.

In the “Main Menu” mode, the user can use the arrow keys to navigate the menu, the Enter key to select, and the Backspace to move them back in the menu. Apart from these keys and “R” and “Q”, all other keys are deactivated.

In the “Pause” mode, the user can only use the “R”, “P”, “Q”, and “Shift” + “Q” keys. The “P” key moves the user back into the “Game mode”.

In the “Game” mode, the user can use all available keys. Arrow keys will change the weapon’s aim, while the “W”, “A”, “S”, “D” keys will navigate the player’s character. The “P” key will move the game into “Pause” mode, and the “R”, “Q”, and “Shift + Q” keys will keep their functionality. The Space Bar will shoot the weapon.

## Site Adaptation Requirements

a) The data initialization will be done in the same way for every deployment since there will be no difference when the application will be deployed into identical Java Virtual Machines. During the menu state, the texture files and menu music will be retrieved from the aspects file of the game. During the play state, the map and player figures will be rendered from the selected pre-existing map file before the game starts. There will be additional components which will procedurally be required, like sound and visual effect files. This data will be rendered real-time since their timing and locations are unknown beforehand.

b) Since our application will not change functionality in different areas of deployment, we will not have any specific mission-related features.

## Product Functions

The program will allow users to start the game, pick an avatar for themselves in that game, and choose a map to play on. While playing, users may choose a weapon to attack other players with, attack other players with the weapon, and move around on the map. Players may save the current game if it is incomplete, and load earlier game instances to play.

## User Characteristics

The game requires the user to have basic knowledge on how to use a keyboard. Other than that, no technical expertise, special knowledge, or extensive experience is required. The game is designed to be intuitive and easy to understand for new players.

## Constraints

Developers must create the game in Java.

## Assumptions and Dependencies

We are expecting that the game is designed for computers with regular functioning keyboards with ISO or US layout which possesses the required keys for users to communicate the game.

We are expecting that the game is designed for multi-color and multi-dimensional screens with a size such that gameplay is user-friendly.

We are expecting that the game is developed using an up-to-date Java version to avoid version conflicts.

## Apportioning the Requirements

If the up-to-date Java version of the development time requires different hardware and software constraints (like macOS losing support), requirements may change at the time.

If the up-to-date computer communication technologies render the use of a keyboard, speaker, or display interfaces obsolete; different types of I/O with the user should be implemented.

# Specific Requirements

This is where you provide the details of the results of your analysis work.

## External Interface Requirements

# Appendixes

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

# Index