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*

1.1 Purpose _____ 1

1.2 Scope of Project _____ 1

1.3 Glossary _____ 1

1.4 References _____ 1

1.5 Overview of Document _____ 2

2 Overall Description _____ *2*

2.1 Product Perspective _____ 2

2.1.1 System Interfaces _____ 2

2.1.2 User Interface _____ 2

2.1.3 Hardware Interfaces _____ 3

2.1.4 Software Interface _____ 3

2.1.5 Communications Interface _____ 3

2.1.6 Memory Constraints _____ 3

2.1.7 Operations _____ 3

2.1.8 Site Adaptation Requirements _____ 4

2.2 Product Functions _____ 4

2.3 User Characteristics _____ 4

2.4 Constraints _____ 4

2.5 Assumptions and Dependencies _____ 5

2.6 Apportioning the Requirements _____ 5

3 Specific Requirements _____ *5*

3.1 External Interface Requirements _____ 5

3.1.1 User Interfaces _____ 5

3.1.2 Hardware Interfaces _____ 5

3.1.3 Software Interfaces _____ 7

3.1.4 External Interface Requirements _____ 7

3.2 Functional Requirements _____ 7

3.2.1 Menu State _____ 7

3.2.1.1 Arrow Keys _____ 7

3.2.1.2 Enter Key _____ 7

3.2.2 Choice State _____ 7

3.2.2.1 Arrow Keys _____ 7

3.2.2.2 Enter Key 7
3.2.2.3 Letter and Number Keys 7
3.2.3 Game State 7
3.2.3.1 Arrow Keys 7
3.2.3.2 Space Key 7
3.2.3.3 WASD Keys 7
3.2.3.4 Q Key 8
3.2.3.5 R Key 8
3.2.3.6 P Key 8
3.2.3.7 Shift + Q Keys 8
3.2.4 Save State 8
3.2.4.1 Letter and Number Keys 8
3.2.4.2 Enter Keys 8
3.2.4.3 Q Keys 8
3.2.5 Pause State 8
3.2.5.1 P Key 8
3.2.5.2 R Key 8
3.2.5.3 Q Key 8
3.2.6 Load State 8
3.2.6.1 Arrow Keys 8
3.2.6.2 Q Key 8
3.2.6.3 Backspace Key 8
3.2.7 Tutorial State 8
3.2.7.1 Arrow Keys 8
3.2.7.2 Backspace Key 9
3.2.7.3 Q Key 9
3.3 Performance Requirements 9
3.4 Design Constraints 9
3.5 Software Design Attributes 9
3.5.1 Reliability 9
3.5.2 Availability 9
3.5.3 Security 9
3.5.4 Maintainability 9
3.5.5 Portability 9
3.6 Other Requirements 9

4 Appendixes_____ 10

5 Index_____ 10

Analysis Report

EmojiStrike

1 Introduction

1.1 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.

1.2 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.

1.3 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.

1.4 References

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

1.5 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.

2 Overall Description

2.1 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.

2.1.1 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.

2.1.2 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.

2.1.3 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.

2.1.4 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

2.1.5 Communications Interface

N/A

2.1.6 Memory Constraints

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

2.1.7 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.

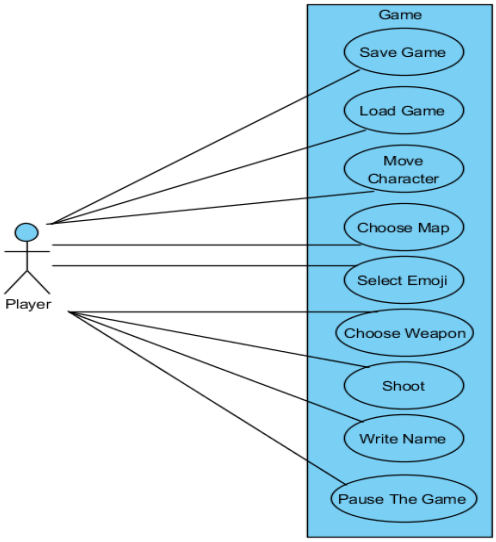
2.1.8 Site Adaptation Requirements

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.

2.2 Product Functions

Figure 1 Use Case Diagram



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.

2.3 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.

2.4 Constraints

Developers must create the game in Java.

2.5 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.

2.6 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.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The system starts a window in Java and waits for keyboard inputs from the user to change the game state. Different commands work in different states, and so not all keys work in every part of the game. For example, the ASDW commands will not work for the menu state.

3.1.2 Hardware Interfaces

a) Keyboard

b) To allow the user to enter commands to be interpreted by the game

c) N/A

d) N/A

e) N/A

f) Real-time

g) Changes will be made visible on the monitor screen and audible through the speakers

h) N/A

i) N/A

j) N/A

k) N/A

l) End messages.

a) Speakers

b) To allow the user to hear sounds emitted by the game for a better user experience

c) Source is game events

d) N/A

e) N/A

f) Real-time

g) Changes made by keyboard influence sounds produced

h) N/A

i) N/A

j) .mp3, .wav

k) N/A

l) End messages.

a) Monitor

b) To allow the user to view changes in the gameplay and to see the visualization of their commands.

c) Source is game events

d) Valid range, accuracy, and/or tolerance

e) Units of measure

f) Real time

g) Changes will be made visible on the monitor screen and audible through the speakers

h) N/A

i) N/A

j) .jpg, .png, etc

k) Command formats

l) End messages.

3.1.3 Software Interfaces

N/A

3.1.4 External Interface Requirements

N/A

3.2 Functional Requirements

3.2.1 Menu State

3.2.1.1 Arrow Keys

The system shall highlight the available options depending on which keys are pressed. Up key moves the selection up, down key moves it down.

3.2.1.2 Enter Key

The system shall run the highlighted option.

3.2.2 Choice State

3.2.2.1 Arrow Keys

The system shall highlight the available options depending on which keys are pressed. It will loop through the options.

3.2.2.2 Enter Key

The system shall run the highlighted option.

3.2.2.3 Letter and Number Keys

In the Emoji-selection window, the system shall take the textual input from the user and save it as the name of that player.

3.2.3 Game State

3.2.3.1 Arrow Keys

The system shall calculate the direction where the weapon will be fired and represent an arrow in that direction.

3.2.3.2 Space Key

The system shall fire the weapon and display the animations, and calculate all updates to display on the screen (characters hurt, terrain destroyed, remaining players, new spaces moved)

3.2.3.3 WASD Keys

Update the location of the player on the map by the predetermined move amount for the character. The amount of location update may change by the powerups acquired by the character. The character may strafe left, right, and jump. Strafing and jumping physics are calculated by the physics engine and will stop if a horizontal or vertical collision occurs.

Detect the collision on the map and use integrated physics to execute the necessary action. No collision with terrain executes the falling action and continues until a collision is provided.

3.2.3.4 Q Key

The system shall terminate.

3.2.3.5 R Key

The system shall quit this instance of the game state and move to the menu state.

3.2.3.6 P Key

The system shall move from the Game state into the Pause state.

3.2.3.7 Shift + Q Keys

The system shall first open the Save State and then terminates.

3.2.4 Save State

3.2.4.1 Letter and Number Keys

The system shall store the input.

3.2.4.2 Enter Keys

The system shall make a new file with the stored data of the current game, and the name of the and then return to the state it came from.

3.2.4.3 Q Keys

The system shall terminate.

3.2.5 Pause State

3.2.5.1 P Key

The system shall move from the Pause state into the Game state.

3.2.5.2 R Key

The system shall quit this instance of the game state and move into the menu state.

3.2.5.3 Q Key

The system shall terminate.

3.2.6 Load State

3.2.6.1 Arrow Keys

The system shall change the highlighted saved game on the load game screen. If there is no saved game, no action will be taken by the system.

3.2.6.2 Q Key

The system shall terminate.

3.2.6.3 Backspace Key

The system shall move back into the Menu State.

3.2.7 Tutorial State

3.2.7.1 Arrow Keys

The system shall scroll through the tutorial screen displayed to the user.

3.2.7.2 Backspace Key

The system shall return to the Menu state.

3.2.7.3 Q Key

The system shall terminate.

3.3 Performance Requirements

The game consists of only one terminal. The game is played only on one computer, with no outside communication.

The game supports anywhere from one to ten simultaneous players.

Keyboard input will be used to alter the state of the game, which will be reflected via changes in monitor and speakers.

Data transactions will hopefully be handled in real-time since this system is rather light. The user should not have to wait for the game.

3.4 Design Constraints

Limitations imposed by Javax or Swing, for the GUI.

Limitations imposed by the Java Sound API.

Limitations imposed by the Java KeyboardListener used by the GUI component.

3.5 Software Design Attributes

3.5.1 Reliability

The client should have a Java Runtime Environment 8 installed since the system will run on that. The reliability of the system depends on the JVM, and so it should not crash or hang for any reason other than operating system failure.

3.5.2 Availability

The system is available to everyone with Java Runtime Environment.

3.5.3 Security

No need for security.

3.5.4 Maintainability

Software has no real maintainability requirements.

3.5.5 Portability

Since it will run on the Java Virtual Machine, the system is highly portable. It has no other portability considerations.

3.6 Other Requirements

There are no other requirements.

4 Appendixes

//We will update this part in future revisions

5 Index

//We will update this part in future revisions