# Requirments and Analysis Document

Gabriel Wallin, Omar Oueidat, Erik Strid, Olof Enström

May 2017

## Version

Version: 3.0
Date: 17 May 2017
Author: Revised by Omar Oueidat
This version overrides all previous versions.

# Contents

**1 Introduction** **1**

1.1 MiniMiner . . . 1

1.2 Definition, acronyms and abbrevations . . . 1

1.3 Scope of application . . . 1

**2 Requirements** **2**

2.1 User interface . . . 2

2.2 Funcitonal requirements . . . 2

2.3 Non-functional requirements . . . 3

2.3.1 Usability . . . 3

2.3.2 Performance . . . 3

2.3.3 Supportability . . . 3

2.3.4 Implementation . . . 3

**3 Use cases** **4**

3.1 Use case listing . . . 4

**4 Domain model** **5**

4.0.1 Class responsibility . . . 5

**5 References** **6**

# 1 Introduction

## 1.1 MiniMiner

The main reason behind the creation of miniMiner is the common interest of platform games. All four members of the group liked the idea of a platform-digging game, and therefore, the concept of miniMiner emerged. The essential goal of the application is entertainment and allowance of "micro-breaks".

## 1.2 Definition, acronyms and abbrevations

- Miner - The player
- Hull - The miners health, which decreases when he hits something or crashes
- Fuel - The gas of which the miner uses to be able to move around libGDX - The library used to create the game.
- MVC - Model-View-Controller, a design pattern that organises the program in a structure in a way that avoids high coupling and dependencies
- Drilling - The act of which the player will dig a hole into the ground
- Gear - A collective word for Fuel and Hull

## 1.3 Scope of application

The application will not be able to save any data or restore any previous used data when terminated. Game controllers will only include a touchpad (and a drill button), this means that the game will not be optimal for desktop usage. The user will not be able to modify the map or store, it is a non-dynamical world built once. Furthermore, sound effects can be implemented into the game.

The game is developed with the mindset of extensibility. More extensions and complex features should be easily added in the final product.

# 2 Requirements

## 2.1 User interface

The application is based on a 2D-platform GUI, with the user digging downwards into a dirt-like environment. The UI will contain a joystick in the bottom right corner that controls the miner. A shop will also be added to the game which interacts with the user's drilling machine.

## 2.2 Funcitonal requirements

The user will be able to move, dig and fly until the fuel gauge is empty. On the surface, the user is allowed to sell materials, purchase new equipment, as well as refuel and repair the vehicle. The world in which the player will move in is gravity-based, so there will always be a downward-facing force forcing the vehicle back down to solid ground.

#### The user should be able to

1. Start a game or change options in the starting menu
    - (a) There should be a starting game screen in which the player can chose to start the game or see through options.
2. Play through the game which involves
    - (a) Moving around the map
        - i. Moving the touchpad should move the miner into a desired position
    - (b) Diggin through the ground
        - i. Using a button, the miner will dig through the ground to collect minerals and score
    - (c) Gather minerals
        - i. Sell minerals
    - (d) Upgrade fuel and hull and drill parts
        - i. Using cash the miner gets from minerals, he will be able to upgrade the gear to improve its functionality
    - (e) Take damage
        - i. Hitting anything in high speed will result in decrease of hull(health) if this is 0, the game is over
    - (f) Fill fuel and fix the drill
        - i. The miner will be able to fill the fuel or repair the drill so he can continue the game, otherwise he will die and the game will be over

## 2.3 Non-functional requirements

### 2.3.1 Usability

The game focuses greatly on usability. The game language is English centered.

The user will get a brief rundown of how the program is used and further knowledge will demand some intuition from the user.

### 2.3.2 Performance

The game will be constructed in a way which will reduce the amount of memory used to process the game which will be very important for a mobile phone user. Input from the user should be almost instantaneous with very little delay(if any at all), and will respond immediately graphically as well. Since it is a offline based game, there will be no network and therefore no latency.

### 2.3.3 Supportability

The program is mainly focused to be supported for an android mobile device, but will also be supported for a desktop computer. The GUI is constructed in a way to support a mobile screen but will work on a desktop accordingly.

However, the joystick will not be effective in a desktop playing environment.

### 2.3.4 Implementation

The program will use the Java Environment in which the user running the program will be required to have a device that can run Java programs.

# 3 Use cases

## 3.1 Use case listing

@usecase doc

## 4 Domain model

Link to UML: https://drive.google.com/open?id=0B2Q$_zO7v9KTdR1VSTy1PWFVVS2M$

#### 4.0.1 Class responsibility

## 5 References

GÖR BIBLOTEK OCH IEEETran