# Requirements and Analysis Document for Maze (RAD)

## Contents

# 1. Introduction

Here we introduce a short overview of the project.

### 1.1 Purpose of the system

The aim of the project is to create a computer game which consists of a player getting through a labyrinth. On the way the player encounters obstacles, such as difficult questions and monsters.

### 1.2 General characteristics of application

Only one player at a time can play. The game has three save slots where three different players can save their games. The application does not save half-finished games. The player gets points for correct answers and sometimes he/she gets a key that they can use for the final door. The player has a time limit of one hour for each level. The game cannot be paused. The time becomes points and the player gets a total score on every map.

### 1.3 Scope of application

It should be possible to play two different maps with increasing difficulty. Further, it should be possible to create, select and delete save slots and see highscores for the game.

### 1.4 Objectives and success criteria of the project

1. It should be possible to play two levels.
2. It should also be possible for three different players to save their games. (There should be three save slots)
3. It should also be possible to see a high score over all players that have played the game.

### 1.5 Definitions, acronyms and abbreviations

- GUI, graphical user interface.
- Java, platform independent programming language.

# 2. Requirements

### 2.1 Functional requirements

The player should be able to:

1. Make a new player (saving name, character type, and high score per level)
2. Load a previous player
3. Play the game with two different levels
4. See highscores.
5. Exit the program. If the player is in the middle of a game it will not save. But if he/her has finished it will save the score and level.

## 2.2 Non-functional requirements.

### 2.2.1 Usability

High priority.

### 2.2.2 Reliability

NA

### 2.2.3 Performance

Any action initiated by the player should not exceed 5 seconds in response time

### 2.2.4 Supportability

The application will at the beginning support PC but the GUI should easily be modified to other platforms like Apple (Iphone,Ipad), Google (any google phone and tablet) and Windows (any windows phone and tablet). The time it will take to adapt the GUI should not exceed one month of work.

### 2.2.5 Implementation

The application will be implemented in Java. All hosts have to have the JRE installed.

### 2.2.6 Packaging and installation

The program contains resources such as image and text files for the maps and the info saved for the players.

### 2.2.7 Legal

We do not have any lawyers on the team.

## 2.3 Application models

### 2.3.1 Use case model

See appendix.

### 2.3.2 Use cases written/priorities

For written description, see separate page. Use cases in order of priority.

1. Move
    a. Answer questions
    b. move onto a wall
    c. Feed monster / want to enter monster square
    d. Unlock doors / enter door square
    e. Open chest / enter chest square
2. Choose new save slot
3. Load game / choose used save slot
4. View highscore
5. Return to main menu

### 2.3.3 Analysis model

See appendix.

### 2.3.4 User interface

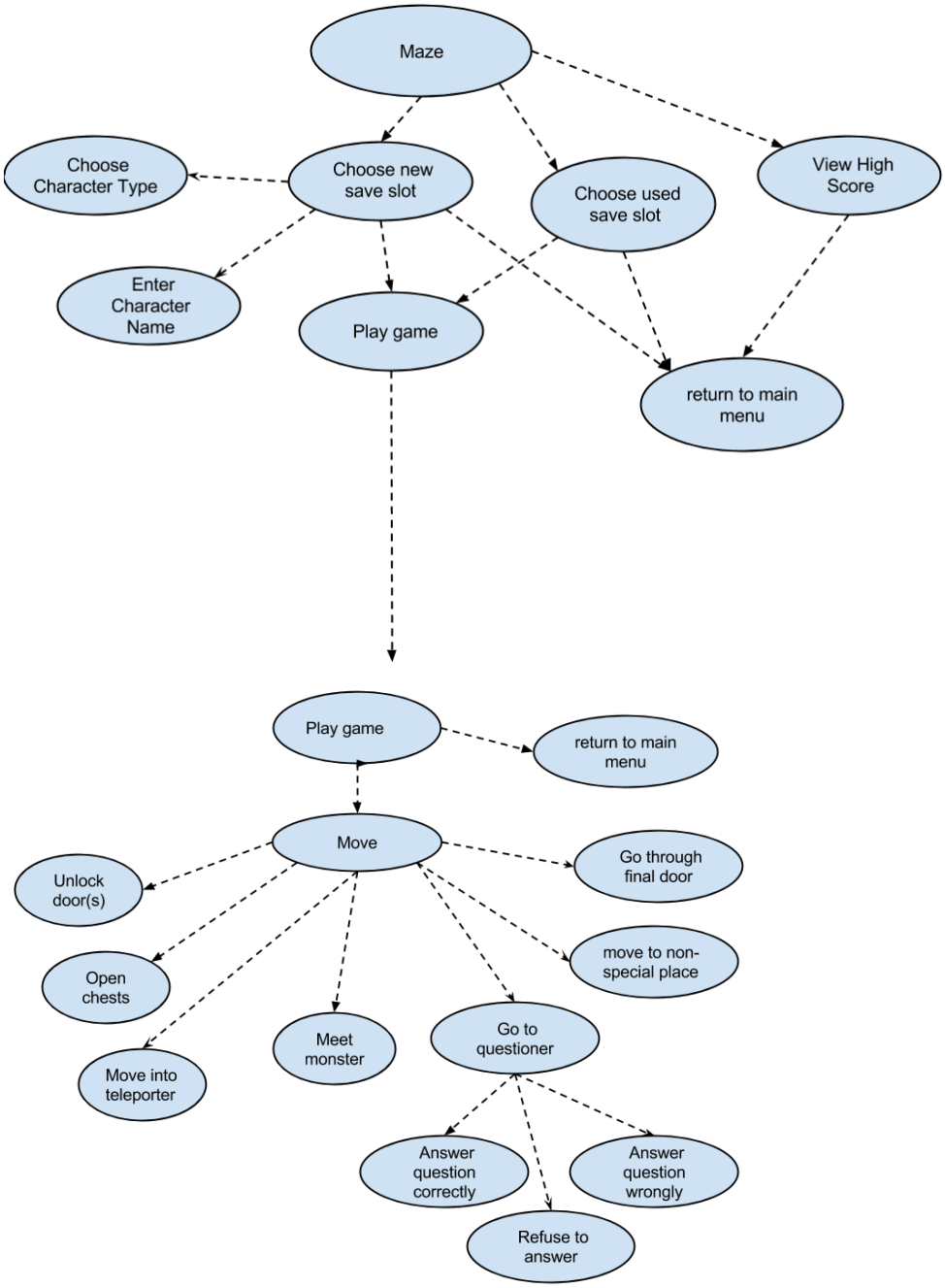
See appendix.

## 2.4 References

None

# APPENDIX

Use cases
Overview.




**Use case texts**

# Use Case: Move to empty square

**Summary**: This is how the player moves his/her character on the map.
**Priority**: High
**Participators:** Actual player

| Player | System |
|---|---|
| Wants to move to adjacent empty square | |
| | Moves player to new square |

## Use Case: Attempts to move onto wall

**Summary**: This is how the player interacts with walls.
**Priority**: High
**Participators:** Actual player

| Player | System |
|---|---|
| Wants to move to adjacent wall square | |
| | Does nothing |

## Use Case: Walk up to questioner

**Summary**: This is how the player interacts with questioners.
**Priority**: High
**Participators:** Actual player

| Player | System |
|---|---|
| Walks up to questioner | |
| | Asks whether user would like to answer a question |

Flow 1 Yes

| Answers yes | |
|---|---|

| | |
|---|---|
| | Asks question |

Flow 1.1 Answers correctly

| | |
|---|---|
| Selects correct answer | |
| | Awards points. If user does not have exit key, gives exit key. Otherwise, randomly awards an apple, a key or more points. TODO: character type effects? |

Flow 1.3.1.2 Answers incorrectly

| | |
|---|---|
| Selects wrong answer | |
| | Takes points. Depending on type of questioner: Wizard moves character back to first square, thief randomly steals an apple or a key (providing the player has one), warrior?? |

Flow 1.3.2 No

| | |
|---|---|
| Declines to answer | |
| | Empties text box. |

# Use Case: Walk up to monster

**Summary**: This is how the player deals with monsters.
**Priority**: High
**Participators:** Actual player

| Player | System |
|---|---|
| Player attempts to move onto monster's square | |
| | If player has an apple, removes it and makes monster stand aside (square will in future react like an empty square). Moves player onto monster's square. |

| | |
|---|---|
| | Otherwise, system gives player a message informing them that they need an apple to pass the monster. |

## Use Case: Open a chest

**Summary**: This is how the player opens a chest.
**Priority**: High
**Participators:** Actual player

| Player | System |
|---|---|
| Steps onto square containing chest | |
| | Randomly generates 1 or no apple, 1 or no key and ? to ? points to give to the player. In future, this square will behave like an empty one. |

## Use Case: Finish the game

**Summary**: This is how the player finishes the game.
**Priority**: High
**Participators:** Actual player

| Player | System |
|---|---|
| Moves to final gate | |
| | If player has exit key, exits the game. Saves player's high score (if higher than previous), updates high score board (if necessary), shows level selection screen. |

## Use Case: Choose new save slot

**Summary**: Player starts a new game with a new name.
**Priority**: Medium

| Player | System |
|---|---|
| Clicks on save slot | |
| | Shows make-new-character screen |
| Enters information (optional) and clicks start | |
| | Saves player's information. Starts new game at level 1. |

## Use Case: Load game / Choose old save slot

**Summary**: Player starts a previous game with an old name.
**Priority**: Medium

| Player | System |
|---|---|
| Clicks on the save slot. | |
| | Shows the select level screen. |
| Selects level. For every level the player has finished, the next is also selectable. | |
| | Starts new game at chosen level. |

## Use Cases: high score

**Summary**: Player wants to see high score board.
**Priority**: Low

| Player | System |
|---|---|
| Clicks button for high score board (main menu). | |
| | Displays high scores. |

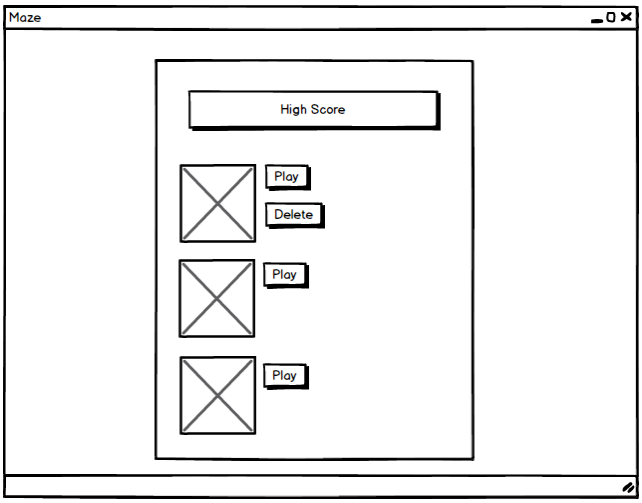
## Use Case: return to main menu

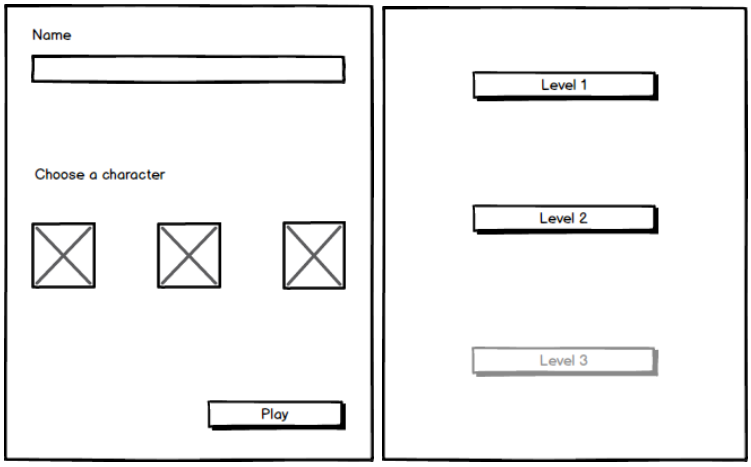
**Summary**: Player wants to return to the main menu.

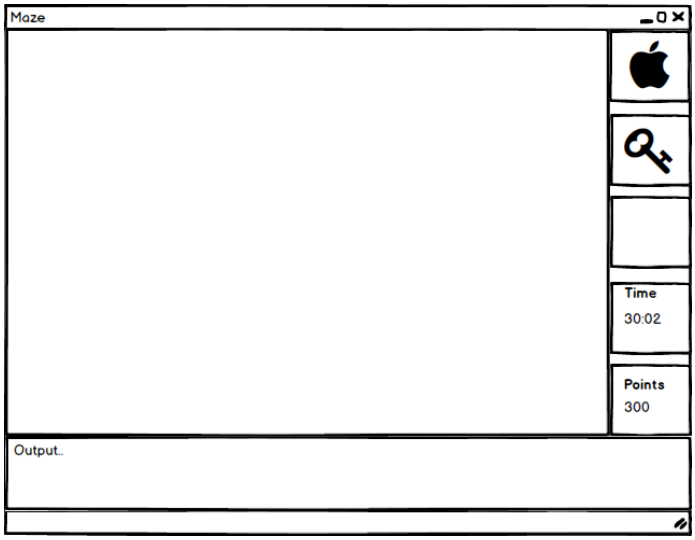
**Priority**: Medium

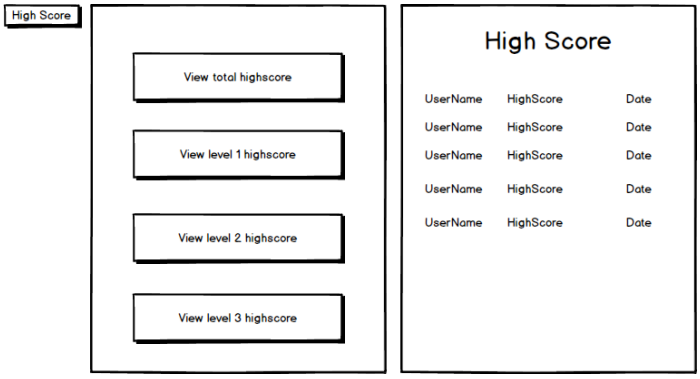
| Player | System |
|---|---|
| Clicks relevant button (from any screen other than main menu) | |
| | Aborts game (if any) and shows main menu. |

## GUI




Name
Choose a character
Play
Level 1
Level 2
Level 3


Maze
Time
30:02
Points
300
Output..


High Score
View total highscore
View level 1 highscore
View level 2 highscore
View level 3 highscore
High Score
UserName HighScore Date
UserName HighScore Date
UserName HighScore Date
UserName HighScore Date
UserName HighScore Date

## Analysis model

Preliminary analysis model (possible to update).