Requirements and Analysis Document for the Epic Game of Awesome project (RAD)

Contents

Version: 2.0

Date: 2015-05-28

Author: Emil Axelsson, Erik Karlkvist, Rebecka Reitmaier, Hampus Rönström.

1 Introduction

The project is about creating a two dimensional platformer. It is supposed to be a fun game for everyone to enjoy and spend time with when there is nothing else to do. In order to give the game its unique feel the character can grow and shrink. This opens up a dozens of potential puzzle for a user to solve. It should take some effort to complete each map, but you will be rewarded for it.

1.1 Purpose of application

Create a fun platforming game for everyone to enjoy.

1.2 General characteristics of application

It's a two dimensional platformer with a character the user controls. It is based around puzzle solving, jumping and avoiding dangers. The character can collect stars to grow or shrink. This creates a set of challenges the user has to go through to win the game. In order to achieve a higher score the user has to finish each level in a short time, however this is optional.

1.3 Scope of application

It is singleplayer only. There is no networking and it is a standalone desktop app run by Java.

1.4 Objectives and success criteria of the project

There is at least two levels. The character is able to grow and shrink. Time is counted and presented at the end of each level to represent a score. There is at least one dangerous obstacle. The character is able to move in x- and y-axis.

1.5 Definitions, acronyms and abbreviations

- The user - the real life person that plays the game
- Character - the center of the game, a player controlled object
- Level - contains a map and everything else visible to the user
- Map - the layout where the character moves
- Stars - an object the character picks up to grow or shrink
- Exit-door - the end of the map, this is where the user should aim to reach
- Spikes - dangerous objects, if touched resets the character to the start of the map

2 Requirements

2.1 Functional requirements

- Start
- Pause
- Setting
- Levels Menu
- Select level
- Volume Control
- Move
- Jump

2.2 Non-functional requirements

2.2.1 Usability
It should be easy to understand for new users and intuitive. The menus are logical and resemble already known menus. In order to achieve these goals tests will be performed by non computer professionals.

2.2.2 Reliability
The game is as bug free as possible. As in the usability this will be tested by non computer professionals. It will also be tested by testing tools.

2.2.3 Performance
The game is fast and responsive. The goal is to never annoy the user by unnecessary clicks or sluggish movements. The user is always in control and the character does not do actions the user has not asked for.

2.2.4 Supportability
N/A

2.2.5 Implementation
The application will use Java JDK 1.8, Gradle, and be supported by the libGDX library.

2.2.6 Packaging and installation
Every user is required to have Java JRE installed to play. They have to download the game, containing all pictures and music, in order to play it. Apart from this there is no installation required and they will only need to launch the exe file.

2.2.7 Legal
N/A

2.3 Application models

2.3.1 Use case model

See appendix 1.0 Use cases.

Nouns:

- Player
- Menu
- Level
- Block
- Obstacle(spikes)
- Star(big/small)
- Controller
- Exit-door
- Ball

2.3.2 Use cases priority

Highest

- Start
- Move
- Jump

Mid

- Start
- Level selector

Low

- Settings
- Sound

2.3.3 Domain model

See appendix 2.0 Domain model

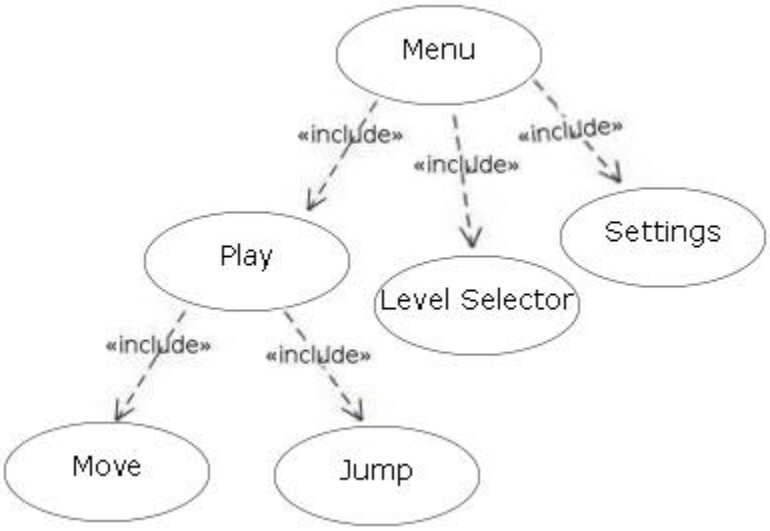
2.3.4 User interface

The user interface uses the resolution 1280x720. The GUI is not themable. The application is resizable but with no guarantee for flawless functionality. See Appendix 3.0 GUI for mockups of the menus and the game.

2.4 References

APPENDIX

1.0 Use case



Use Case: Menu

Summary: This is where the player chooses what he/she want to do.
Priority: High
Extends:
Includes:
Participators:

Normal flow

Flow 1.1: User press the start-button

	Actor	System
1.1.1	Press start in main menu	
1.1.2		Starts game

Flow 1.2: User press the level-button

	Actor	System
1.2.1	Press levels in main menu	
1.2.2		Open level selector

Flow 1.3: User press the settings-button

	Actor	System
1.3.1	Press settings in main menu	
1.3.2		Open settings menu

Alternative flow

There is no alternative flow

Exceptional flow

There is no exceptional flow

Use Case: Select level

Summary: This is where the player chooses the current level. Requires being in the “Level select”-menu.
Priority: Medium
Extends:
Includes:
Participators:.Level

Normal flow

Flow 2.1: Normal movement without obstruction

	Actor	System
2.1.1	Press a level in the level selector	
2.1.2		Start the selected level

Alternative flow

2.2 Level not unlocked

	Actor	System
2.2.1		Nothing happens

2.3 Press back-button in level selector:

	Actor	System
2.3.1	Press back in the level selector	
2.3.2		Open start menu

Exceptional flow

There is no exceptional flow

Use Case: Settings

Summary: This is where the user can change settings
Priority: Low
Extends:
Includes:
Participators:

Normal flow

Flow 3.1: Press the “Change controls”-button

	Actor	System
3.1.1	Press a level in the Controls	
3.1.2		Stars the change-control menu.

Flow 3.2: Press the “Volume”-button

	Actor	System
3.2.1	Press a level in the level selector	
3.2.2		The label besides the “Volume”-button will be writable. Write the volume you’d like

Flow 3.3: Press “Reset all”

	Actor	System
3.3.1	Press a “Reset all”-button.	
3.3.2		Resets the highscores and change controls to default.

Alternative flow

There is no alternative flow

Exceptional flow

There is no exceptional flow

Use Case: Play

Summary: This is where the game is played, a level. The character is controllable in this state.
Priority: High
Extends:
Includes: Jump, Move
Participators:

Normal flow

Flow 4.1: Character moves in y-axis

	Actor	System
4.1.1	Press jump button	
4.1.2		Character jumps

2.2 Character moves in x-axis

	Actor	System
4.2.1	Press move button	
4.2.2		Character moves in chosen direction

Alternative flow

4.3 Pause**:**

	Actor	System
4.3.1	Press pause button	
4.3.2		Open pause menu

4.3 Restart**:**

	Actor	System
4.3.1	Press restart button	
4.3.2		Character is put back in starting position

Exceptional flow

There is no exceptional flow

Use Case: Move

Summary: This is how the player, or actor, moves their Character in the world. UC Start must happen before this.
Priority: High.
Extends:
Includes:
Participators: Character

Normal flow

Flow 5.1: Normal movement without obstruction

	Actor	System
5.1.1	Hold left- or right-key	
5.1.2		Character moves to the left or right
5.1.3	Release key	
5.1.4		Character stops moving

Alternative flow

Flow 5.2: Character collide with a wall

	Actor	System
5.2.1		Character stops moving

Flow 5.3: Character collide with a lethal obstacle

	Actor	System
5.3.1		Reset character position
5.3.2		Reset time
5.3.3		Reset stars

Flow 5.4: Character takes a big star

	Actor	System
5.4.1		Character grows big

Flow 5.5: Character takes a small star

	Actor	System
5.5.1		Characther shrinks

Flow 5.6: Character collides with the exit

	Actor	System
5.6.1		Display score screen

Flow 5.6.1: Score screen

	Actor	System
5.6.1.1	Press “Next Level”	
5.6.1.2		Load next level

Flow 5.6.2: Score screen

	Actor	System
5.6.2.1	Press “Retry”	
5.6.2.2		Load this level

Flow 5.6.3: Score screen

	Actor	System
5.6.3.1	Press “Home”	
5.6.3.2		Goes to home screen

Exceptional flow

There is no exceptional flow

Use Case: Jump

Summary: This is how the player, or actor, jumps with their Character in the world.
Priority: High
Extends:
Includes:
Participators: Level

Normal flow

Flow 6.1: Character jumps

	Actor	System
6.1.1	Press up-key	
6.1.2		Ball moves in the y-axis

Alternative flow

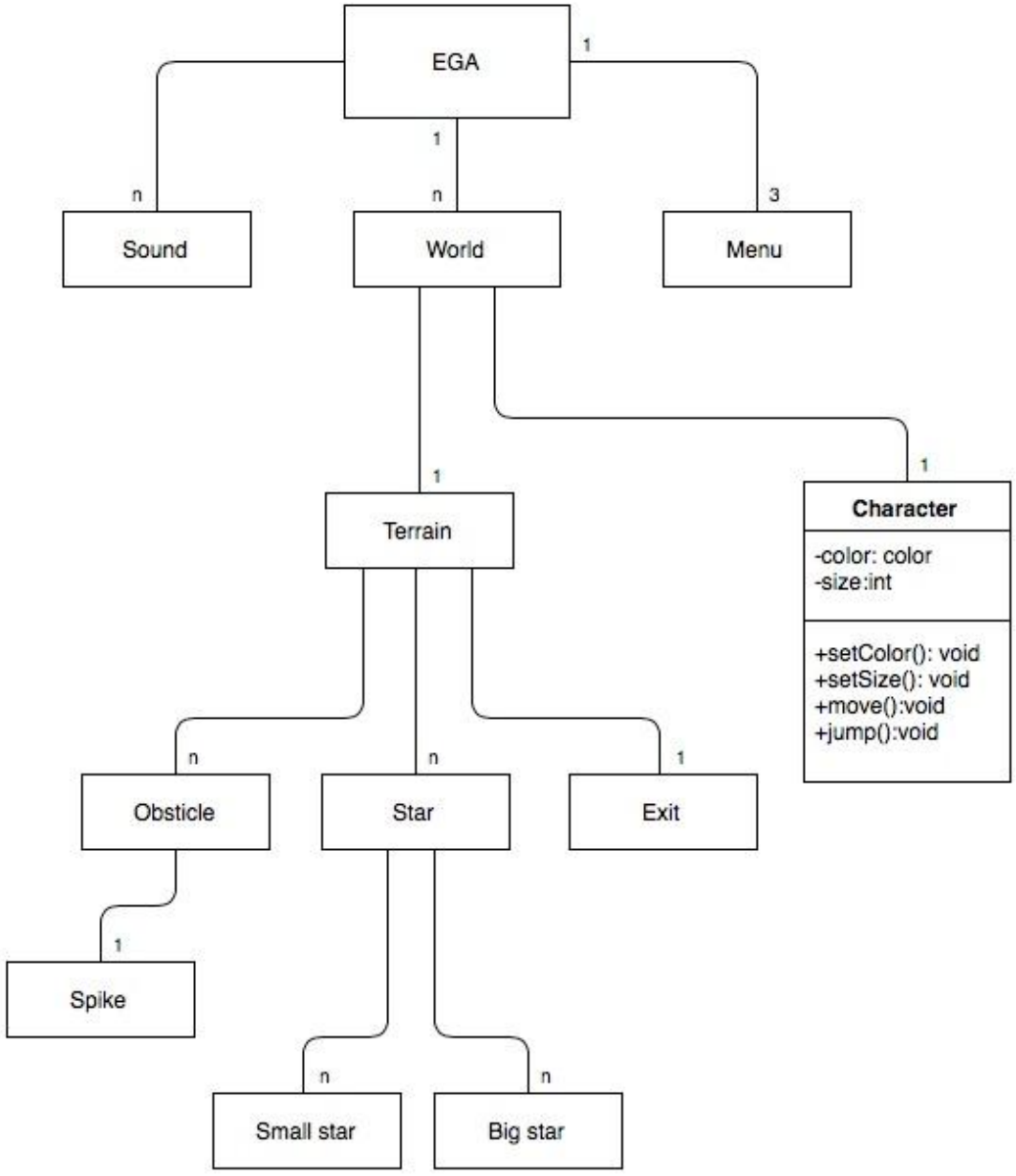
Flow 6.2: Character moves rigth or left in jump

	Actor	System
6.2.1	Press up-key and rigth-/leftkey	
6.2.2		Ball moves in the x- and y-axis

Exceptional flow

There is no exceptional flow

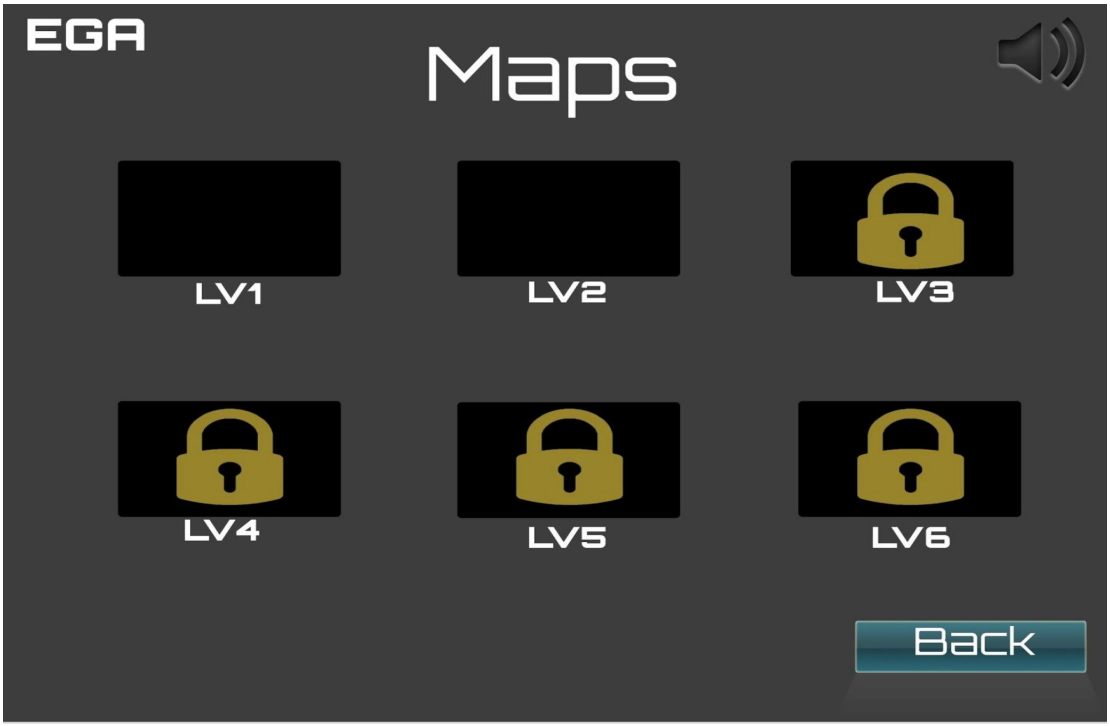
2.0 Domain model

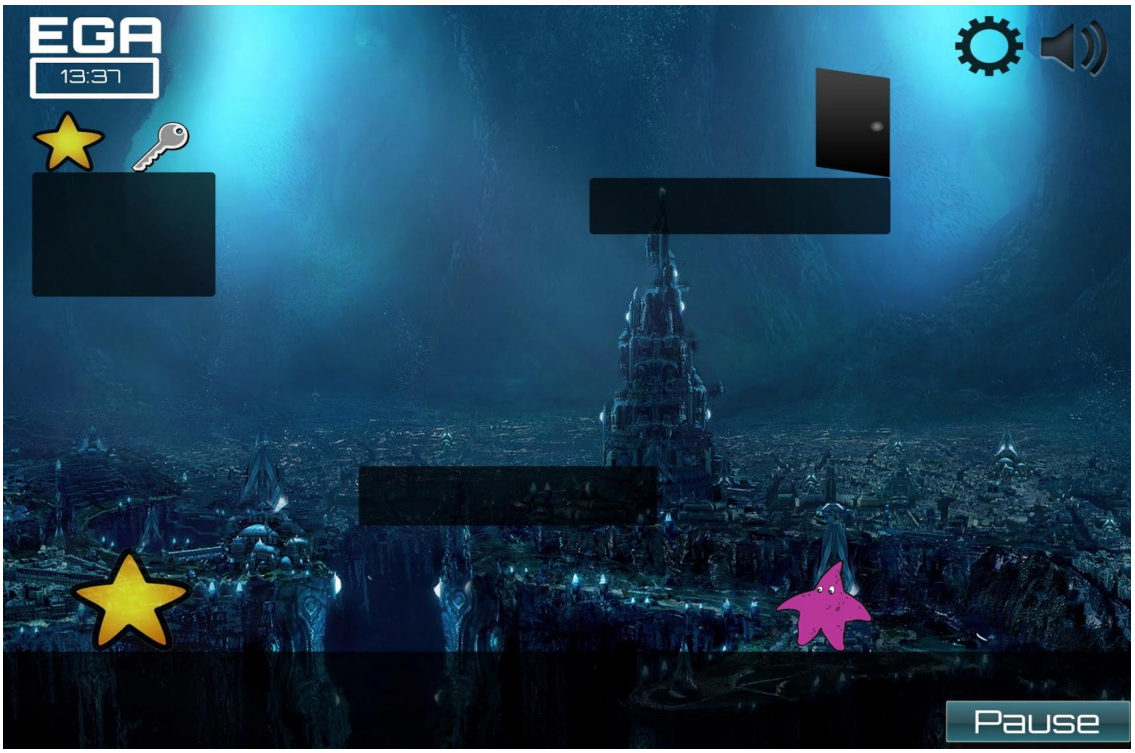


.

3.0 GUI

Mockups of the preliminary GUI




EGA
13:37
Pause