Requirements and Analysis Document for group 16

Erik Sjöström, Filip Labe, Jonatan Källman, Sarosh J. Nasir

May 17, 2017
v1.0

1 Introduction

Our project is a game about defeating monsters. You do this by clicking on the monsters with your finger. When defeated the monsters drop gold which can be used to buy upgrades. There are three upgrades that can be bought, upgrades to your damage, upgrades which provide damage every second or upgrades which provide gold every second. When the player has defeated enough
monsters the player can progress to a new area and defeat monsters in a different scenery.

1.1 Definitions, acronyms, abbriviations

2 Requirements

2.1 User interface

2.2 Functional requirements

- Attack
- Open map
- Use map
- Buy upgrades
- Go home
- Improve home
- Check stats

2.3 Non-functional requirements

2.3.1 Usability

This will be a mobile game so we should put extra emphasis on the game being as intuitive as possible.

2.3.2 Performance

The game should be light on resources.

2.3.3 Implementation

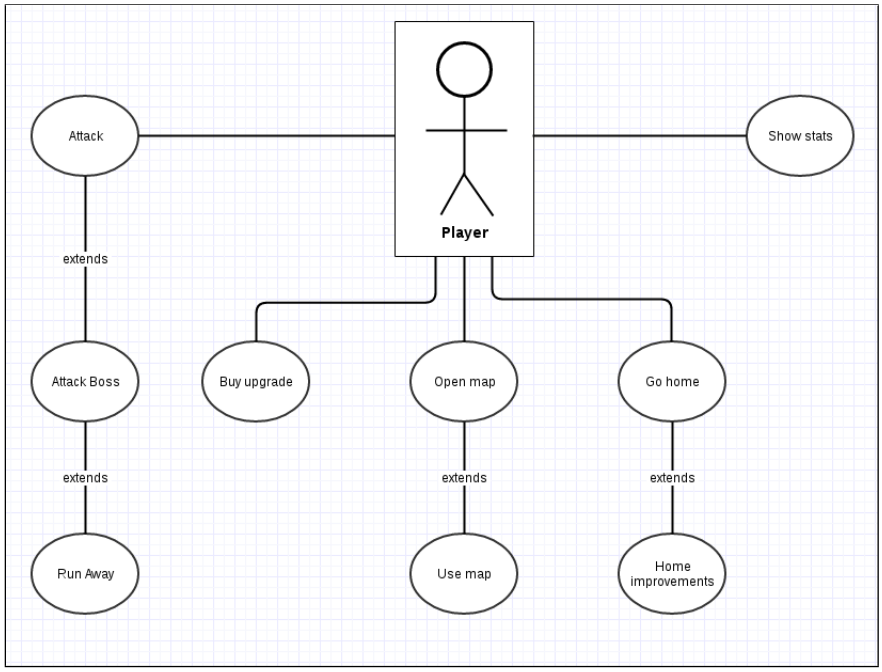
The application will be written in Java.

2.4 Packaging and installation

The application will be distributed as an *.apk-file*.

3 Use cases

3.1 Usability model



3.2 Use case listings

3.2.1 Use case: Open app

Summary: The user opens the app
Priority: High
Extends:
Includes:
Participators: User
Normal flow of event: *The user opens the app and the game displays the main menu.*

	Actor	*System*
1	*User clicks app icon*	
2		*Load game*
3		*Display main menu*

3.2.2 Use case: Attack

Summary: The user clicks the screen and the game calculates new health for the monster.
Priority: High
Extends: undefined
Includes: undefined
Participators: Player
Normal flow of event: *The player attacks the monster and the monster survives.*

	Actor	*System*
1	*Player clicks screen*	
2		*Calculate new health for monster*

Alternate flow: *The player attacks the monster and the monster dies. The level is not cleared.*

	Actor	*System*
1	*Player clicks screen*	
2		*Calculate new health for monster*
3		*Replace dead monster with new monster*
4		*Player gets money*

Alternate flow: *The player attacks the monster and the monster dies. The level is cleared.*

	Actor	*System*
1	*Player clicks screen*	
2		*Calculate new health for monster*
3		*Replace dead monster with new monster*
4		*The option to change level is made avliable*
5		*The player gets money*

3.2.3 Use case: Attack boss

Summary: The player is fighting a boss
Priority: High
Extends: Attack
Includes:
Participators: Player
Normal flow of event: *The player attacks the boss and defeats it within the time frame*

	Actor	*System*
1	*Player clicks screen*	
2		*Calculate new health for boss*
3		*Unlock next level*
4		*Player gets money*

Normal flow of event: *The player attacks the boss and does not defeat it, but is still within the time frame.*

	Actor	*System*
1	*Player clicks screen*	
2		*Calculate new health for boss*

Alternate event: *The player does not defeat the boss within the time frame*

	Actor	*System*
1		*Reset boss*

3.2.4 Use case: Open map

Summary: The player clicks the map button Priority: Medium
Extends: undefined
Includes: undefined
Participators: Player
Normal flow of event: *The player clicks the map button*

	Actor	*System*
1	*Player clicks on the button*	
2		*Displays map page*

3.2.5 Use case: Use map

Summary: The user tries to move to a different level.
Priority: High
Extends: undefined
Includes: undefined
Participators: Player
Normal flow of event: *The player clicks on the level they want to move to, the level is unlocked.*

	Actor	*System*
1	*Player clicks on the level*	
2		*Checks that the level is unlocked*
3		*Loads new level*

Alternate flow of event: *The player clicks on the level they want to move to, the level is locked.*

	Actor	*System*
1	*Player clicks on the level*	
2		*Checks that the level is unlocked*

3.2.6 Use case: Show stats

Summary: The use clicks the stats button
Priority: low
Extends: undefined

Includes: undefined
Participators: Player
Normal flow of event: *The player clicks the stats button*

	Actor	*System*
1	*Player clicks on the button*	
2		*Displays stats page*

3.2.7 Use case: Go home

Summary: The user wants to go home
Priority: Low
Extends: undefined
Includes: undefined
Participators: Player
Normal flow of event: *The player clicks the home button*

	Actor	*System*
1	*Player clicks on the button*	
2		*Displays the player home screen*

3.2.8 Use case: Home improvements

Summary: The user wants to improve their home
Priority: low
Extends: undefined
Includes: undefined
Participators: Player
Normal flow of event: *The player wants to buy an upgrade, has enough money.*

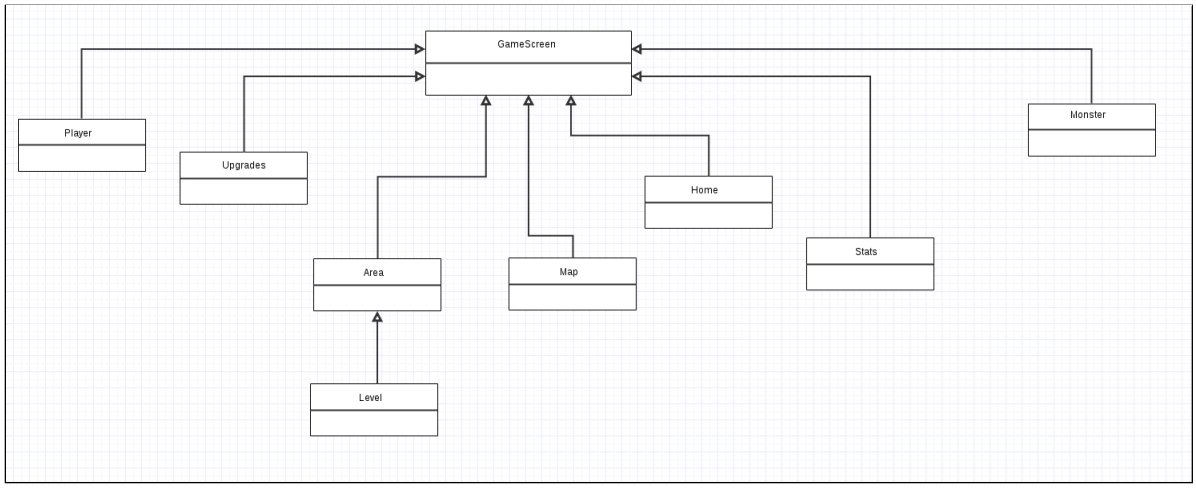
	Actor	*System*
1	*Player clicks on a buy upgrade button*	
2		*Deduct money from player*
3		*Apply upgrade to player*

3.2.9 Use case: Buy upgrade

Summary: The user is at the shop and wants to buy an upgrade.
Priority: Medium
Extends: undefined
Includes: undefined
Participators: Player
Normal flow of event: *The player clicks on the upgrade and has enough money.*

	Actor	*System*
1	*Player clicks on the upgrade*	
2		*The upgrade is applied to the player*

4 Domain model



4.1 Class responsibilities

4.1.1 Game screen

Combines all other classes into one unified interface.

4.1.2 Player

The actual player. Has money, an amount of damage, an amount of damage per second and an amount of money per second.

4.1.3 Monster

A bad guy. Has health points, and an amount of money to drop.

4.1.4 Stats

A view for the players stats.

4.1.5 Map

A view for a map over the areas that a player can travel to.

4.1.6 Upgrades

A view over the upgrades that a player can buy.

4.1.7 Home

The players home. Here the player can upgrades which earn passive money.

4.1.8 Area

An area, has a name, and a list of levels associated with that area.

4.1.9 Level

Determines how hard the monsters are, and how much money they drop.

5 References

No references yet.