RAD

*Requirements and Analysis Document for **Falling***

Version: 1.0
Date: 2016-05-29
Author: Group 18

This version overrides all previous versions.

1 Introduction

1.1 Purpose of application

The project aims to create a game that simulates skydiving, with game features such as a points system that makes it an enjoyable game.

1.2 General characteristics of application

The application is a single player first person virtual reality mobile game for Android where the player simulates performing a parachute jump. Each jump consists of an initial stationary phase, a free fall phase, a parachute phase and a landing.

There are balloons placed along the way that the player can collect for points, and mines that should be avoided. The ground the player should land on is a sandy island in the ocean, and landing in the ocean, or without pulling the parachute, will result in “game over”.

The game will be played using Google Cardboard Virtual Reality-headset (Google, 2016a) and an Android phone. The player will steer by moving his/her head and tapping the screen.

1.3 Scope of application

The application contains only one level and one game mode. There are no menus, nor any other game content.

1.4 Objectives and success criteria of the project

For the project to be considered a success the following criterias must be fulfilled:

- It should be possible to perform a jump (see definition at 1.5).
- It should be possible to start a new jump when one is completed.
- The simulated world should look and behave realistically at all stages and heights of a jump. This includes both graphical and simulated properties.

1.5 Definitions, acronyms and abbreviations

- *VR* – virtual reality, specifically the use of a mobile device placed inside Google Cardboard glasses
- *Jump* – a jump is going through all the phases: pre-jump, free falling, parachute falling and landing.

2 Requirements

2.1 Functional requirements

The player should be able to:

1. Execute a jump
    a. Jump off plane
    b. Move in the air while falling by looking around
    c. Pull parachute
    d. Move while having the parachute out by tilting their head.
    e. Land
2. Collect objects
    a. Balloons, which will give the player points
    b. Mines, which will reset the player's bonus multiplier
3. Start a new jump

2.2 Non-functional requirements

2.2.1 Usability

As *Falling* is a mobile game the game should be easy to play without too much explanation. This implies that the game controls should be as intuitive as possible, and that the game (instructions) will be short and in English.

2.2.2 Reliability

The application is not a critical system and has no specific reliability requirements. Problems with reliability across different Android phones is not a major issue.

2.2.3 Performance

As this is a *VR*-application special attention must be exercised to avoid "lag" (uneven and low framerate). Lag in *VR* can lead to *simulator sickness* which is very uncomfortable for the user (Google, 2016b).

To make sure the application runs smoothly and Android-devices it is important to consider:

- The complexity of 3D models
- The resolution of textures

More complexity in the models and higher resolution if the textures can lead to more GPU processing which is very limited on a smartphone. The rendering in general needs to be simple and optimized to make sure it runs smoothly.

2.2.4 Supportability

The application must be separated from its platform requirements to enable future changes in application scope, platform change, or other major changes. The game will be written mainly for Google Cardboard on Android (see 2.2.5 for more information), but the majority of the code should be agnostic of all platform requirements.

There should be automated unit tests covering as much of the application code as possible. However, games like this application are not very suited for such tests, and much of the testing will have to be done manually.

2.2.5 Implementation

The application will be written in Java using the libGDX game engine/library (Badlogic Games, 2016). To incorporate VR Google Cardboard will be used. More details and motivation can be found in section 2.1 of the SDD for *Falling*.

2.2.6 Packaging and installation

As the application is developed for Android it will be distributed as an *.apk*-file (Android application package file). This file contains the whole application, including assets, and can be installed into an Android phone directly.

Since the desktop version is only for development no end-user distribution of this executable will be performed.

2.2.7 Legal

Because of limited time and resources some graphical assets protected under copyright law will be used. This will however not be an issue: since the application is produced solely for educational purposes, the usage of the mentioned assets can be considered fair use and therefore noninfringing.

2.3 Application models

2.3.1 Use case model

See APPENDIX.

2.3.2 Use case priority

1. Fall
2. Drop (start)
3. Move
4. Pull parachute
5. Fall (with parachute)
6. Move (with parachute)
7. Land
8. Turn parachute
9. Restart game
10. Pick up object

2.3.3 Domain model

See APPENDIX for UML diagram representing the domain model.

2.3.4 User interface

The application is a game controlled mainly by movement and does therefore not make use of a normal user interface. Most user interface will be represented by in-game objects; some text will however be presented on the screen in a normal fashion. Special precautions must be made to make sure the user interface is visible on most Android phone screens. See APPENDIX for an example of user interface as in-game objects.

2.4 References

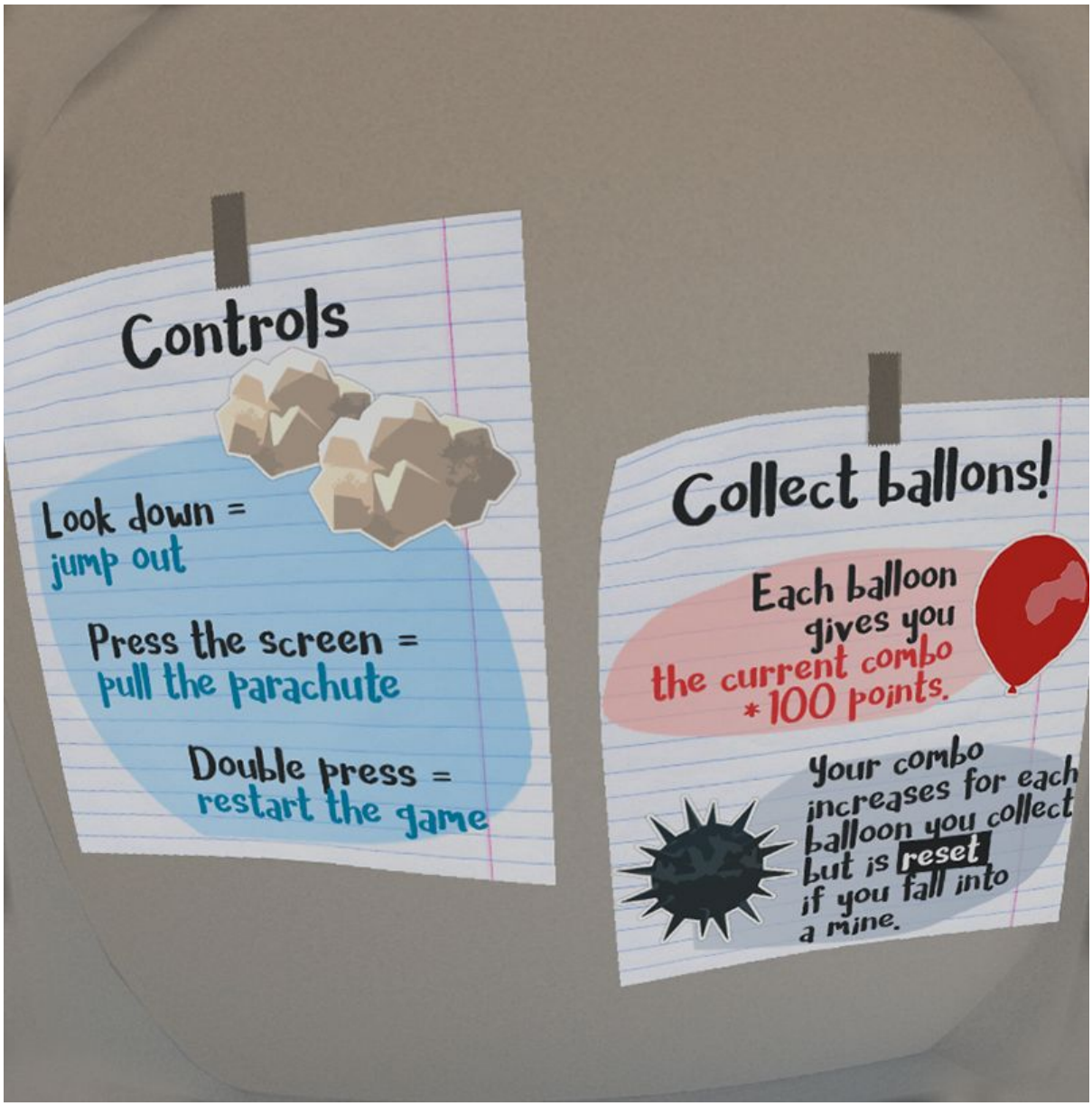
Badlogic Games (2016). *libGDX*. Accessed from https://libgdx.badlogicgames.com/ at 2016-05-25.

Google (2016a). *Google Cardboard – Google VR*. Accessed from https://vr.google.com/cardboard/ at 2016-05-26.

Google (2016b). *Designing for Google Cardboard*. Accessed from https://vr.google.com/cardboard/ at 2016-05-26.

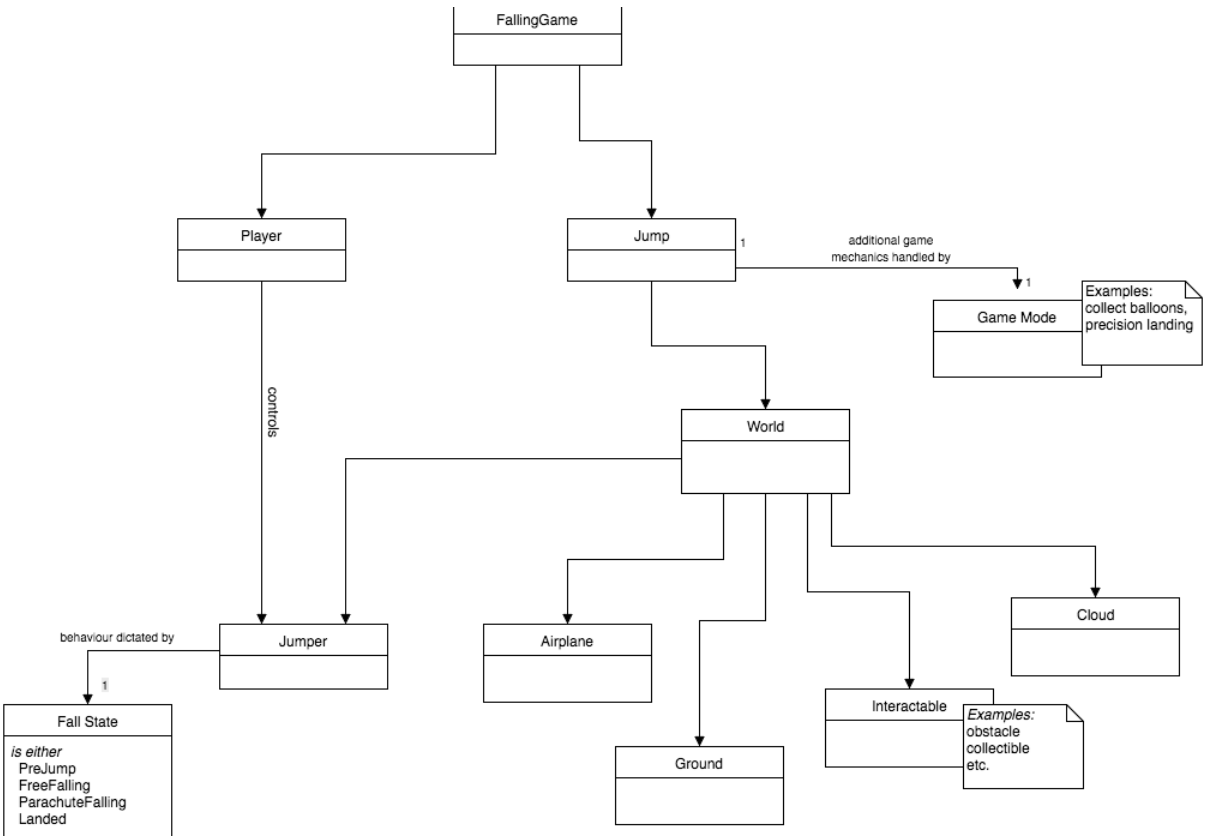
APPENDIX

GUI



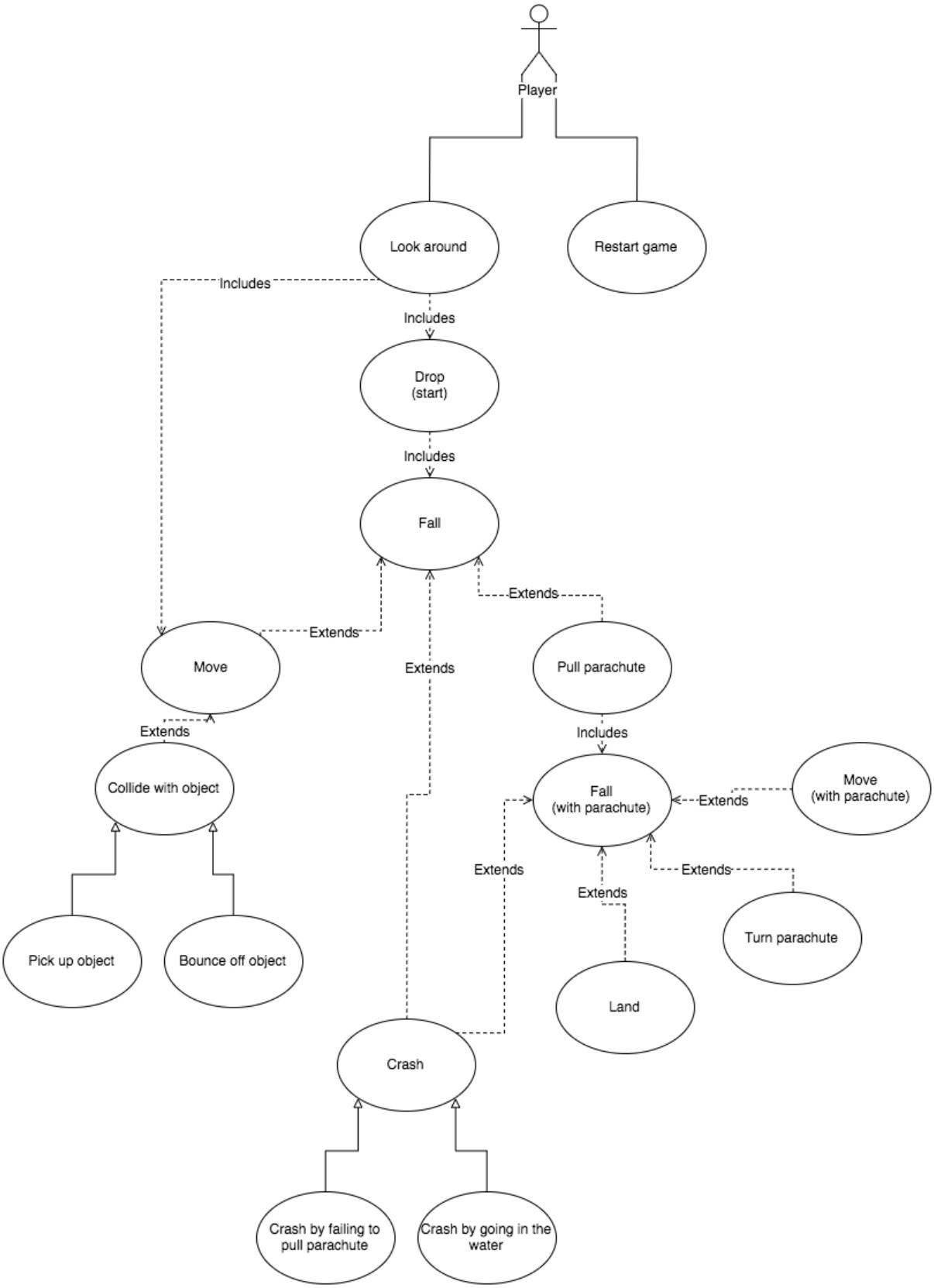
The figure shows how in-game objects (papers in this case) are used to build up the graphical user interface.

Domain model



The above UML diagram represents the domain model of the application.

Use case model



The UML above is a diagram of all the implemented use cases.

Use cases

Use case nr. 1

DROP (start)

Summary: The player jumps out of the plane, by looking down. (Can only be done once per jump)

Priority: High

Extends:

Includes: FALL

Participators: Player

Normal flow of events

	Actor	**System**
1	Player tilts head downwards, looking out of the plane	
2		Player jumps out of the plane
3		Falling speed increases until terminal velocity is reached

Use case nr. 2

FALL

Summary: The player continuously falls toward the ground.

Priority: High

Extends:

Includes:

Participators: Player

Normal flow of events

	Actor	**System**
1		The player moves toward the ground

Use case nr. 3

MOVE

Summary: The player moves in the direction they are looking during free fall.

Priority: High

Extends: FALL

Includes:

Participators: Player

Normal flow of events

	Actor	**System**
1		The player moves in the direction which they are looking

Use case nr. 4

PICK UP OBJECT

Summary: The player collides a collectible object, which increases the score, and is notified with a sound

Priority: Low

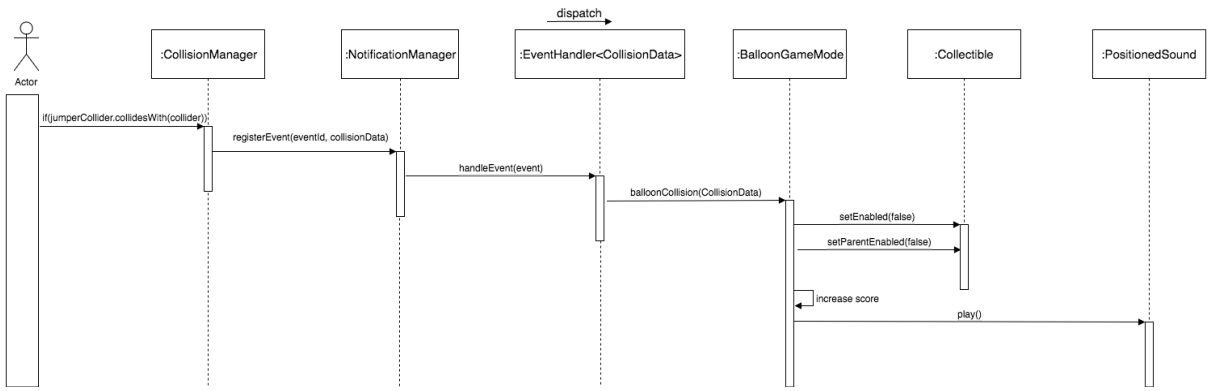
Extends: COLLIDE WITH OBJECT

Includes:

Participators: Player

Normal flow of events

	Actor	**System**
1	Player touches collectible object	
2		Remove object
3		Increase score
4		Play sound



A sequence diagram of this particular use case.

Use case nr. 5

PULL PARACHUTE

Summary: The player pulls the parachute and the speed is changed. (Can only be done once per jump)

Priority: High

Extends: FALL

Includes: FALL (with parachute)

Participators: Player

Normal flow of events

	Actor	**System**
1	Presses the screen (while in free fall)	
2		Change state to parachute state
3		The speed is reduced

Use case nr. 6

FALL (with parachute)

Summary: A slower fall toward the ground.

Priority: High

Extends:

Includes:

Participators: Player

Normal flow of events

	Actor	System
1		Moves the player toward the ground

Use case nr. 7

MOVE (with parachute)

Summary: The player moves forward during parachute fall.

Priority: High

Extends: FALL (with parachute)

Includes:

Participators: Player

Normal flow of events

	Actor	System
1		The player moves forward

Use case nr. 8

TURN PARACHUTE

Summary: The player tilts their head while falling with the parachute and the direction of the parachute changes

Priority: Medium

Extends: FALL (with parachute)

Includes:

Participators: Player

Normal flow of events

	Actor	**System**
1	Tilts their head (and the phone by extension)	
2		Changes the direction of movement

Use case nr. 9

LAND

Summary: When the player reaches the ground, if the parachute has been pulled, the player slows to a stop.

Priority: High

Extends: FALL (with parachute)

Includes:

Participators: Player

Normal flow of events

	Actor	**System**
1	Player reaches the ground	
2		Change state to landed state
3		Immediately stops falling, and forward motion slows to a stop

Use case nr. 10

RESTART GAME

Summary: Restarts the game

Priority: Medium

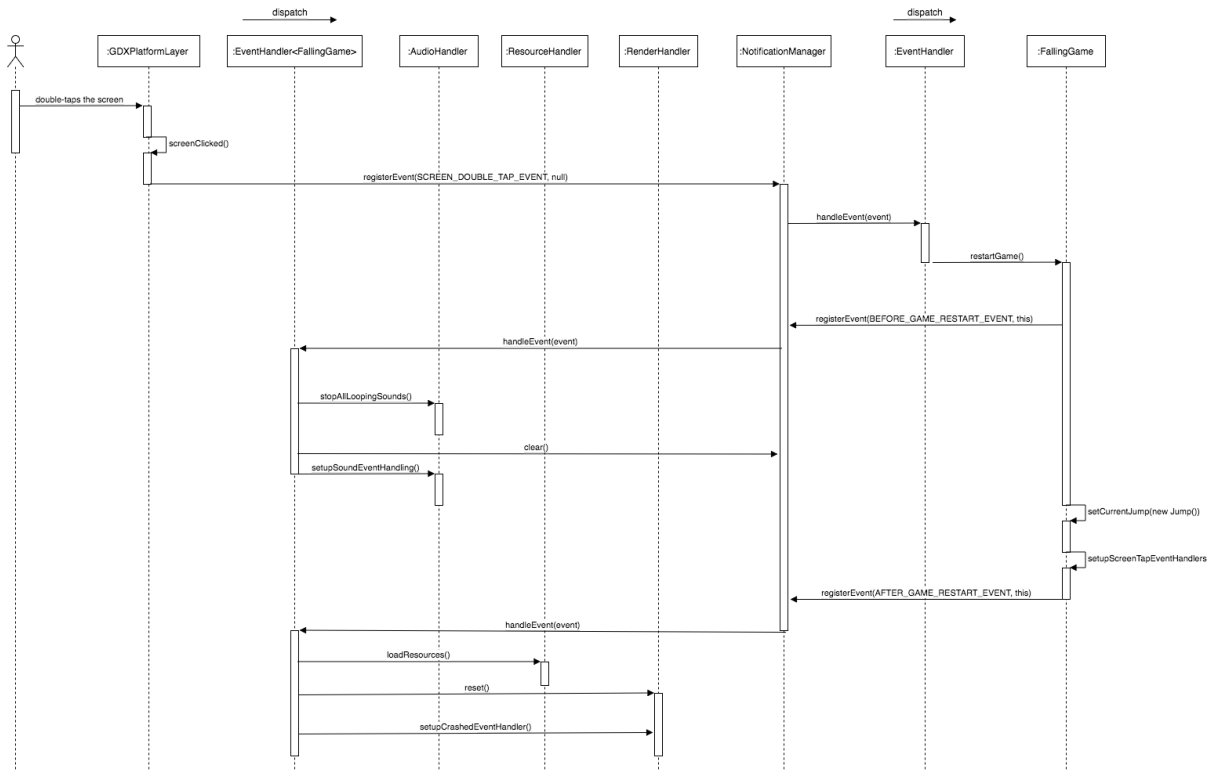
Extends:

Includes:

Participators: Player

Normal flow of events

	Actor	System
1	Player double-taps the screen	
2		The game restarts



A sequence diagram of this particular use case.