RoboJump – System Specification

Project Name	AhPrSt
Project Manager	Fabian Ahammer
Document Owner	Christian Steyrer
Created on	
Last modified	04.02.2018
State	X in Work vorgelegt Released
Dokumentablage	SystemSpecificationTemplate.doc

Content

1 Initial Situation and Goal

1.1.1 Initial Situation

There are many jump and run games on the market, but in most cases, the player must invest a lot of time to achieve something.

Most games have a huge disadvantage in the time aspect, the levels often have a playing time of 15-30 minutes.

Game:	**Average time for a level:**	**Platforms:**
Donkey Kong	10 minutes	Nintendo platforms
Super Mario Galaxy	35 minutes	Nintendo Wii
Shovel Knight	15 minutes	Nintendo (3DS, WiiU , Switch) PlayStation (3, 4) Xbox One, Fire OS, Windows, macOS, Linux
New Super Mario Bros.	6,6 minutes	Nintendo Wii, WiiU

All these Games have an average playing time over 10 minutes, expect Super Mario Bros. but that is only available on Nintendo platforms.

Robo Jump will be oriented on some of these games and tries to take the positive aspect and to improve the negative ones.

An important aspect is that the game is available to most people, therefore it will be mainly available to PC and not to consoles.

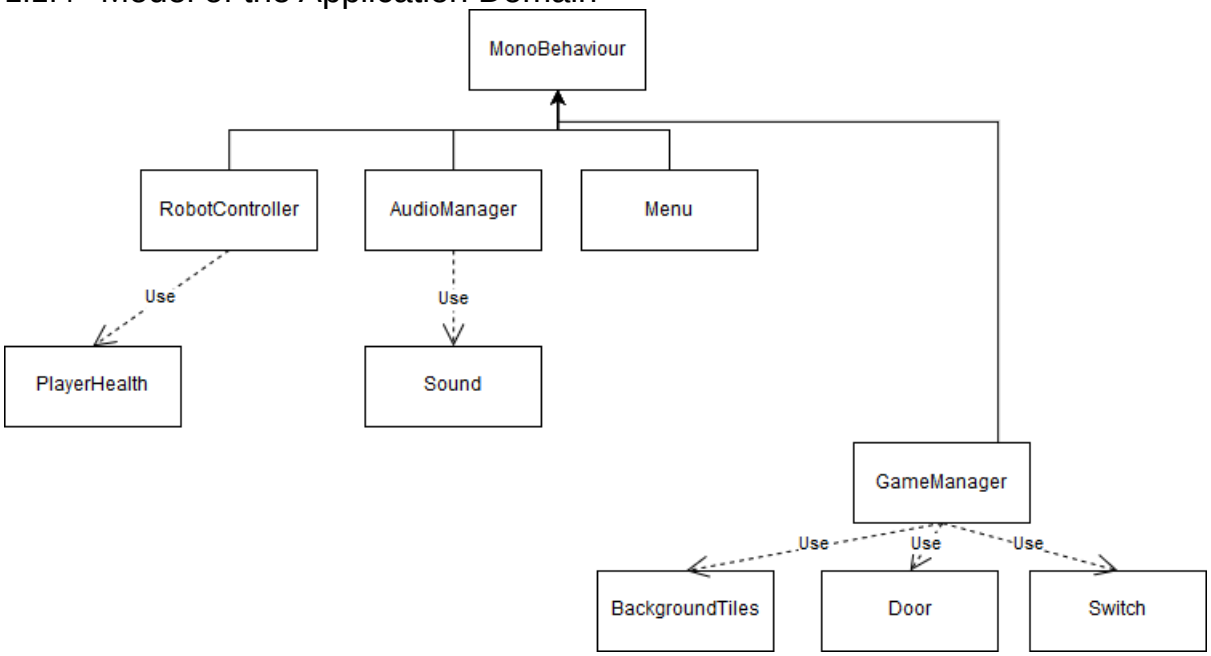
So Robo Jump will have the advantage that the levels are playable in under 5 minutes and with that it is perfect for casual gamers which love to play in breaks and in short waiting times.

1.1.2 Application Domain

Robo Jump is mainly for students, which sometimes play some games in breaks.
Commonly these breaks are around 5-15 minutes so it highly important that that the levels have low playing times around 2-3 minutes.

1.1.3 Glossary

1.1.4 Model of the Application Domain



1.1.5 Goal Definition

Our goal is to create a game which someone can use to kill waiting times or breaks, so the playing time should be reduced to 2-3 minutes per level.

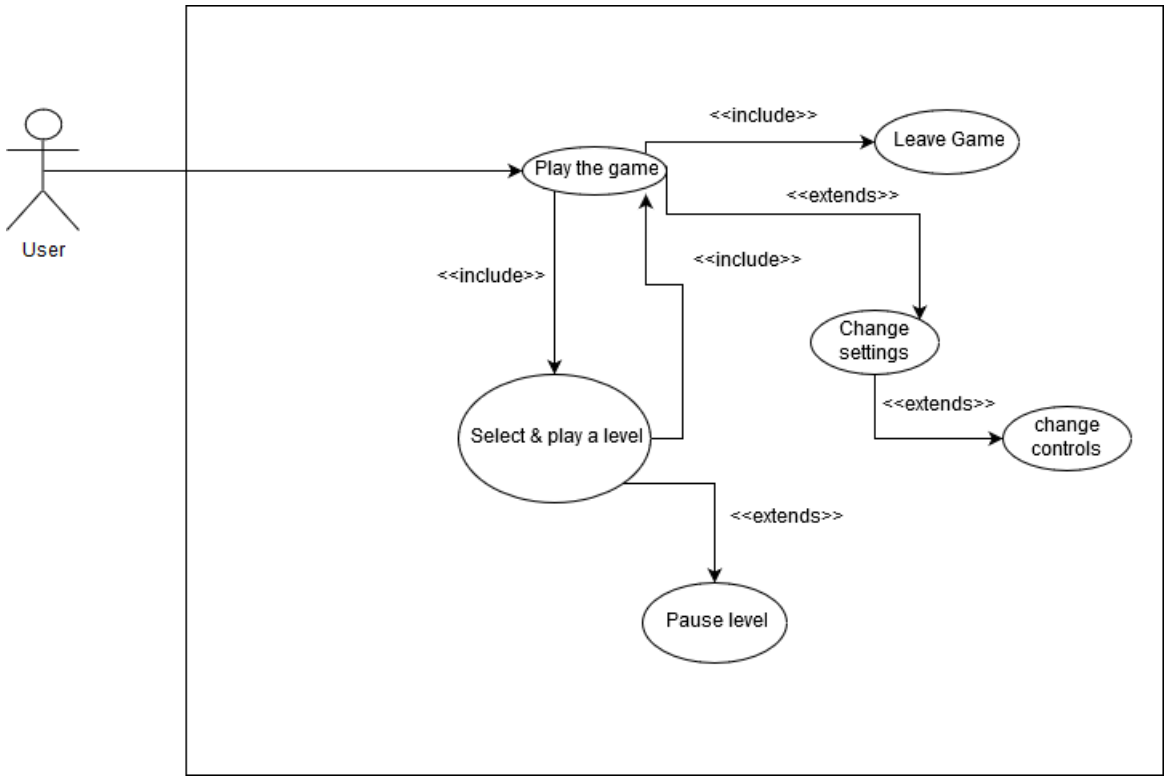
When finished it will be published as a Unity game for Windows and possibly later for Mac OS and Linux.

The game should be relatively easy to use as it will contain a short tutorial and user-friendly and known controls.

2 Functional Requirements

2.1.1 Use Case Diagram

Overview of all use cases in a use case diagram:



2.1.2 Use Case Details

2.1.2.1 Use Case ID 1 (Play the game)

From the Main Menu the player calls the other use cases.

Main Menu:

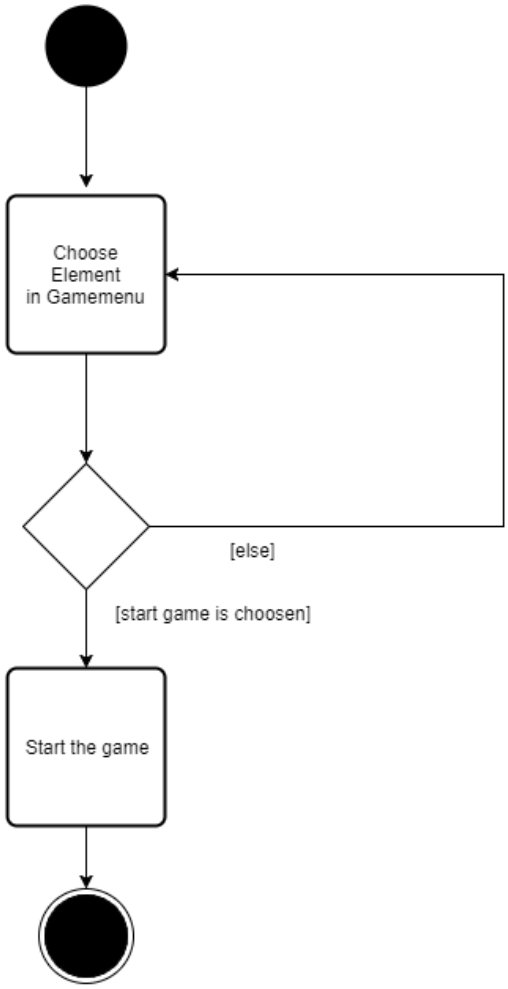
Input field	Calls
Play button	Use Case 2
Settings button	Use Case 5
Exit button	Use Case 4

Characteristic Information:

Goal:	This is the central Use Case which simply say that the player currently uses the program.
Precondition:	The condition for this Use Case is that the game does not run currently.
Postcondition:	When this Use Case will be executed the player is in the main menu
Involved User:	Role name: “Player”
Triggering Event:	The Use Case will be executed when the program is started.

Standard scenario:

Step	User	Activity
1	Player	choose game in menu
2	System	starts the game
3	Game	shows the main menu



2.1.2.2 Use Case ID 2 (Select & play a level)

To play the game the player needs to select and play a level.

GUI to call the use case:

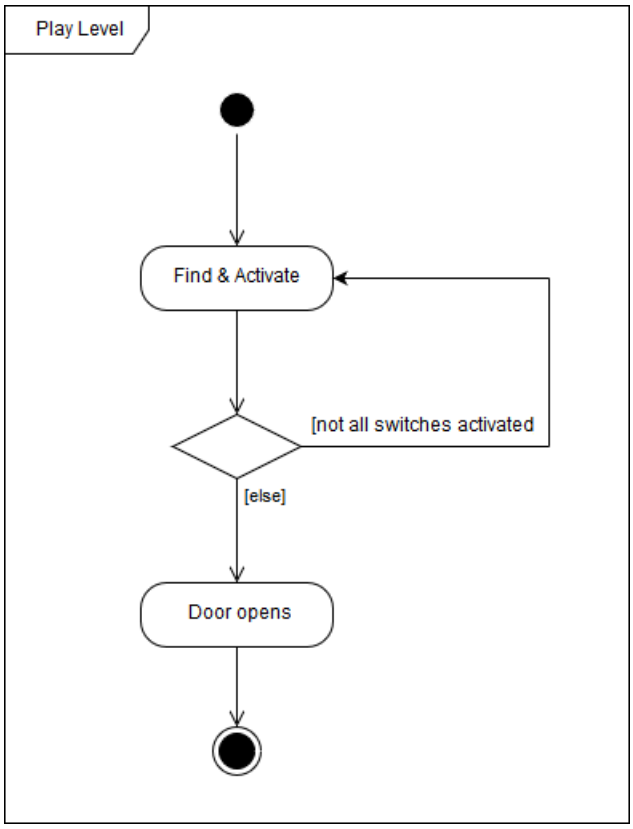
Characteristic Information:

Goal:	The goal of this Use Case is to give the player information on his progress and to get the information which level he wants to play.
Precondition:	The condition for this Use Case is that the player is currently in the main menu.
Postcondition:	When this Use Case will be executed the player is at the level start from the chosen level.
Involved User:	Role name: “Player”
Triggering Event:	The Use Case will be executed when the player enters the level menu and then start and play a level.

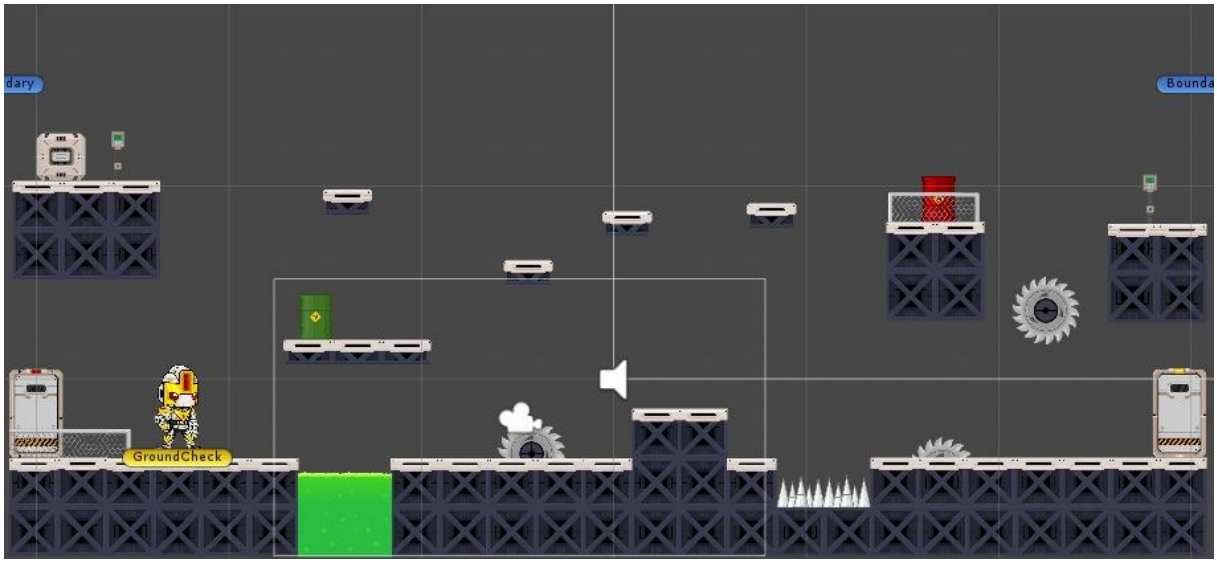
Standard scenario:

Step	User	Activity
1	Player	choose level in menu
2	Game	starts the level
3	Player	find and activate switches
4	Game	opens the door (level goal) if all switches are activated

Workflow of the levels:



Screenshot of the first level:



2.1.2.3 Use Case ID 3 (Pause level)

While playing a level the player has the possibility to pause it.

Characteristic Information:

Use Case ID 3:

Goal:	The goal of this Use Case is that the player has the possibility to freeze the game actions, so that he can continue after a break.
Precondition:	The condition for this Use Case is that the player currently plays a level
Postcondition:	When this Use Case will be executed the level pause, this means that all actions freeze.
Involved User:	Role name: “Player”
Triggering Event:	The Use Case will be executed when the player hits the “ESC”-button.

Standard scenario:

Step	**User**	**Activity**
1	Player	hit “ESC”-button
2	Game	freeze the level
3	Game	open pause menu
4	Player	hit “ESC”-button again
5	Game	unfreeze the level

2.1.2.4 Use Case ID 4 (Leave Game)

The game can be leaved every time during playing.

Characteristic Information:

Goal:	The goal of this Use Case is to leave the game.
Precondition:	The condition for this Use Case is that the game currently runs.
Postcondition:	When this Use Case will be executed the game saves and then close the program.
Involved User:	Role name: “Player”
Triggering Event:	The Use Case will be executed when the player hits the “Exit”-button.

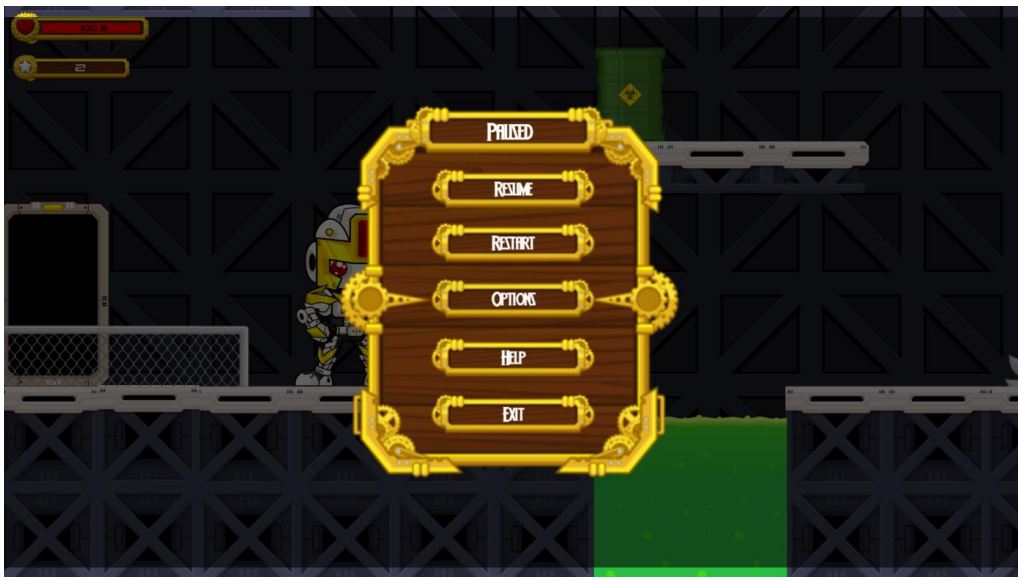
Standard scenario:

Step	**User**	**Activity**
1	Player	hit “Exit”
2	Game	Save data
3	System	Close game

2.1.2.5 Use Case ID 5 (Change settings)

It is possible to change the settings.

GUI to call the use case:



Characteristic Information:

Goal:	The goal of the Use Case is to change settings.
Precondition:	The condition for this Use Case is that the player is currently in the main or pause (ESC) menu.
Postcondition:	When this Use Case will be executed the player will have the possibility to change game.
Involved User:	Role name: “Player”
Triggering Event:	The Use Case will be executed when the player hits the settings button.

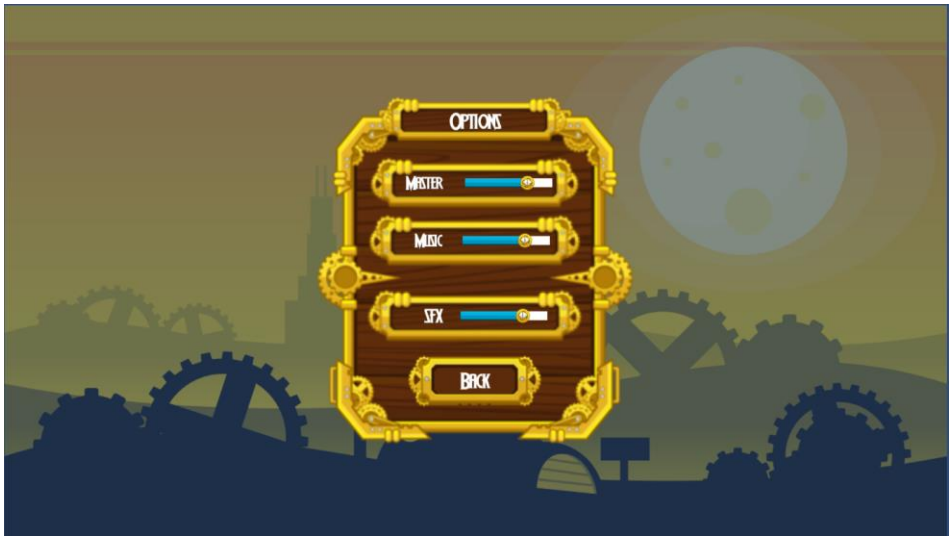
Standard scenario:

Step	**User**	**Activity**
1	Player	hit “options”-button
2	Game	open options

2.1.2.6 Use Case ID 6 (Change controls)

In the settings it is possible to change controls

GUI to call the use case:



Characteristic Information:

Goal:	The goal of this Use Case is to change game controls.
Precondition:	The condition for this Use Case is that the player needs to be in the settings menu.
Postcondition:	When this Use Case will be executed the game-controls could be on other values.
Involved User:	Role name: “Player”
Triggering Event:	The Use Case will be executed when the player changes the sliders positions.

Standard scenario:

Step	**User**	**Activity**
1	Player	change position of sliders
2	Game	change game-controls

3 Non-functional Requirements

Type USE: Usability requirement

To make Robo Jump as useable as possible it must guarantee some criteria.

- The game should not require more than 1 GB memory.
- The levels of the game should not have a long loading time because nobody will wait more than about 5 seconds.
- The appearance should be oriented on other games, so players do not have to spend a lot of time on learning game irrelevant controls.

Type EFFIC: Efficiency requirement

As before mentioned in the usability requirements the game levels must load in 5 seconds.

Type MAINT: Maintenance and portability requirements

As it is planned to translate the game to German when it is finished it should be especially maintainable at this. Eventually it will be available on other platforms, so the controls should be capable to handle other input methods.

Type SEC: Security requirement

As Robo Jump does not use online services and it is not planned to add online features in the future, it is not required to be the best secured program.

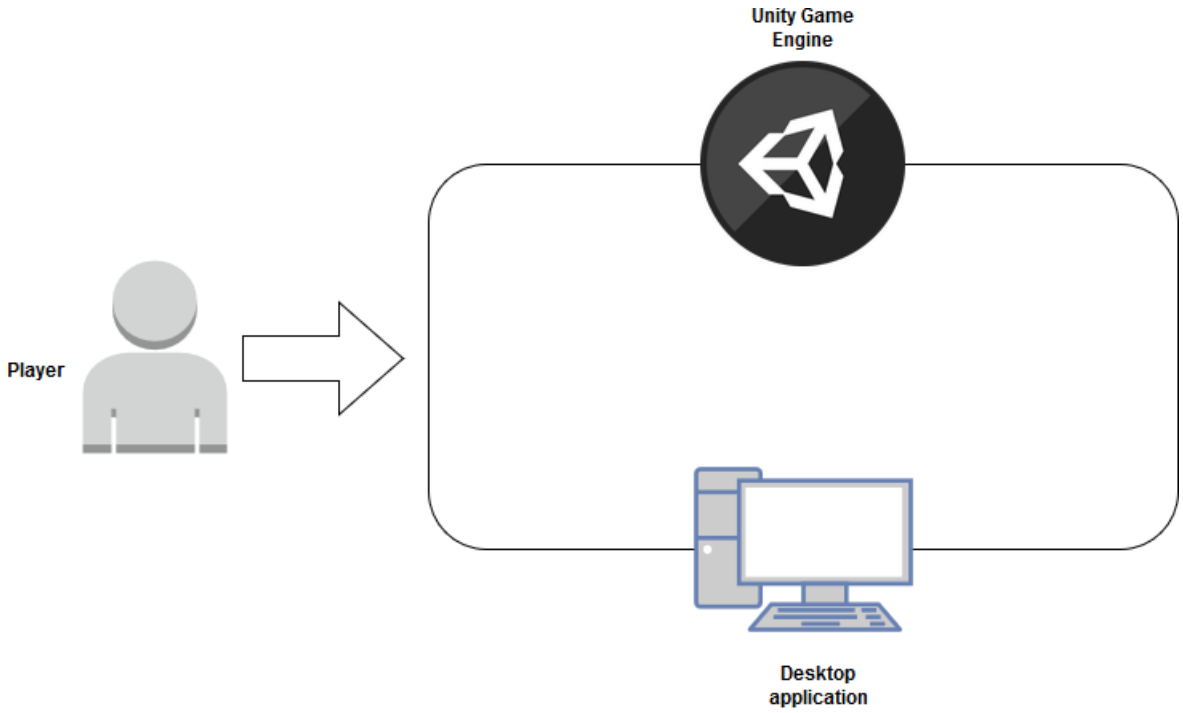
Type LEGAL: Legal requirement
Laws or standards, which could be important, are not known yet.

4 Quantity Structure

The main data of Robo Jump will be the game assets for each level.

Conclusion
We expect that the total size of Robo Jump will be around 1 GB disk space.

5 System Architecture and Interfaces



The Unity Game Engine is a main part of the system, it powers the Desktop application which is finally used by the player to play the game.

6 Acceptance Criteria

Use case 1

Test Step	**Expected Behaviour**
Start game	Main menu should be shown

Use case 2

Test Step	**Expected Behaviour**
Load levels	All levels can be loaded
Play levels	All levels are playable

Use case 3

Test Step	**Expected Behaviour**
Pause level	Level should be freezed

Use case 4

Test Step	**Expected Behaviour**
Close game	Game should save the current state

Use case 5

Test Step	**Expected Behaviour**
Open settings menu	All available settings should be shown

Use case 6

Test Step	**Expected Behaviour**
Change controls	Game should recognize and change settings