Requirements and Analysis Document for Alesk The Unicorn

Version: 1.0
Date: 2018-04-13
Author: Johannas Soldater
This version overrides all previous versions.

1 Introduction

Alesk the Unicorn is a fast paced, whimsical 2D platform game where the player controls Alesk, a unicorn on the hunt for lunch boxes. On his way he can find different powerups that he will use to minimize the time spent on each level.

The game is a focused experience, meant to be played in short sessions whenever there is time to kill. Therefore the different levels will be independent of each other and self contained units. The goal of each level will be to minimize the time spent to collect all lunchboxes while avoiding the enemies spread throughout the levels or falling off the world.

1.2 Definitions, acronyms and abbreviations

- Player character = Alesk The Unicorn itself
- Level = A standalone instance of the game, containing all game entities such as blocks, player, enemies etc. They do not need to be interconnected or coherent.

2 Requirements

2.1 User interface

2.1.1 The graphical user interface of the game

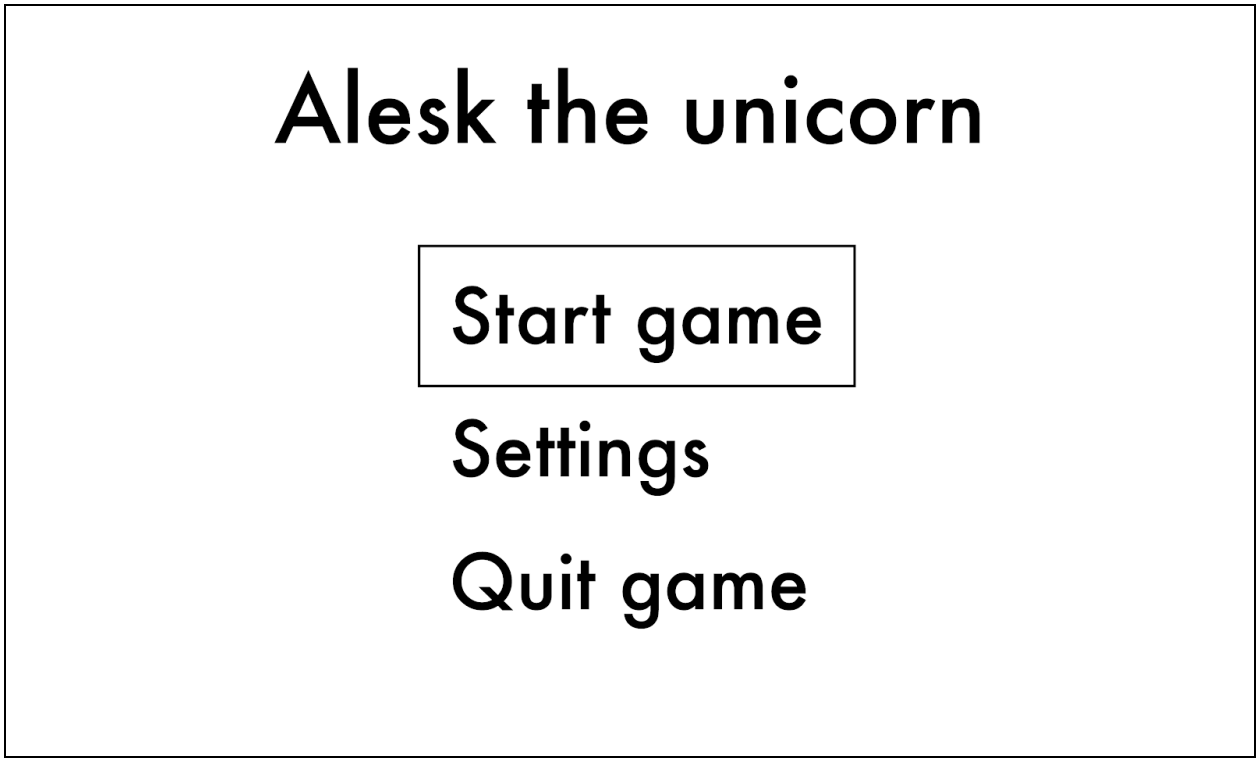
The game itself will contain a 2D visualization of all game elements, with Alesk the unicorn in the center (the player character). The interface will also contain from left to right:

- The player health, visualized by a unicorn horn
- The time spent on the level
- The total amount of lunchboxes on this level, and the amount of collected lunchboxes.

2.1.2 The menu

The menu is supposed to be a visual tool for the user to access different parts of the application, such as selecting a level to play, editing levels or quitting the application.

The main menu will contain the option to start the game, quit the game and change settings.



The select level menu will contain a list of available levels. While navigating through the levels with the up and down keys a preview of the current level will be shown to the right.

BILD

2.2 Functional requirements

What will the user be able to do? Write a list of use case names (id's) in the language of the customer. The specific flows for each use case is recorded below.
Specify a use cases in priority order.

1. Start the game by selecting a level and playing it
2. Browse between different levels and seeing small previews of them
3. Move the player character in a level:
    a. Movement sideways with the left/right arrow keys
    b. Jumping with the space bar
    c. As a consequence, the player character may die
        i. The user may choose to replay the level or go back to the main menu
    d. As a consequence, if the player character picked up the last lunch box, the user will have completed the level.
        i. The user may choose to replay the level, quit to the main menu or select another level.

4. Complete a level by picking up the last lunchbox
    a. Enter a signature to save the score
5. Quit the level
6. Change the settings:
    a. Configure key mappings.
    b. Turn off music.
    c. Turn off sound effects.
    d. Switch between fullscreen / windowed
7. Quit the game (exit the application)

2.3 Non-functional requirements

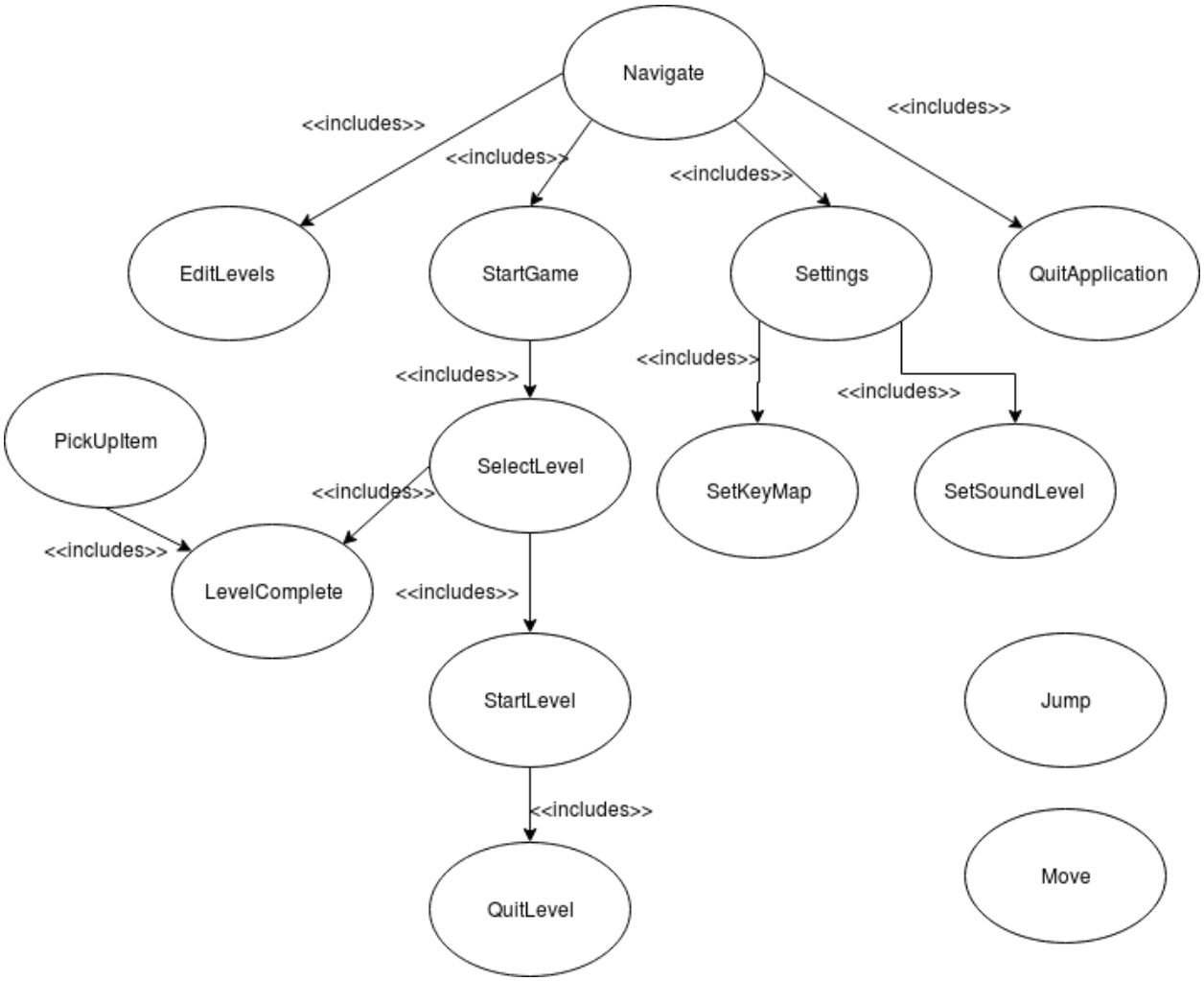
The game is meant to play the game in short bursts. All the interaction should be intuitive so that no explanations are needed.

Only environmental requirements is Linux, Mac OS or Windows computers with Java installed.

The application will be delivered as folder containing:

1. A Java Archive Runnable containing:
    a. The compiled game code
    b. All sounds and graphical assets
2. A folder containing all the levels in JSON-format

3 Use cases



3.1 Use case listing

UC: 1 Jump

Summary: The *Player Character* jumps into the air!
Priority: High
Participators: The *Player Character*

Normal flow of events: The *Player Character* jumps

	Actor	System
1	Presses ‘Space’ key or ‘Up’ arrow key	
2		*Player Character* moves upwards for a while and then falls down again
2.1 Hit a *Block*		The *Player Character* doesn’t move beyond the *Block*
2.2 Hit a *Spider*		See TakeDamage
2.3 Hit a pick up item		See PickUpItem
2.4 Doesn’t land on anything		*Player character* dies See EndLevel

UC: 2 Move

Summary: The *Player Character* moves
Priority: High
Participators: The *Player Character*

Normal flow of events: The *Player Character* moves on top of a collection of *Blocks*

	Actor	System
1	Holds down the ‘Left’ or ‘Right’ arrow key	
2		Shows the *Player Character* moving along the top of the collection of *Blocks* in the direction of the key pressed
2.1 Hit a *Block*		The *Player Character* doesn’t move beyond the *Block*
2.2 Hit a *Spider*		TakeDamage
2.3 Hit a *Lunchbox*		See PickUpItem 1.1
2.4 Hit a *PowerUp*		See PickUpItem 1.2

Alternate flows: The *Player Character* moves beyond the current collection of *Blocks*

	Actor	System
2.5 Moves beyond		The *Player Character* falls down

collection of *Blocks*		
2.5.1 Lands on a *Spider*		See 1.2
2.5.2 Lands on a pick up item		See 1.3
2.5.3 Lands on other collection of *Blocks*		See 1.1
2.5.4 Doesn't land on anything		*Player Character* dies See EndLevel

UC: 3 StartGame

Summary: The user navigates from the *Main Menu* to the *Select Level* screen
Priority: High
Extends: Navigate
Includes: SelectLevel
Participators: The user

Normal flow of events: The user selects and clicks the 'Start Game' button in the *Main Menu*

	Actor	System
1 User selects and clicks the 'Start Game' button	See Navigation	
2		See SelectLevel

UC: 4 SelectLevel

Summary: The user enters the *Select Level* screen and selects a *Level*
Priority: High
Extends: StartGame
Includes: StartLevel
Participators: The user

Normal flow of events: The user selects a *Level*

	Actor	System
1		A list of *Levels* are read in from files
2		List of *Levels* available are shown on the screen First *Level* in list is highlighted
3		Display preview/level info and *Highscores*
4	Use arrow keys or click with the mouse to highlight a *Level*	
5		Highlight the *Level* and display a small preview of it
6		Display preview/level info and *Highscores*

Alternate flow of events: At any point in time the user selects the 'Go Back' button and presses the 'Enter' key

	Actor	System
1		A list of *Levels* are read in from files
2		List of *Levels* available are shown on the screen First *Level* in list is highlighted
3		Display preview/level info and *Highscores*
4	Selects the ‘Go Back’ button	
5		Highlight ‘Go Back’ button
6	User presses the ‘Enter’ key	
7		The screen is replaced with the *Main Menu*

Alternate flow of events: At any point in time the user presses the ‘Escape’ in the *Level Select* Screen

	Actor	System
1		A list of *Levels* are read in from files
2		List of *Levels* available are shown on the screen First *Level* in list is highlighted
3		Display preview/level info and *Highscores*
6	The user presses the ‘Escape’ button	
7		The screen is replaced with the *Main Menu*

UC: 5 StartLevel

Summary: The user presses the ‘Start Game’ button and the *Level* is started
Priority: High
Extends: SelectLevel
Participators: The user

Normal flow of events: The user starts the level

	Actor	System
1	The user presses the ‘Start Game’ button from the *Level Select* screen	

2		The screen is replaced with the *Game* screen and the *Level* is started

UC: 6 QuitLevel

Summary: The user quits the *Level* from the *Game Screen*
Priority: High
Extends: StartLevel
Participators: The user

Normal flow of events: User presses Escape and presses Yes to exit to the Select Level screen

	Actor	System
1	User presses ‘Escape’	
2		Show a YES/NO dialogue, the ‘NO’ button is highlighted by default
	User presses the ‘Y’ key or clicks the ‘YES’ button or selects the ‘YES’ button and presses the ‘Enter’ key	
		Shows the *Select Level* screen

Alternate flow of events: User presses Escape and presses No and stays at the Game Screen

	Actor	System
1	User presses the ‘Escape’ key	
2		Show a YES/NO dialogue, the ‘NO’ button is highlighted by default.
	User presses the ‘N’ key or clicks the ‘NO’ button or selects the ‘NO’ button and presses the ‘Enter’ key	
		Shows *Game* Screen

UC: 7 QuitApplication

Summary: The user quits the application from the *Main Menu*
Priority: High
Extends: Navigate
Participators: The user

Normal flow of events: The user presses the ‘Quit Game’ button from the *Main Menu*

	Actor	System
1. User selects the ‘Quit Game’ button	See Navigate	
2		The system quits the application and closes the window

UC: 8 EditLevels <To be decided>

Summary: The User starts the game.
Priority: Low
Extends: Navigation
Includes: <To be decided>
Participators: The user

Normal flow of events: To be decided

UC: 9 Settings

Summary: The user enters the *Settings* screen
Priority: Low
Extends: Navigate
Includes: SetKeyMap, SetSound
Participators: The user

Normal flow of events: The user enters the *Settings* screen changes something and then leaves the *Settings* screen

	Actor	System
1 The user highlights the 'Settings' button from the *Main Menu*	See Navigate	
2		Show *Settings* screen
2.1 The user sets the sound	See SetSound	
2.2 The user sets the key map	See SetKeyMap	
	User selects the 'Go Back' button	
		Highlight the 'Go Back' button
	User presses the 'Enter' button	
		The screen is replaced with the *Main Menu* screen

Alternate flow of events: At any point in time the 'Go Back' button and leaves the *Settings* screen

	Actor	System
6	The user clicks on the 'Go Back' button	
7		The screen is replaced with the *Main Menu* screen

UC: 9.1 SetKeyMap (To be decided)

Summary: The user changes the key mappings in the *Settings* screen
Priority: Low
Extends: Settings 2.1
Includes:
Participators: The user

Normal flow of events

	Actor	System

UC: 9.2 SetSound (To be decided)

Summary: The user is in the *Settings* screen
Priority: Low
Extends: Settings 2.2
Includes:
Participators: The user

Normal flow of events:

	Actor	System

UC: 10 Navigate

Summary: The user starts the application
Priority: High
Includes: StartGame, EditLevels, Settings, QuitApplication
Participators: The user

Normal flow of events: The user highlights an option in the *Main Menu* with the arrow keys or mouse and presses the ‘Enter’ key or clicks the left mouse button to replace the screen with the selected option screen

	Actor	System

1	The user presses an arrow key	
2		Highlights the relevant menu option
3	The user presses the ‘Enter’ key	
4		Replaces the screen with the selected option screen

Alternate flow of events: User marks Menu Item with the mouse and clicks to show Menu Item screen

	Actor	System
2	The user hovers over the mouse over a menu option	
3		Highlights the relevant menu option
4	The user presses the left mouse button	
5		Replaces the screen with the selected option screen

UC: 12 LevelComplete

Summary: The user completes the current *Level*
Priority: High
Extends: TakeDamage, PickUpItem
Includes: SelectLevel
Participators: The *Player Character*

Normal flow of events: User completes a *Level*, inputs name and presses the ‘Enter’ button or clicks the ‘Select Level’ button

	Actor	System
1 User won		Show a prompt to write a signature
2	User enters a signature using the keyboard, and presses the ‘Enter’ key to confirm	
3		Show *Complete Level* screen, the ‘Select Level’ button is highlighted by default
4	Presses ‘Enter’ or	See SelectLevel

	clicks ‘Select Level’	

Alternate flow of events: User marks the ‘Retry’ button and presses the ‘Enter’ key or clicks the ‘Retry’ button

	Actor	System
3	Highlight the ‘Retry’ button with arrow keys or hovers with mouse	
4		Highlight the ‘Retry’ button
5	Presses the ‘Enter’ key or clicks on the ‘Retry’ button	
6		Shows *Game* Screen

UC: 14 PickUpItem

Summary: The *Player Character* has landed on an item that can be picked up
Priority: High
Includes: CompleteLevel
Participators: The *Player Character*

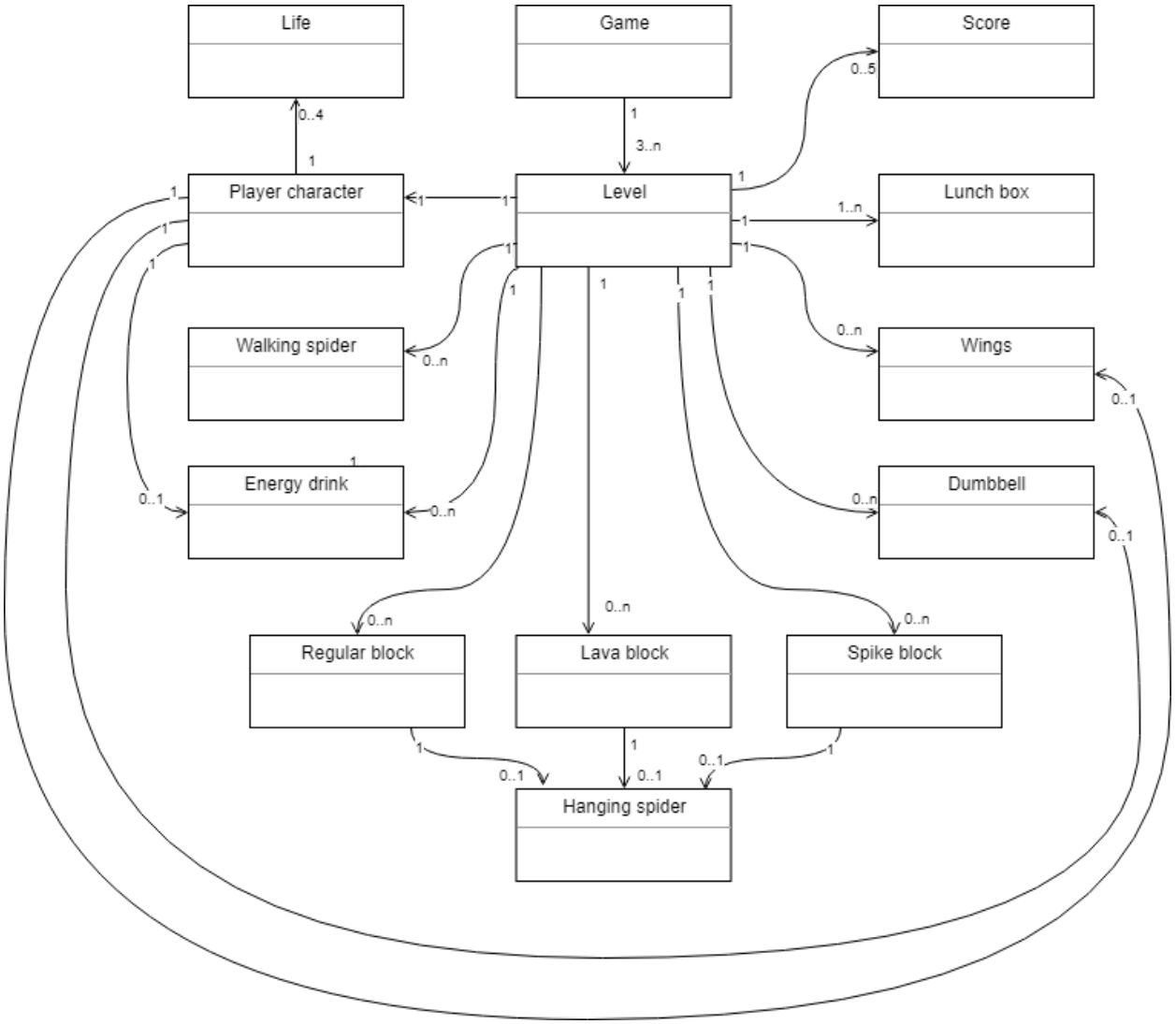
Normal flow of events: The *Player Character* moves into a *Lunchbox* and/or a *PowerUp* without ending the *Level*.

	Actor	System
1.1	Moves into a *Lunchbox*	Pick up animation shown GUI element showing number of *LunchBoxes* left decreased
1.2	Moves into *PowerUp*	Pick up animation shown, start player animation with pickup.

Alternate flow: The player character moves into a lunchbox that was the last one left.

	Actor	System
1.1 The last *Lunchbox* on the *Level* is picked up		Player wins See CompleteLevel

4 Domain model



4.1 Class responsibilities

Explanation of responsibilities of classes in diagram

Game: Is the overall representation of the game and holds all the levels and performs some logic on the level.
Level: A container with all the objects for a level.
Player Character: The unicorn holding any picked up items, i.e Energy drink, Wings or dumbbell
Life: Represents the amount of life left with a piece of unicorn horn

Score: Represents the time it took for a player to finish the level.
Lunchbox: An item possible to be picked up by the player.
Walking Spider: A representation of an adversary in the game.
Energy Drink: An item possible to be picked up by the player.
Wings: An item possible to be picked up by the player.
Dumbbell: An item possible to be picked up by the player.
Regular block: Represents a block, may hold a hanging spider
Lava block: Represents a block of lava, may hold a Hanging spider.
Spike block: Represents a block with spikes, may hold a Hanging spider.
Hanging spider: A representation of an adversary in the game belonging to a block.

5 References