Requirements and Analysis Document for Morf

Version: 2.0
Date: 31-05-2015
Author: Lage Bergman, Harald Brorsson, Christoffer Karlsson, Gustav Bergström

This version overrides all previous versions.

1. Introduction

1.1 Purpose of application

The project aims to create a two dimensional platform game with a puzzle aspect. For definitions, terms and rules of the game see 1.4, Objectives and success criteria of the project.

1.2 General characteristics of application

The application will be a desktop, stand alone (non-networked), single-player application with a graphical user interface for the Windows/Mac/Linux platforms.

The application will be a real time platformer game where the player controls a character in a two-dimensional world. The game is focused around different levels. A levels is chosen through the main menu each time the application is started. When a level is completed the next one starts automatically. At any time, when in a level, the player can open a menu and choose to return to the main menu or exit the game. The levels have no time limit.

1.3 Scope of application

The project is limited to a few playable levels only. The game can only be played by one player at a time. There will be no enemies or monsters.

1.4 Objectives and success criteria of the project

Basic player character movement will be working.
Interaction with blocks in the world (freezing water, melting ice etc.) will be working.
You will be able to complete at least one level.
A good graphical interface and original textures will be implemented.
The game will save progress and scores.

1.5 Definitions, acronyms and abbreviations

All definitions and terms regarding the core Morf game are as defined in the *RULES* section.

- GUI, graphical user interface.
- Java, platform independent programming language.

- JRE, the Java Runtime Environment. Additional software needed to run a Java application.
- Host, a computer where the game will run.
- Player, the person who is playing the game.
- Player Character, the character that the player controls.
- Level, a stage where the goal is to take the character from one point to another.
- Level grid, a grid for representing positions in a level.
- Block, a type of environment placed on a specific position in the grid.

2. Requirements

2.1 Functional requirements

The player should be able to:

- Select a level to play
- Exit the application
- Pause the game
- Return to start screen while playing
- The player character should be able to:
    - Walk
    - Jump
    - Fly
    - Freeze water
    - Melt ice
    - Evaporate water
    - Fall
    - Die
    - Pour water
- Change the application settings (sound volume etc.)

2.2 Non-functional requirements

2.2.1 Usability

Normal users should be able to play the game within a short period of time. An easy level containing all game components will be implemented to ensure that the player knows of all basic functionality available in the game.

2.2.2 Reliability

NA

2.2.3 Performance

Any actions initiated by the player in any of the menus should not exceed a 2 second response time in worst case. During actual gameplay immediate reaction is a must.

2.2.4 Supportability
The application should be built for a desktop environment.

2.2.5 Implementation
To achieve platform independence the application will use the Java environment. All hosts must have the JRE installed and configured.

2.2.6 Packaging and installation
The application will be delivered as a zip-archive containing:

1. A file for the application code (a standard Java jar-file).
2. All needed resources, icons and images, etc.

2.2.7 Legal
NA

2.3 Application models

2.3.1 Use case model
See APPENDIX for UML diagram and textual descriptions.

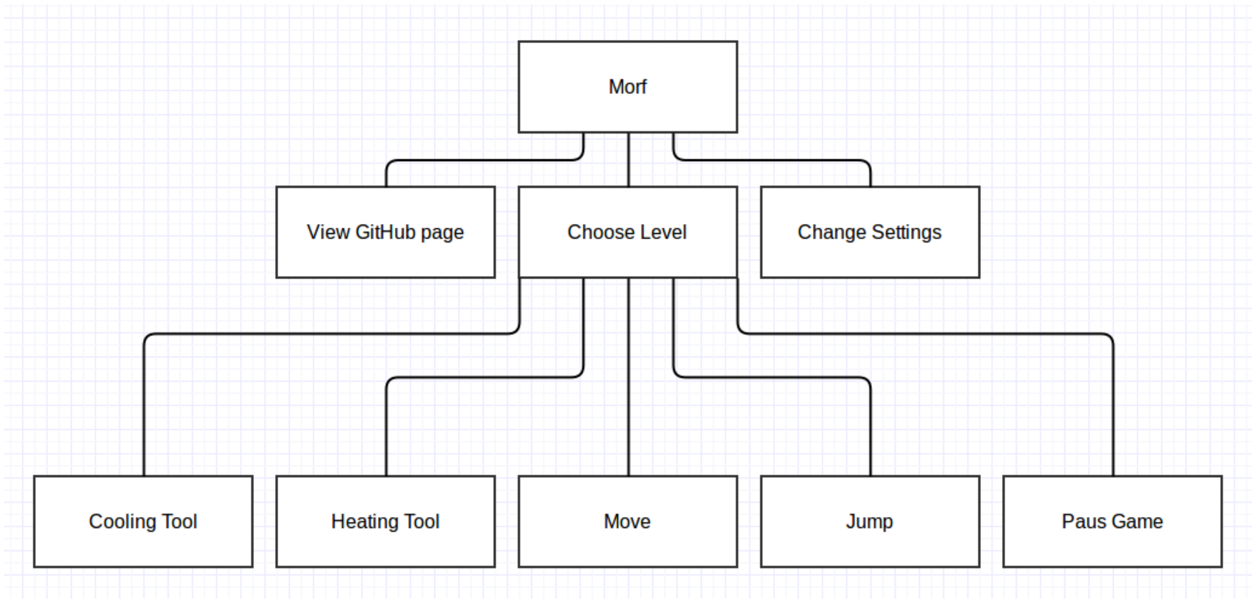
2.3.2 Use cases priority
1. Move (right/left)
2. Fall
3. Jump
4. Die
5. Pour water
6. Freeze water
7. Melt ice
8. Boil water
9. Swim
10. Fly on vapor

2.3.3 Domain model
See APPENDIX.

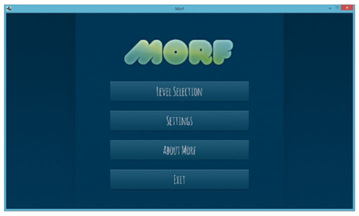
2.3.4 User interface
The application will run in fullscreen all the time and it will not be possible to resize the window. There will be three different views, one menu view, one game view, and one pause view. The menu view will let the user start the game, change settings and exit the application. The game view will present graphics for the ongoing game. The pause view will be a minimized version of the menu view, shown on top of the game view when the game is paused.

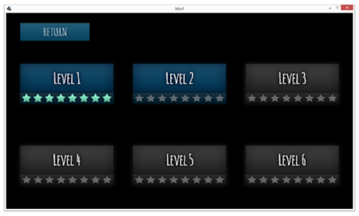
APPENDIX

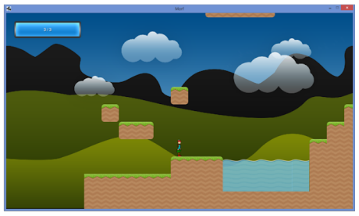
Use Case diagram

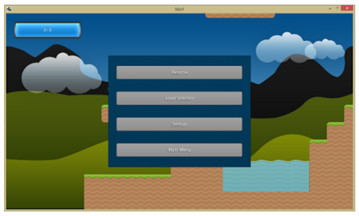


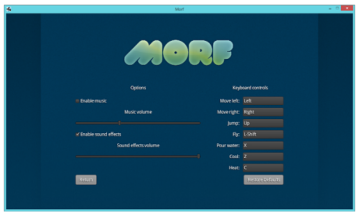
GUI











Options

Enable music

Music volume

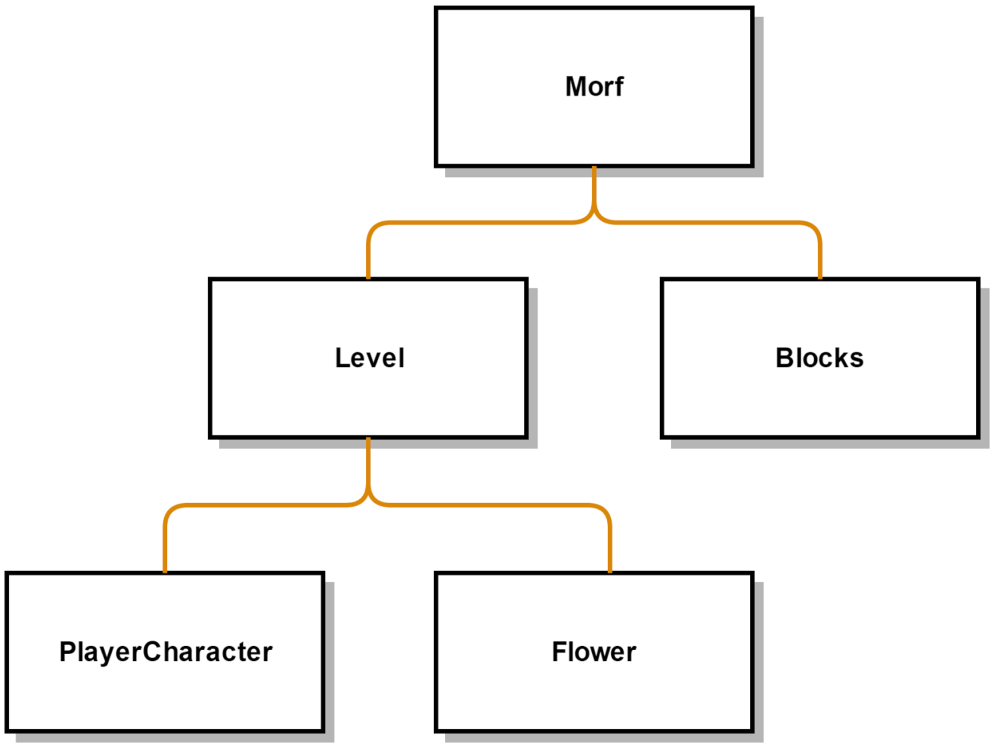
✓ Enable sound effects

Sound effects volume

Keyboard controls

Move left:	Left
Move right:	Right
Jump:	Up
Fly:	L-Shift
Pour water:	X
Cool:	Z
Heat:	C

Domain Model



Use Cases

Use case: Move

Summary: The most basic movement command in the game. The player uses this to move the player character right and left. UC StartGame precedes this.
Priority: high
Extends: -
Includes:
Participators: The player

Normal flow of events:

	Actor	System
1	User presses direction key	
2		The player character moves in the selected direction for as long as the key is pressed.

Alternate flows:
Flow 2.1: Something is in the way

	Actor	System
2.1.1	User presses direction key	
2.1.2		Checks if character move direction is empty.
2.1.3		If not empty, move animation is played, but player character does not move.

Flow 3.2: SPikes is in the way (i.e spikes)

	Actor	System
2.2.1	User presses direction key	
2.2.2		Checks if character move direction contains spikes.
2.2.3		Player characters death animation is played.
2.2.4		The game restarts at the current level.

Use case: Jump

Summary: One of the most basic movement commands in the game. The player uses this to jump over obstacles and is the main way for the player to move the player character vertically. UC StartGame precedes this.
Priority: high
Extends: -
Includes:
Participators: The player

Normal flow of events:

	Actor	System
1	User presses and holds jump key	
2		The player character moves up the maximum amount allowed. Holding the jump key when the player character has started falling does

		nothing.

Alternate flows:
2.1: Something is in the way

	Actor	System
2.1.1	User presses jump key	
2.1.2		Checks if area above player character are empty
2.1.3		If not, jump animation is played until the player releases the key, but the player character is not moved.

Use case: spikes

Actor	System
User is near spikes	
	if player character touches spikes death animation is played.

Use case: Pour water

Summary: This is one of the main puzzle mechanics in the game. This creates a block of water in front of the player character that the player can manipulate to progress in the level. UC Select Level precedes this.
Priority: high
Extends: -
Includes:
Participators: The player

Normal flow of events:

	Actor	System

1	User presses pour water key	
2		Water can animation is played.
3		Water is drained from water can.
4		Water is created in front of the player character.

Alternate flows:

Flow 3.1 Player character pours water on flower

	Actor	System
3.1.1		Remaining water is counted and added to player score.
4.2.3		The next level is started.

Flow 4.1 The water can is empty

	Actor	System
4.1.1		Player character death animation is played
4.2.3		The level is restarted.

Use case: Cooling Tool

Summary: This is one of the main puzzle mechanics in the game. This causes the player character to use the cooling tool which interacts with whatever is in front of the player character. UC Select Level precedes this.
Priority: medium
Extends: -
Includes:
Participators: The player
Normal flow of events:

	Actor	System

1	User presses cooling key	
2		Cooling tool animation is played.
3		Nothing happens.

Alternate flows:
Flow 3.1 There is water in front of the player character

	Actor	System
3.1.1		Water in front of the player character is frozen and becomes solid ice.

Alternate flows:
Flow 3.1 There is vapor in front of the player character

	Actor	System
3.1.1		Vapor in front of the player character is condensed and becomes liquid water.

Alternate flows:
Flow 3.1 There is ice in front of the player character

	Actor	System
3.1.1		Nothing happens.

Use case: Heating Tool

Summary: This is one of the main puzzle mechanics in the game. This causes the player character to use the heating tool which interacts with whatever is in front of the player character. UC Select Level precedes this.
Priority: medium
Extends: -
Includes:
Participators: The player

Normal flow of events:

	Actor	System
1	User presses heating key	
2		Heating tool animation is played.
3		Nothing happens.

Alternate flows:
Flow 3.1 There is water in front of the player character

	Actor	System
3.1.1		Water in front of the player character is vaporized.
3.1.2		The steam starts to move up.

Flow 3.2 There is ice in front of the player character

	Actor	System
3.2.1		Ice in front of the player character is melted and becomes water.

Flow 3.3 There is vapor in front of the player character

	Actor	System
3.1.1		Nothing happens.