RAD

Version: 0.1

Date: 25/3

Author: grupp 16

This version overrides all previous versions.

1 Introduction

1.1 Purpose of application

The purpose of the project is to create an entertaining platform game wherein the player controls a character, running and jumping to move from one point to the next.

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 game flow provides continuous feedback. The character of the game will carry a flashlight which will provide almost all of the light in the game, therefore the game will have a dark background and the character can barely be seen.

The game will provide a certain amount of levels which will vary in difficulty. To get to the next level, the gamer must complete a number of puzzles, for example collect keys or other objects to unlock doors and so on. There will be several obstacles in the different levels, like enemies who will try to kill the character and chasms the character can fall into. The enemies will only be able to approach the character when light falls on them, so the gamer can move without fright of them when not pointing the flashlight at them.

1.3 Scope of application

One player game. The player can save their progress. Four save-slots. Will have sound. Multiple light sources, with light sources as part of levels.

1.4 Objectives and success criteria of the project

There will be at least two levels with complete functionality. At least a single light source.

1.5 Definitions, acronyms and abbreviations

FPS - frames per second
JRE - the Java Run time Environment. Additional software needed to run an Java application.
GUI - Graphical User Interface
Viewport - Screen space in which the game world is drawn, as opposed to interface controls.

2 Requirements

2.1 Functional requirements

The player should be able to:

1. Start a game.
2. Create a save file.
3. Move the character around (includes jumping and crawling).
4. Move the flashlight around.
5. Pick up certain items from the world.
6. Move certain items around in the world.
7. Go through doors to access new levels.
8. Save their progress to the save file.
9. Load a active save file.
10. Exit the application.

2.2 Non-functional requirements

2.2.1 Usability

“Learning by doing”-approach, some kind of instruction of the controls and some in-game-tips of how to play.

2.2.2 Reliability

N/A

2.2.3 Performance

The game should be efficient enough to reliably generate at least 30 FPS on modern desktop or laptop computer systems.

2.2.4 Supportability

The application should be build in such a way that porting it to another device should not be more work than approximately 1 man-month. As such it should be built following the principle that the model of the game should be detached from it’s view and input method.

2.2.5 Implementation

To achieve platform independence the application will use the Java environment. All hosts must have the JRE installed and configured. The application needs to be installed on all hosts where it will run (possibly downloaded).

2.2.6 Packaging and installation

JAR-file

2.2.7 Legal

NA

2.3 Application models

2.3.1 Use case model

UML and a list of UC names (text for all in appendix)

2.3.2 Use cases priority

1. Walk Right
2. Jump
3. Fall
4. Die
5. Move flashlight
6. Enemy moving
7. Crouch
8. Crawl
9. Go trough door
10. Pick up item
11. Drop item
12. Use item

2.3.3 Domain model

See appendix.

2.3.4 User interface

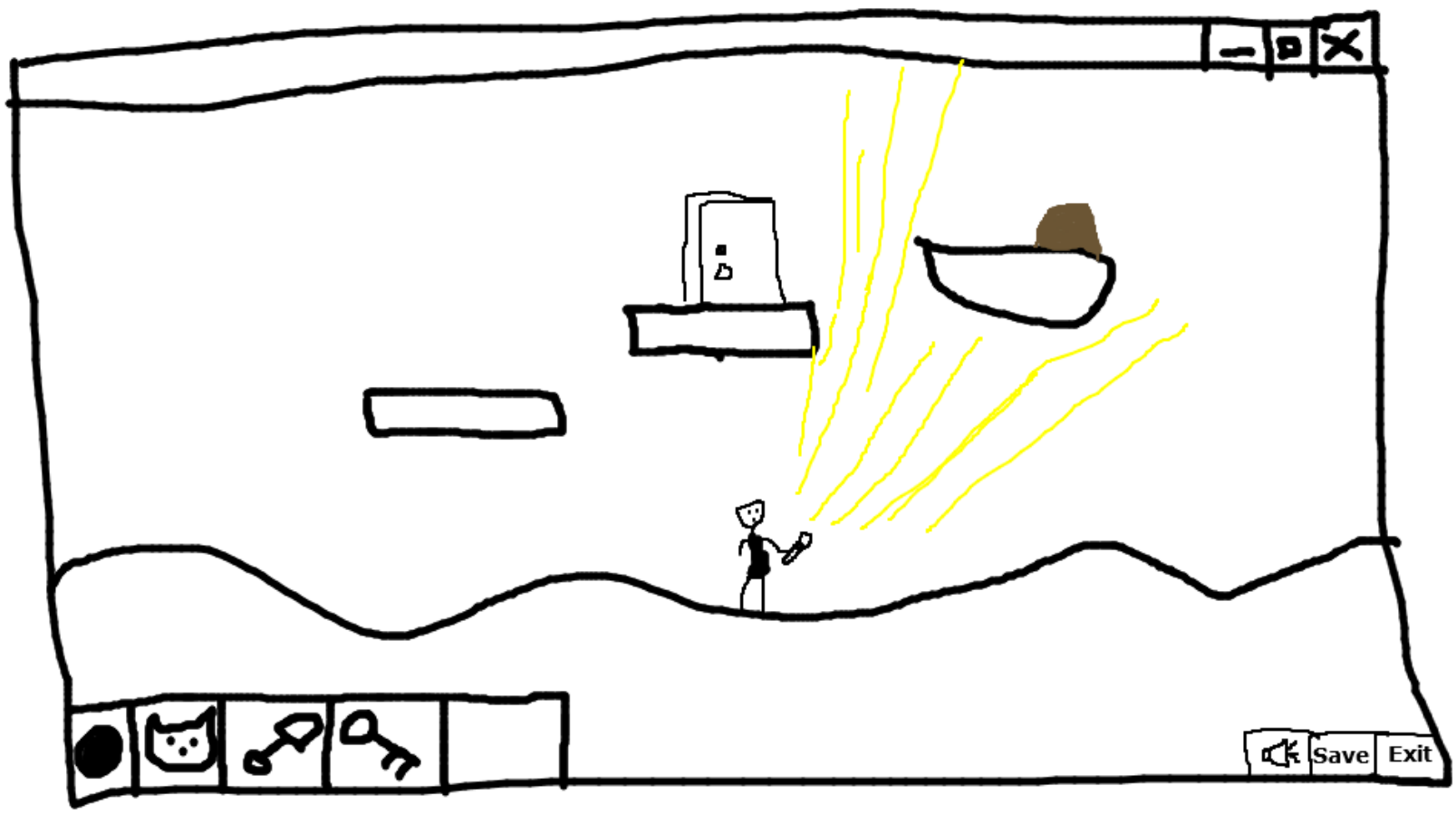
The game will use a simple interface, with most of the screen space consisting of the viewport. Interface elements include an inventory, in which the items the player is currently carrying are shown, and meta controls such as saving, exiting, and controlling sound volume.

2.4 References

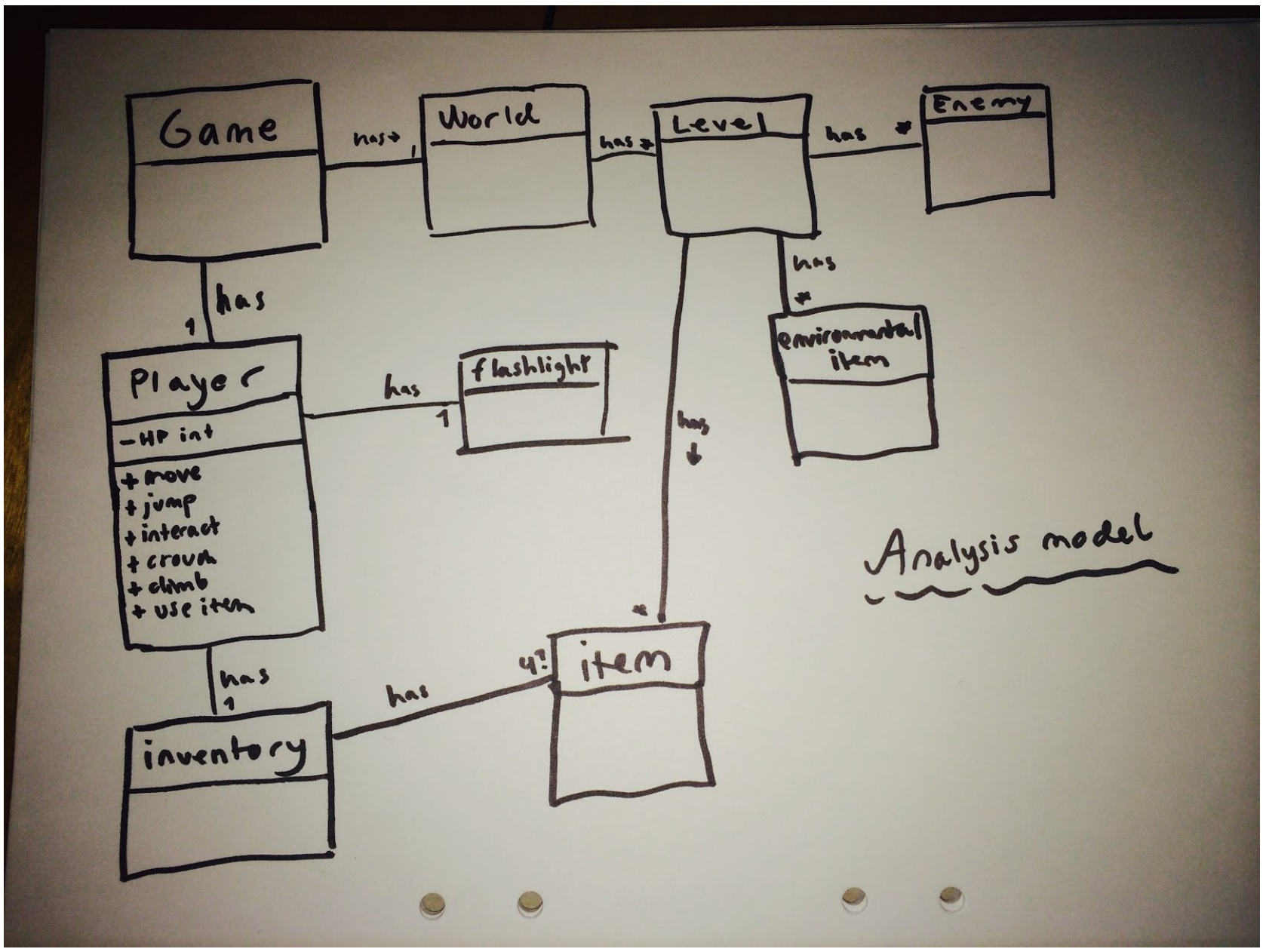
APPENDIX
GUI
Domain model
Use case tex

APPENDICES

Appendix A



Superduperpreliminärt skissförslag



first version of analysis model

Appendix B

Use Cases:

Use Case: WalkRight

Summary: This is how the player moves right in the game area. WalkLeft is equivalent but the player presses the left button and walks to the left instead.

Priority: high

Participators: Player

Normal flow of events:

	Actor	System
1	Player presses and holds the right button	
		Character moves to the right.

Alternate flow

Flow 2.1 (Walks against a wall)

	Actor	System
1	Player presses and hold the right button	
		Keeps the character still

Flow 2.2 (Walks against a enviromental object)

	Actor	System
1	Player presses and hold the right button	
		Moves the character and the enviromental object right.

Use Case: MoveFlashlight

Summary: This is how the player sees things clearly in the dark setting, by moving around their flashlight.
Priority: medium

Participators: Player

Normal flow of events:

	Actor	System
1	moves cursor	

		illuminates the new area, where the cursor is pointing.

Use Case: Jump

Summary: The ability for the player to make the character jump.

Priority: high

Includes: Fall

Participators: Player

Normal flow of events:

	Actor	System
1	Presses the jump button	
		Launches the character upwards
		Lets the character Fall (see other use case)

Use Case: Fall

Summary: Describes what happens anytime there isn't any collision-area below the character. A lot of different actions can lead to this.

Priority: high

Participators: Player

Normal flow of events:

	Actor	System
1	Places the character without any	

	collision-area below it	
		Accelerates the character towards the bottom of the screen.

Use Case: Pick up Item

Summary: This is how the Player picks up items in the world.

Priority: Low

Participators: Player

Normal flow of events:

	Actor	System
1	Presses the “interact”-button when the character is ontop of a item	
		Adds the item to the character’s inventory

Exceptional flow:

	Actor	System
1	Presses the “interact”-button when the character is ontop of a item	
		Displays “the inventory is full” on screen.

Use Case: Crouch

Summary: Lets the player make the character crouch to avoid obstacles.

Priority: medium

Participators: Player

Normal flow of events:

	Actor	System
1	Presses the crouch button	
		Displays the character as crouched

Use Case: Crawl

Summary: Lets the player make the character crawl to avoid obstacles and get through tight spaces.

Priority: Medium

Participators: Player

Normal flow of events:

	Actor	System
1	Holds down the crouch button while pressing one of the movement buttons	
		Displays the character crawling along the floor

Use Case: Die

Summary: The character dies.

Priority: high

Participators: Player

Normal flow of events:

	Actor	System
1	Walks or Falls into a enemy or a dangerous object	
		Removes a HP from the character
		If character is out of HP, display “Game Over”

Use Case: Enemy Moving

Summary: Some of the enemys will only move if they are shined on by the flashlight

Priority: medium

Participators: Player

Normal flow of events:

	Actor	System
1	Moves the flashlight over a enemy.	
		Makes the enemy move towards the player

Use Case: Use Item

Summary: Lets the player use a picked up item at the appropiate place.

Priority: low

Participators: Player

Normal flow of events:

	Actor	System
1	Presses the “use Item” button.	

		If able to use the item at that position, uses the item. Otherwise displays “Item not usable here”

Use Case: Drops Item

Summary: Lets the player drop a item so that he can pick up another one

Priority: low

Participators: Player

Normal flow of events:

	Actor	System
1	Presses the “drop item” button.	
		Drops the active item on the ground besides the player.

Use Case: Go through door

Summary: Lets the player go from the world into a level

Priority: medium

Participators: Player

Normal flow of events:

	Actor	System
1	Presses the “interact” button in front of a door.	
		Shifts the view to the level corresponding to the door.

RAD

Version: 0.1

Date: 25/3

Author: grupp 16

This version overrides all previous versions.

1 Introduction

1.1 Purpose of application

The purpose of the application is pure entertainment. The application is a platform game.

1.2 General characteristics of application

The application will be a desktop, standalone (non-networked), single-player application with a graphical user interface for the Windows/Mac/Linux platforms. The game flow provides continuous feedback. The character of the game will carry a flashlight wich will provide almoust all of the light in the game, therefore the game will have a dark background and the character can barely be seen.

The game will provide a certain amount of levels wich will vary in difficulty. To get to the next level, the gamer must complete a number of puzzles, for example collect keys or other objects to unlock doors and so on. There will be several obsticles in the different levles, like enemies who will try to kill the character and chasms the character can fall in to. The enemies will only be able to approach the charachter when light falls on them, so the gamer can move without fright of them when not pointing the flashlight at them.

1.3 Scope of application

One player game. The player can save their progress. Four save-slots.

1.4 Objectives and success criteria of the project

There will be at least two levles with complete functionality.

1.5 Definitions, acronyms and abbreviations

2 Requirements

In this section we specify all requirements

2.1 Functional requirements

Create a list of high level functions here (from the use cases).

2.2 Non-functional requirements

Possible NA (not applicable).

2.2.1 Usability