Requirements and Analysis Document for Adlez

Version: 1.0

Date: 29/5 - 2016

Michel Folkemark, Pontus Thomé, Viktor Engström, Martin So

1 Introduction

Adlez is a project where the aim is to create a computer based 2D game which is inspired by the famous game, “Zelda: A link to the past”. The game will be a single player adventure RPG (role playing game) where you play as the main character and try to defeat your enemies.

The project is going to be implemented in the programming language, Java in an Object Oriented way.

1.1 Purpose of application

The application has to be simple to use and understand since it should be suitable for everyone. The game should be playable on almost all modern computers since it won’t require the latest hardwares. It will also be an offline game which means that you can play it even though you don’t have an internet connection, as long as the application already is installed on your platform.

1.2 General characteristics of application

The application will be compatible for desktop with the following platforms, Windows/Mac/Linux. No need for internet connection to play.

An adventure game in a fairly large world with one main hero. The hero fights enemies in his journey and gains money/experience and items from them which he then can use to gain new abilities and stats. The game will be very open, which means that you do not have to wait for any turns, you can freely walk wherever you want.

1.3 Scope of application

The application is a single player game with enemies and friendly characters that are controlled by the computer. It is possible to save the game so that the player later can continue from where he/she left of.

1.4 Objectives and success criteria of the project

The success criteria is for the user to be able to control the hero, explore a world, fight enemies, and gain new abilities and items.

1. The player should be able to control a character and move around in an area.
2. The player should be able to encounter non-friendly characters/creatures and fight them using an array of weapons/abilities.

3. The player should be able to visit several areas with varying appearances, such as caves, buildings etc.
4. The player should be able to encounter friendly characters and interact with them.

1.5 Definitions, acronyms and abbreviations

GUI, graphical user interface.
Java, platform independent programming language.
OOP, Object Oriented Programming, a very commonly used programming paradigm.
RPG, Role Playing Game.
NPC, Non Player Character.

2 Requirements

2.1 Functional requirements

Game setup

- A user should be able to:
    - start a new game
    - save the current game
    - load a saved game

Character actions

- A user should with their character be able to:
    - move in four directions
    - attack both melee and ranged
    - cast a spell
    - use a skill
    - pick up gold and items
    - equip items
    - open inventory
    - interact with NPCs, such as having a conversation, buy and sell items

Enemies actions

- An enemy should be able to:
    - locate the player’s character when the enemy is visible on the screen
    - attack and damage the player’s character
    - drop gold and items

2.2 Non-functional requirements

2.2.1 Usability

The starting screen of the game should be quite simple, easy to start a new game or load a saved game. Controlling the characters movements and actions should also be simple, there should only be a few keys to press and a few buttons to press with the mouse. To learn all the movements and actions should be very quick to learn.

2.2.2 Reliability

The application should run without crashing for at least 1 hour at a time.

2.2.3 Performance

The game should be able to run smoothly on most computers. It should react to input from the user rather instantly and only take time to load in cases such as starting the game or entering a new map area in the game.

2.2.4 Supportability

Needs Java 1.8 install. The game will be designed to run on a personal computer with Windows/Mac OS/Linux installed.

2.2.5 Implementation

The game will be implemented using the Java programming language as well as the game-development framework libGDX, which will allow the game to easily be deployed for different systems.

2.2.6 Packaging and installation

A README-file with instructions is in the package.

2.2.7 Legal

We will not use Project Adlez for any commercial intentions and it will not be published. It is a school project with the aim to fulfill the requirements to pass the course TDA367.

2.3 Application models

2.3.1 Use cases

What actors that can use which use cases can be seen in the use case diagram in the appendix, Use case diagram. Scenario of the use case **Start new game** can be seen in the appendix, Start game. Scenario of the use case **Move** can be seen in the appendix, Move. Sequence diagram of the use case **Move** can be seen in the appendix, Move.

Game setup

1. Start new game
2. Save game
3. Load game

Character

4. Move
5. Melee attack

6. Range magic attack
7. AOE magic attack
8. Interact
9. Loot chest
10. Regenerate mana
11. Interact with Friendly NPC *(not yet implemented)*
12. Change area
13. Buy *(not yet implemented)*
14. Sell *(not yet implemented)*
15. Open inventory
16. Equip item *(not yet implemented)*

2.3.2 Use cases priority

High

- Start new game
- Move
- Melee Attack

Mid

- Range magic attack
- AOE magic attack
- Interact
- Loot chest
- Regenerate mana
- Change area
- Equip item *(not yet implemented)*
- Open inventory
- Save game
- Load Game

Low

- Interact with friendly NPC *(not yet implemented)*
- Buy *(not yet implemented)*
- Sell *(not yet implemented)*

2.3.3 Domain model

A graphical view of the domain model of the application can be seen in the appendix, Domain model.

Adlez

Represents the whole game and contains two entities, **Player** and **Area**.

Area

Represents the a level in the game and everything a level contains which is,

- **Enemies**, represents a character that will attack the **Player.**

- **Friendly characters**, represents a character that will not attack the **Player** and that the **Player** can interact with.
- **Area connectors**, represents an object that the **Player** can use to move between **Areas**.
- **Walls**, represents a non destructible object that confines the **Area**.
- **Chests**, represents a chest that the **Player** can loot for **Items**.
- **Mana fountains**, represents an object where the **Player** can regenerate it's mana.
- **Obstacles**, represents an obstacle that can be attacked and destroyed.

Player

Represents a character that the user can control. It can move around in areas and attack enemies and objects and interact with characters and objects. It can also move between areas, and is the only entity that can do so.

Item

Represents an item that the **Player** can use and benefit from in some way. **Items** can be a **Weapon** which the **Player** can equip to do more damage, or an **Armor** that the **Player** can equip to take less damage.

2.3.4 User interface

Graphical representations of the user interface can be seen in the appendix, GUI.

The game will at least have three main screens:

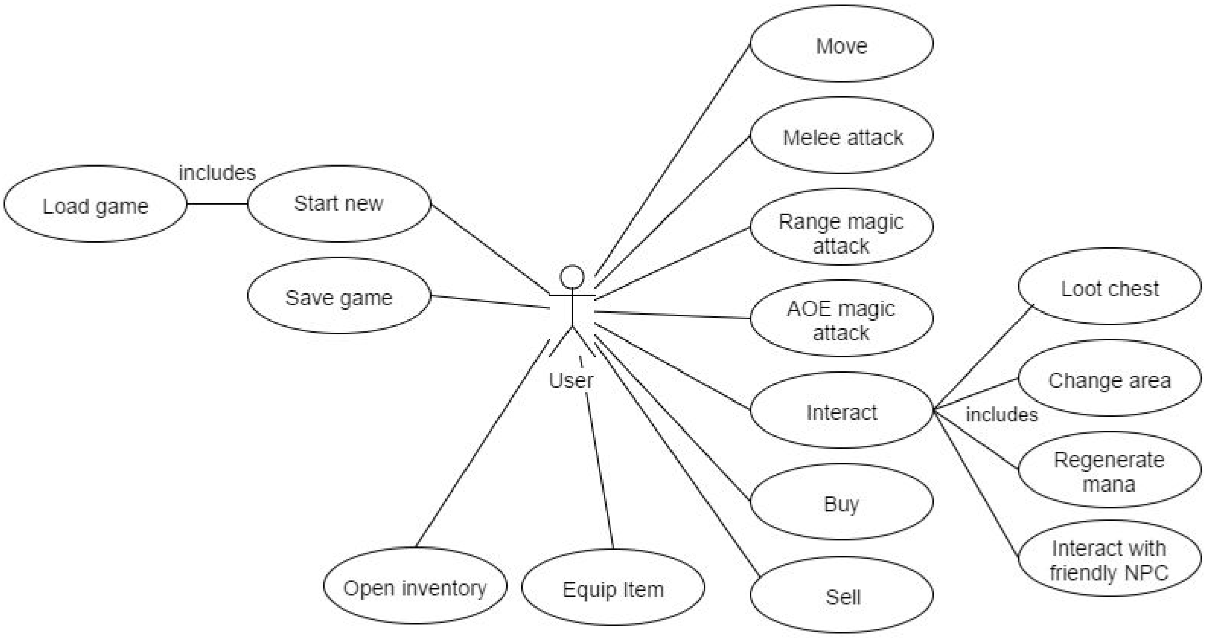
1. The starting screen: Will have buttons for starting the game along with buttons for options and instructions that may lead to subscreens.
2. The playing screen: Where the user actually plays the game. User interface elements such as a health bar, what weapons are equipped, a map etc. will be present.
3. The menu screen: Where the user will be able to do things such as save and/or quit the game as well as browse an inventory of weapons and items. (The inventory could possibly be integrated in the playing screen)

2.4 References

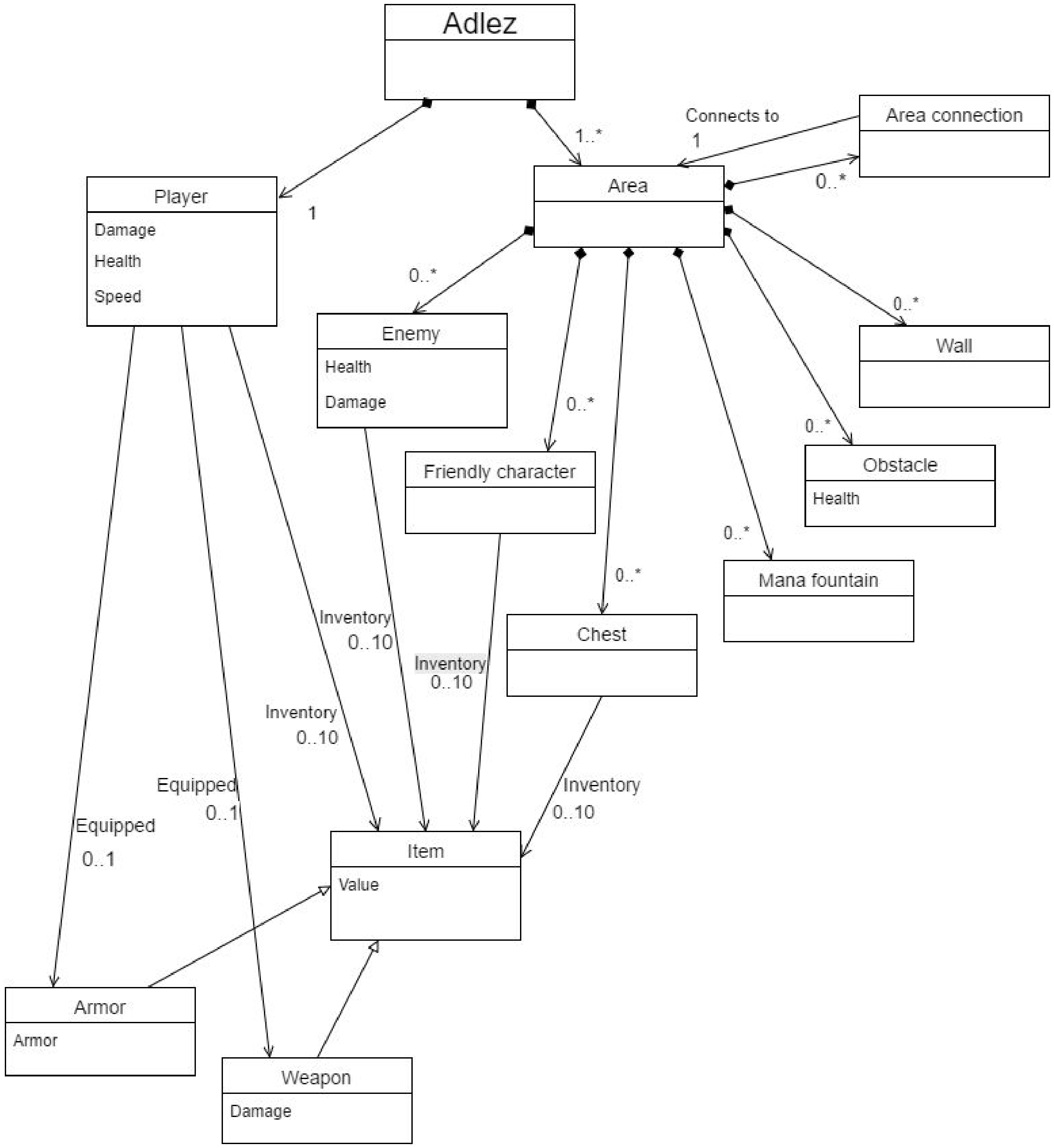
APPENDIX

GUI

Use case diagram



Domain model



Use case scenarios

1. Start new game

Priority: High
Extends: -
Includes: - 2. Load game
Participators: Player

Normal flow

Starts the game

	Actor	System
1)	Clicks on new game	
2)		Launches a new game

Alternate flows

Flow 1.1: Player clicks on Load Game option

	Actor	System
1.1.1)	Clicks on load game	
1.1.2)		Go to use case 2. Load game

2. Load game

Priority: Mid
Extends: -
Includes: - 1. Start new game
Participators: Player

Normal flow

Loads a saved game

	Actor	System
1)		Launches the saved game

Alternate flows

Flow 1.1: No saved game found

	Actor	System
1.1.1)		Launches a new game

4. Move

Priority: High
Extends: -
Includes: -
Participators: Player

Normal flow

Move in direction of key pressed

	Actor	System
1)	Player presses an arrow key	
2)		Moves the player's character in direction pressed

Alternate flows

Flow 2.1: Player is next to an obstacle and moves towards the obstacle

	Actor	System
2.1.1)		The character does not move but turns the player's character in the direction pressed

6. Range Attack

Priority: Mid
Extends: -
Includes: -
Participators: Player

Normal flow

Damage enemy or obstacle

	Actor	System
1)	Player click range attack key	
2)		Player’s mana is reduced by the mana cost of the range attack.
3)		Enemy or obstacle is within range and the enemy or obstacle loses the amount of health from the damage the player’s attack did.

Alternate flows

Flow 3.1: Killed enemy or destroyed obstacle

	Actor	System
3.1.1)		The damage received from the player’s attack reduces the enemy’s or obstacle’s health to zero and the enemy or obstacle is removed from the game.

Flow 3.2: Enemy or obstacle is not within range

	Actor	System
3.2.1)		-

Flow 2.1: Player is out of mana

	Actor	System
2.1.1)		-

8. Interact

Priority: Mid
Extends: -
Includes: Loot chest, Regenerate mana, Interact with friendly NPC, Change area
Participators: Player

Normal flow

Player stands in front of a Chest

	Actor	System
1)	Player click the interact key	
2)		Go to use case 9. Loot chest

Alternate flows

Flow 2.1: Player stands in front of a ManaFountain

	Actor	System
2.1.1)		Go to use case 10. Regenerate mana

Flow 2.2: Player stands in front of a FriendlyNPC

	Actor	System
2.2.1)		Go to use case 11. Interact with friendly NPC

Flow 2.3: Player stands in front of a AreaConnection

	Actor	System
2.3.1)		Go to use case 12. Change area

Flow 2.4: Player stands in nothing interactable

	Actor	System
2.4.1)		-

9. Loot chest

Priority: Mid
Extends: -
Includes: -
Participators: Player

Normal flow

Player loots all the items

	Actor	System
1)		All the items are transferred from the chest to the Player.
2)		The chest is displayed as opened.

Alternate flows

Flow 1.1: Player's inventory is full

	Actor	System
1.1.1)		-

Flow 1.2: Player's inventory becomes full while looting the chest

	Actor	System
1.2.1)		Items are transferred to the player until the player's inventory is full.

10. Regenerate mana

Priority: Mid
Extends: -
Includes: -
Participators: Player

Normal flow

Player regenerates mana

	Actor	System
1)		The player’s mana is set equal to its maximum mana.

11. Interact with friendly NPC

Priority: Mid
Extends: -
Includes: -
Participators: Player

Normal flow

	Actor	System
1)		- Not yet implemented

12. Change area

Priority: Mid
Extends: -
Includes: -
Participators: Player

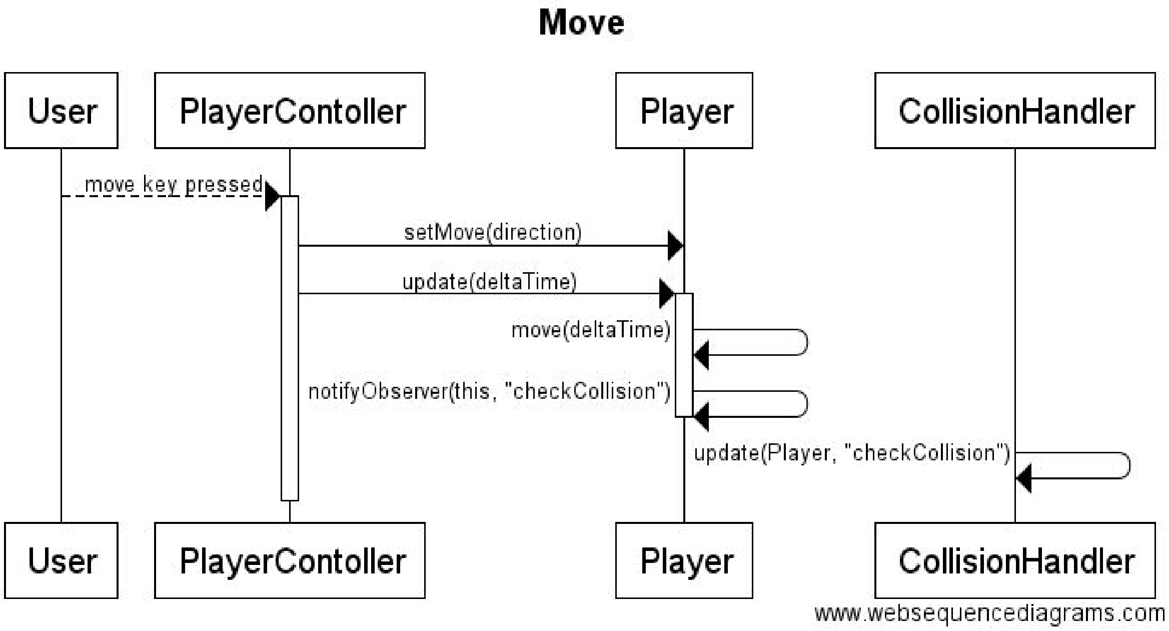
Normal flow

Player regenerates mana

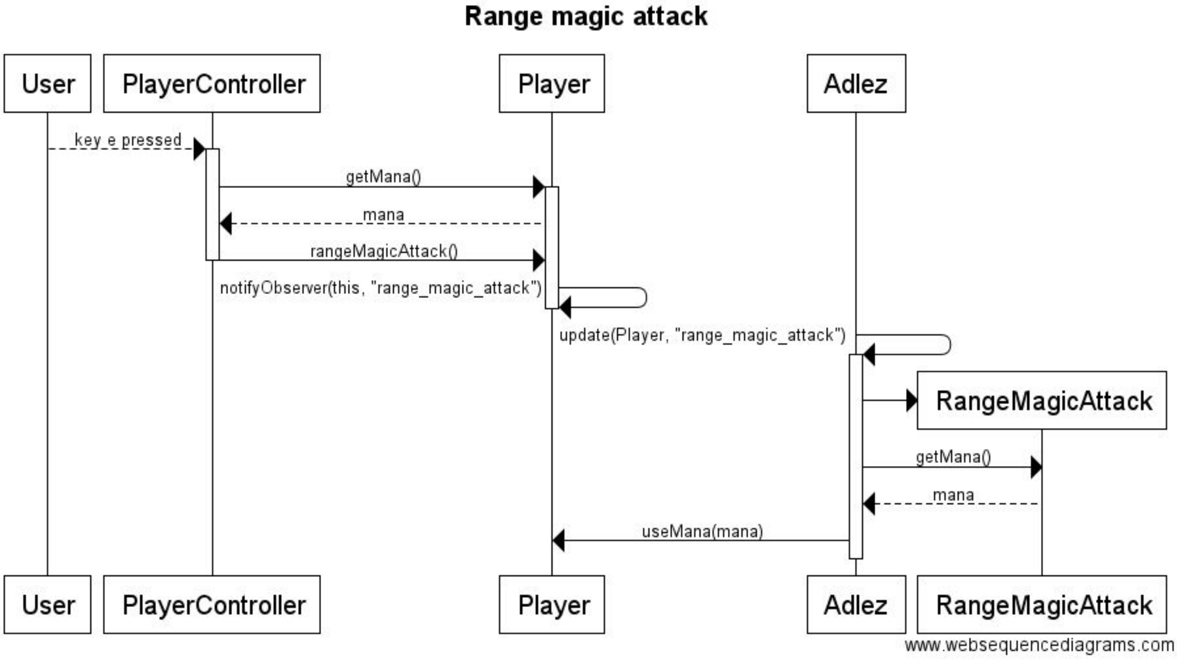
	Actor	System
1)		The player is transferred to the specific area that the AreaConnection is connected to.

Sequence diagrams

4. Move



7. Range magic attack


Range magic attack
User
PlayerController
Player
Adlez
key_e pressed
getMana()
mana
rangeMagicAttack()
notifyObserver(this, "range_magic_attack")
update(Player, "range_magic_attack")
RangeMagicAttack
getMana()
mana
useMana(mana)
User
PlayerController
Player
Adlez
RangeMagicAttack
www.websequencediagrams.com