WLAN-Opp-based RPG

Bachelor Thesis

Nicolas Imhof

January 13, 2013

Advisors: **Sacha Trifunovic**
Supervisor: **Prof. Dr. Bernhard Plattner**

Computer Engineering and Networks Laboratory, ETH Zurich

Abstract

During this bachelor thesis, a role playing game, short RPG, for android smartphones has been created. The player can choose one of three different character classes, fulfill quests, and improve his or her character. For communication with other players, the application makes use of opportunistic networks by using the WLAN-OPP service. This way the game serves as basis to gather data about the efficiency and usefulness of WLAN-OPP. To ensure that enough data is collected the application needs to be popular and therefore the final goal is to put it on the android market. To test the user acceptance a beta version of the game has been made available for four months. In total 17 people downloaded and played the game. This beta test gave some additional insights and was source for many changes after its evaluation.

Acknowledgments

First of all, I would like to thank all people that participated in the beta test of the game. The collected usage data and crash reports helped to debug the game and improved it in various ways. Thank also goes to the person behind the pseudonym "Lorc" for his large amount of freely usable black-and-white pictures. [1] Drawing so many icons for the game by myself would not have been feasible in addition to the game implementation. Special thank goes to my advisor Sascha Trifunovic for his extraordinary support, ideas and insightful discussions and the provided smartphones for testing purposes and last but not least to Prof. Dr. Bernhard Plattner, my supervisor.

Contents

Chapter 1

Introduction

Computer games are an inherent part of modern society. While in the past games were played mostly on the console or the personal computer, playing on mobile devices becomes more and more popular. There are over ten billion downloads of apps from the Android market. While statistics [2] show that a good quarter of those apps are games, there is still a deficit of free, good and textbased RPGs on the Android market. To go against this issue an RPG was created with the present work.

Modelling 3D figures or drawing images takes a lot of time. Therefore, a text based RPG offers the advantage that one can fully concentrate on the actual game content. Playing against the artificial intelligence can be fascinating, but multiplayer offers more diversity and a different kind of challange. Using direct device to device communication, the multiplayer part can be realized without the need of a fixed infrastructure like a server. Players don't need any internet connection and can play anywhere by forming Ad-Hoc networks.

The following chapter will go more into detail about WLAN-OPP, the service that enables the device to device connectivity, as well as other related work. Afterwards in Chapter 3 the basics of the game are introduced. The subsequent Chapter 4 depicts the design of the game and gives a concrete impression of game mechanical elements such as quests or battles. While the following Chapter 5 shows how important parts of the application have been implemented, the Chapter 6 evaluates the data from the beta test conducted with 17 people and shows which results have been derived from it. It also shows that the game has become quite complex. Then, in Chapter 7, some ideas are given on how the game could be further improved and extended. Finally, Chapter 8 summarizes what has been reached with the application and concludes that, in spite of the already wide content, it would still be a good idea to spend more time on the game and add more content, before it is put on the android market.

Chapter 2

Related Work

2.1 WLAN-OPP

WLAN-OPP is a service application developed at the ETH Zurich, that allows mobile devices to form opportunistic networks. In opportunistic networks, devices can connect and disconnect at any time and they offer the advantage that no fixed infrastructure is needed. Unfortunately, most mobile devices are not capable of Ad-Hoc networking, so opportunistic networks are not often used. Also using Bluetooth is not an option for opportunistic networking because it is too limited due to its short range. The solution proposed in the paper of *Sascha Trifunovic et al.* [3] is an alternative way that makes use of common smartphone features and realizes opportunistic networks by using existing WiFi networks as well as the access point mode of mobile devices. The proposed solution offers even some advantages over conventional Ad-Hoc networks; such as less battery power consumption. In the present work, the WLAN-OPP service is successfully and actively used for the communication between the players.

2.2 Comparable Games

The android market offers heaps of games to play. The more sophisticated games are mostly not for free and if they are then they offer the possibility to spend money ingame. Of course there are too many games out there than could have been tested in a search for related work. So the search was reduced to read the description of comparable games and only download and test those that are for free. The fact that only games were sought-after that are text based role playing games, reduced the range a lot. In the end there were some games found that had similar properties to the intended work but there was no exact match.

In the following three games are listed that share some commonalities with the created game.

Aardwolf RPG [4] is a purely text based RPG game that requires internet connection to play and that simulates a world where the players can interact with each other. The game requires the user to type a lot of text as actions, which seems quite inconvenient.

Wizard's Choice [5] is basically just a thrilling story that can be read and in which one has to make wise decisions in order to not die before having reached the end of the story. In this game there are no items that can be collected, there is no multiplayer part, and no character to level up. While the first story can be downloaded for free, the sequels demand a small charge of the player.

Shakes and Fidget [6] is originally a browser game but is also available as an app. It's the game that probably resembles the most to the present work. It has similar character classes and offers text based quests. However, the player is not allowed to make decisions to change the direction of the quest. It is also not possible for the player to trade with other players or to craft items. The game offers the possibility to buy "mushrooms" with real money, which then serve as currency in the game to buy equipment.

Chapter 3

Game

The created game belongs to the game genre of role playing games, short RPG. In general a role playing game lets the user control one or several game characters in a virtual world. Usually players can solve quests, gain levels and improve their game characters, which is something that can have a strong fascination on people, so that they spend a lot of time on the game. Depending on the game the actual goal can be to solve the main quest or sometimes it is to just make the game character as strong as possible. In other words: The journey ist the reward.

3.1 Gameplay

The player starts by creating a new character. Therefore, he or she has to choose a name and one of three available character classes. After the creation of a game character, one can actually start playing. At the beginning the character does not have any gold, which serves as currency in the game, and the inventory hardly contains any items. The few start items that there are depend on the chosen character class. The warrior's inventory starts with a rusty sword and armor as well as some coal which can be used for forging. The mage starts with a feeble wand, an old dirty mage robe and some empty phials which can be used to create potions. The thief has a dull dagger and a scale armor from the beginning, as well as some picklocks.

The player should first equip his or her game character with the weapon and the armor from the inventory and then choose one of the available quests. By solving the quest, the player usually gains experience and rewards like gold or new items. Often, there is an adversary to fight in a quest. In those battles the character usually loses health. It is then necessary to wait for the character's health bar to refill; otherwise the player takes an increased risk to fail in the battle of the next quest. When fighting, not only health is lost, but also the equipped weapon and armor lose durability and can break. Broken items can be repaired by the local blacksmith for gold. Gold can also be used to buy needed items from the merchant. For example a warrior can buy an iron ingot to forge a new weapon with his blacksmith skill.

The experience that is obtained in quests makes the character level up when a certain threshold is reached. That way, the character grows stronger and stronger and with every new level better items can be produced and equipped and new quests become eventually available.

3.2 Character Classes

The three character classes are warrior, mage and thief. Each of them has two unique skills and benefits slightly different from the attributes strength, intelligence and agility.

3.2.1 Warrior

Warriors have a lot of strength, which is why they can make use of heavy weapons such as iron swords and battle axes and often wear heavy armors that offer strong protection. With the skill "mining" the warrior is able to extract materials from ores, such as gems or metals. The gems can be sold to mages and the metals are often used by the warriors by their second skill "blacksmith", which allows them to forge new equipment like weapons or armors. Another advantage of this skill is that the warrior can repair broken equipment by himself which often saves a lot of money.

3.2.2 Mage

Mages are not strong but highly intelligent which allows them to make use of powerful magic wands as weapons. Because of their lack of strength, they usually wear robes or light chain armors. The skill "enchanting" allows a mage to create new magic wands and to extract magic powder from gems which can then be used to enchant equipment to permanently provide additional benefits like increased attributes. Furthermore, with "alchemy" a mage knows how to mix magic potions. Depending on the used ingredients, a potion can increase attributes for some time or regenerate health when consumed. It is even possible to generate potions with permanent effects, but the ingredients needed for this are very seldom and expensive.

3.2.3 Thief

Thieves compensate their deficit of strength with high agility which greatly allows them to evade attacks. They often use daggers as weapons and wear chain or scale armors. With the "lock picking" skill, a thief is able to open chests and to get the treasures that have been locked inside. Other classes like the warrior or mage have no other choice than to either sell a locked item or to ask a thief for help. The second skill of the thief, toxicology, allows to create poisons and antidotes. Poisons can be used on weapons to inflict additional damage to an adversary on hit or to decrease the target's attributes. Unlike enchantments, poison effects on weapons loose their effect after some battles and need to be reapplied.

3.3 Quests

The quests are structured by four categories. The category **simple quest** contains the basic quests that can be solved alone and without the need of any special items. The strength of the enemies encountered in the quests is adapted to the level of the player but can vary in difficulty. The category **group quest** contains quests that are obviously intended for more than one player and the category **special quest** offers quests that force the player to get certain items or to look for another player with a certain skill. The last category is the **main quest**. It's a sequence of quests that tell a story and which are not dynamically adapted to the level of the player. At the moment the main story can usually be completed around level twelve.

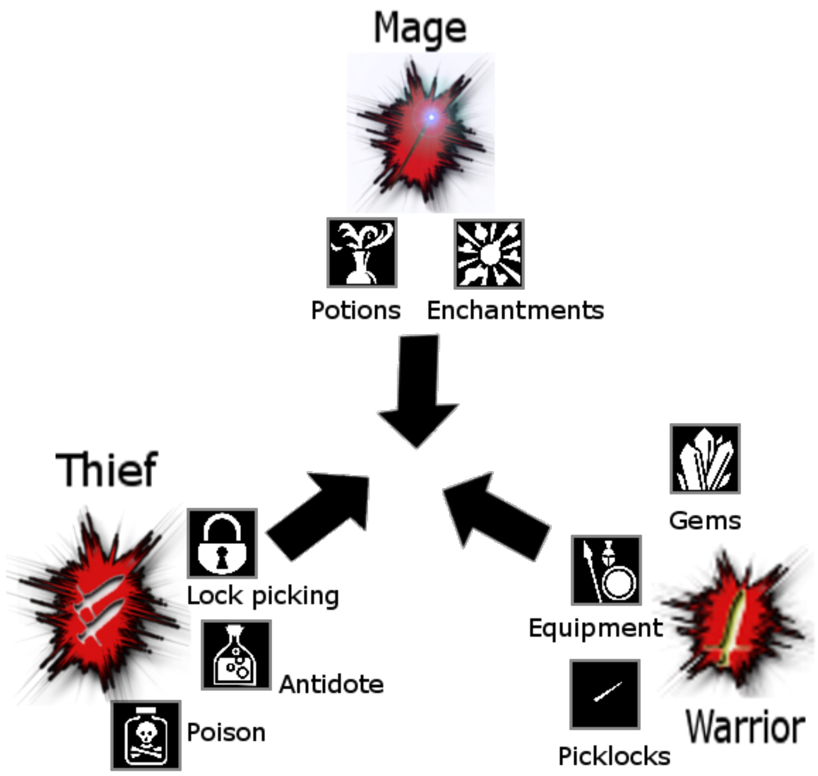
A concrete example of a quest is presented later in the design chapter in the section 4.2.4.

3.4 Multiplayer

The game offers various possibilities to interact with other players. There is a global chat where the players' messages address all locally connected players, as well as the possibility to have private chat conversations. Players can have duels to compare their strength and they can exchange items and gold by trading. Trading is an essential part of the game. Since all character classes need different resources and services from other classes, a player that trades can improve the character's skills faster and is usually richer. The figure 3.1 shows the essential trade dependencies.

Note, that there could also be a second diagram drawn with inverted arrows, where all the items are shown that the character classes need as ingredients for their skills.

Figure 3.1: Trade dependencies



Chapter 4

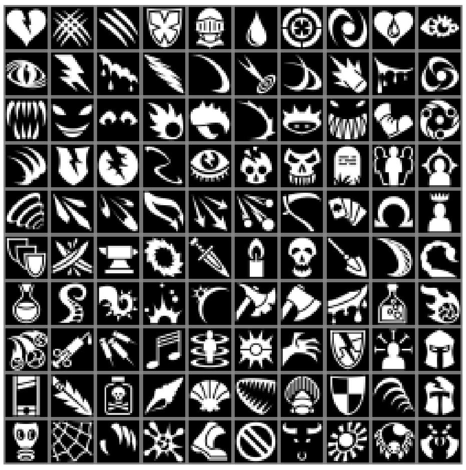
Design

4.1 Appearance

The first intention, when creating the game, was to focus on content and functionality and not to spend much time on its appearance. Because drawing images is a time consuming work, the game should be a simple looking text based RPG. However, during implementation this view changed and time was then spent on designing own buttons and backgrounds with GIMP [7], a freely available graphics editing programm.

Luckily, a huge collection of about 700 small and freely available RPG black-and-white pictures (Figure 4.1) where then found in the internet. Those pictures served perfectly as icons for the game and the fact that there were so many of them had the advantage of homogeneous style. The pictures are under a license [8] which allows modification and the commercial use of them.

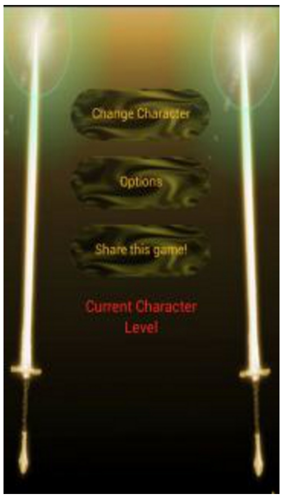
Figure 4.1: Some pictures of the found icon set.



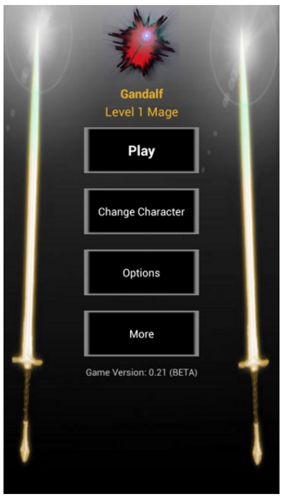
The found icons improve the optical representation of the game remarkably. However, the quadratic style as well as the black and white colors of the new icons did not fit with the existing design of self made, rounded brown buttons.

Therefore, the buttons and backgrounds were redesigned and the current style was finally established.

Figure 4.2: Comparison of old and new design.



(a) Old design



(b) New design

4.2 Navigation

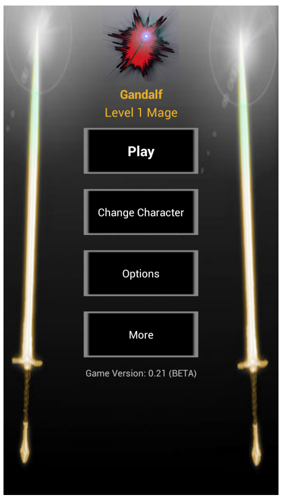
This chapter shows the possible interactions of the player with the user interface and the transitions between the activities.

4.2.1 Starting Screen

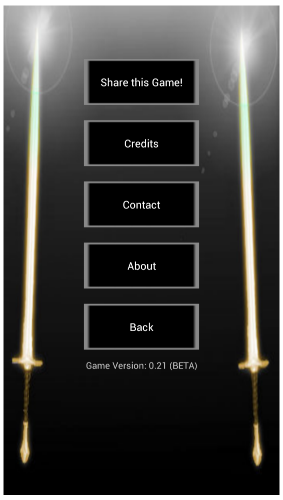
In figure 4.3a the starting screen of the game is shown. When the game is started, the screen is faded in slowly using an animation. At the top of the screen, information about the currently selected game character is shown. If the game is started for the first time the screen looks different, because no game character exists. The play button is then gone and the **Change Character** button is renamed to **Create Character** and does directly lead to the character creation instead of showing the list of created characters. At the bottom of the screen, the current game version is displayed. If a new game version is available, then an additional **Download** button and a red text informing about it, is visible. A click on that button directly starts the download of the new version.

The button **Options** opens a new activity that allows the player to change certain parameters like disabling the sending of crash reports or the checking for updates on game start. The button **More** leads to a new activity whose layout is displayed in figure 4.3b. Again there are several buttons which all open new activities. The activity of the first button **Share this Game!** helps to easily share the game, by providing the link for the download and the important hint that "unknown sources" needs to be activated in the phone settings. In addition, there is a picture of a QR code with the same link encoded, so that the user does not need to enter the link in his or her phone if a scanner is available.

Credits shows information about who was involved in the creation of the game and **Contacts** encourages the player to give feedback by sending an email or to register at the official game forum. Just for fun, a dancing android man has been added to this activity. A click on the **About** button explains the original purpose of the game; that it was created as a bachelor thesis and that it is intended to gather data for WLAN-OPP. The **back** button finishes the activity, so that the starting screen appears again.



(a) Main screen



(b) More

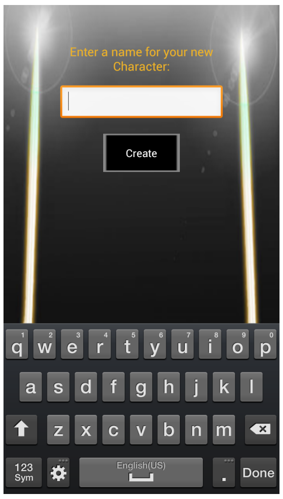
4.2.2 Game Character Creation

To create a new character the player is first asked to enter a name for the new character (Figure 4.3c) and then to choose a character class (Figure 4.3d). Of course the chosen name has to comply with a certain minimal and maximal length.

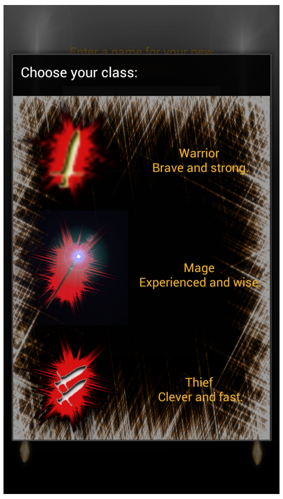
4.2.3 The PlayActivity

With a click on the **Play** button the **PlayActivity** is started. (Figure 4.4) This is the central activity from which the player can reach everything. The yellow long bar at the top shows the character's health bar. Below the health bar the current character's name, in this case "Gandalf", and the current level is indicated. The very small progress bar below the level displays the experience of the character. When it is full, the character receives a level up.

Figure 4.3: Creating a new game character.

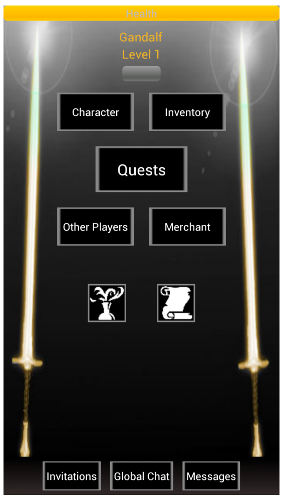


(c) Choosing a name for the game character.



(d) Deciding for a character class.

Figure 4.4: The PlayActivity.



Various buttons are located below the experience progress bar. In the following each button and its connected activities are explained.

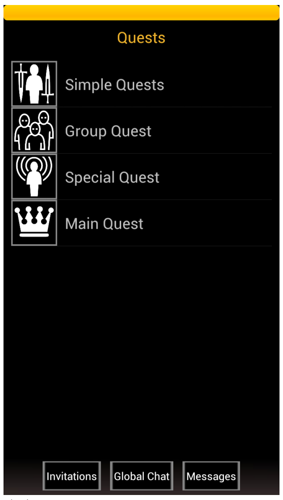
4.2.4 Quests

The **Quest** button is centered in the middle of the PlayActivity's layout and has increased size to signal its importance to the user, since the quests are the heart of the game. A click on it opens a new activity that shows the four quest categories in list. (Figure 4.5a) A click on one of the categories opens again a new activity with a list, which shows the available quests of that category. (Figure

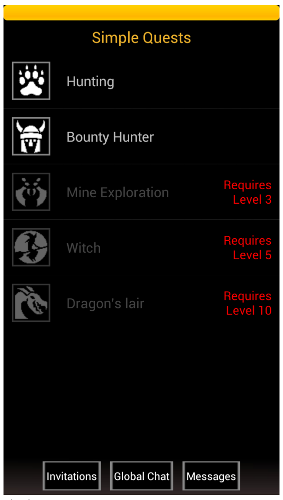
4.5b) Not all quests are straight away available from the beginning. Some of them require the game character to have reached a certain level. That's the reason why in the picture only the quest "Hunting" and "Bounty Hunter" are executable. The others are greyed out. After a quest has been done it takes a while, depending on the quest, until a quest of the same type can be done again. The quest is then greyed out and simplified information about when the it will be available again, like for example "Less than ten minutes", is displayed next to it. A click on the quest shows the exact waiting time.

In the category special quest there is an additional button **Change Quest** which allows the player to discard the current quest to get a new one after a certain time. This has been added because sometimes the special quests are hard to fulfill. With the possibility to change the quest, the player can decide if he or she wants to keep trying it or to change it for a new quest that might better suit the current situation.

Figure 4.5: Quests



(a) The four quest categories.



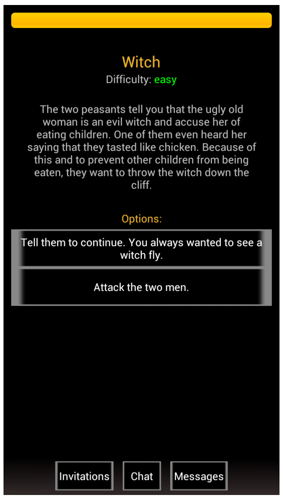
(b) The list of simple quests.

In the following a short example of a quest is given.

Example Quest

A quest always starts with some text telling a short story, optionally followed by a description of the encountered adversary and/or the possible reward. At the bottom part of the quest are the options listed that the player can choose. In the figure 4.6a the player can choose to either attack the two angry peasants, that want to throw an old woman down a cliff, or he can decide to just observe the situation. Depending on the player's choice the quest develops in a different direction. If the player decides to attack the two men, then the quest will end with a battle. Otherwise, by picking the passive option, the player might see the witch falling down the cliff and receives an annoying witch curse that lowers the character's attributes for some time. (See figure 4.6b) The outcome of a decision on a quest is not always deterministic. Sometimes there are several possibilities on how the quest continues when a certain option is picked. For example, it can also happen that the witch transforms the two angry peasants with her black magic into chickens and then attacks the player. (See figure 4.6c) Note that a click on the witch's icon reveals some additional information.

Figure 4.6: Witch quest.


Witch
Difficulty: easy

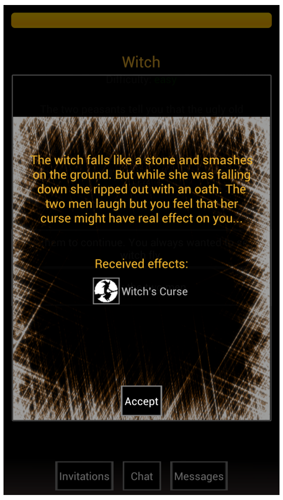
The two peasants tell you that the ugly old woman is an evil witch and accuse her of eating children. One of them even heard her saying that they tasted like chicken. Because of this and to prevent other children from being eaten, they want to throw the witch down the cliff.

Options:

Tell them to continue. You always wanted to see a witch fly.

Attack the two men.

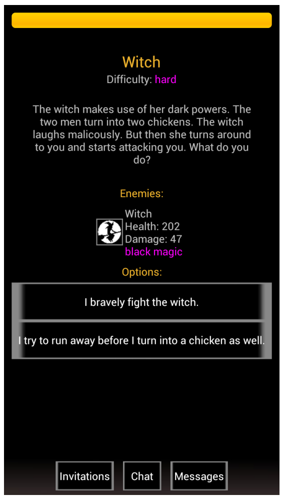
(a) Making decisions in quests.


The witch falls like a stone and smashes on the ground. But while she was falling down she ripped out with an oath. The two men laugh but you feel that her curse might have real effect on you...

Received effects:

Witch's Curse

(b) Getting cursed by the witch.


Witch
Difficulty: hard

The witch makes use of her dark powers. The two men turn into two chickens. The witch laughs malicously. But then she turns around to you and starts attacking you. What do you do?

Enemies:

Witch
Health: 202
Damage: 47
black magic

Options:

I bravely fight the witch.

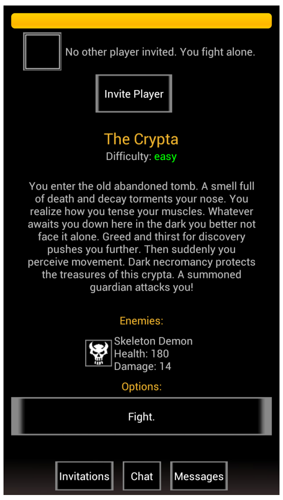
I try to run away before I turn into a chicken as well.

(c) Getting attacked by the witch.

Group Quests

In a group quest (Figure 4.7) the player can click on the **Invite Player** button to look for another quest participant. Group quests can also be done alone, but obviously it is much easier with the support of another player.

Figure 4.7: Group quests offer the possibility to invite other players.


No other player invited. You fight alone.

The Crypta
Difficulty: easy

You enter the old abandoned tomb. A smell full of death and decay torments your nose. You realize how you tense your muscles. Whatever awaits you down here in the dark you better not face it alone. Greed and thirst for discovery pushes you further. Then suddenly you perceive movement. Dark necromancy protects the treasures of this crypta. A summoned guardian attacks you!

Enemies:

Skeleton Demon
Health: 180
Damage: 14

Options:

Fight.

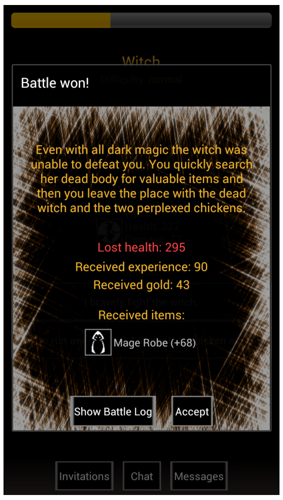
4.2.5 Battles

Sooner or later the player will have to fight an adversary in a quest. The outcome of a battle is calculated automatically, so the player can not take any action during a battle. Depending on the quest and the player's decisions in the quest, the player or the adversary has the first attack.

After a battle has taken place, the player gets informed about the result. (Figure 4.8a) The player sees directly how much health was lost because of the battle, how much experience gained, gold found and which items were received as a reward. With a click on the button **Show Battle Log**, more details about the battle can be read. (See figure 4.8b) Important information, such as critical attacks, evaded attacks, broken equipment or received poison, is highlighted with colors.

Battles can also include more then two entities. In figure 4.8c the player has to fight against several spiders.

Figure 4.8: Battles



Battle won!

Even with all dark magic the witch was unable to defeat you. You quickly search her dead body for valuable items and then you leave the place with the dead witch and the two perplexed chickens.

Lost health: 295

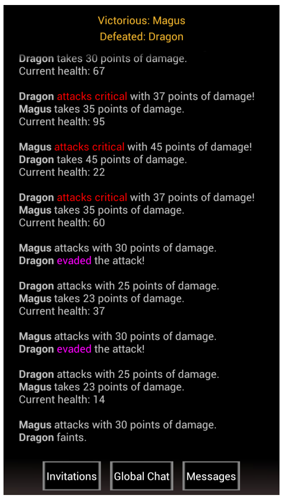
Received experience: 90

Received gold: 43

Received items:

Mage Robe (+68)

(a) Getting the reward.



Victorious: Magus
Defeated: Dragon

Dragon takes 30 points of damage.
Current health: 67

Dragon attacks critical with 37 points of damage!
Magus takes 35 points of damage.
Current health: 95

Magus attacks critical with 45 points of damage!
Dragon takes 45 points of damage.
Current health: 22

Dragon attacks critical with 37 points of damage!
Magus takes 35 points of damage.
Current health: 60

Magus attacks with 30 points of damage.
Dragon evaded the attack!

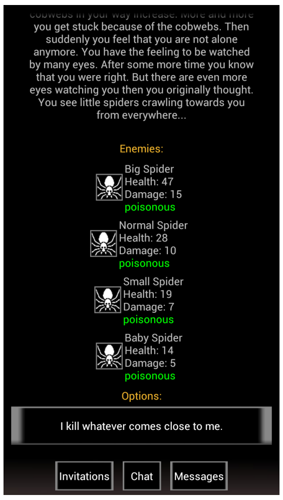
Dragon attacks with 25 points of damage.
Magus takes 23 points of damage.
Current health: 37

Magus attacks with 30 points of damage.
Dragon evaded the attack!

Dragon attacks with 25 points of damage.
Magus takes 23 points of damage.
Current health: 14

Magus attacks with 30 points of damage.
Dragon faints.

(b) The battlelog.



cobwebs in your way increase. More and more you get stuck because of the cobwebs. Then suddenly you feel that you are not alone anymore. You have the feeling to be watched by many eyes. After some more time you know that you were right. But there are even more eyes watching you then you originally thought. You see little spiders crawling towards you from everywhere...

Enemies:

Big Spider
Health: 47
Damage: 15
poisonous

Normal Spider
Health: 28
Damage: 10
poisonous

Small Spider
Health: 19
Damage: 7
poisonous

Baby Spider
Health: 14
Damage: 5
poisonous

Options:

I kill whatever comes close to me.

(c) Multiple adversaries.

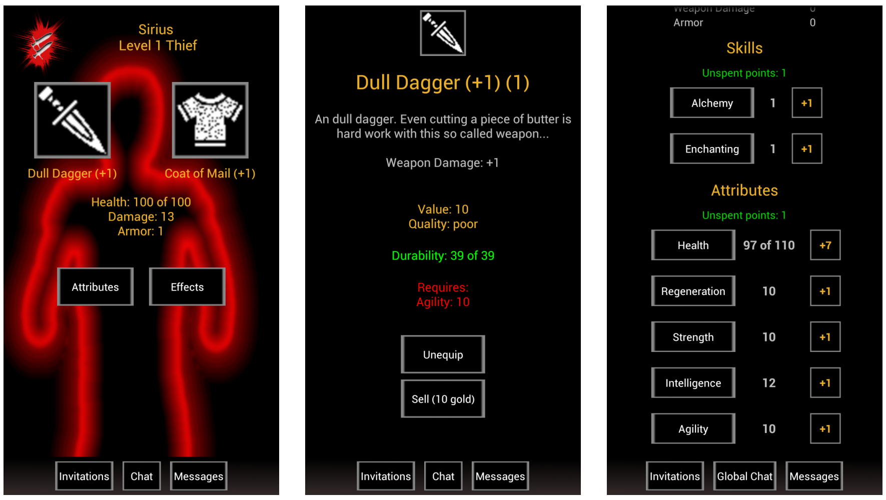
4.2.6 The Game Character

By pressing the **Character** button information about the character is displayed as visible in figure 4.9a. The two icons show the currently equipped weapon and armor. In this case it's a dull dagger and a coat of mail. A click on one of the icons reveals more information about the item and allows interactions such as unequiping the item, selling it to the merchant or repairing it. (Figure 4.9b)

The button **Effects** opens a new activity with a list of all effects that are currently influencing the character. The activity opened by the button **Attributes** allows the player to examine the values of all attributes and skills of the character. The figure 4.9c shows the level of the two skills of a mage (alchemy and enchanting) and the five base attributes health, regeneration, strength, intelligence and agility. A click on an attribute reveals more exact information about it. So for example a click on the attribute regeneration informs the user about how long it takes to regenerate the character to full health.

The small buttons with the yellow numbers on the right side in figure 4.9c are only visible after the player reached a level up and therefore earned an attribute point. Attribute points can be spent on attributes by clicking on those small buttons. Note, that the picture does not show all information about the character. So for example the two attributes **Accuracy** and **Evading** below **Agility** are not shown.

Figure 4.9: Clicking on the **Character** button reveals more information.



An dull dagger. Even cutting a piece of butter is hard work with this so called weapon...

(a) A Character. (b) Item description. (c) Character attributes.

4.3 Attributes

Game characters have various attributes that determine the character's strength in battles. On every third level up, the character's regeneration increases by one point. On every single level up, the maximum health increases by ten and the player gets one attribute point which can be spent on either health, regeneration, strength, intelligence or agility. It is the player's choice which attribute he or she wants to improve by spending the point. If the player decides to spend the point on health then the maximum health of the character is increased by seven. The other attributes only get raised by one.

In the following a short overview of the different attributes is given.

Health
: Health is lost in battles and regenerated slowly over time. A battle is lost if the health drops to zero.

Regeneration
: Regeneration determines how fast a character regenerates health after a battle.

Strength
: A certain amount of strength is often a requirement for wearing equipment.

Besides this, it increases the chance for a critical strike (a more powerful attack) for warriors and thieves. Additionally, warriors' attacks become stronger with every point of strength.

Intelligence
Intelligence is very important for mages. It allows them to make use of powerful magic wands and increases their attack damage and chance on a critical strike. For warriors and thieves it is of no importance in a fight, but it has a good effect on their skills. An intelligent warrior is more successful in mining and has a better chance to forge equipment of high quality and an intelligent thief has fewer difficulties to pick a lock and generates deadlier poisons.

Agility
With more agility a character has a bigger chance to evade attacks as well as to successfully hit an enemy with the own attack. For thieves agility is of big importance because it directly increases the attack damage.

Evading
Evading improves the character's chance to completely evade an attack of an adversary.

Accuracy
Accuracy is the counter part of evading. It decreases the chance that an adversary manages to evade an attack of the game character.

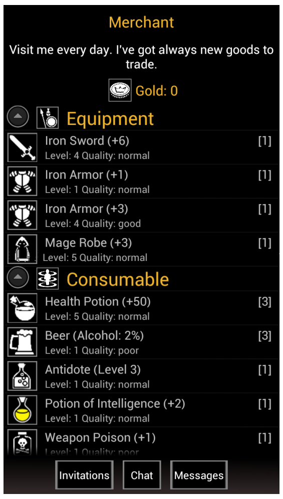
Armor
A game character does not come with armor by itself. It is only acquired by equipped items. In a battle, armor helps the character to survive longer by decreasing the taken damage of adversarial attacks.

Weapon Damage
Like armor, the weapon damage is only acquired by equipped items. Weapon damage directly increases the damage dealt by the attacks of the game character.

4.3.1 Inventory And Merchant

The **Inventory** button opens the game character's inventory, where all items are shown in an expandable list that are currently in the character's possession. The activity opened by a click on the **Merchant** button is very similar. The difference is that it shows the items that the merchant is offering for sale. (Figure 4.10) In the list, the items are represented by their name, their level and the quality. On the right side, there is a number indicating how many exemplars of the item are available. A click on an item of the list creates a new activity that reveals all the information of the selected item and shows buttons for interaction. If the click was on an item of the merchant there is actually just a **Buy (X gold)** button visible, where X is the amount of requested gold. Otherwise, if the item is in the player's inventory, then, depending on the item, are various options available. The item can be sold to the merchant, equipped/unequipped, consumed, used on a weapon, repaired, lock picked and opened.

Figure 4.10: The merchant.


Merchant

Visit me every day. I've got always new goods to trade.

Gold: 0

Equipment

Iron Sword (+6) Level: 4 Quality: normal	[1]
Iron Armor (+1) Level: 1 Quality: normal	[1]
Iron Armor (+3) Level: 4 Quality: good	[1]
Mage Robe (+3) Level: 5 Quality: normal	[1]

Consumable

Health Potion (+50) Level: 5 Quality: normal	[3]
Beer (Alcohol: 2%) Level: 1 Quality: poor	[3]
Antidote (Level 3) Level: 1 Quality: normal	[1]
Potion of Intelligence (+2) Level: 1 Quality: normal	[1]
Weapon Poison (+1) Level: 1 Quality: poor	[1]

4.3.2 Other Players

The button **Other Players** shows all other human players that are currently connected over the same network as the player.

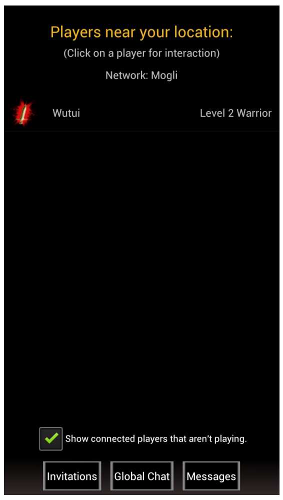
In figure 4.11a the player is connected with another player called Wutui that is playing with a level two warrior. The player can also see to which network the phone is currently connected. This is very helpful because if two player want to play together but are not in the same network, they won't see each other in the game. The checkbox at the bottom allows the player to see all the neighbor devices that have been found by WLAN-OPP in the same network and which are running the WLAN-OPP service. If the checkbox is unchecked, only the neighbors are listed that have installed the game and that are currently running it. A click on a player in the list reveals more information about that player's character and displays options for interaction.

In figure 4.11b the player can send a private text message to Wutui, have a duel to see who is stronger or start a trade as shown in figure 4.11c. If the player decides to do such an action, a message is sent to the other player which creates a popup that appears in a fluent animation. If the action was a trade request, then the button **Invitations**, which is visible on most layouts and located at the bottom left, changes it's name to **Invitations (1)** where the number in the brackets is the number of currently unanswered invitations such as trade requests or quest invitations. A click on that button shows a list of the invitations.

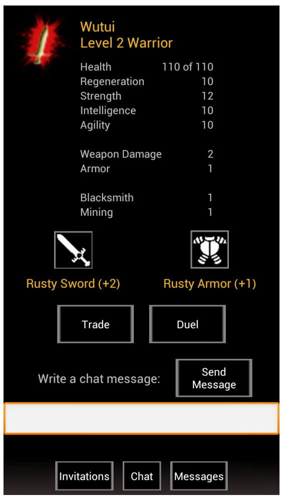
The **Message** button at the bottom right behaves in a very similar manner but it shows a list of chat conversations with other players.

The **Global Chat** button between those two buttons changes its color if there are unread messages that have been sent over the global chat. A click on that button opens the GlobalChatActivity which shows the global chat messages and has a text field and a send button to allow the user to send messages to all connected players. It also shows the number of connected players, so that the player knows how many people will receive the message when it is sent.

Figure 4.11: Other players


Players near your location:
(Click on a player for interaction)
Network: Mogli
Wutui Level 2 Warrior
✓ Show connected players that aren't playing.

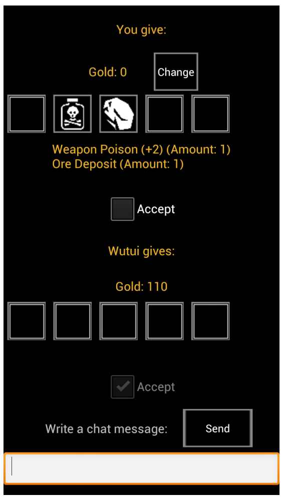
(a) Connected with a warrior named Wutui.


Wutui
Level 2 Warrior

Health	110 of 110
Regeneration	10
Strength	12
Intelligence	10
Agility	10
Weapon Damage	2
Armor	1
Blacksmith	1
Mining	1

Rusty Sword (+2)
Rusty Armor (+1)

(b) Possible interactions with Wutui.


You give:
Gold: 0
Weapon Poison (+2) (Amount: 1)
Ore Deposit (Amount: 1)
Accept
Wutui gives:
Gold: 110
✓ Accept

(c) Wutui offers 110 gold for two items.

Trade

If a player clicks on an invitation, which is a trade request, then the TradeActivity is started for both of them. Each player sees ten empty item slots. Five of them belong to one player, the other five to the other player. By clicking on one of the own empty item slots, the inventory of the player is opened and an item can be selected for the trade. A click on a slot that is not empty lets the user change the amount of the selected item that he or she wants to trade. A click on an item slot of the other player shows, if not empty, information about that item. Besides items, also gold can be traded. A simple button allows the user to enter the amount of gold that he or she wants to give to the other player. To complete the trade, both players have to click on "Accept". Whenever a player changes an item or the amount of gold, both accept checkboxes are reset.

The players can communicate directly in the TradeActivity by sending chat messages. The exchanged messages are displayed below the white field where the player can enter a chat message.

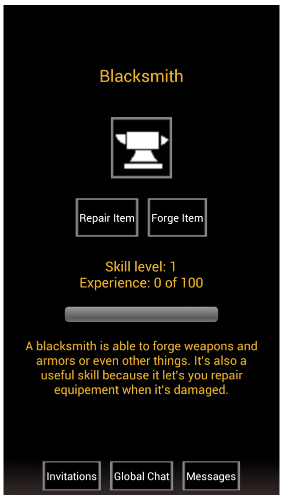
4.3.3 Skills

With a click on one of the two icons below the buttons in the PlayActivity a new activity with information about the selected skill is opened. As seen in figure 4.12a the skill level and experience is displayed as well as a short description of the skill. In the following the skills of the character classes are presented in more detail.

Blacksmith

A click on the button **Forge Item** shows an expandable list with blacksmith recipes. (Figure 4.12b) Note, that not all recipes are available from the beginning. Most of them require a certain skill level. When the player selects a recipe, he or she can choose to see the description which shows information about the item that will be produced by the recipe, as well as the required resources. Conveniently, missing resources are colored in red. If the player decides to use a recipe the ForgeActivity is opened showing an animated fire and its temperature. In figure 4.12c the forging activity is shown. In this case an Iron Sword is being produced. By pressing the hammer button, the quality of the produced item can be improved. A sword of a higher quality deals more damage and can be sold for more gold. Of course you can not just hammer forever until the maximum quality is reached. Instead there is a chance depending on skill level to break the item. Before the item breaks a warning appears. The player can still try to hammer more but then there is the risk that all the materials and all the work was in vain. Additionally you can add coal to the fire to increase the temperature which decreases the risk to break the produced item when hammering. Like this a player can try to create an item of high quality by spending additional resources.

Figure 4.12: Skill Blacksmith

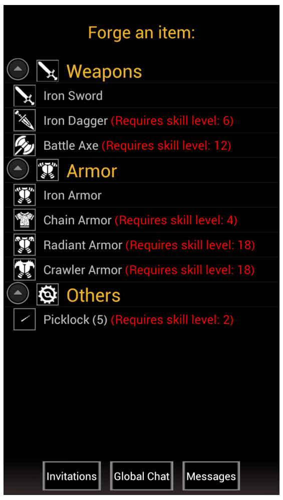


Blacksmith

Skill level: 1
Experience: 0 of 100

A blacksmith is able to forge weapons and armors or even other things. It's also a useful skill because it let's you repair equipement when it's damaged.

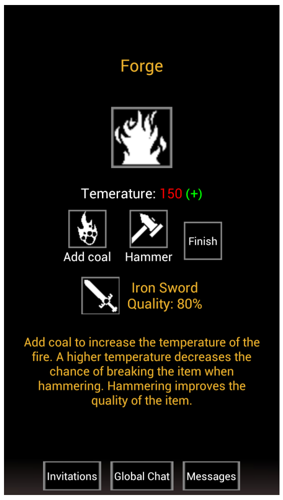
(a) The blacksmith skill.



Forge an item:

Weapons
Iron Sword
Iron Dagger (Requires skill level: 6)
Battle Axe (Requires skill level: 12)
Armor
Iron Armor
Chain Armor (Requires skill level: 4)
Radiant Armor (Requires skill level: 18)
Crawler Armor (Requires skill level: 18)
Others
Picklock (5) (Requires skill level: 2)

(b) Blacksmith recipes.



Forge

Temerature: 150 (+)

Iron Sword
Quality: 80%

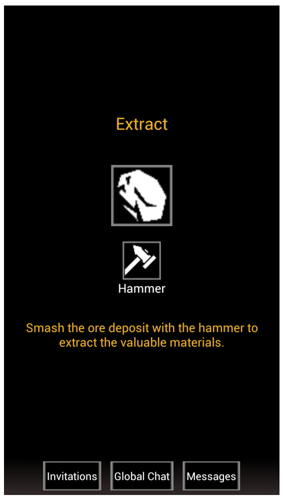
Add coal to increase the temperature of the fire. A higher temperature decreases the chance of breaking the item when hammering. Hammering improves the quality of the item.

(c) Forging an item.

Mining

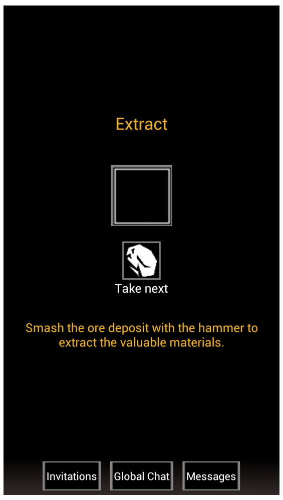
The design of the mining skill is quite simple. If the player has an item of type "Ore Deposit" in the inventory then this skill can be used to extract materials from it. For each "Ore Deposit" the player can click the hammer icon three times. After that, the item is destroyed and the next item can be taken for extraction, if available. (Figure 4.13b) On every click on the hammer the player has a chance to extract a material. In figure 4.13c the player was lucky and extracted an item.

Figure 4.13: Skill Mining



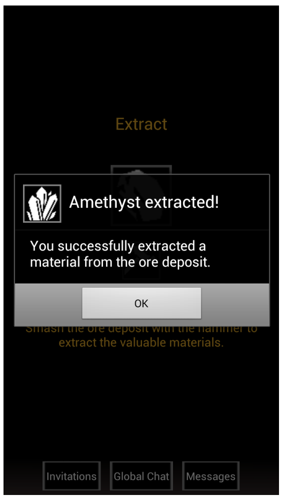
Smash the ore deposit with the hammer to extract the valuable materials.

(a) Extracting.



Smash the ore deposit with the hammer to extract the valuable materials.

(b) Taking the next.



Amethyst extracted!

You successfully extracted a material from the ore deposit.

Smash the ore deposit with the hammer to extract the valuable materials.

(c) Successful extracting.

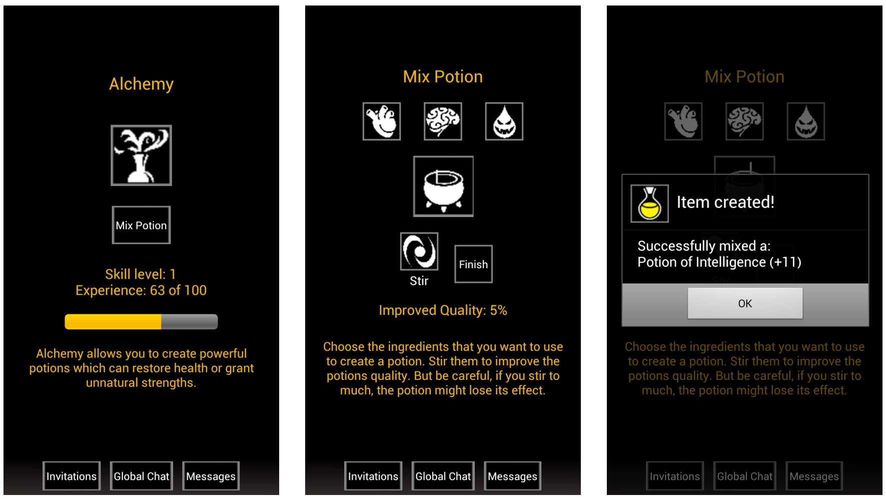
Alchemy

Alchemy is a very complex skill. It allows a mage to mix all sorts of potions by mixing all sorts of ingredients. Potions can increase attributes for some time or regenerate health when consumed. It is even possible to generate potions with permanent effect, but therefore ingredients are needed that are very rare and expensive. What distinguishes alchemy substantially from other skills like enchanting or blacksmith is that there are no recipes. Instead, the player has to figure out by himself which combination of ingredients lead to which potion.

The figure 4.14b shows the mixing of a potion by combining the items heart, brain and witch blood which results in a potion of intelligence (+11) as visible in figure 4.14c.

The resulting potion of a combination of ingredients is not hard coded. Every ingredient has certain tendencies to favor certain attributes. For example the ingredient "heart" will increase the tendency that the resulting potion will increase the player's health or regeneration, but it does not contribute much to intelligence or agility. So the combination of three hearts will definitely generate a good potion of health. But when combining a heart with two brains, the brains will dominate the effect and lead to a potion that increases the attribute intelligence. However, it will be less effective as if the combination consisted of three brains.

Figure 4.14: Skill Alchemy



Skill level: 1
Experience: 63 of 100

Alchemy allows you to create powerful potions which can restore health or grant unnatural strengths.

Improved Quality: 5%

Choose the ingredients that you want to use to create a potion. Stir them to improve the potions quality. But be careful, if you stir to much, the potion might lose its effect.

Item created!

Successfully mixed a:
Potion of Intelligence (+11)

(a) Skill description. (b) Adding ingredients. (c) Potion created.

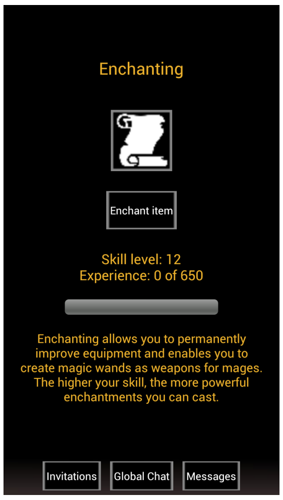
Enchanting

Enchanting grants the possibility to permanently improve a weapon or armor. Enchanted items provide an additional attribute bonus, health or armor. Only one enchantment can be active on an item at the same time. With enchanting a mage can also create magic wands which can then serve as weapons. The required materials for enchanting are gained by extracting magic powder and essences from gems like amethysts or diamonds. The figure 4.15b shows an iron armor being enchanted with +23 Health.

Lock Picking

Sometimes the player finds locked items in quests and as a thief the player can try to unlock them. To do this, the player simply clicks on the locked item in the inventory. The description of the item appears and the player has an additional button **Pick Lock (50%)** where the number in the brackets describes the chance to be successful. (Figure 4.16a) To unlock an item a thief needs a picklock. Unfortunately picklocks are very fragile and on every attempt to unlock an item the used picklock might break. So if the player decides to pick the lock of an item, either a success message or a message informing about the broken picklock is shown. With a higher skill level, the chance to break a picklock decreases and it gets easier to open even difficult locks. Picklocks can either be bought by the merchant or produced by a warrior with the skill "blacksmith". If the thief successfully unlocked an item, the **Pick Lock** button disappears and a new button **Open** is shown. (Figure 4.16b) The item can now be opened by anybody to see what is inside.

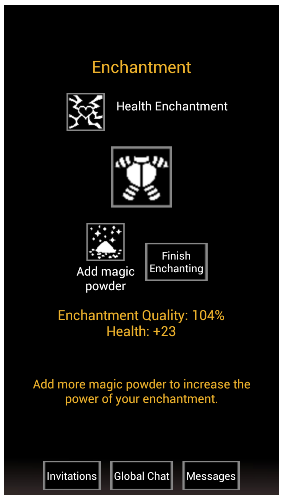
Figure 4.15: Skill Enchanting


Enchanting

Skill level: 12
Experience: 0 of 650

Enchanting allows you to permanently improve equipment and enables you to create magic wands as weapons for mages. The higher your skill, the more powerful enchantments you can cast.

(a) Skill description.


Enchantment

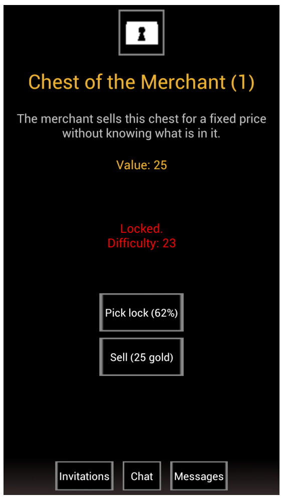
Health Enchantment

Enchantment Quality: 104%
Health: +23

Add more magic powder to increase the power of your enchantment.

(b) Enchanting an item.

Figure 4.16: Skill Lock Picking

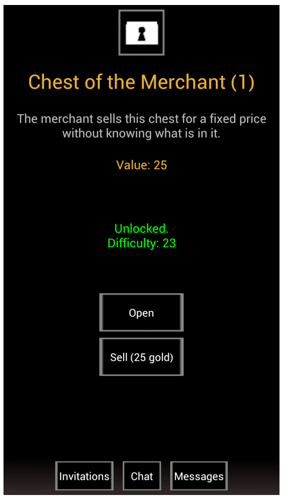

Chest of the Merchant (1)

The merchant sells this chest for a fixed price without knowing what is in it.

Value: 25

Locked.
Difficulty: 23

(a) Locked chest.


Chest of the Merchant (1)

The merchant sells this chest for a fixed price without knowing what is in it.

Value: 25

Unlocked.
Difficulty: 23

(b) Unlocked.

Toxicology

The second skill of the thief, toxicology, which allows to create poisons and antidotes is mostly similar to alchemy, except that the player can not mix three arbitrary ingredients but has to seclect a recipe, like in the blacksmith skill.

Chapter 5

Implementation

5.1 General

The application was implemented using eclipse as software development environment together with the Android plug-in. For the implementation of the game only one additional library was used besides the WLAN-OPP library. The additional library is the Gson 2.2.1 library [9] which helps to automatically map fields of classes to JSONObjects and vice versa. This functionality was really useful because many data structures needed a conversion to a JSONObject in order to be able to send them to other devices.

In general one could say that the code of the application is low coupled which is mostly due to its distribution with the Android activities. Parts can easily be changed without having an effect on others. A metrics plug-in for eclipse revealed about the application, that there is a total number of 23979 lines of code and 172 Java classes in total.

5.2 Server Structure

In order that the game could be easily shared and updated, a server structure was created. With the link *http://n.ethz.ch/student/nimhof/MyRPG/download.php* the latest version of the game can be downloaded. The used PHP code offers the advantage that it counts how many times the file was downloaded and stores the result in another file. The link for the WLAN-OPP service, which needs to be downloaded seperately, is provided in the game as soon as the player accesses a multiplayer component.

Furthermore the server contains the files *checkForUpdates.php, createUsageReport.php* and *createCrashReport.php.* The first one, when accessed by the game, returns the number of the latest game version. Like this the game can compare its own version to the one available on the server. The second file is used to store information at the server about how long and especially how the player played the game. For every player, a separate folder is created, which contains a subfolder for each game character. The last file handles the creation of crash reports. Here the files do not need to be ordered by the game characters. Instead, a daily folder is created. More details about the usage and crash reports is revealed in section 6.1.1 and 6.1.2, respectively.

5.3 Multiplayer

The communication with other devices is handled by the classes of the *connection* package. If information needs to be sent to another device, then the *Connection* and *Protocol* class are directly used. The *Connection* class offers the public methods to send a message to a single IP address or to all connected neighbors. The message is then sent by an instance of the class *SendTask*, which implements the interface *Runnable* so that it can be started as thread, since the GUI thread is not allowed to open a TCP connection. All the messages that can be sent and interpreted correctly by the game are generated by the static methods of the *Protocol* class. For instance, if the player wants to attack another player, then the following code will generate the message and send it to the other device.

```
String message = Protocol.getAttackPlayerMessage();
Connection.sendMessage(IP, message);
```

The *Protocol* class takes care of creating the message. In this case, the message is encoded as JSONObject with three parameters. (Figure 5.1) The first parameter serves as identity for the type of the message. It is always encoded with the key *vCMD*. The second parameter encodes the information about the player's character. This information is needed by the other device to calculate the duel. The last parameter is a random number which will be used as a random seed in the calculations of the battle. Note that no result of a battle is transmitted. Instead, both devices calculate the battle and therefore a number needs to be transmitted so that both use the same random seed. The message is received by an instance of the *ReceiveTask* class which continuously waits for TCP connections. The *ReceiveTask* is an extension of the *AsyncTask*. The received message is given as parameter to the inherited method *publishProgress*, so that it can be handled by the GUI thread in the inherited *onProgressUpdate(String... values)* method. This method then passes the message further to the *Protocol* class, which extracts the key *vCMD* to understand what type of message it is. Then the message is passed to the corresponding method, in this case *handleAttackMessage()*, which then calculates the battle and sends an answer message back to the originator of the message. The originator then extracts the information of the other player's character and calculates the battle as well.

Figure 5.1: The code of the Protocol class to generate an attack message.

```

public static String getAttackPlayerMessage(){
  PlayerCharacter p = Game.getCurrentCharacter();
  if(p == null) {
    logger.logError("getAttackPlayerMessage() own Character is null
        .");
    return null;
  }
  JSONObject attacker = p.getCharacterViewAsJSON();

  JSONObject json = new JSONObject();
  try {
    json.put(vCMD, ATTACK_MESSAGE);
    json.put(vCHAR, attacker);
    json.put(vRANDOMSEED, (long)(Math.random() * Long.MAX_VALUE));
  } catch (JSONException e) {
    logger.logError(e);
  }
  return json.toString();
}
```

5.4 WLAN-OPP Integration

During the development of the game also WLAN-OPP improved and evolved continuously. At the beginning certain devices were not able to connect to others or the connection was just unidirectional. Step for step, with newer versions, those problems disappeared. Also the integration of the WLAN-OPP service in the own application changed a lot.

For a long time the code for registering a content observer and to start the service had to reside in the game application. Now an application just needs to include the WLAN-OPP library that handles this part and offers a clear interface and dedicated methods.

The figures 5.2 and 5.3 show two code pieces for the service without the library while figure 5.4 shows the much simpler code that uses the library.

For the application to actually start and stop the service just the code in figure 5.5 is needed in a super activity from which all activities inherit. The reason for the delayedStop() is that the service should not be stopped during a transition from one Android activity to another.

5.5 Quests Integration

The quests are the most important part of the game. They need to be available in a large number and have to be diversified. Hard coding the quests is not an option. A structure was needed that allows creating quests in an easy way but also grants enough freedom. Players should be able to make decisions in quests, to fight enemies, to get items and to receive effects. This has been realized by mainly three classes: Quest, QuestPage and QuestPageOption.

The Quest class serves as basis and contains information like the type of quest (simple, group, special, main), its title, level, difficulty, required level, icon and so on. Also game characters, items and effects can be added to the

Figure 5.2: Code before the library

```java

/**
 *  Initializes the content observer that calls the method
      updateNeighbors() when the data changes.
 */
public static void startNeighborUpdates(Activity activity){
  if (neighborCursor == null || neighborCursor.isClosed()){
    final ContentResolver resolver = activity.getContentResolver();
    final String[] projection = { "ip", "device_id" };
    neighborCursor = resolver.query(
      Uri.parse("content://ch.ethz.csg.burundi.NeighborProvider/
          dictionary"),
      projection, null, null, null);
  }

  //register a ContentObserver to get informed about changes.
  neighborObserver = new ContentObserver(new Handler())
  {
    @Override
    public void onChange(boolean selfChange)
    {
      super.onChange(selfChange);
      updateNeighbors();
    }
  };

  activity.getContentResolver().registerContentObserver(
    Uri.parse("content://ch.ethz.csg.burundi.NeighborProvider/
        dictionary"),
        false, neighborObserver);
}
```

Figure 5.3: Code before the library

```java

/**
 *  Gets the new list of neighbors.
 */
public static void updateNeighbors(){
  ArrayList<Neighbor> new_neighbors_list = new ArrayList<Neighbor
      >();

  neighborCursor.requery();

  neighborCursor.moveToFirst();
  while (!neighborCursor.isAfterLast()){
    String ip = neighborCursor.getString(neighborCursor.
        getColumnIndex("ip"));
    String id = neighborCursor.getString(neighborCursor.
        getColumnIndex("device_id"));
    Neighbor temp = new Neighbor(ip,id);
    new_neighbors_list.add(temp);
    neighborCursor.moveToNext();
  }

  neighborCursor.deactivate();

  //handle the changes
  updateNeighborsList(new_neighbors_list);
}
```

Figure 5.4: Code that uses the library.

```java

controller = WlanOppController.getInstance(activity, new Handler())
    ;
try{
  observer = NeighborObserver.getInstanceWithTask(new Handler(),
      activity,
  new NeighborObserverTask() {

    @Override
    public void run(List<Neighbor> neighbors) {
      updateNeighbors(neighbors);
    }

  });
}
catch(Exception e){
  return;
}
```

Figure 5.5: Starting and stopping the service.

```java

public static void onResume(){
  if(controller != null)  {
    controller.cancelDelayedStop();
    controller.startWlanOpp();
  }
  if(observer != null) observer.register();
}

public static void onPause(){
  if(controller != null)controller.stopDelayed();
  if(observer != null)observer.unregister();
}
```

quest in this class. The Quest class further contains a set of QuestPages and knows which one of them is currently to display. The QuestPages correspond to single pages in the quest and store parts of the quest's story. An instance of a QuestPage can also be configured to display an adversary or reward that has previously been added to an instance of the Quest class. In addition the QuestPages contain the QuestPageOptions which define the possible options that the player can choose when doing the quest. The QuestPageOptions store what happens if the player chooses that option. This can either be a fight, the distribution of items, the receiving of effects or the end of the quest.

Of course this implementation of the quests imposes certain restrictions. For example so far it is not possible to create QuestPageOptions that are only available if the player's character's intelligence is high enough or that have a different outcome depending on an attribute of the character at the time the quest is done. But the classes can easily be extended to support these features if needed.

The actual generation of the quests is done by the class QuestGenerator. Whenever a quest is solved, this class replaces that quest with a fresh one. When generating a new quest, the player's level is taken into account and the new quest is initialized with adequate enemies and rewards. To add some variance in the quest difficulty, some random variance is added to the used level in the quest initialisation.

5.6 Battle Calculations

This section gives additional details on the calculations that occur during a battle.

5.7 Evading and Accuracy

A game character's chance to evade an attack depends on the defence value which is tested against the hit rating of the attacker. The defence value is the combination of the attribute agility and evading, while the hit rating is calculated by adding the agility and accuracy of the attacker. The code in figure 5.6 shows the method for obtaining the chance to evade.

Figure 5.6: Evading system

```java
/**
 * Calculates the chance for the defender to evade the attack of
    the attacker.
 * Chance go from 0 to 30%
 * @param attacker
 * @param defender
 * @return       The chance to evade for the defender times 100.
    So 1400 means 14%.
 */
public static int getChanceToEvade(GameCharacter attacker,
    GameCharacter defender){

  int hitRating = attacker.getAgility().value()+attacker.
    getAccuracy().value();
  int defenceValue = defender.getAgility().value()+defender.
    getEvading().value();

  if(hitRating == 0) hitRating = 1;

  int total = defenceValue + hitRating;
  int evadeSuccessBorder = defenceValue*3000/total;

  return evadeSuccessBorder;
}
```

The goal of the system is to get a percentage ranging from zero to 30 which is achieved by the multiplication by 3000 instead of 10000 in the last step. Note that the value 10000 represents 100%. The 30% border is to avoid game characters evading every attack which could be frustrating for the unlucky attacker. The code below shows the previously used system.

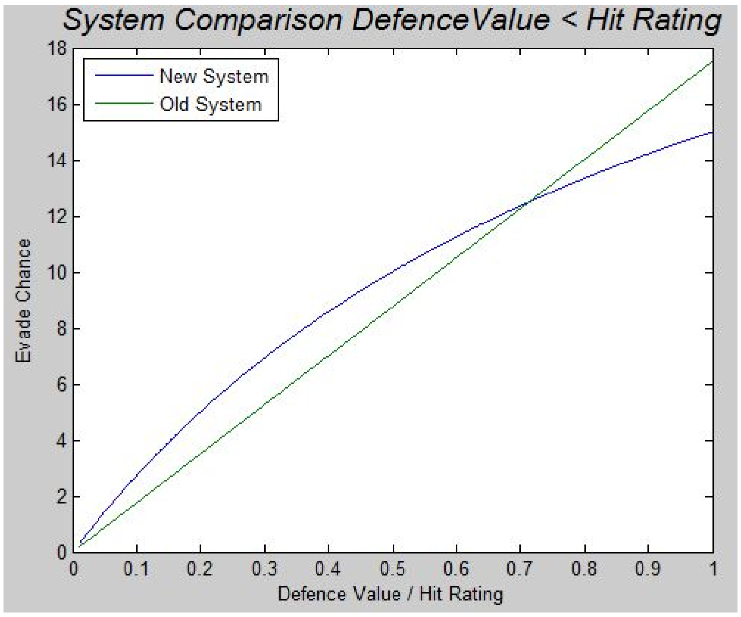
```java
  if(hitRating >= defenceValue){
    evadeSuccessBorder = defenceValue * 1750 / hitRating;
  }else{
    evadeSuccessBorder = 3250 - hitRating * 1500 / defenceValue;
  }
```

Note, that in both system the actual evade chance only depends on the relation of the defence value to the hit rating. In figure 5.7 the graphs for the two systems are plotted. Figure 5.7a shows the graphs with the defence value being smaller then the hit rating and figure 5.7b shows the opposite.

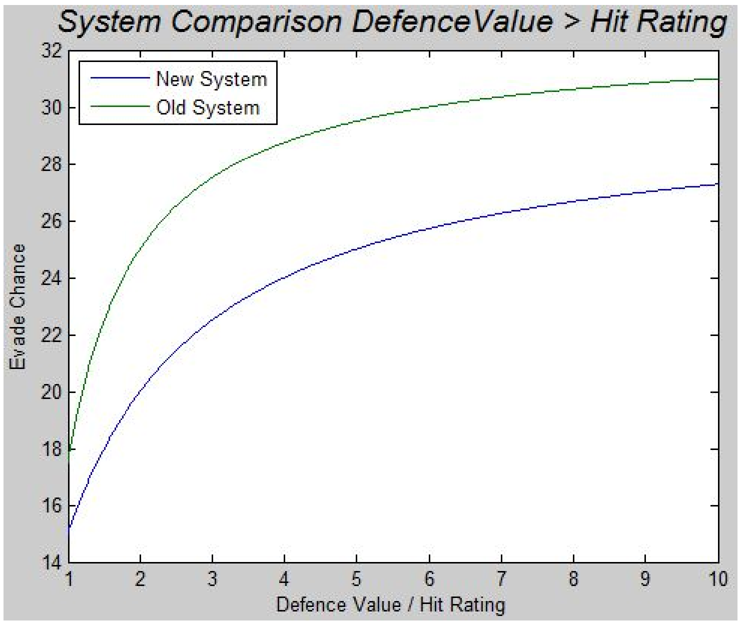
Since it is up to the player to spend points on attributes, the system has to deal with defence values and hit ratings that might differ substantially in dimension. A thief on level one hundred is probably going to have ten times more agility than a mage. Using a linear system to calculate the chance to evade, for example by using the calculation *evadeChance=Math.max(0,Math.min(30, defenceValue-hitRating))* is therefore not a good idea, because that way it doesn't matter anymore if the mage has ten or twenty agility. The mage will always have the worst possible chance if he is not close to the thief's agility.

Comparing the two previously described systems in figure 5.7, the new system was preferred because increasing the defence value or hit rating has more effect on the actual evading chance, when the value that it is compared to is of much bigger size.

Figure 5.7: Evade Chance Graphs



(a)



(b)

5.8 Critical Strikes

The chance for a critical strike is calculated by the simple formula **criticalHitRating - defenderLevel * 7 / 10**, where the criticalHitRating is equal to the attribute strength for a warrior or thief and intelligence for a mage. Because of this calculation, the chance for a critical strike grows continuously and is not bounded. On higher levels, the chance can exceed 100%. Of course this is not a good solution for long, but to reach the level where it gets problematic takes a very long time. For example with level 30 the chance is around 60% and reaching that level can take up to four months. Nevertheless, this system should be replaced and maybe a counter attribute should be added.

Chapter 6

Evaluation

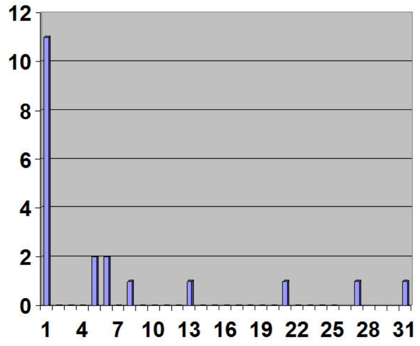
The game has grown quite big and has reached a certain complexity. There are three character classes with a total of six skills and seven attributes, not counting armor and weapon damage. There are sixty different items and thirteen quests with nineteen quest adversaries. In the quests, a character can get poisoned by three different poisons or receive one of three other effects, like a burn wound. Items have various values like quality and durability. Players can interact in various ways with each other. They can have a duel, do group quests together and exchange items and gold.

The following sections inform about the beta test. They show how the data was acquired and what results were derived from that data and the feedback of the beta testers.

6.1 Beta-Test

From October 2012 until the end of January 2013 a small beta test was conducted. In total, 17 people downloaded and played the game. While one player's three characters reached level 31, 27 and 21 and the player is still playing, others stopped playing earlier. Another player reached level 13 with the first character and level eight with a second character. Four players reached level five or six. The remaining did not play long enough to increase in level.

Figure 6.1: Character level distribution.



To get interesting information about how and how long the people played the game, the application stores usage data, generates crash reports and sends them to a server.

6.1.1 Usage Reports

The usage reports are generated on a (at most) daily basis. The data is sent anonymously and contains only data about the player's ingame actions, the game characters, game version and the device that it is played on. The IMEI number is only transmitted as a hash value. The figure 6.2 shows a real usage report that has been received by the server. Note, that for better readability, the information about the game character has been partially omitted.

Figure 6.2: usage-report-11-02-13

```


USAGE REPORT
GAME VERSION: 0.15
MANUFACTURER: HTC
PRODUCT: htc_ace
MODEL: HTC Desire HD A9191
DEVICE: ace
API VERSION: 10
Level: 33
CharacterClass: Mage


AccountID=1351487258447
AccountIMEIHash=e24be5cd
Email=null
gameVersion=0.15
potionsMixed=57
enchantments=29
wardsCreated=25
simpleQuestsDone=851
specialQuestsDone=9
mainQuestsDone=3
daysPlayed=81
merchantVisited=210
itemsBought=156
PlayerCharacter={"regeneration":10,"merchant":{"inventory":{"
     array_equipment":...
PlayerCharacterName=abc
```

6.1.2 Crash Reports

Not only usage data is sent to the server. Likewise, whenever the game crashes a crash report is generated and sent to the server. Consider figure 6.3. The crash report consists of four parts.

The first part shows information about the player's device, the level and character class and the game version. The second part contains the thrown exception that caused the application to stop. It follows the PlayerActionLog which gives insights on what the player was doing before the application crashed. It is cleared every time the game is started and it logs every click of the player.

The last part of the crash report is an internal error log. That error log contains all the "half expected" errors, like for example when a method is called with a null reference as an argument which should actually never be null. One could actually talk of violated pre- and postconditions.

In the given example of a crash report the error was quite easy to figure out. As visible from the PlayerActionLog, the player pressed a page option in a quest quickly two times in series so that the action was handled twice. This led to a null pointer exception when it was pressed the second time.

6.1.3 Results

The beta test gave some useful insights. It showed that the game can have an addictive influence on people and therefore has potential. It clearly demonstrated, that additional game content is needed to encourage the users to keep on playing after having reached level five since at level five all simple quests were accessible for the beta testers. In the newer version, an additional quest for level ten has already been added. Furthermore the usage reports gave clues on how long it takes to reach a certain level. For example the usage report in 6.2 showed that it took the player 81 days to reach level 33.

To summarize, the most heard critic was that there are not so many quests. Especially at the start of the game when the most used category "simple quests" only offers two distinct quests.

Another conclusion, which was drawn from the usage reports, was that the special quests were done quite rarely. In the example usage report (Figure 6.2) the player did only nine special quests but about 851 of the simple ones. Of course, a higher number of simple quests was expected, but not in this proportion. To come up with a compensation for this, the special quests were made more attractive. They now provide better rewards and are easier to solve.

The beta test revealed various other balancing issues. Since its end a lot of balancing was done. The drop rates of items in the quests have been adapted as well as the partially random generation of the items that are sold by the merchant. The merchant did not offer enough equipment for sale, so that classes like the thief or mage, which can not create all the needed equipment by themselves, could not buy reasonable equipment. Also the prices of items have been adapted. Certain items, like for example potions and antidotes, were too expensive.

Another issue was that the repair costs for equipment was too high. Especially items of higher quality were not economical.

The beta test also showed that the scaling of the player's strength on higher levels (e.g. level 30) compared to the encountered enemies in the quest was too strong. One problem was that the health/damage balance slid from health to damage, so that the fights became shorter which led to a decreased difficulty in fights where the player attacked first and an increased difficulty in the other case. Furthermore, some of the poisons, that the character could suffer from after a fight, were just too strong.

In addition, the collected data also showed that the players did hardly ever make use of the multiplayer part. The reasons for this probably are that in the used game version was no group quest integrated and that the game is not widely spread.

Figure 6.3: crash-report-19-12-12-03-53-15.txt

```

CRASH REPORT
GAME VERSION: 0.15
MANUFACTURER: HTC
PRODUCT: htc_bravo
MODEL: HTC Desire
DEVICE: bravo
API VERSION: 8
Level: 5
CharacterClass: Thief

java.lang.NullPointerException
java.lang.NullPointerException
   at rpg.questbasis.Quest.doFight(Quest.java:393)
   at rpg.questbasis.Quest.handleOptionClicked(Quest.java:584)
   at rpg.activities.QuestActivity.handleOptionClicked(QuestActivity
        .java:371)
   at rpg.activities.QuestActivity.access$1(QuestActivity.java:364)
   at rpg.activities.QuestActivity$2.onClick(QuestActivity.java:296)
   at android.view.View.performClick(View.java:2408)
   at android.view.View$PerformClick.run(View.java:8817)
   at android.os.Handler.handleCallback(Handler.java:587)
   at android.os.Handler.dispatchMessage(Handler.java:92)
   at android.os.Looper.loop(Looper.java:144)
   at android.app.ActivityThread.main(ActivityThread.java:4937)
   at java.lang.reflect.Method.invokeNative(Native Method)
   at java.lang.reflect.Method.invoke(Method.java:521)
   at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(
        ZygoteInit.java:858)
   at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:616)
   at dalvik.system.NativeStart.main(Native Method)

PlayerActionLog:
rpg.activities.MainActivity: onClickStartPlayActivity
rpg.activities.PlayActivity: onClickStartOtherPlayersActivity
rpg.activities.PlayActivity: onClickStartCharacterActivity
rpg.activities.CharacterActivity: onClickStartAttributesActivity
rpg.activities.PlayActivity: onClickStartInventoryActivity
rpg.activities.PlayActivity: onClickShowFirstSkill
rpg.activities.PlayActivity: onClickStartQuestActivity
rpg.activities.MultiQuestActivity: onItemClick Simple Quests
rpg.activities.QuestListActivity: onItemClick Hunting
rpg.activities.QuestActivity: handleOptionClickedSearch actively
     for the beast in the woods.
rpg.activities.QuestActivity: handleOptionClickedSearch actively
     for the beast in the woods.

ERROR_LOG:
13_Dec_19_53_20 rpg.questbasis.Quest: : LinkedList enemies in Quest
      contains an enemy that is null.
13_Dec_19_53_21 rpg.questbasis.Quest: : LinkedList enemies in Quest
      contains an enemy that is null.
```

When the application is on the Android market, it is important that new versions will not destroy the players' existing game characters. In this sense the beta test gave some experience in updating the game without losing old characters.

Chapter 7

Future Work

There is lot that can be done to improve the user's playing experience. The number of possibilities to add new game content is nearly limitless. Obviously one can add additional quests, enemies, new items and the recipes to craft them as well as completely new character classes. Doing so, one should still have in mind that a new player should not be bombarded straight away from the beginning with too many things.

7.1 Concrete Ideas

In this section, some more complex ideas, which might be worth integrating in the game, are listed.

7.1.1 Talent trees

Well known from many games, talent trees usually force the player to make decisions on how to specialize his or her game character. Each character class could have its own talent tree which would also increase the differences between the character classes and therefore make it more appealing to try playing several characters. As a concrete example a warrior could have a talent which converts the next attack to a critical strike whenever the target evaded his or her attack. And as alternative there could be a talent that decreases the damage taken from magical attacks. Like this, the first talent would give the player better chances in a fight against agile adversaries like thieves, while the second one would obviously help against adversaries that make use of magic attacks like mages.

7.1.2 Lands

There could be a map in the game with several lands or kingdoms. The lands all provide different quests and offer different shopping possibilities. The player could then explore new lands and move from one land to another. The new lands could become available with the progress in the main quest.

7.1.3 Location Based Gaming

The game could make use of the location of the user. One could design quests that force the user to visit a certain position in the real world to fulfill them. A highly interesting idea would be to take companies into account. For example, there could be daily extras like a free in game muffin that regenerates the player's health and which would be available when he or she enters a Starbucks.

Chapter 8

Conclusion

The goal of this thesis was to create a game that makes use of opportunistic networks with the WLAN-OPP service and which could be deployed on the Android market. The game is in a reasonable state that would allow such a deployment. The graphical appearance of the game is better than originally intended and the game has become quite complex: Players can create and level up a character, solve quests and interact with other players. It can be played alone or with other people for quite a while. In addition, a server infrastructure is used to receive usage data and crash reports.

Even though the game contains a lot, it would still be a good idea to spend some more time on improving it and on adding even more content. That way bad ratings on the Android market can possibly be avoided, since the main critic from the beta testers was that they wanted more quests. Nevertheless, the beta test also showed that the game has potential and that it will help to gather usage data for the WLAN-OPP service. This is even done if the multiplayer components are not actively used by the player since the service is started together with the game as long as WLAN-OPP is installed.

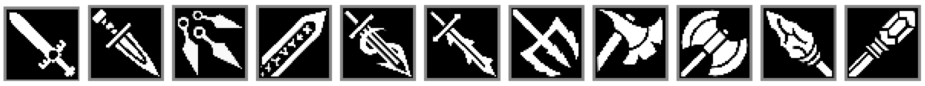
Appendix A

Items Overview

This document gives a short overview on the different items that can be found or created in the game. The number between the brackets indicates the amount of different items of this category. Note that the number of potions is higher than the number of visible pictures. This is due to the fact that there are several potions using the last picture. (Actually one for each attribute.)

A.1 Equipment (16)

Weapons (11)



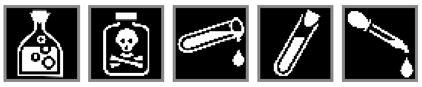
Armors (6)

A.2 Consumable (17)

Consumables like potions can be used to get temporary or even permanent positive effects on attributes. Poisons can be used on weapons to improve their damage or to weaken the enemy. Antidotes can cure from illnesses which reduce the characters efficiency. Beer is a fun item which is not at all useful

Potions (11)

Weapon Poisons and Antidotes (5)



Fun Items (1)

A.3 Ingredients (24)

Ingredients can be used and processed with the player's characters skills. In general, the end product is a new item of the type of the category equipment or consumable. However, it can also be an improvement of an existing item. In the following, all ingredients are listed and grouped by the skill for which they are useful.

Mining (1)

Blacksmith (4)

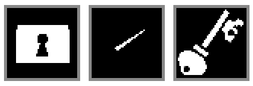
Enchanting (4)

Alchemy (12)

Toxicology (4)

A.4 Other Items (3)

In this category are all items that do not really belong to a category. Actually, one could argue that picklocks or even chests belong to the ingredient category but since a chest is opened instead of really transformed into a new item and because a picklock is more like a tool, they are categorized as "Other Item".



Bibliography

[1] Lorc's collection of rpg icons. *http://opengameart.org/content/700-rpg-icons.*

[2] Statistics on the android market. *http://android-developers.blogspot.ch/2011/12/closer-look-at-10-billion-downloads.html.*

[3] Dominik Schatzmann Franck Legendre Sacha Trifunovic, Bernhard Distl. Wifi-opp: Ad-hoc-less opportunistic networking. *http://www.csg.ethz.ch/people/sachat/papers/Trifunovic2011WiFiOpp.pdf*, 2011.

[4] Aardwolf. *http://www.aardwolf.com/.*

[5] Wizard's choice volume 1. *https://play.google.com/store/apps/details?id=wizardsChoiceV1.toucher&hl=de.*

[6] Shakes and fidget. *http://www.sfgame.de/.*

[7] Gimp, graphics editing program. *http://www.gimp.org/.*

[8] Icon license. *http://creativecommons.org/licenses/by/3.0/legalcode.*

[9] Gson library. *http://en.wikipedia.org/wiki/GSON.*