Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Battle of the Smartphones

Bachelor Thesis

Romina Som

somr@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Gino Brunner, Pascal Bissig, Simon Tanner
Prof. Dr. Roger Wattenhofer

July 17, 2017

Abstract

A smartphone's performance is about more than just processing power. Other important factors include camera quality, WiFi performance and the accuracy of its sensors.

This bachelor thesis describes an app which combines all of these factors into a game that can be played between two friends. The game is a battle between two smartphones and consists of six rounds which compare the two smartphones. At the end of the battle, the users know which phone is better.

The six rounds test the WiFi antenna, the CPU, the gyroscope, the battery capacity and the camera. There are two different camera rounds, one round tests how nice the pictures of the camera are and the other round tests how fast the camera can take a picture.

The app does not require any external equipment, and does not even need a controlled environment. The only requirement it has is an Internet connection and if possible a WiFi network nearby, but the users do not need to be logged into that network.

Contents

Chapter 1

Introduction

1.1 Motivation

There are numerous benchmark apps in the Google Play Store, but most of them only test one thing. This can be the CPU, the GPU, the GPS, the RAM, OpenGL, or other things, but there is almost no benchmark that tests more than two or three different aspects of the phone. There is especially no interactive benchmark at all, meaning in every benchmark app you press a “Start” button then wait some time and then you have your score. One goal of this thesis is to create an app which is more interactive, and where you do not simply benchmark your phone. The app should be a game which you can play against your friend to find out which device is better.

Some companies do sophisticated analysis of camera performances (e.g. DxO-Mark [1]), but this requires expensive specialized equipment. WiFi performance can be tested, but it involves manual setup and requires careful controlling of the test environment. The second goal of this thesis therefore was to develop a method to measure the performance of these components without requiring specialized equipment.

The last goal of this thesis was to create a real-world benchmark, meaning that the scores represent the values you get in everyday use.

1.2 Related Work

As already mentioned above, there are numerous benchmark apps in the Play Store. One of the most known is AnTuTu Benchmark [2]. AnTuTu tests the CPU and GPU performance, the RAM speed and the user experience. Geekbench [3] has specialized in CPU performance. The most known app for testing the Internet speed is Speedtest.net from Ookla [4]. It measures the throughput for downloading and uploading data to a server. For measuring real-world performance there is also the app DiscoMark [5] which measures the start times of different apps.

Chapter 2

Implementation

The following chapter will show how the app is structured and explain the different parts of the app. It will also show how the server works, and how the database on the server is used.

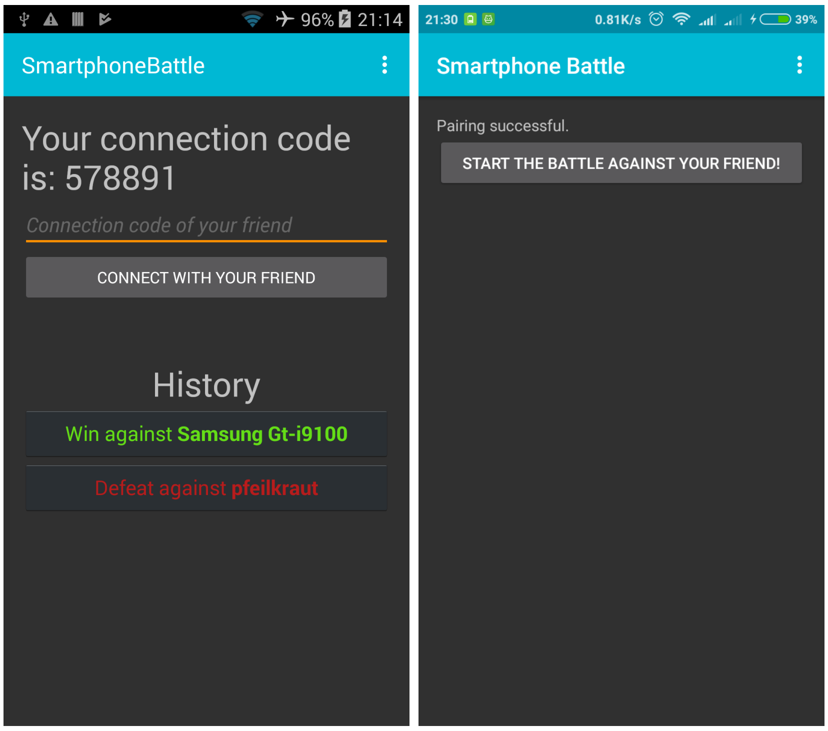
The Minimum SDK Version to use our app is 16, which corresponds to Android 4.1 Jelly Bean [7]. This means that about 98% of users [8] can use the app. To make the app available to an even wider audience, a German translation is available.

While we wanted to be able to identify the users, the decision was made against a login with the Google Account, so the user can stay more anonymous. Instead, it is possible for the user to set a nickname. This nickname does not need to be unique, and the user can change it anytime in the preferences. When the app is opened for the very first time, it asks for a nickname, but it is possible to leave that field empty. The nickname will then be the build name of the phone, for example “Samsung Gt-i9100”.

To tell the user how to use the app, a tutorial is displayed while playing the first battle. After the first battle is completed, the tutorial is turned off, but can be turned on again by the user in the settings.

2.1 Main Pairing Screen

As you can see in Figure 2.1a, a connection code is displayed to the user. This are the last 6 letters of the internal identity which gets set the first time the user opens the app. This identity is sent to the server to identify the device, but is never visible to the user. When the user clicks the button, his identity and the connection code of the opponent is sent to the server, and a battle ID is received from the server if the pairing worked. For the pairing to work, both users need to click the “PAIR” button within ten seconds. If the user does not input a valid connection code (e.g. only three letters) or there is no one registered with that connection code, a button is displayed which tells them to try again.


96% 21:14
SmartphoneBattle
Your connection code is: 578891
Connection code of your friend
CONNECT WITH YOUR FRIEND
History
Win against Samsung Gt-i9100
Defeat against pfeilkraut
21:30
0.81K/s
39%
Smartphone Battle
Pairing successful.
START THE BATTLE AGAINST YOUR FRIEND!

(a) Screen when you open the app (b) Successfully paired

Figure 2.1: Screenshots - Main Pairing Screen

Below that, the user can see a history of the last ten completed battles. When the user clicks on an old battle, the Final Overview screen of that battle is displayed.

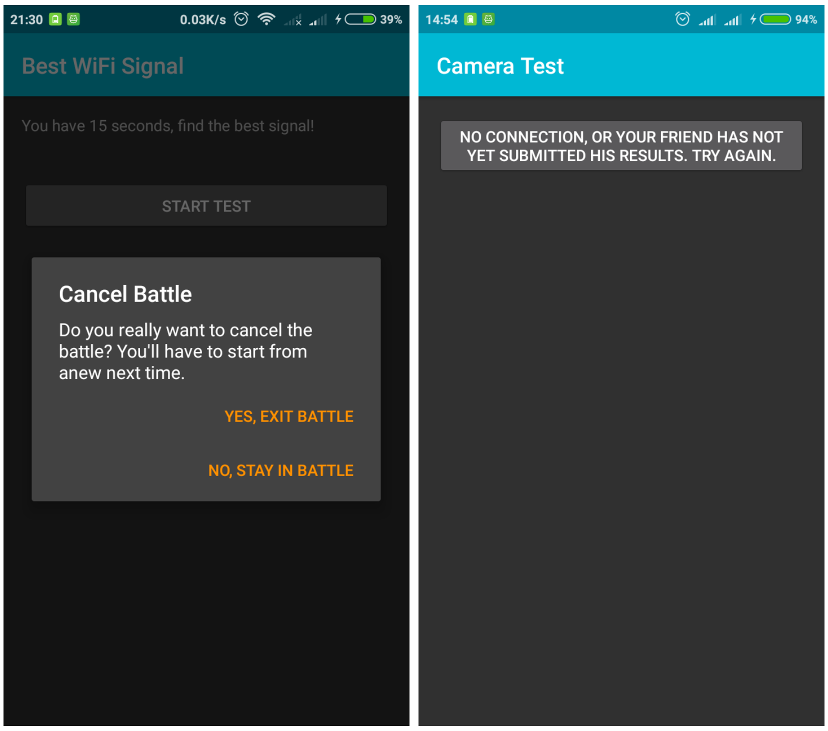
If the pairing was successful, a button is displayed to start the battle (Figure 2.1b).

2.2 Battle Rounds

The user can play a battle against a friend. Each battle consists of six different rounds. In the current version of the app the user cannot choose which rounds to do, but has to do all six.

When the user is in a battle and presses the back button, an alert dialog pops up and asks the user if he would really like to exit this battle (Figure 2.2a), because once the user left it is impossible to resume the battle.

To save the result of a battle, the class Battle is used. The results of all rounds are stored there and then written to the internal memory of the phone to save the result for later use, e.g., for the history.



You have 15 seconds, find the best signal!

Cancel Battle

Do you really want to cancel the battle? You'll have to start from anew next time.

NO CONNECTION, OR YOUR FRIEND HAS NOT YET SUBMITTED HIS RESULTS. TRY AGAIN.

(a) Back Button Overwrite (b) No Connection

Figure 2.2: Screenshots

If the two opponents do not send results within some time, a button is displayed for the user to try again (Figure 2.2b).

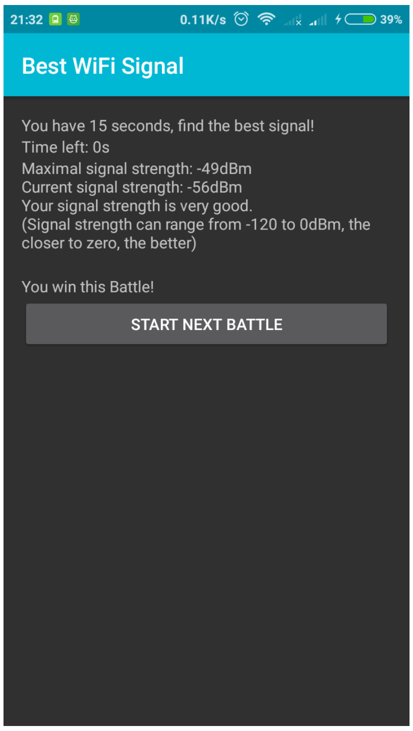
It is always possible that both users win or both users lose a round, for example when their scores are equal, or when both run into an error.

In the following subsections, each round is explained in the order the user does them. The order is chosen such that similar rounds (e.g., the two Camera rounds) are not directly after each other, and that the non-interactive rounds are between interactive ones.

2.2.1 Best WiFi Signal

In this test, the users have 15 seconds and need to find the WiFi signal with the highest signal strength. The countdown, as well as the current and maximal signal strength found, are displayed to the user (Figure 2.3). To read the signal strength of surrounding WiFis the users need to have WiFi enabled, and on Android 6+ also activated the location services. If they have disabled either of them, they are prompted to enable them.

This test also checks if the phone supports 5GHz WiFi. If this is the case,



Best WiFi Signal

You have 15 seconds, find the best signal!
Time left: 0s
Maximal signal strength: -49dBm
Current signal strength: -56dBm
Your signal strength is very good.
(Signal strength can range from -120 to 0dBm, the closer to zero, the better)

You win this Battle!

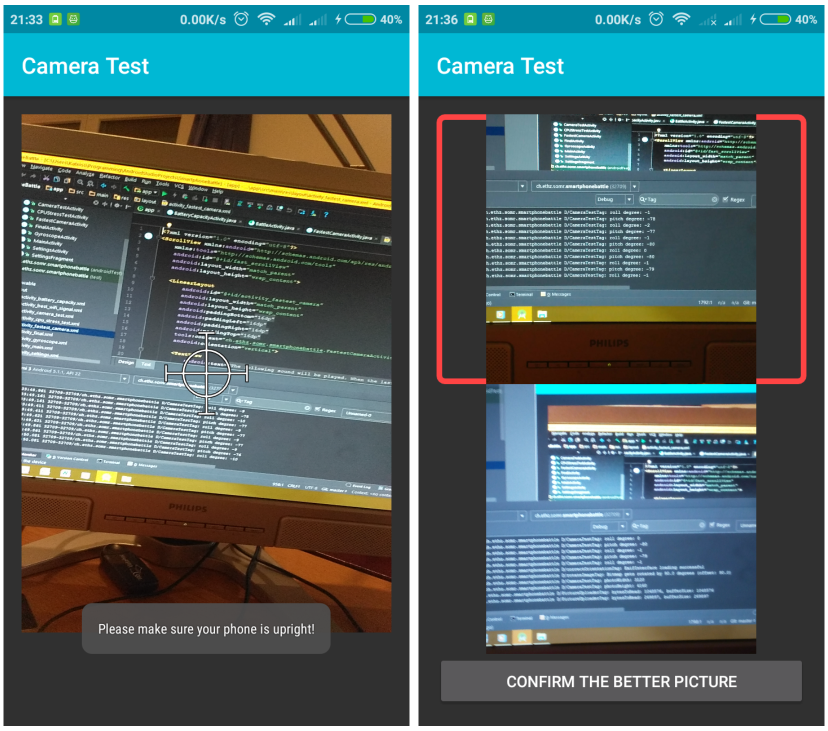
Figure 2.3: Screenshot - Best WiFi Signal Test

the user gets credited with 5dBm, i.e.,1 if the phone supports 5GHz WiFi and finds a signal with a strength of -50dBm, the score of this user for this round will be -45.

The user with the higher score wins this round.

2.2.2 Camera Test

In this test both players have to agree on a target, and then take a picture of this target. There is a cross-hair in the middle of the screen, to simplify this process. The camera takes the highest resolution picture the phone supports, and then sends this picture to the server. The server saves the picture at a randomly generated address and writes this address into the database. The address also gets sent to the opponent's phone. Both pictures are then displayed side-by-side and you can zoom and scroll them. You then have to tap on the better picture and your choice gets sent to the server (Figure 2.4b). The server then decides who wins (for more detailed information see Section 2.4.2) and sends the result back to both devices. Because the pictures are sent with not much compression, some non-negligible data usage can arise, and the user is warned of that with an alert dialog each time he starts a camera test round.


21:33
0.00K/s
40%
Camera Test
Please make sure your phone is upright!
21:36
0.00K/s
40%
Camera Test
CONFIRM THE BETTER PICTURE

(a) Check that phone is vertical (b) Screen to select better picture

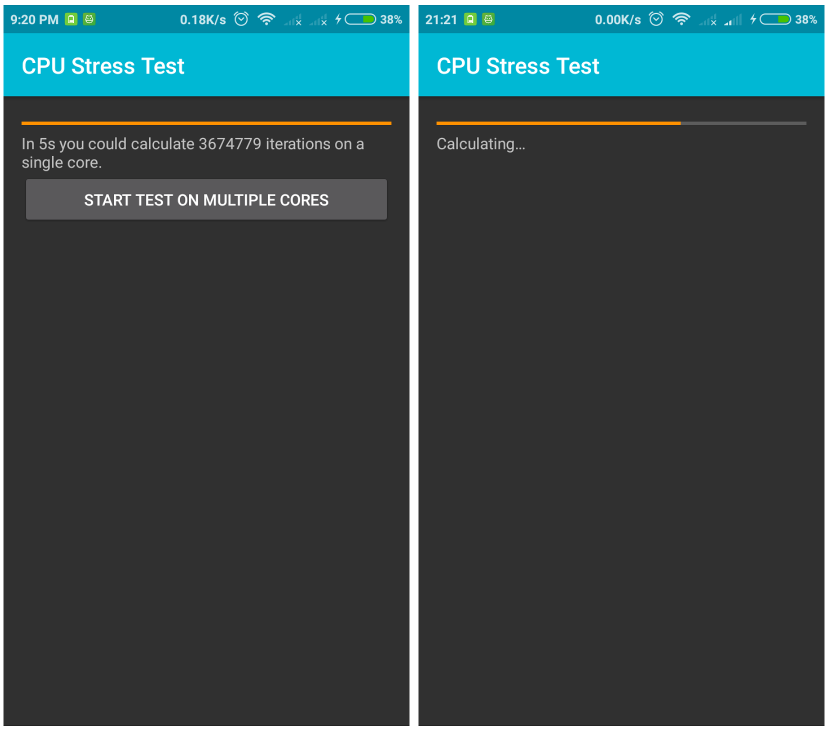
Figure 2.4: Screenshots - Camera Test

The camera also checks if the phone is vertical, so the photos both users take are even more similar. If the phone is not vertical, a message is displayed which only disappears once the phone is upright (Figure 2.4a). The button to take a picture is only displayed when the phone is vertical.

2.2.3 CPU Stress Test

The CPU Stress Test is the usual CPU test: in this test the user just has to click a start button, then the CPU calculates iterations of a "useless Algorithm" during five seconds, and outputs how many iterations it was able to calculate in this five seconds. The useless Algorithm calculates a number iteratively, similarly to the Gauss-Legendre Pi calculation algorithm. The problem with this algorithm was that the algorithm got too precise too fast and the variable containing pi did not change anymore. Java noticed that and optimized away the rest of the calculations. Because of that the algorithm was altered to not calculate pi anymore, but some other, "useless" number.

This whole test is done twice, once on a single core and once on multiple scores, i.e., in as many threads as the CPU has cores (Figure 2.5a). To see how



In 5s you could calculate 3674779 iterations on a single core.

Calculating...

(a) After the single core test (b) Progress bar while calculating

Figure 2.5: Screenshots - CPU Test

much time has already passed, a progress bar is shown (Figure 2.5b).

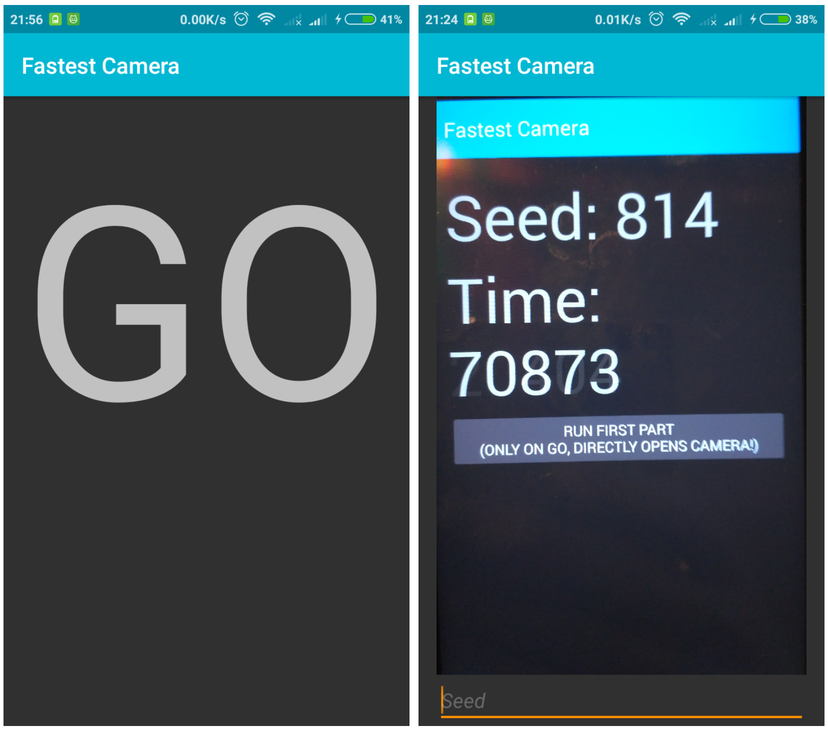
To calculate who wins this round, the score from the single core and from the multi-core test are added to each other.

2.2.4 Fastest Camera

This round works a bit like an old, western-style duel. One user starts a countdown, and on “GO” (Figure 2.6a) the opponent has to take his phone, click a button to open the camera and take a picture of the screen of the user as fast as possible. Then the roles are reversed and the other person has to take a photo as fast as possible. The user who is faster wins the duel and therefore this round.

As the countdown is not only visual but has also sound, an audio-test is performed at the start of the battle. This is a mock countdown, to show the user how the countdown works and also to test that their music volume is high enough. After you tested the sound, you can either start the real test, or retest the sound if it did not work the first time.

To determine how long the opponent had to take the picture, two numbers are shown on the screen of the user. The two numbers are a “seed” and a “time”.



(a) Countdown (b) Blur effect of time visible

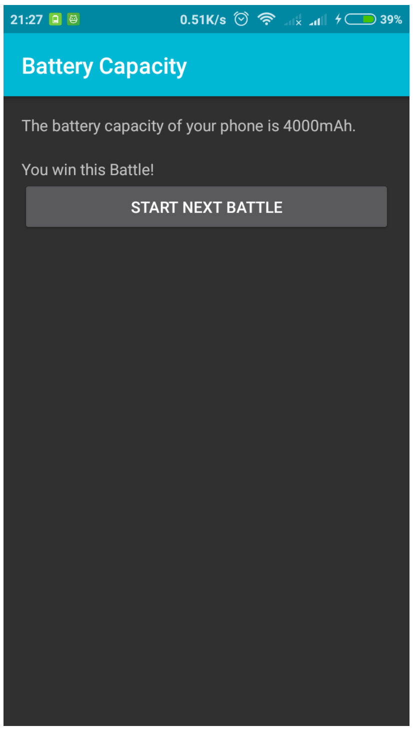
Figure 2.6: Screenshots - Fastest Camera Test

The seed always stays the same, but the time gets updated every 250ms. To know how much time has passed, one can just take the remainder of the division of the time by the seed. This remainder is the number of 250ms intervals that have passed since starting the countdown.

The refresh rate is only 250ms and not faster because otherwise you often have two numbers simultaneously in the photo because the exposure time of the camera is too long. You can already see this effect in Figure 2.6b. After you took the picture, you have to write down the two numbers and then your time is displayed.

2.2.5 Battery Capacity

Like the CPU test, the Battery Capacity round is a non-interactive one, where you just have to click the “START” button, and then your battery capacity is read programmatically and displayed (Figure 2.7). The user with the higher capacity wins this round.


21:27 0.51K/s 39%

Battery Capacity

The battery capacity of your phone is 4000mAh.

You win this Battle!

Figure 2.7: Screenshot - Battery Capacity Test

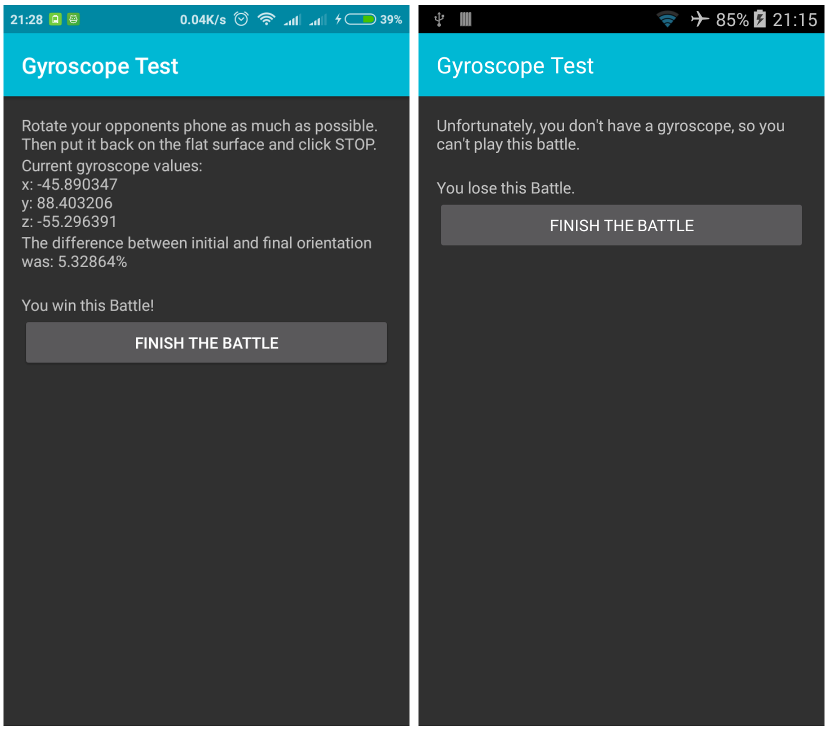
2.2.6 Gyroscope Test

At the beginning of this round, both phones should be placed on an even surface. After clicking the START button each player takes the phone of his opponent, rotates the device as much as he pleases, puts the phone back into the initial position and clicks STOP. It is then calculated how much the initial orientation differs from the current orientation (Figure 2.8a). The user with the smaller difference wins this round.

If a phone does not have a gyro sensor, his user automatically loses the round (as it is always better to have a sensor than not to have one, Figure 2.8b). If both phones do not have a gyroscope, both lose this round.

2.3 Final Overview Screen

In the final overview, you see who won each round (Figure 2.9). The results are read from and written to the internal memory of the phone, so you can look at them again later (in the history).


21:28 0.04K/s 39%

Gyroscope Test

Rotate your opponents phone as much as possible. Then put it back on the flat surface and click STOP.
Current gyroscope values:
x: -45.890347
y: 88.403206
z: -55.296391
The difference between initial and final orientation was: 5.32864%

You win this Battle!

85% 21:15

Gyroscope Test

Unfortunately, you don't have a gyroscope, so you can't play this battle.

You lose this Battle.

(a) Usual screen (b) No gyroscope available

Figure 2.8: Screenshots - Gyroscope Test

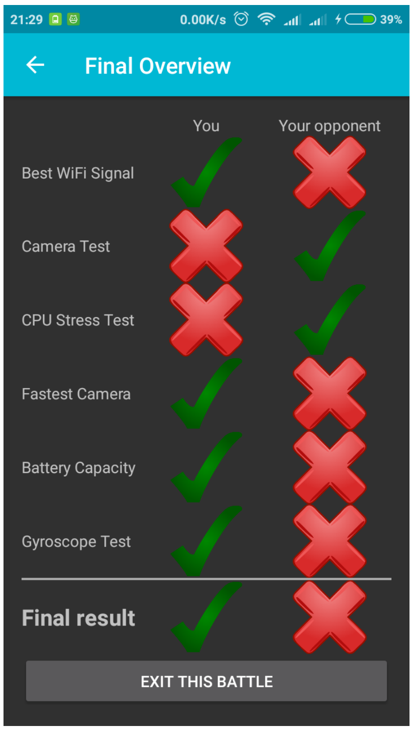
2.4 Server

The server consists of three different parts. There is a database, in which the results of the battles are saved, then there is a PHP script which mainly exists to communicate between the app and the database and finally there is also a privacy policy, which consists of an English and a German html file and a CSS file for style.

The privacy policy is required by Google because the app uses the camera permission. This permission is needed to take a photo in the camera test round. The photos are stored indefinitely on the server to make further evaluations possible. The app also needs the location permission to be able to scan the surrounding WiFi networks for the WiFi test. The Privacy Policy is linked in the Play Store and also in the preferences of the app.

2.4.1 Database

The MySQL database smartphone_battle consists of two tables: USERS and BATTLES.



	You	Your opponent
Best WiFi Signal	✓	✗
Camera Test	✗	✓
CPU Stress Test	✗	✓
Fastest Camera	✓	✗
Battery Capacity	✓	✗
Gyroscope Test	✓	✗
Final result	✓	✗

Figure 2.9: Screenshot - Final Overview

The USERS table saves everything concerning a single user (see Table A.1). This includes his ID, when he registered for the first time, when he was last seen, his nickname if he set one and information about his phone. During the user's first battle, information about his camera and gyroscope is gathered and also saved here.

The Android version of the device is not saved in the table USERS, but in the BATTLES table, as it is possible for the user to update the phone without losing his battle data, meaning that information could change between different battles.

In the BATTLES table everything about a battle is saved (see Table A.2). For each battle there are two entries in the database, one for each player. There is a BATTLE_ID which uniquely identifies the entry. The columns MY_ID, OTHER_ID and OTHER_BATTLE_ID define who plays the battle against whom. Then there is information about when the battle started, if it is finished and of course the results and information gathered of each round.

2.4.2 PHP Script

The PHP script communicates between the app and the database. It fills the database with the results sent by the app, reads the result of the opponent from the database, calculates who wins this round and sends this back to the app. The script is always accessed with GET requests, except for the photo uploading.

Every time the app is opened, a connection to the server is established by sending the own ID and information about the device, which the script writes into the USERS database if they do not yet exist there.

When the user sets the nickname, or changes it in the preferences, the nickname is sent to the script which then updates this user's database entry.

When the user wants to start a new battle, he sends his ID, the connection code and information about his operating system to the script. The script makes a new entry in the BATTLES database with this information. The connection code is used to search for the opponent's entry in the USERS database as it corresponds to the last 6 letters of the opponent's ID. The script then pairs the two opponents, writes information about each other into the two BATTLES entries and finally returns the own BATTLE_ID and the nickname of the opponent. The pairing only works if both users want to pair to each other and they press the pairing button within five seconds of each other.

After each round of a battle, the app sends his current BATTLE_ID and the results of this round to the server, which updates the BATTLES entry and sends back if this phone won the round and if the other phone won the round. It is necessary to send both because it is possible that both players win or lose a round, e.g., when both have the same score they both win, or when both had an error they both lose.

The only part where a POST request is used, is when the app sends the photo of the camera test to the server. The server generates a random name of 80 letters length where it saves the picture. It writes this name into the database, looks for the name of the picture of the opponent in the database and sends this name back to the app. The server uses a cryptographically secure random function for generating the name to prevent attackers from guessing the name of the next picture if they have one picture's name. The photo is only accessible by direct link and the folder it is placed in is not accessible.

The calculation of who won the Camera Test is a bit more complicated, which is why it is shortly explained here. Each score starts at 1. If a out of memory error occurred, 1 is subtracted. This can happen when loading the two pictures into memory to display them, as the pictures are displayed in the highest quality possible, but this is sometimes repairable and does not always lead to a failure of the round. If you said your own picture is better, 1 is added. If the opponent said your picture is better, 1 is added as well. The player with the higher score wins, but if a fatal error occurred with one phone, this one loses anyways.

If any error occurs while executing the script, the server responds with the corresponding HTTP status code. As there was no status code appropriate for an SQL error [9] and the use of "HTTP/1.1 500 Internal Server Error" seemed too generic, the code "HTTP/1.1 418 I'm a Teapot" was used for this specific error. Even though 4xx codes are typically used for client-side errors, we used the 418 code as it has the advantage that the SQL error will never get confused with something else because 418 is usually not used.

Chapter 3

Evaluation

Because all test were designed by ourselves, we need to make sure that they are fair and it is possible to reproduce results. We especially tested the "CPU Stress Test" to make sure it is balanced. We also tested the "Battery Capacity" test to verify that we read correct information from the phone.

3.1 CPU Stress Test

As the content of the iterations of the CPU test were defined more or less randomly, we needed to make sure the deviation over a few different executions was not too big. To test this, we run the CPU test five times on each of the six different test devices and measured the number of iterations the device was able to calculate (see Figures 3.1 and 3.2).

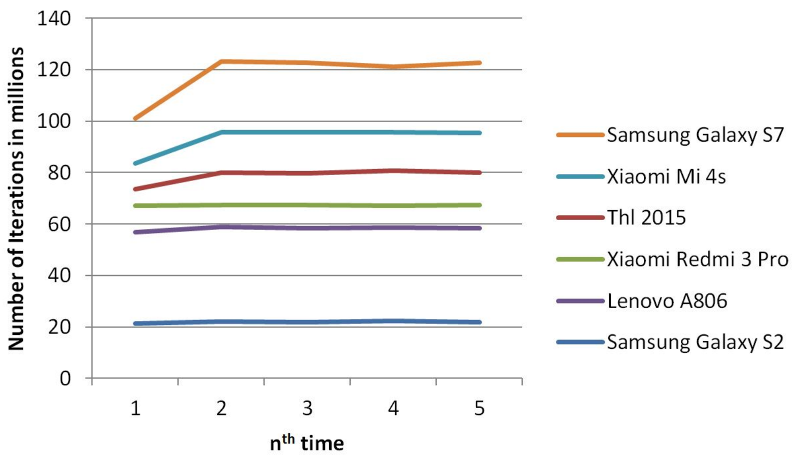


Figure 3.1: Number of iterations on a single core in five seconds, measured five times. On most of the six devices, a clear increase of the number of iterations between the first and the second time can be seen. This is probably because the CPU is already warm when doing it the second time and can optimize better.

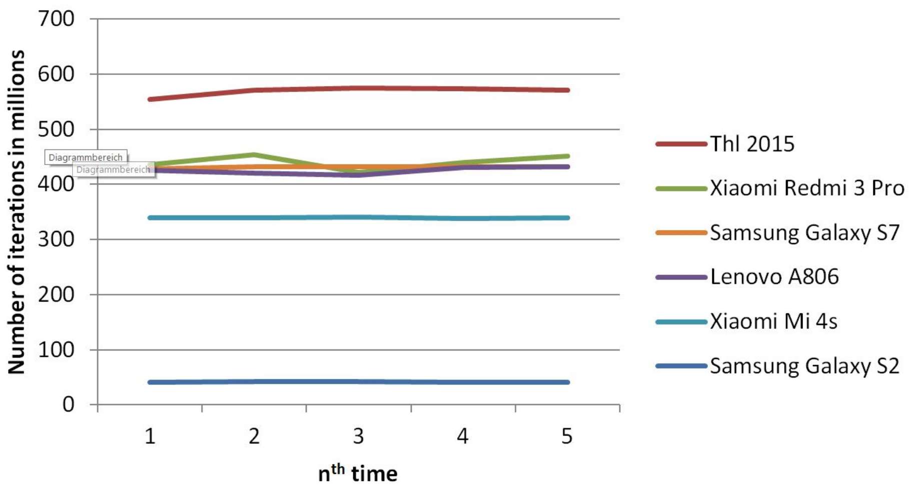


Figure 3.2: Number of iterations on multiple cores in five seconds, measured five times. The number of threads corresponds to the number of cores the device has, but is at least two.

If you calculate the relative standard deviation from the average number of iterations for each device and take the mean of those six numbers you get 3.47% deviation for single core execution and 3.63% deviation for multi core execution. This means that the test is quite consistent and it is possible to reproduce the results.

3.2 Battery Capacity

As the battery capacity can only be read programmatically, we wanted to make sure that the number we get is the correct capacity. This was checked because we noticed that the capacity our app displays is not always the same as the capacity one can read in the official specifications of the phone. To make sure our app gets the correct value we compared this value to the capacity displayed in the system and the battery capacity found in the official specifications of the phone.

As it is apparent in Table 3.1, the battery capacity we find is always equal to the capacity found in the system (if we found it) but this value is not necessarily equal to the capacity in the official specifications of the phone. However, this method of getting the battery capacity programmatically is the best we can do without contacting an external database which has information of the official specifications of all the phones. We did not want to do this because we want the app to work on all devices, not only the ones in the database and we did not find a database which contains this information on all test devices.

In all of the cases except one (discussed in the next paragraph), the battery capacity our app finds is equal to the one displayed in the system. Even though the capacity given by the manufacturer sometimes slightly differs from the capacity displayed in the system, we are as good as the system itself, which is the best we can do without an external database.

In one case (see Table 3.1, “Lenovo A806” entry) the system and the GO Battery Saver did not display any results. It is noticeable that in this case, our app displays a result which is completely different from the manufacturer’s data. This is probably due to lacking information, as programmatically accessed attributes of the phone get saved in a file by the manufacturer when producing the smartphone.

Phone	Android Version	Official Battery Capacity	Battery Capacity in System	Battery Capacity we found
Samsung Galaxy S2	4.1.2	1650mAh	(1650mAh)	1650mAh
Thl 2015	4.4.4	2700mAh	(2500mAh)	2500mAh
Xiaomi Redmi 3 Pro	5.1.1	4100mAh	4000mAh	4000mAh
Lenovo A806	4.4.2	2500mAh	-	1000mAh
Samsung Galaxy S7	6.0.1	3000mAh	3000mAh	3000mAh
Xiaomi Mi 4s	5.1.1	3260mAh	3210mAh	3210mAh

Table 3.1: Comparison of different measurements of the battery capacity. You can see the capacity according to the official specifications of the manufacturer, the battery capacity displayed in the system and the capacity displayed in our app. Because the battery capacity is not always displayed in the system, we used the app “GO Battery Saver” [10] in these cases, which are identified with braces around the value.

Even though we now have the battery capacity of the phone, it is hard to tell what that really means for real world performance, i.e., how long the battery lasts. Expertreviews [6] did a comprehensive test of a lot of different smartphones. They measured the SOT (Screen-on-time) of the phones in airplane mode and on a screen brightness set to 170cd/m^2. Then they continuously played some endlessly looped scenes and measured how long the battery lasts. As you can see in Figure 3.3 and 3.4, there is no apparent dependency between the SOT and the battery capacity. The SOT depends more on the type of phone you have and the operating system it runs. Even though we know that, we still decided to leave the battery capacity round in the battle, because in general one can still say that the bigger the battery capacity, the better as the phone probably still lasts longer when not using it.

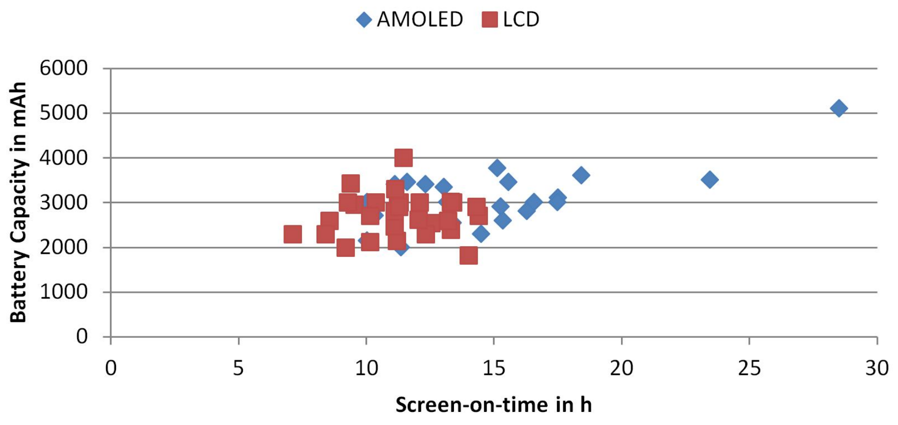


Figure 3.3: SOT (in hours) versus battery capacity (in mAh) of different Android phones. There is no apparent dependency between the SOT and the battery capacity. SOT value is from [6], battery capacity from official specification.

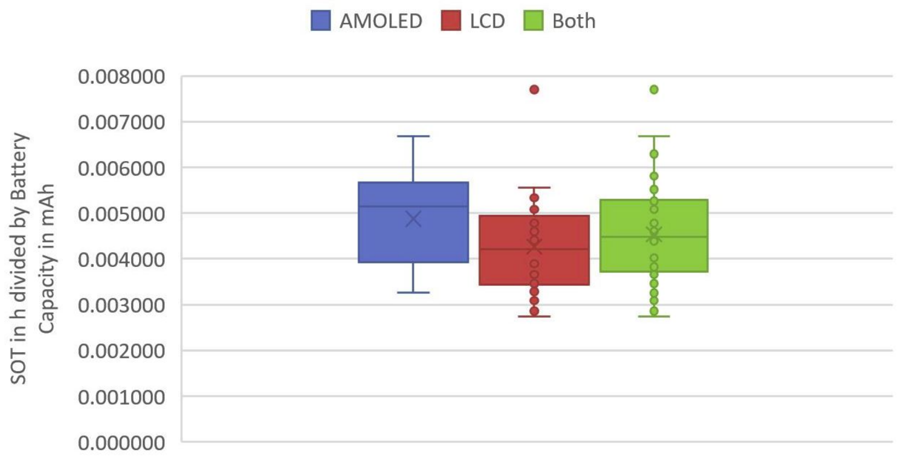


Figure 3.4: Box-plot of the SOT divided by the battery capacity (same data as in Figure 3.3). This plot suggests that there is a difference between AMOLED and LCD displays, but if this is due to the different power consumption of the two display types, or due to the fact that most AMOLED displays have Touch Wiz as they are mostly Samsung devices we do not know.

Chapter 4

Conclusion and Future Work

The app we implemented during this thesis implements the three points mentioned in the introduction: the app does not need any external equipment, two thirds of the rounds are interactive so the user gets the feeling of playing a game and the app is mostly a real-world benchmark. This is achieved especially with the two camera tests. According to the test users, the battle is interesting and fun to play, which hopefully leads to a bigger user base in the future, so more data about different phones can be gathered.

An interesting result we found was that the camera of a CHF150 phone (Xiaomi Redmi 3 Pro) was better than the camera of the four times as expensive Samsung Galaxy S7 (see Figure 4.1). As we could at first not believe this, we took some other picture with the pre-installed camera app, but the result was the same. The camera of the Xiaomi phone is apparently better than the camera of the Samsung Galaxy S7. Of course there could be a problem with this particular Samsung device, or it could just have a dirty camera lens. We would need more data to detect this type of possibly incorrect result.

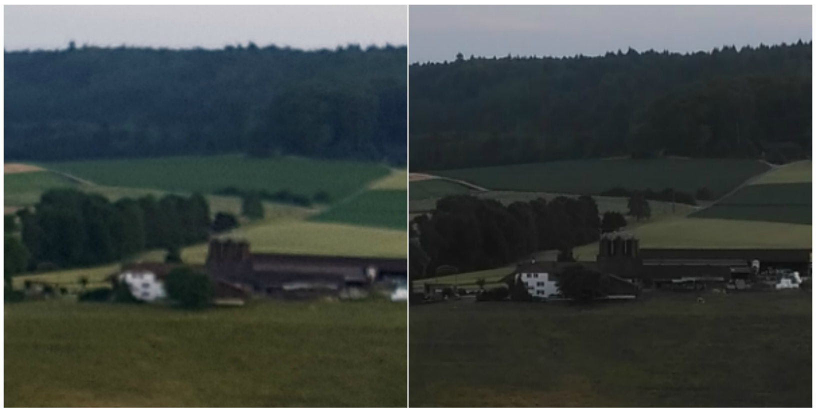


Figure 4.1: Part of a picture taken with the Samsung Galaxy S7 (left) and with the Xiaomi Redmi 3 Pro (right), zoomed by the same amount. It is clearly visible that the camera of the Xiaomi phone has more detail.

A problem with the WiFi test is, that to look if a phone supports 5 GHz WiFi, a method called is5GHzBandSupported() [11] is used. Firstly, this method was only introduced in API 21, which equals Android 5.0 Lollipop. Second, as this method is programmatically, it means that if the method returns false, this does not mean that this phone does not support 5 GHz WiFi. So there are some phones which would have 5 GHz WiFi, but do not get the bonus points because they fall into one of those two cases.

While designing the Battery Capacity test, the idea came up to measure the percentage of battery used during the battle. Unfortunately this does not work, as the battery level can only be read in whole percents and this was too imprecise, as the battle should not use more than one percent of the battery. Another problem with such a test would be to know how much of the energy consumption comes from the battle and how much from the background apps. Therefore we decided to use just the battery capacity as the score, even though this does not necessarily mean that the phone can stay awake longer (as discussed in Section 3.2).

4.1 Future Work

There are a lot of ways how one could extend the existing app. For example it is easily possible to add new rounds. If a lot more rounds are added, it would be nice to let the users choose which of those battles they would like to do.

Some rounds that could be added are for example a test on how fast the RAM is, how fast the access to the main memory is, or a test about screen brightness. For the screen brightness test you make the screen as bright as possible and then hold the two phones together in a way that the light sensor of one phone is on the screen of the other phone. This should then be done twice.

It would also be nice, if the users could actually see their scores of each round in the final overview. At the moment the users only see who won which round, but no scores from the rounds itself.

If one would like to make more evaluations about the past battles, this is also easily possible, as the whole data is saved in the database. Even the pictures are saved, so they could be used for further analysis, for example to find out why people think a certain picture is better. If there is enough data, it may also be possible to find out how different phone manufacturers calibrate their signal strength and if there are differences between different manufacturers.

Bibliography

[1] DxOMark Mobile
https://www.dxomark.com/Mobiles
Accessed 13-February-2017.

[2] AnTuTu Benchmark
https://play.google.com/store/apps/details?id=com.antutu.ABenchMark
Accessed 13-February-2017.

[3] Geekbench 4 by Primate Labs Inc.
https://play.google.com/store/apps/details?id=com.primatelabs.geekbench
Accessed 13-February-2017.

[4] Speedtest.net by Ookla
https://play.google.com/store/apps/details?id=org.zwanoo.android.speedtest
Accessed 13-February-2017.

[5] DiscoMark Benchmark by disco.ethz
https://play.google.com/store/apps/details?id=ch.ethz.disco.gino.androidbenchmarkaccessibilityrecorder
Accessed 13-February-2017.

[6] Best phone battery life 2017: The BEST smartphones tested
http://www.expertreviews.co.uk/mobile-phones/1402071/best-phone-battery-life-2017-the-best-smartphones-tested
Accessed 23-February-2017.

[7] Android Codenames, Tags, and Build Numbers
https://source.android.com/source/build-numbers
Accessed 01-July-2017.

[8] Android Platform Versions
https://developer.android.com/about/dashboards/index.html
Accessed 01-July-2017.

[9] List of HTTP status codes by Wikipedia
https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
Accessed 01-July-2017.

[10] GO Battery Saver and Power Widget by GOMO Go
https://play.google.com/store/apps/details?id=com.gau.go.launcherex.gowidget.gopowermaster
Accessed 18-February-2017.

[11] Android WifiManager
https://developer.android.com/reference/android/net/wifi/WifiManager.html#is5GHzBandSupported()
Accessed 01-March-2017.

Appendix A

Documentation of the MySQL Database

This appendix is the more precise documentation of the MySQL database which saves all the user and battle information.

A.1 Table USERS

Name	Type	Use
ID	varchar(36)	Key of this table, user ID
REGISTERED_SINCE	datetime	Time this entry was made
LAST_SEEN	datetime	Last time the user opened the app
NICKNAME	varchar(20)	Nickname of the user if he set one
DEVICE_NAME	varchar(50)	Name of the device, is read programmatically by the app
DEVICE_BRAND	varchar(50)	Brand of the device, is read programmatically by the app
HARDWARE	varchar(50)	Hardware name of the device, is read programmatically by the app
RAM	bigint(20)	Size of the RAM in bytes
CAMERA_MAX_WIDTH	int(11)	Width of the largest photo (area-wise) the Camera
CAMERA_MAX_HEIGHT	int(11)	Height of the largest photo (area-wise) the Camera
CAMERA_FOCAL_LENGTH	float	Focal length of the Camera
HAS_GYROSCOPE	tinyint(1)	If this device has a gyroscope

Table A.1: Overview of the USERS table.

A.2 Table BATTLES

Name	Type	Use
BATTLE_ID	bigint(20)	Key of this table, is auto-incremented
MY_ID	varchar(36)	User ID of the user playing this battle
OTHER_ID	varchar(36)	User ID of the opponent. This column is actually redundant, but was left in here as it makes the pairing much easier.
OTHER_BATTLE_ID	bigint(20)	Battle ID of the opponent's battle entry
APP_VERSION	varchar(50)	Version of the app of this user
ANDROID_SDK_VERSION	varchar(50)	Android version of the phone of this user
START_TIME	datetime	Time this entry was made
FINISHED	tinyint(1)	If this battle is already finished
CANCELED	tinyint(1)	If this battle was canceled or finished correctly
FIRST	tinyint(1)	If this user goes first in the Fastest Camera round in this battle
BATT_ERROR_OCCURRED	tinyint(1)	If there occurred an error during the Battery Capacity round
BATT_CAPACITY	int(11)	Battery Capacity of this device in mAh
BATT_WIN	tinyint(1)	If this user wins the Battery Capacity round
WIFI_MAX_SIGNAL	int(11)	Maximal signal strength recorded during the Best WiFi Signal round
WIFI_5GHZ	tinyint(1)	If this device supports 5GHz
WIFI_SCORE	int(11)	Calculated score of the Best WiFi Signal round
WIFI_WIN	tinyint(1)	If this user wins the Best WiFi Signal round
CAME_ERROR_OCCURRED	tinyint(1)	If a fatal error occurred during the Camera Test
CAME_OUT_OF_MEM	tinyint(1)	If a Out Of Memory error occurred during the Camera Test
CAME_MY_WIN	tinyint(1)	If the user chose his own picture as being the better one
...		

Name	Type	Use
CAME_SCORE	int(11)	Calculated score of this round
CAME_PHOTO_PATH	varchar(100)	Path to the photo taken of this user
CAME_WIN	tinyint(1)	If this user wins this round
CPUT_ERROR_OCCURRED	tinyint(1)	If an error occurred during the CPU Test
CPUT_SINGLE	tinyint(1)	Number of iterations calculated during the Single Core phase
CPUT_MULTI	tinyint(1)	Number of iterations calculated during the Multi Core phase
CPUT_SCORE	int(11)	Calculated score of this round
CPUT_WIN	tinyint(1)	If this user wins this round
FAST_ERROR_OCCURRED	tinyint(1)	If an error occurred during the Fastest Camera Test
FAST_TIME	bigint(20)	Time the user had to open the camera and take the picture
FAST_WIN	tinyint(1)	If this user wins this round
GYRO_ERROR_OCCURRED	tinyint(1)	If an error occurred during the Gyroscope round
GYRO_DIFFERENCE	float	Difference between the initial and final position of the device
GYRO_WIN	tinyint(1)	If this user wins this round
SCORE	int(11)	Calculated score of the whole battle
WIN	tinyint(1)	If this user wins this battle

Table A.2: Overview of the BATTLES table.