Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Semester Thesis

Development of a Location Based Game for Android Phones

Thomas Bürli
Sara Pfenninger

January 24, 2012

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

Computer Engineering and Networks Laboratory, ETH Zurich

Abstract

The goal of this semesterthesis is to design, implement and evaluate a location-based game for Android. The idea is to draw a virtual figure with the physical location of the players. A figure consists of dots which need to be submitted by walking to the different locations and drawing a figure like this. When all dots of a figure are submitted, a score is displayed showing how accurate the drawn figure is.
To evaluate our game the power usage as well as the data traffic are analyzed. Concerning the data traffic, we can state that it is kept very small. The total of data sent and received for sumbitting one figure is around 1MB. About 96% of this is used for displaying the map. Also, the energy footprint is reasonably small when the screen is not always on. It is comparable to other application.

Acknowledgments

Our utmost gratitude goes to Prof. Dr. Bernhard Plattner for giving us the opportunity to be part of the CSG ("Communication Systems Group") research. We would also like to express our appreciation to our advisors Bernhard Distl and Sacha Trifunovic for many insightful conversations during the development of our Android application in this semesterthesis, their helpful comments on the text, their kindness and most of all, for their patience. It has been an honour and a great pleasure to work with you.

Contents

1. Introduction **5**

2. The Game **6**
2.1. Overview of the Game 6
2.1.1. Game Flow 7
2.1.2. Scoring system 8

3. Design and Implementation **11**
3.1. Client 11
3.1.1. Flow diagram 11
3.2. Server 13
3.2.1. Storage: Database 13
3.2.2. Logic: Webserver 15
3.3. Client & Server Communication 15

4. Evaluation **16**
4.1. Screenshots and Game Evaluation 16
4.2. Power Usage 18
4.3. Data traffic 19

5. Future Work **20**
5.1. Improvement possibilities 20
5.2. Extension possibilities 20

6. Conclusion **22**

A. Installation **23**
A.1. Downloading Android Application 23
A.2. Installation of the server 23
A.2.1. Certificate 24
A.3. Setting up the client 25
A.3.1. Include Server Certificate 25

B. Roadmap **27**

Bibliography **28**

1. Introduction

In the last years, smartphones have become more and more popular and the growth continues. As the smartphone, mobile apps have begun their triumph. The most downloaded apps are games [1]. There are many different sort of games, some of them us the location of the user. But location based gaming is far from being used at its full potential. So this project is a step in this direction. A location based game can motivate friends, families or just a single person to go outside, get some exercise and having fun at the same time.

The goal of this semesterthesis is to design, implement and evaluate a location based game. To make the game interesting for the user it should have a strong incentive to play. The idea of the game is that the users see themselves on a map and can draw virtual figures in the world. Every player gets a figure consisting of dots and lines which he has to draw. By walking around and submitting the dots of the figure at specific locations, the figure is drawn on the map. As soon as all dots of the given figure are submitted, the task is accomplished and a score is derived depending how accurate the drawn figure is.

After having designed the game, the goal is to develop it as an Android application with the corresponding server infrastructure. At the end the resource consumption of the game is evaluated such as how much battery the game uses and the amount of network traffic. Power usage is an important factor since the user usally does not have the possibility to charge the cell phone when he is in transit. The data traffic also matters since the application should not use up all bytes available from the users data plan.

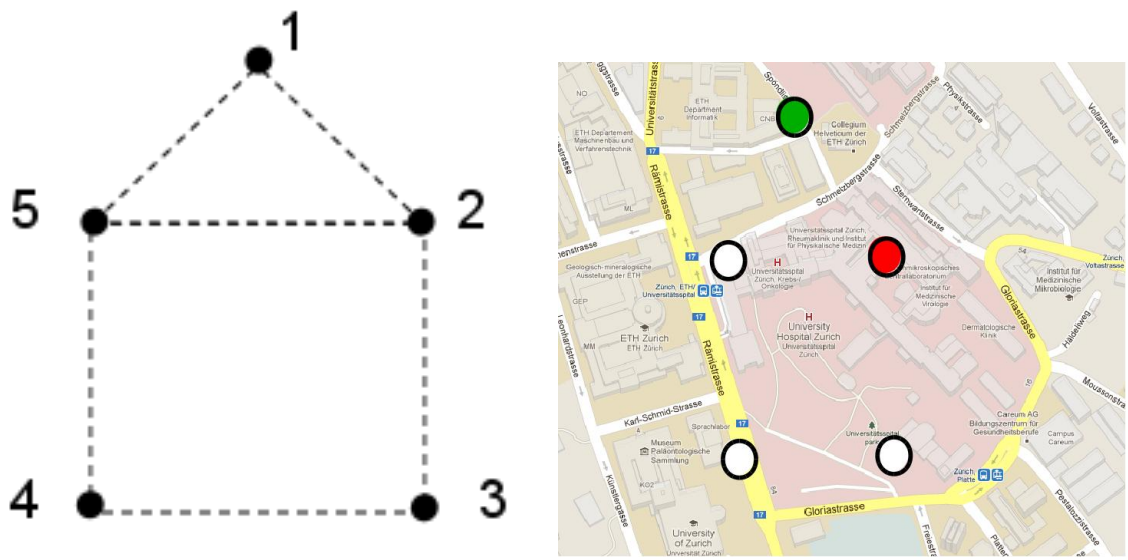
[1] `http://www.apple.com/euro/itunes/charts/apps/top10appstorepaid.html`
`https://market.android.com/details?id=apps_topselling_paid`

2. The Game

In this chapter, we will describe how our developed location-based game called "DrawMe" works and how it is designed.

2.1. Overview of the Game

The idea is to draw a virtual figure with the physical location of the players. A figure consists of dots and lines as shown in figure 2.1a. A possible version of how to draw the house is shown in figure 2.1b.



(a) Figure which has to be drawn

(b) How the figure could be placed

Figure 2.1.: Idea of the game

A player can play the game either alone or collaborate with other people in a group. If there is only one person playing the game he has to walk to all the different locations by himself.

To illustrate the idea of the game, an example is shown:

The player is currently standing where the green dot is in figure 2.1b and submits point number 1 which is the top of the roof. Then he walks down and submits the right upper corner of the house point number 2. He continues like this until all points are submitted.

If for example 5 players are in a group, every player could be responsible for a specific dot. So group member 1 submits point number 1, the second player walks to the coordinates of point number 2 and so on. They can coordinate by phone or SMS and afterwards submit at the same time.

2.1.1. Game Flow

The flow of game is presented in figure 2.2 in a very simplified way.

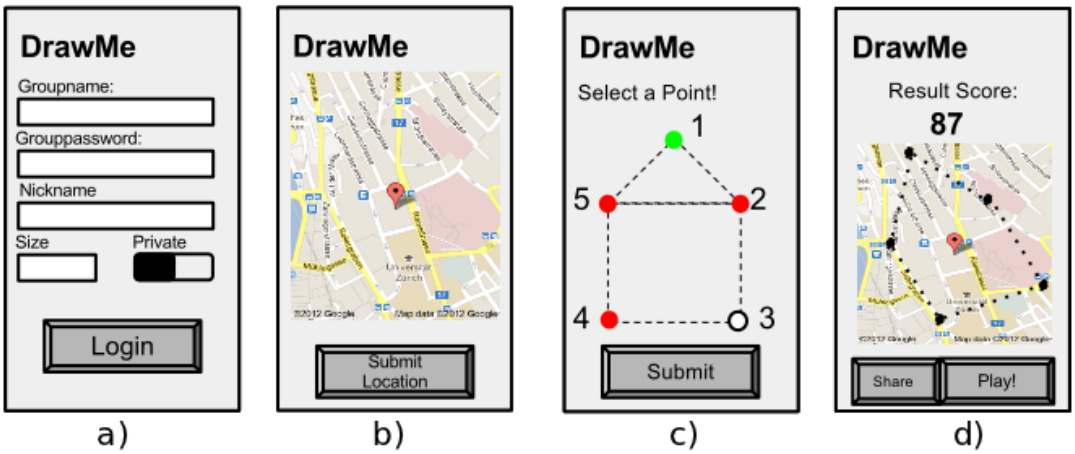


Figure 2.2.: Game Flow

Figure 2.2a shows how to create a new group with a groupname, a grouppassword and a maximal groupsize. A user then distributes the group password to all his group members so that they can play the game together. A player can be part of several groups. As soon as a player is part of a group he gets the figure which was assigned to this group.

In figure 2.2b a player can see his location on a map and can submit the location as a dot (usually a corner) of the figure.

If a player hits the "Location submit" button, he can see the status of each dot (figure 2.2c). A red dot shows that someone has already submitted a location for

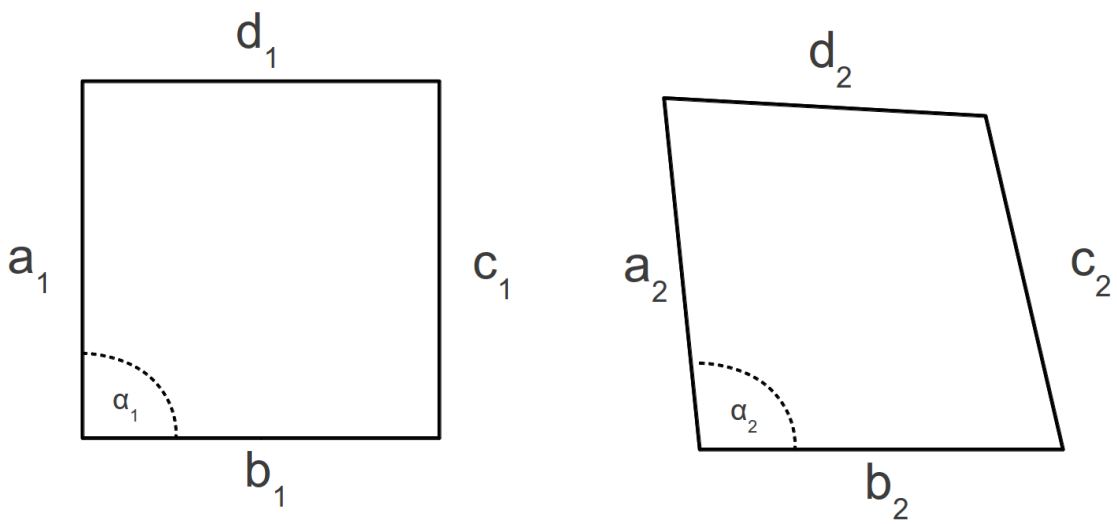
this point, a white dot is a dot which still has to be submitted. By clicking on a red dot the player can see who has submitted this dot. If the user clicks on a white dot, the dot is selected and turns green. By hitting the submit button, the location gets sent to the server.

If all the dots are submitted, the result gets displayed (figure 2.2d). On the result page, each player can see the reached score and the drawn figure.

2.1.2. Scoring system

The score respectively the ranking is the main motivation behind the game so it is an important part of the game. To derive the score for a finished game two main factors are taken into account, the lengths of the edges and the angles.

First we have a look how to derive the score for the edges.



(a) The given (original) figure

(b) The drawn figure

Figure 2.3.: Scoring illustration

At the beginning, the reference ratio has to be determined. For that all lengths are summed up for each figure. For example, in figure 2.4 the circumference of both figures is derived as follows:

$$sum_1 = a_1 + b_1 + c_1 + d_1$$
$$sum_2 = a_2 + b_2 + c_2 + d_2$$

And then the reference ratio is calculated:

$$ref = \frac{sum_1}{sum_2}$$

The errors are accumulated $diff$ and the score is derived in percentage how good the length ratios are: $score_{length}$.

$$diff = \frac{|a_1 - ref * a_2|}{a_1} + ... + \frac{|d_1 - ref * d_2|}{d_1}$$

$$score_{length} = \exp\left(-FACTOR_{scale} * \left(\frac{diff}{num}\right)\right) * 100$$

num is the number of lines. $FACTOR_{scale}$ is a scaling factor to adjust the scoring. The higher the $FACTOR_{scale}$ is set the more difficult it is to get the full score. In our implementation a value between 2 and 4 is for instance a reasonable value for $FACTOR_{scale}$. However, theoretically $FACTOR_{scale}$ can be any value within the interval (0,inf).

In addition to the length ratios, also the angles are compared. Since the angles are independent of the size of the drawn figure, the angles can be directly compared.

$$diff = \frac{|\alpha_1 - \alpha_2|}{\alpha_1} + ... + \frac{|\delta_1 - \delta_2|}{\delta_1}$$

$$score_{angle} = \exp\left(-FACTOR_{scale} * \left(\frac{diff}{num}\right)\right) * 100$$

Here is num the number of angles in the figure. For both, the angles and the lengths, a score is given which counts 50 percent each for the final score.

Example

The following example illustrates how the score is calculated. Figure 2.4a shows the given figure and figure 2.4b represents the drawn figure. The units of the lenghts are not relevant in this example.

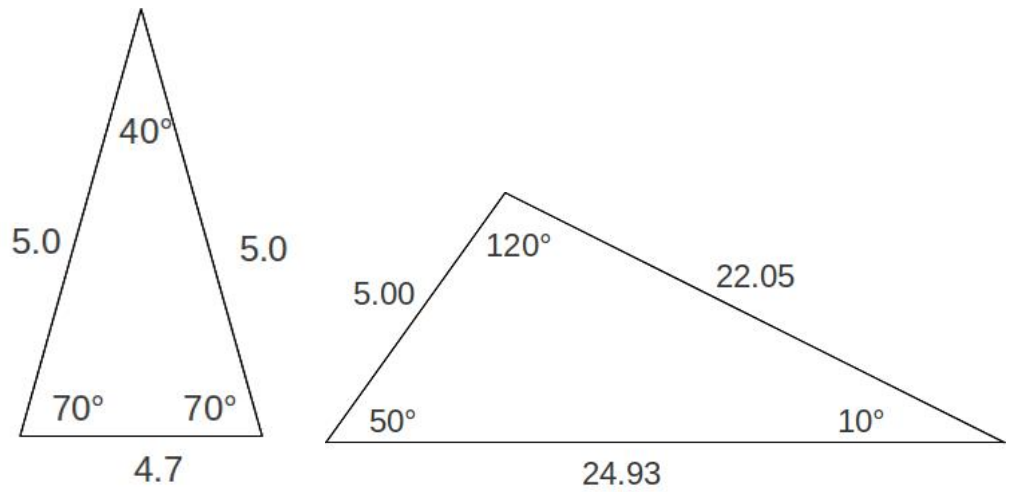
To derive the lenth we calculate the circumferences of both triangles.

$$sum1 = a_1 + b_1 + c_1 + d_1 = 14.7$$

$$sum2 = a_2 + b_2 + c_2 + d_2 = 51.98$$

The reference value becomes then:

$$ref = \frac{sum_1}{sum_2} = 0.28$$



(a) The given (original) figure

(b) The drawn figure

Figure 2.4.: Scoring example

Then we calculate the difference:

$$diff = \frac{5 - 0.28 * 5}{5} + \frac{5 - 0.28 * 22.05}{22.05} + \frac{4.7 - 0.28 * 24.93}{24.93} = 0.57$$

$$score_{length} = \exp\left(-4 * \left(\frac{0.57}{3}\right)\right) * 100 = 46.77$$

To calculate the score of the angles we compare the difference of the different angles.

$$diff = \frac{|\alpha_1 - \alpha_2|}{\alpha_1} + ... + \frac{|\delta_1 - \delta_2|}{\delta_1} = \frac{|70 - 50|}{70} + \frac{|40 - 120|}{40} + \frac{|70 - 10|}{70} = 3.14$$

$$score_{angle} = \exp\left(-2 * \left(\frac{3.14}{3}\right)\right) * 100 = 12.33$$

The final score can then be determined as follows:

$$score_{final} = score_{length}/2 + score_{angle}/2 = 29.55$$

This score is not that high but when looking at the drawn triangle in figure 2.4b we can see that it is not so nicely drawn.

3. Design and Implementation

The application consists of two parts, a client and a server. The client is the interface to the user implemented in Android 2.2 (API Level 8). The client can easily be replaced by another client. All the different smartphones act as clients. On the other side is the server which is responsible for the main logic and storage. The main logic is implemented in PHP and for the storage we use a SQL database.

3.1. Client

The client is the interface of the game for the user who must be able to login, see himself on the map, submit points or similar actions in order to play the game.
Our design philosophy is to store nothing on the client, so everything is saved in the database of the server. Therefore, if the client needs any information such as the correctness of the password or the already submitted points, it asks the server.
However for usability reasons there is an exception. User data such as the id of the user are passed directly from one activity to the next, so that the user does not have to login when switching the screen.

3.1.1. Flow diagram

On the client side, the flow looks as shown in 3.1. The rectangles represent all 13 activities which are also the screens in the application. The black arrows are buttons on the screen and the red arrows show the behaviour when pressing the back button on the device. The program is designed that every task, such as login or register has its own activity. The overview activity is the main screen of the application. Here, the user can see his teams which he is part of. From the overview screen the user has also access to the highscore and a list of all his played games. If the user wants to play he can select a game. He also has the possibility to join or create a new team and after that start a game.

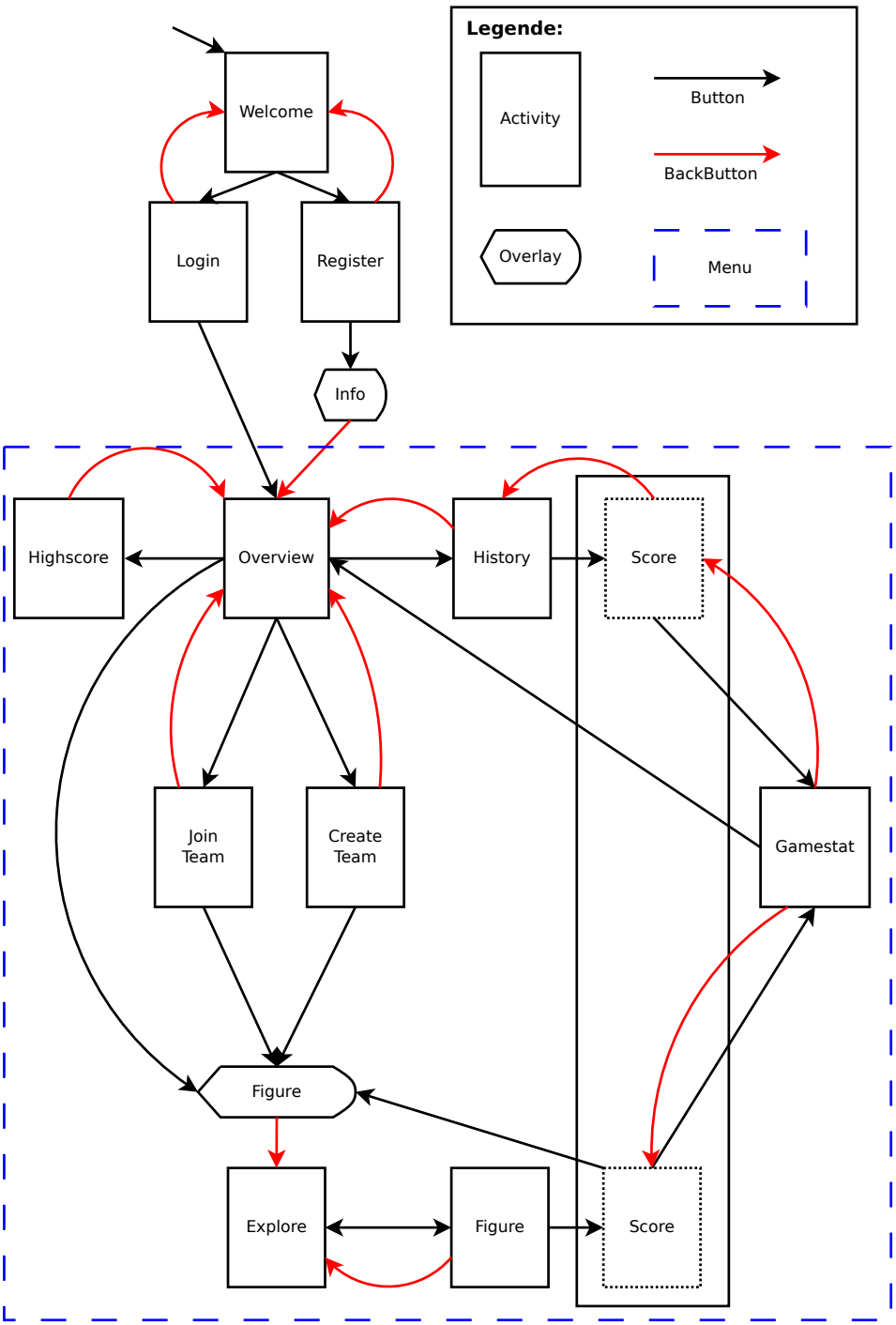


Figure 3.1.: Architecture for program flow showing the activity files representing the different screens

3.2. Server

There are mainly two parts for the server, the logic and the storage of the data of the game. The logic is programmed in PHP and the storage is a SQL-Database. The server coordinates the game. The status of the dots of the figure, the scores, the team names and other important information about the game is all saved on a database on the server.

3.2.1. Storage: Database

In figure 3.2 one can see the entity relationship of the different tables in the database. SQL is used for the database.
In the following the different tables from figure 3.2 are described.

- **User**: The user represents a player with a name, an email address, a password, and other attributes.
- **Team**: A user has to be in a team to play the game. There can also be just one user in a team.
- **User in Team**: This table links the user with the team, it shows who is in which team. A user can be in several different teams.
- **Point**: A point is described by posX and posY which are the coordinates on the screen.
- **Line**: A line consists of two points.
- **Figure**: A figure is represened by several points/lines.
- **Game**: This table specifies a game which a team is playing with a particular figure.
- **Location**: A location is given by the longitude and latitude of the GPS coordinates where a point was submitted. It's assigned to a game and point.

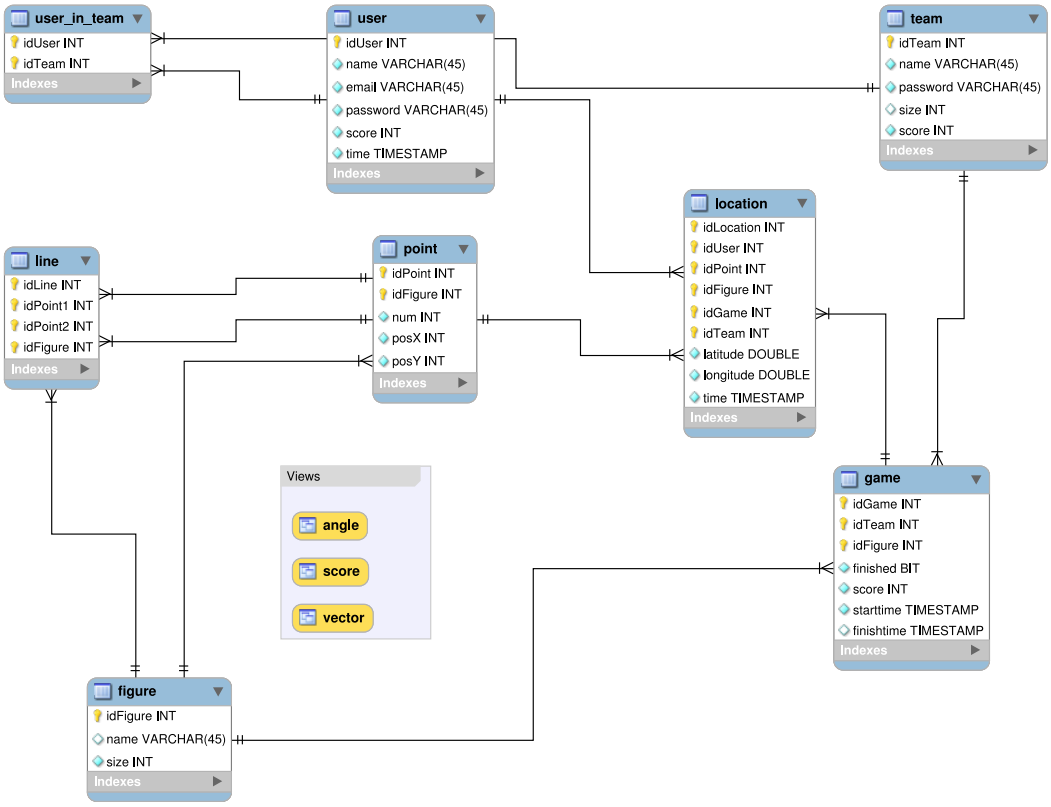


Figure 3.2.: Entity relationship diagram

3.2.2. Logic: Webserver

The server is responsible for the logic of the game. It runs on an apache2 webserver and is written in php. In general there is a php file on the server for every screen shown in figure 3.1 which processes the items sent from the according activity.
The php files all use the same error codes. The codes are chosen such that they are in accordance with the HTTPS error codes so that 1xx is informational, 2xx is a success, 3xx redirection, 4xx a client error and 5xx a server error.
The Android client uses HTTP POST to send data to the server. The server on his side, also supports HTTP GET request which is useful for debugging and testing the webapplication. In fact the game could be played by using the webbrowser. When the server gets a request it responds with a JSON (JavaScript Object Notation) object. The JSON format was chosen because it provides human-readable data interchange which is very helpful for debugging. In addition to that, Android supports JSON natively.

3.3. Client & Server Communication

The client sends the data to the server which then processes it and sends a JSON object back to the client what is schematically drawn in figure 3.3.
The communication between the server is secured with HTTPS. This means that the client and the server use a TLS/SSL connection in order to ensure privacy. In addition the password is also saved encrypted in the database.

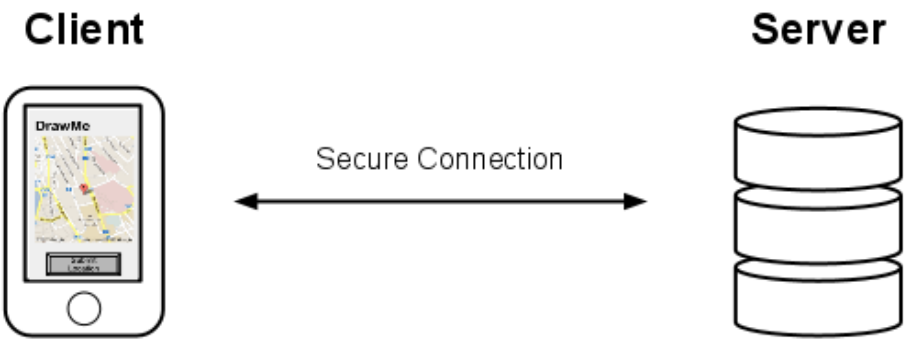


Figure 3.3.: General Architecture

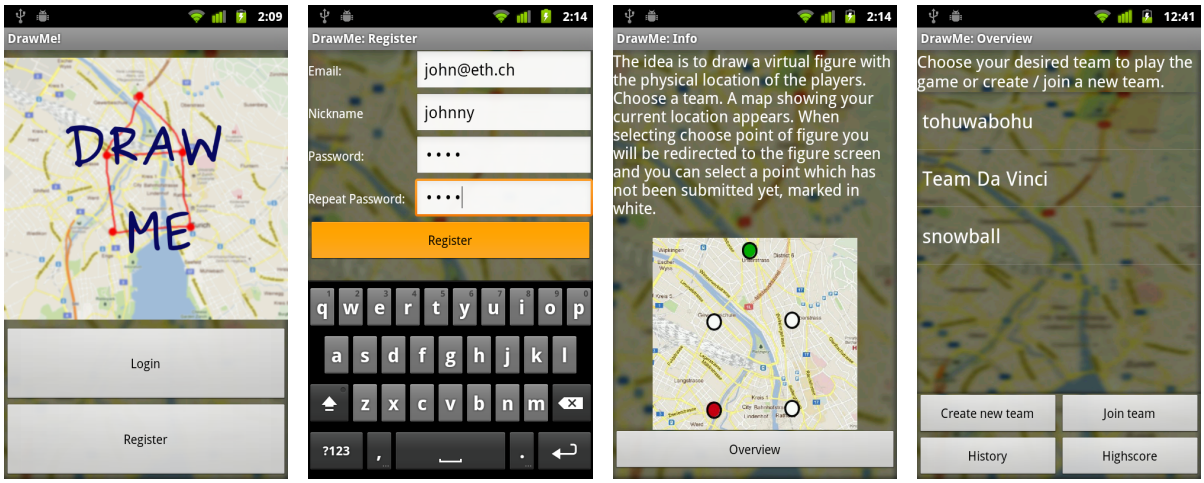
4. Evaluation

We will evaluate our game and analyze two important factors: The power usage as well as the data traffic. Power usage is important because the user is not willing to charge his phone every two hours and also might not have the possibility to do so, when he is for example on a hiking trip or playing the game in a forest. The second important factor is how much data is sent to and from the servers. The user very likely uses a data plan when he is playing outside and there is usually only a limited amount of data he can download within a month. It is therefore crucial to minimize data traffic.

4.1. Screenshots and Game Evaluation

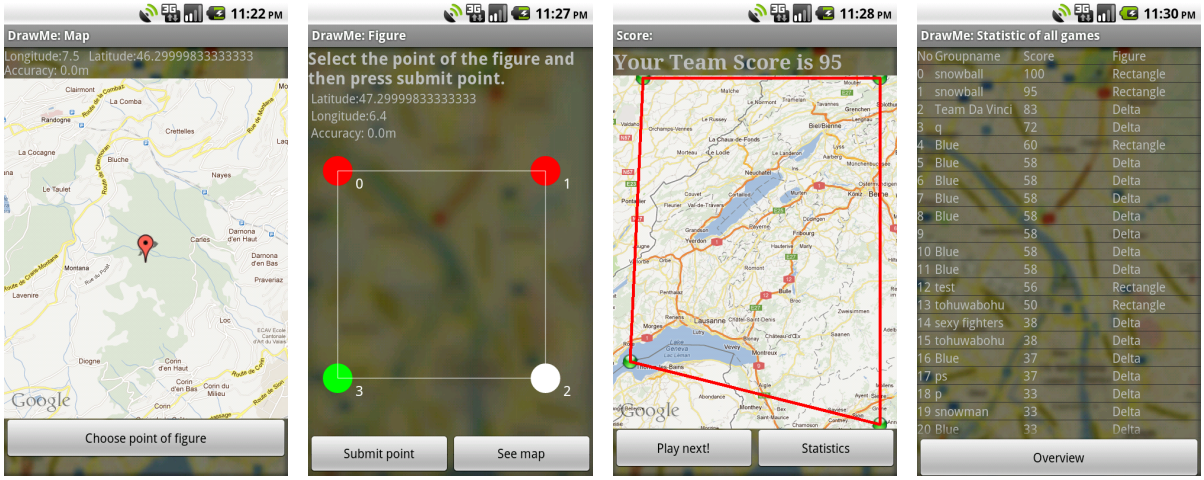
Figure 4.1 and figure 4.2 show how the result looks like. First there is the "Welcome" screen and then the "Registration" screen. The third screen shows the "Overview" which represents the main menu. A group can now be selected or a new one can be created. Also the history can be seen or the highscore. By doing so, the game can be started and a map with the current location is shown. Whenever the user wants to submit a point, he presses the "choose point of figure" button and then can select the according point of the figure. If all points are submitted the drawn figure and the reached score are shown. Eventually the highscore is being displayed.

A problem is that if there is bad reception for GPS such as in a building, a wireless spot is used to locate where the user is currently standing. Depending on how far the user is from such a spot or how many satellites can be located, the accuracy varies quite a bit but what can only be influenced by having better reception or standing closer to an access point. Another problem is that it is possible to cheat by sending arbitrary GPS coordinates using the emulator.



(a) Welcome (b) Registration (c) Information (d) Overview

Figure 4.1.: Screens of the programmed application



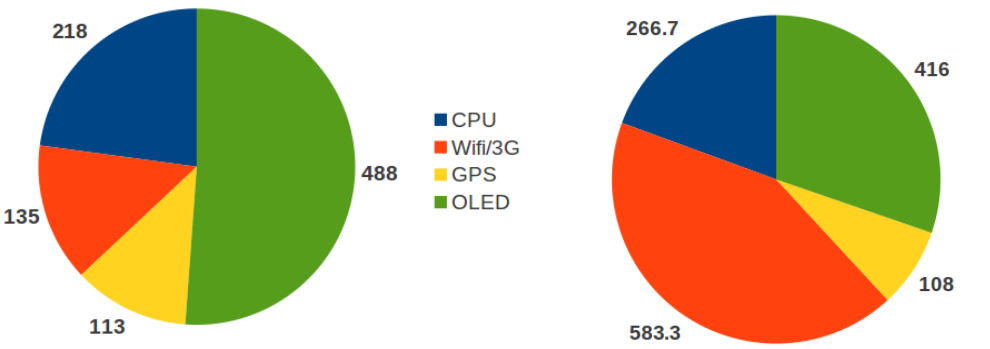
(a) Map (b) Figure (c) Score (d) Highscore

Figure 4.2.: Screens of the programmed application

4.2. Power Usage

To monitor the power usage of our application, we used a tool provided by the University of Michigan. It was developed by Mark Gordon, Lide Zhang and Birjodh Tiwana and is called PowerTutor.[1]

The measurements are taken by playing the game several times during 1 minute. Playing means observing google maps, walking around and submitting all points without leaving the phone at rest.
The power used when there is only WiFi switched on is showed in figure 4.3a. Figure 4.3b shows the power usage when also 3G is used. The numbers of the pie chart indicate the power usage in milliwatt (mW). We can see that the power consumption using 3G is much higher, 1375 mW, than when using WiFi which uses 954 mW. The battery life would be only about 3 hours when playing the game without a pause. A large part is used for the screen which consumes about 450mW independant of WiFi or 3G and is also independant of the application. However it is not a realistic scenario that a user would play the game with the screen on all the time.
The application is designed that it only uses power when it runs in the foreground. So when the game is running in the background, the game does not use any power. Therefore the power usage is fine and comparable to other applications or normal phone usage.



(a) Power Usage with WLAN

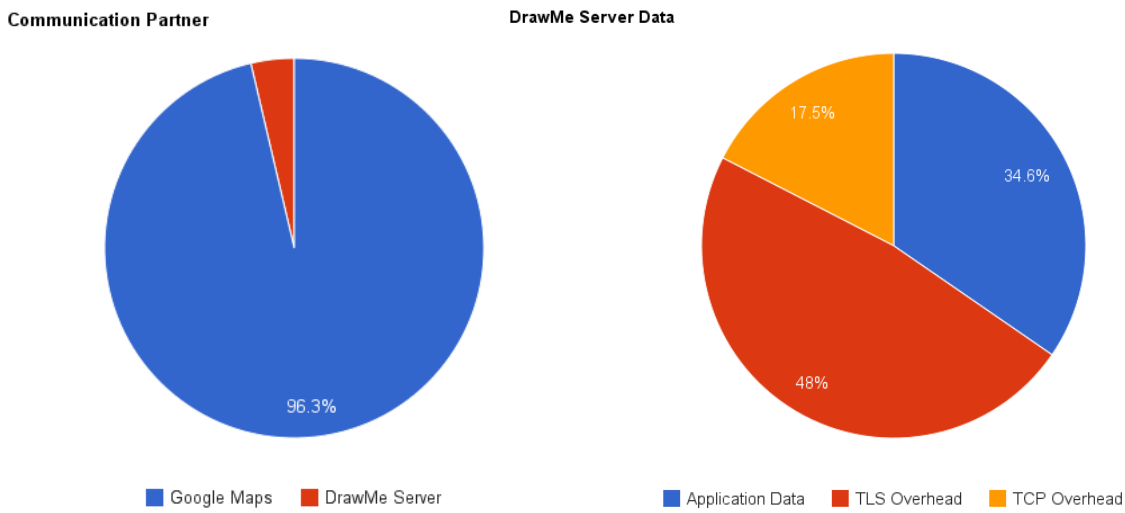
(b) Power Usage with 3G

Figure 4.3.: Power Usage when just using wireless (left) and when also having 3G switched on

4.3. Data traffic

To analyze how much data is sent to the server from the phone, we completed the following steps several times: Login, joining a team, checking the location on the map and then submitting all the points of the figure in raduius of 100m and finally reading the score. We used the Android emulator and whireshark[1] to get the amount of traffic which has been sent and received. At the end the average was taken from several rounds.

The total data which is sent and received is around 1MB. We can see in figure 4.4a that 96.3% is from google maps server for displaying the maps and only 3.7% is from our server (DrawMe server).This is about 40KB of the total data sent.
The piechart on the right side in figure 4.4b shows the data traffic distribution of the DrawMe server in more detail. We can see that establishing a secure connection including the handshake uses more data than what the size of the actual application data is.



(a) Data Traffic: Communication Partner

(b) Data Traffic: Server Data

Figure 4.4.: Data Traffic analysis

[1] http://www.wireshark.org/

5. Future Work

There are different ideas what could be done in the future to improve the application or to extend it with some nice and helpful features which make it more enjoyable to play the game.

5.1. Improvement possibilities

One improvement possibility is to make the scoring system fairer. It would be nice to also include the difficulty of the figure, size of the drawn figure, group size, or the elapsed time. The scoring system could also be made more accurate by using not just planar but spherical coordinates by taking into account that people from all over the world are participating. However this would make it considerably harder to derive a fair score.
Furthermore different rank lists could be displayed such as the largest figure or the figure on the highest altitude.
To reduce the network traffic, only one TCP connection should be established to encrypt the packets. Like this there will only be one SSL handshake.
Concerning the problem that it is possible to send arbitrary GPS coordinates using the emulator one could check which wireless spots are within reach. After a person has submitted a GPS coordinate, all wireless spots which can be reached from this location are determined. If they match the ones reported from the phone, it is likely that the GPS coordinate is correct. If none of them matches it is likely that the person did GPS spoofing.

5.2. Extension possibilities

There are different possibilities to extend the game with features. One such feature would be to introduce different difficulty levels so that it is possible to get to more difficult levels the more often one plays it. To make the game easier, one could also display all or a part of the already submitted dots on the map itself so that a player can check where the points so far got submitted. To make the game more difficult, one could limit the time span from the first submitted dot to the last, so it can only be done in groups. It would probably also be nice to have some more rankings such as lists like group ranking, ranking for each figure with highest score, biggest drawing and fastest group, nicest pics in free drawing.

It could also be possible to create special events/challenges likes biggest Swiss cross

or the first group which have a dot in every main city or to be able to create new figures which have to be drawn by other groups.

Other ideas include that it is possible to communicate through the application so that a boss of a group can coordinate the game. He could mark the location on the map showing who should submit a specific point at which location. Each player can then see the marks. A nice feature would also be to see the other users from the same team to better coordinate and make the game easier. To make the game more exciting one could implement easter eggs or put different surprises at certain locations and when a person passes by close to these coordinates, he or she can catch the surprise.

6. Conclusion

As can be seen from the screenshots in the evaluation of the game, we designed and successfully implemented the game in Android. There are lot of features which also could have been implemented but because of time constraints we had to make a selection. A known problem is that it is possible to do GPS spoofing for example by using an emulator. There are different possibilities to make it harder but it is difficult to get rid of the problem and to make it fully secure.

Concerning the power usage, the battery theoretically only holds for about three hours. However the amount of power consumption is comparable to when using non-stop a webbrowser. A disadvantage is that the game is probably more often played outdoors where there is no wireless access so that 3G is used which uses a bit more energy. However, since the game will run most of the time in background the energy footprint is feasible and the battery will last much longer.

Analysing the data traffic reveals that most of the traffic is used for the google maps server, about 96% are sent from the google server and only about 40 KB from our own server. Having a closer look at our own server shows that most of it is used for establishing a secure connection with certificates. However the big advantage is that the traffic is sent over a secure connection so that nobody else can listen and collect private data.

A. Installation

A.1. Downloading Android Application

The URL to download the Android application is: http://people.ee.ethz.ch/∼tbuerli/drawme.apk
By scanning the following barcode it can be directly installed on the phone.

Figure A.1.: Barcode for downloading the application

A.2. Installation of the server

This howto is for an ubuntu installation.
First a PHP and a MySQL server must be installed on the linux server. This is done as follows:

```
$ sudo apt-get install apache2
$ sudo apt-get install php5-mysql
$ sudo apt-get install libapache2-mod-php5
$ sudo apt-get install mysql-server
```

The following command activates the ssl module of apache:

```
$ sudo a2enmod ssl
```

In /etc/apache2/ports.conf the following section has to be uncommented:

```
<IfModule mod_ssl.c>
    Listen 443
</IfModule>
```

Now a new site can be created:

```
$ sudo cp /etc/apache2/sites-available/default-ssl
/etc/apache2/sites-available/drawme-ssl
```

At this point the PHP server installation is finished. The site should now be enabled and the server can now be run by:

```
$ sudo a2ensite drawme-ssl
$ sudo /etc/init.d/apache2 restart
```

Next, the PHP files should be copied to the following folder: "/var/www/final". After that the database has to be created in MySQL by using the script schema.sql.

A.2.1. Certificate

CA.pl is be used to creat the certificates. Make a new directory in which the certificates will be created.
Create the CA directories and files:

```
$ /usr/lib/ssl/misc/CA.pl -newca
## Enter PEM pass phrase: <secret passphrase 1>
## Verifying - Enter PEM pass phrase: <secret passphrase 1>
## Country Name (2 letter code) [AU]:CH
## State or Province Name (full name) [Berkshire]:Zurich
## Locality Name (eg, city) [Newbury]:Zurich
## Organization Name [My Company Ltd]:ETH Zurich
## Organizational Unit Name (eg, section) []:CSG
## Common Name []:DrawMe Root Zertifikat
## Email Address []:youremail@ethz.ch
## A challenge password []:
## An optional company name []:
## Enter pass phrase for ./demoCA/../cakey.pem: <secret passphrase 1>
```

This creates a new CA in the directory demoCA. The CA's self-signed certificate is in demoCA/cacert.pem and its RSA key pair is in demoCA/private/cakey.pem.

Create a certificate request:

```
$ /usr/lib/ssl/misc/CA.pl -newreq
## Enter PEM pass phrase: <secret passphrase 2>
## Verifying - Enter PEM pass phrase: <secret passphrase 2>
```

Common Name must be equal with the FQDN, the remaining details as above.
Sign the certificate request:

```
#/usr/lib/ssl/misc/CA.pl -sign
## Enter pass phrase for ./demoCA/../cakey.pem: <secret passphrase 1>
## Sign the certificate? [y/n]:y
## 1 out of 1 certificate requests certified, commit? [y/n]y
```

In certain situations, e.g., where your certificate and private key are to be used in an unattended SSL server, you may wish to not encrypt the private key, i.e., leave the key in the clear.

```
$ openssl rsa -in newkey.pem -out newkey_unsecure.pem
```

Move following files:

```
$ mv newcert.pem /etc/ssl/certs/drawme_cert.pem
$ mv newkey_unsecure.pem /etc/ssl/private/drawme.key
```

Modify the /etc/apache2/sites-available/drawme-ssl:

```
 SSLCertificateFile    /etc/ssl/certs/drawme_cert.pem
 SSLCertificateKeyFile /etc/ssl/private/drawme.key
```

Restart the webserver

```
$ sudo /etc/init.d/apache2 restart
```

Server should now be up and running

A.3. Setting up the client

A.3.1. Include Server Certificate

Download the server CA certificate from the server. They can be found in demoCA/cacert.pem. Convert it to x509 with this command:

```
$ openssl x509 -in cacert.pem  -out cacert.pem
```

Download the BouncyCastle Provider[1] and store it to a known location. Also ensure that you can invoke the keytool command (usually located under the bin folder of your JRE installation).
Now import the obtained certs (dont import the endpoint cert) into a BouncyCastle formatted keystore.
With the following command a new keystore with the password drawme11 will be created CA certificate will be imported:

```
$ keytool -importcert -v -trustcacerts -file
"path_to_cert/cacert.pem" -alias CACERT -keystore "cacert.bks"
 -provider  org.bouncycastle.jce.provider.BouncyCastleProvider
 -providerpath "path_to_bouncycastle/bcprov-jdk16-146.jar"
 -storetype BKS -storepass drawme11
```

[1] http://bouncycastle.org/download/bcprov-ext-jdk16-146.jar

Verify if the certificates were imported correctly into the keystore:

```
$ keytool -list -keystore "Keystore.bks" -provider
org.bouncycastle.jce.provider.BouncyCastleProvider
-providerpath "path_to_bouncycastle/bcprov-jdk16-145.jar"
 -storetype BKS -storepass drawme11
```

Now it is possible to copy the keystore as a raw resource in your Android application under res/raw/ [2].

B. Roadmap

Figure B.1 shows our roadmap.

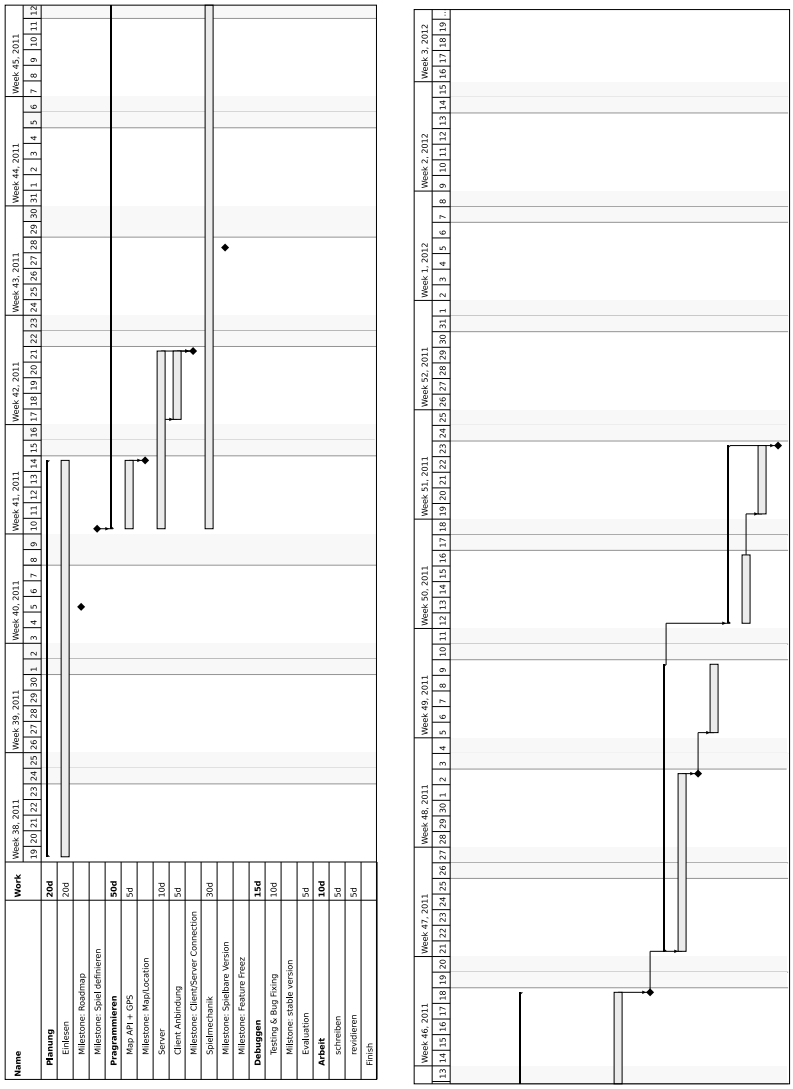


Figure B.1.: Roadmap

Bibliography

[1] Lide Zhang Mark Gordon and Birjodh Tiwana. A power monitor for android-based mobile platforms. Application can be downloaded from: http://ziyang.eecs.umich.edu/projects/powertutor/.

[2] Antoine Hauck. Android: Trusting ssl certificates. Also available as http://blog.antoine.li/2010/10/22/android-trusting-ssl-certificates/.