Mobiler OpenStreetMap J2ME Client

Christian Lins

1 Einführung

Dieses Dokument beschreibt die Entwicklung einer mobilen Kartenapplikation für die *Java Platform, Micro Edition* (J2ME, siehe [Wikc]). Auf Basis der Karten des OpenStreetMap-Projekts erlaubt die Anwendung dem Benutzer aktuelle Straßenkarten auf seinem Mobilgerät anzuzeigen, seine Position mittels eines GPS-Empfängers zu bestimmen und Fehler der Karten direkt vom Mobilgerät dem OpenStreetMap-Projekt zu melden.

2 OpenStreetMap

Das OpenStreetMap (OSM) Projekt[1] hat sich zum Ziel gesetzt eine Weltkarte aufzubauen, die unter einer liberalen Lizenz (Creative Commons Attribution-ShareAlike 2.0, vgl. [RT09], S. 4 u. [Ope]) von jedem verwendet werden darf. Ähnlich wie beim bekannten Wikipedia-Projekt darf jeder ohne Voraussetzungen an dem Projekt teilnehmen und so wird die Karte vollständig von Freiwilligen aus aller Welt aufgebaut und gepflegt (vgl. [RT09], S. 3). Grundlage für die Kartendaten sind GPS-Tracks, die von den Freiwilligen mit GPS-Geräten erstellt und auf der Projektwebseite hochgeladen werden. „Ein GPS-Track ist eine geordnete Liste von Punkten mit Koordinaten, die eine Strecke beschreiben“ (aus [Wike]). Nach diesen GPS-Tracks können dann Wege, Straßen, wichtige Punkte (*Points of Interest* = POIs) und Gebäude in die Weltkarte eingezeichnet werden (vgl. [RT09], S. 25ff).

In Deutschland erreicht OSM in dicht besiedelten Gebieten einen Detailreichtum, der den von kommerziellen Kartendaten (z.B. Tele Atlas oder Navteq) erreicht oder übertrifft. In weniger dicht besiedelten Gebieten ist die Kartenqualität allerdings (noch) nicht den kommerziellen ebenbürtig. Weltweit gesehen hat OpenStreetMap dennoch eine große Bedeutung, denn für einige Entwicklungsländer sind die OSM-Karten überhaupt die einzigen Karten, die digital verfügbar sind. Auch auf Änderungen der Umwelt und Infrastruktur kann das OpenStreetMap-Projekt schnell reagieren. Beispielsweise wurden die Karten von Haiti nach dem letzten schweren Erdbeeben anhand von Satellitenbildern rasch aktualisiert, um den Hilfsorganisationen aktuelle Daten über die Zerstörung der Infrastruktur zu geben.

[1] Siehe http://www.openstreetmap.org/

3 Java Client

Der Java Client ist ein herkömmliches Java-Midlet. Auf die Besonderheiten wird im Folgenden eingegangen.

3.1 Mercator-Projektion der Karte

Das OpenStreetMap-Projekt stellt die gerenderten Karten in der Mercator-Projektion (vgl. [Wikd]) bereit. Diese Projektion bildet die Weltkarte auf einem Zylinder ab (Zylinderprojektion). Dabei wird die Projektion entlang der Zylinderachse (Nord-Süd-Richtung) geeignet verzerrt. An den Polen ist damit die Flächenverzerrung am höchsten. Der Vorteil dieser Projektion ist jedoch, dass die gesamte Weltkarte auf einem Rechteck abgebildet werden kann. Die Verzerrung ist dabei in weitgehend unbewohnten Gegenden der Erde am stärksten und kann daher für das OpenStreetMap-Projekt vernachlässigt werden.

Mit den folgenden Gleichungen kann ein allgemeiner Zusammenhang zwischen den (x, y)-Koordinaten der Mercator-Projektion und den Angaben der geografischen Länge λ und Breite φ hergestellt werden (vgl. [RT09], S. 155):

$x = \lambda - \lambda_0$ mit λ_0 als geografische Länge des Kartenzentrums
$y = \frac{1}{2} \ln(\frac{1+\sin\varphi}{1-\sin\varphi}) = \operatorname{arctanh}(\sin\varphi)$

und die Rückprojektion:

$\lambda = x + \lambda_0$
$\varphi = \arctan(\sinh y)$

3.2 Kachelung

Die Mercator-Projektion macht die OSM-Karte zu einem (sehr großen) Quadrat, mit den folgenden Maßen (vgl. [Wika]):

- x läuft von 0 (180°W) bis $2^{zoom} - 1$ (180°E)
- y läuft von 0 (85.0511°N) bis $2^{zoom} - 1$ (85.0511°S)

Die seltsamen Breitengrade ergeben sich nach $85.0511 = \arctan(\sinh \pi)$.

Die quadratisch projizierte Weltkarte wird in Kacheln (Tiles) aufgeteilt und in unterschiedlichen Detailstufen bereitgestellt.

Die URL eines bestimmten Tiles folgt folgendem Schema:

`http://tile.openstreetmap.org/<zoomlevel>/<x>/<y>.png`

Der Zoomlevel geht von 0 (komplette Weltkarte) bis 18 (höchste Detailstufe). Die Berechnung der Tile-Nummern geschieht nach folgender Vorschrift (vgl. [Wika]):

- Projektion der Koordinaten in die Mercator-Projektion, dazu
  - $x = lon$
  - $y = log(tan(lat) + sec(lat))$
- Transformation von x und y in den Bereich von $0..1$ und Ursprung in nach links oben verschieben
  - $x = (1 + (x/\pi))/2$
  - $y = (1 - (y/\pi))/2$

3.3 Tile-basiertes Rendern der Karte

Über die OpenStreetMap-API ist es möglich, die Kartendaten in einem XML-Format zu exportieren und anhand dieser Vektordaten eine Kartenansicht zu rendern. Dies benötigt jedoch Rechenzeit, die auf einem mobilen Gerät nicht verfügbar ist. Daher verwendet der Client die vorgerenderten Bitmaps (Tiles) der OpenStreetMap-Webseite. OSM verwendet dafür standardmäßig den Mapnik-Renderer und stellt die Tiles unter tile.openstreetmap.org zur Verfügung. Die einzelnen Bitmaps lassen sich nach einem vordefinierten Schema für die verschiedenen Zoomstufen und abhängig von der aktuellen Position des Mobilgeräts per HTTP über das Mobilfunknetz abrufen. Neben dem Standard-Renderer gibt es auch spezielle vorgerenderte Karten, z.B. OpenCycleMap[2] für die Fahrradnavigation (vgl. Abbildung 1).

Die Tile-Server des OSM-Projektes werden komplett aus Spendengeldern finanziert und haben gegen Missbrauch gewisse Nutzungsregeln aufgestellt (vgl. [Wikb])), die vom Java-Client nach Möglichkeit beachtet werden:

- Gültiger User-Agent
- Cachen der Tiles für mindestens 7 Tage
- Maximal zwei parallele Verbindungen zum Tile-Server
- Insbesondere das massenweise Herunterladen für die spätere Offline-Nutzung ist untersagt (exzessives Cache-warming), daher stellt der Client dafür auch keine Funktion bereit.

3.4 Rendern der Tiles

Es ist nicht praktikabel, die Kartenkacheln bei jedem Neuzeichnen auch neu vom OpenStreetMap-Server zu laden. Daher wird die Kachel nach dem erstmaligen Laden im Cachesystem gespeichert. Der Client verfügt über einen kleinen Cache im Speicher, dessen

[2] Siehe http://www.opencyclemap.org/

Abbildung 1: Der Java-Client mit Kacheln von OpenCycleMap

Größe aber durch den geringen Speicher des Mobilgeräts begrenzt ist. Er enthält dennoch alle neun Tiles, die auf dem Bildschirm angezeigt werden. Größer ist der Zwischenspeicher auf der Speicherkarte. Der Zugriff hierauf ist deutlich langsamer als auf den Hauptspeicher, aber im Vergleich meist schneller als der Internetzugriff. Zudem kann dadurch teurer (nicht nur, aber auch im finanziellen Sinne) Internet-Traffic gespart werden.

Das aktuell angezeigte Bild besteht immer aus neun Kacheln, wobei je nach Größe des Displays selten mehr als vier Bilder gleichzeitig angezeigt werden können. Dennoch ist es durch die Anzeige von neuen Kacheln recht einfach, das Scrollen zu implementieren (vgl. Abbildung 2).

3.5 Melden eines Kartenfehlers

Menschen machen Fehler. Auch diejenigen, welche in mühsamer Kleinarbeit die digitalen Karten von OpenStreetMap erstellen und erweitern. Daher ist es nur natürlich, dass sich Fehler in die Karten einschleichen, sei es durch Unwissen oder Unachtsamkeit eines Bearbeiters oder durch die tatsächlichen Veränderungen in der Natur (neue oder geänderte Straßen, etc.). Die Karte selbst zu bearbeiten von einem mobilene Gerät zu bearbeiten dürfte die meisten Nutzer überfordern. Es gibt daher den Dienst *OpenStreetBugs*, quasi

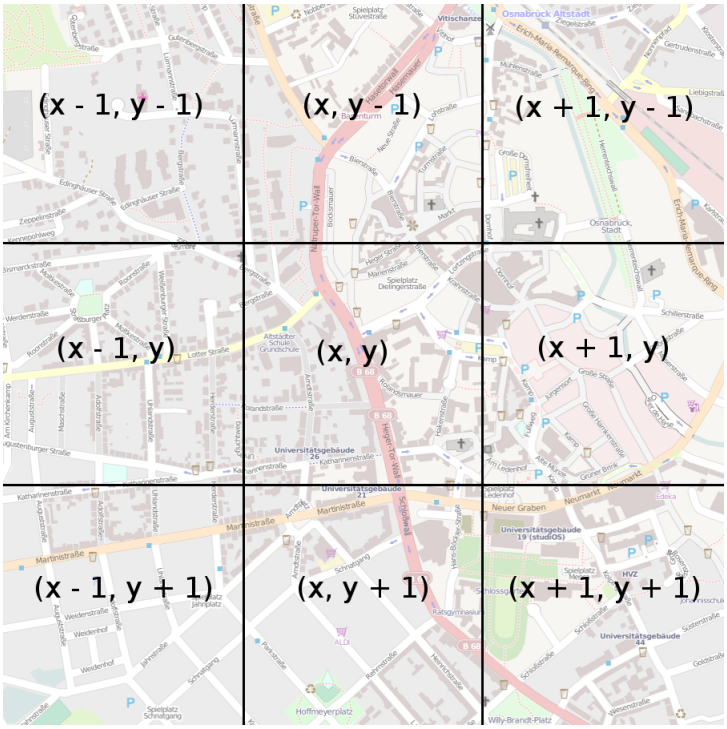

Abbildung 2: Aufteilung der Kartenansicht in Kacheln.

ein Bugtracker für die OSM-Karten. Normalerweise kann ein Benutzer über die Webseite des Dienstes[3] direkt auf der Karte einen Punkt setzen und ihn mit einem Fehlerhinweis versehen. Mapper, die sich in der Gegend auskennen, korrigieren dann die Kartenfehler anhand dieser Hinweise und markieren sie als behoben.

Der Java-Client verfügt über eine Funktion, mit der ein Benutzer von seiner aktuellen Position aus ein Fehlerhinweis an OpenStreetBugs schicken kann. OpenStreetBugs verfügt über kein fest definiertes Interface für Webservices, die benötigten Aufrufe wurden aus dem JavaScript-Quelltext der Seite extrahiert.

Der Aufruf

```
/api/0.1/addPOIexec?lat=8.0&lon=52.0&text=Test&format=js
```

erzeugt einen Fehlerhinweis an der geografischen Position $(8.0, 52.0)$ mit dem Hinweis „Test“.

Der Java Client ist auch in der Lage, die Positionen bereits vorhandener Hinweise von

[3] Siehe `http://openstreetbugs.schokokeks.org/`

OpenStreetBugs zu laden und als Overlay über der eigentlichen Karte anzuzeigen.

Der Aufruf

```
/api/0.1/getBugs?b=36.17496&t=61.03797&l=-9.9793&r=31.54902
```

liefert eine Liste von Hinweisen in dem angegebenen Bereich (b, t, l, r). Die Rückgabe erfolgt in Funktionsaufrufen auf eine JavaScript-Funktion der OpenStreetBugs-Webseite, deren Parameter sich jedoch leicht parsen lassen.

3.6 Positionsbestimmung per GPS

Mobilgeräte mit integriertem GPS-Empfänger stellen die Position des Geräts meist über die Java Location API (JSR-179[4]) zur Verfügung. Da insbesondere Mobilgeräte ohne integrierten GPS-Empfänger diese API nicht bereitstellen auch wenn ihnen über Bluetooth ein externer Empfänger zugeschaltet wurde, hat der Client Methoden integriert, die direkt den NMEA-Ausgabestrom des GPS-Empfängers filtern und so die Position bestimmen. Dazu wird der Bluetooth-GPS-Empfänger über eine serielle Bluetoothverbindung angesprochen. Die dazu nötigen Klassen und Methoden wurden dem Programm *GPS Track*[5] entnommen und angepasst. Die für die Bluetoothverbindung nötige Java-API JSR-82[6] ist auf deutlich mehr Mobilgeräten verfügbar. Vorteil dieser Lösung ist es, dass eine Positionsbestimmung selbst dann möglich ist, wenn das GPS-Gerät keinen Fix auf eine ausreichende Zahl von Satelliten hat (i.d.R.). Die ungefähren Positionen gibt das GPS-Gerät (zumindest bei dem Testgerät mit SirfSTAR III Chip) sofort nach dem Einschalten in den NMEA-Datensätzen des Typs RMC an.
Ein solcher Datensatz hat den folgenden Aufbau:

```
$GPRMC,162614,A,5230.5900,N,01322.3900,E, ...
```

Erkennbar ist der Breitengrad 52.30 N und der Längengrad 13.22 E. Diese Angabe kann sehr exakt bis völlig ungenau sein. Über die Qualität des RMC-Datensatzes gibt der GGA-Datensatz Auskunft. Er enthält Angaben wie z.B. die Anzahl der empfangbaren Satelliten.

Die Position - gleich wie sie bestimmt wurde - wird als blauer Kreis über die Karte gezeichnet und dort alle fünf Sekunden aktualisiert.

[4] Siehe http://jcp.org/en/jsr/detail?id=179
[5] Siehe http://www.qcontinuum.org/gpstrack/
[6] Siehe http://jcp.org/en/jsr/detail?id=82

4 Fazit & Lessons Learnt

Die J2ME Plattform ist inzwischen ausgereift und gut verstanden. Mit NetBeans und Eclipse stehen zwei sehr gute Entwicklungsumgebungen zur Verfügung. Im Internet finden sich schon unzählige Lösungen für nahezu jedes denkbare Problem. Die Entwicklung auf dem Emulator gestaltet sich damit recht einfach.

Problematisch ist jedoch insbesondere die Fehlersuche direkt auf dem mobilen Endgerät. Durch die Vielzahl von Mobiltelefonen und APIs ist es extrem schwer eine komplexe Anwendung zu schreiben, die problemlos auf einem Großteil der Geräte funktioniert. Selbst wenn das Programm abbricht, ist es mangels Debugging-Möglichkeiten nicht möglich den Fehler konstruktiv zu beheben. Das Remote-Debuggen auf dem Gerät (auch on-device debugging) funktionierte mit den getesten Endgeräten nicht.

Zunächst war es nicht möglich, die Software auf dem Sony Ericsson K610i zu starten. Die Virtual Machine verweigert den Dienst mit der Meldung „Ein Anwendungsfehler ist aufgetreten“. Es stellte sich heraus, dass an einigen Stellen im Programm mangels Unterstützung der Java Location API, ein `NoClassDefFoundError` für die Klasse `javax.microedition.location.LocationProvider` auftrat. Erwartet wurde jedoch eine `ClassNotFoundException`, sodass der entsprechende try-catch-Block zunächst ohne Wirkung blieb und die KVM auf dem Gerät abstürzte.

Auf dem zweiten Testgerät, einem Nokia 6300i, startet die Anwendung zwar, war aber durch die ständigen Sicherheitsnachfragen des Systems quasi unbenutzbar. Vor jeder Internetverbindung fragte das Mobiltelefon, ob man denn die Verbindung zulassen wolle. Da die Kartenkacheln vom OpenStreetMap-Server geladen werden, fragt das Gerät bei jedem Scrollvorgang bis zu neun Mal die gleiche Frage. Mangels Zertifikat der Anwendung, lässt sich das ständige Nachfrage nur manuell in den Einstellungen auf eine Nachfrage reduzieren.

Problematisch ist auf beiden Testgeräten der Tilecache auf der Speicherkarte. Ohne gültige Signatur wird der Zugriff auf die Speicherkarte offenbar verweigert (oder zumindest durch ständige Sicherheitsfragen gestört) und nur mit deaktivierten Speicherkarten-Cache ist die Anwendung benutzbar. Ohne diesen Cache müssen jedoch alle Kacheln über das Mobilfunknetz geladen werden, was nicht immer in ausreichender Geschwindigkeit möglich ist. Frustation des Benutzers ist vorprogrammiert.

Eine selbst-zertifizierte Anwendung startet überhaupt nicht, da die Zertifikatskette nicht vollständig ist. Hier zeigt sich auch die größte Schwäche des ganzen Entwicklungssystems: ohne ein CA-signiertes Zertifikat ist es quasi nicht möglich, vollständig taugliche Anwendungen für die J2ME-Plattform zu entwickeln (vlg. [Var]). Kommerzielle Entwickler mag das nicht stören, da diese die Investitionen in Zertifikate nicht scheuen, für OpenSource-Projekte oder Hobbyentwickler, die auf preisgünstigen Endgeräten programmieren möchten, ist dies nicht zu stemmen. Ein einziges Zertifikat genügt jedoch nicht, da nicht jedes Endgerät die gleichen Rootzertifikate vorinstalliert hat. Vermutlich ist je ein Zertifikat von Thawte, Verisign und Java Verified notwendig (vgl. [Pre]). Ein Nachinstallieren von Root-Zertifkaten ist möglich, die MIDP 2.0 Spezifikation verbietet jedoch das Ausführen von Anwendungen, die mit nachträglichen installierten Zertifikaten signiert

wurden, selbst wenn die Zertifikatkette vollständig ist.

Es bleibt abzuwarten, ob Java ME sich weiter als Platform für Mobilgeräte halten kann, zumal die generelle Zukunft von Java nach der Übernahme der Firma Sun durch Oracle ungewiss bleibt.

Literatur

[Ope] OpenStreetMap. Urheberrecht und Lizenz. http://www.openstreetmap.org/copyright. [Online; Stand 13. August 2010].

[Pre] Mihai Preda. Midlet Signing. http://blog.javia.org/midlet-signing/. [Online; Stand 17. August 2010].

[RT09] Frederik Ramm und Jochen Topf. *OpenStreetMap: Die freie Weltkarte nutzen und mitgestalten*. Lehmanns, 2009.

[Var] Various. Generating self-signed certificate. http://discussion.forum.nokia.com/forum/showthread.php?124311-Generating-self-signed-certificate. [Online; Stand 17. August 2010].

[Wika] OpenStreetMap Wiki. Slippy map tilenames. http://wiki.openstreetmap.org/wiki/Slippy_map_tilenames. [Online; Stand 30. Juli 2010].

[Wikb] OpenStreetMap Wiki. Tile usage policy. http://wiki.openstreetmap.org/wiki/Tile_usage_policy. [Online; Stand 30. Juli 2010].

[Wikc] Wikipedia. Java Platform, Micro Edition. http://de.wikipedia.org/w/index.php?title=Java_Platform,_Micro_Edition&oldid=75708498. [Online; Stand 17. August 2010].

[Wikd] Wikipedia. Mercator-Projektion. http://de.wikipedia.org/w/index.php?title=Mercator-Projektion&oldid=77508659. [Online; Stand 7. August 2010].

[Wike] Wikipedia. Track (GPS). http://de.wikipedia.org/w/index.php?title=Track_(GPS)&oldid=77182170. [Online; Stand 17. August 2010].