HTW Chur
Institut für Informations- und Kommunikationstechnologien

OSM Inspector - reloaded

Lukas Toggenburger

Agenda

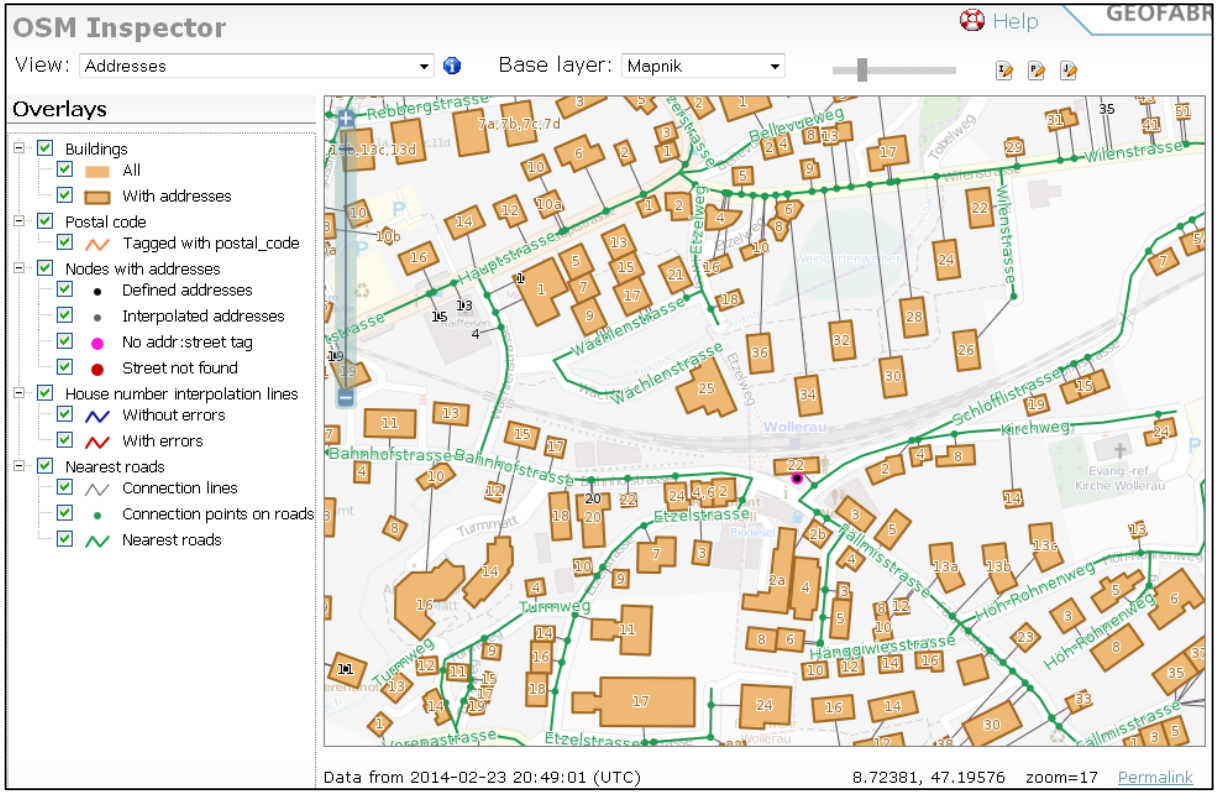
- Ziele der Arbeit
- Einführung in OpenStreetMap und Adress-Tagging
- Ausgangslage
- Implementierung
- Resultate

OSM Inspector

OSM Inspector: Visualisierungs- und Debugging-Werkzeug für OSM der Geofabrik GmbH für die OSM-Community (http://tools.geofabrik.de/osmi/)

Verschiedene Ansichten für verschiedene Aspekte:

- Geometrien
- Tagging
- Küstenlinien
- Routing
- **Adressen**
- …



Ziele der Arbeit

1. Berechnung der Adress-Ansicht des OSMI beschleunigen um weltweite Ansicht zu ermöglichen

2. Überführung des Quellcodes in objektorientierte Programmierung für bessere Wartbarkeit

Was ist OpenStreetMap?

OpenStreetMap ist:

- eine Datenbank für Geodaten (Strassen, Gebäude, Adressen, …)
- mittels «Crowdsourcing» von Freiwilligen erstellt
  - Eine Art «Wikipedia» für Karten
  - Jeder kann und soll mitmachen
  - 1.5 Mio. registrierte Benutzer
- kostenlos benutzbar

Was ist OpenStreetMap?

Komplette Datenbank ist als sogenannte «Planet» Datei verfügbar:

- XML unkomprimiert: 400 GB
- XML komprimiert: 34 GB
- PBF: 24 GB

Drittparteien bieten auch Ausschnitte an.

Datenstrukturen

3 grundlegende Datenstrukturen:

- Nodes (Knoten)
- Ways (Wege = Streckenzüge)
- Relationen

Jede dieser Datenstrukturen wird innerhalb der OSM-Datenbank mit einer ID (positive Ganzzahl) eindeutig gekennzeichnet.

Jede der 3 Datenstrukturen kann beliebige Tags tragen.

Tags

Ein Tag ist ein Paar von zwei Texten:

1. Teil heisst Schlüssel («key»)
2. Teil heisst Wert («value»).

Trennung typischerweise gekennzeichnet durch =

Beispiel:

name=Restaurant Sonne

Regeln:

- Beliebige Keys und Values erlaubt (aber Konsens ist erwünscht!)
- Beliebige Anzahl Tags pro Objekt erlaubt
- Pro Objekt keine mehrfache Verwendung desselben Schlüssels

Node

Einfachste Datenstruktur in OSM.

Ein Node ist ein Punkt mit einem WGS84-Koordinatenpaar (Geografische Länge und Breite)

```
<node id='2498' lat='47.253003617445' lon='8.78535577144'>
    <tag k='amenity' v='restaurant' />
    <tag k='name'    v='Restaurant Sonne' />
</node>
```

Way

Ein Way besteht aus einer Abfolge von Nodes.
Die Nodes werden separat gespeichert und über ihre IDs referenziert.

```
<node id='123' lat='47.253536070794' lon='8.78420646856843' />
<node id='456' lat='47.253490102351' lon='8.78463311534727' />
...
...
<way id='987654'>
    <nd ref='123' />
    <nd ref='456' />
    ...
    <tag k='highway' v='residential' />
    <tag k='oneway' v='yes' />
</way>
```

Die Reihenfolge der Node-Referenzen ist informationstragend.

Relation

Relationen sind Gruppierungen von Datenobjekten (Nodes, Ways, Relationen).

Abhängig vom Typ der Relation

- wird der *Reihenfolge* der Member eine Bedeutung zugemessen
- wird den Membern eine *Rolle* zugewiesen

Anwendungsgebiete von Relationen:

- Multipolygone (Polygone mit «Löchern»)
- Abbiege-Verbote
- Bus-Linien
- Wanderwege
- Adressen (selten)

Adress-Tagging - Karlsruher Schema

Meistverbreitete Methode für Tagging von Post-Adressen

Gebäude (als Way-Objekt):
building=yes
addr:street=Ringstrasse
addr:housenumber=34

Anliegende Strasse (als Way-Objekt):
highway=primary
name=Ringstrasse

Keine Verbindung zwischen adressiertem Objekt und zugehöriger Strasse (ausser Strassenname («Ringstrasse») und geografischer Nähe)

Adress-Tagging - Interpolationslinien

«Relikt» aus der Zeit vor hochauflösenden Luftaufnahmen.

Nur erste und letzte Hausnummer wird getaggt. Zwischenliegende Hausnummern werden entlang einer Hilfslinie interpoliert.

Adress-Tagging - Interpolationslinien

```
<node id='123' lat='47.253536070794' lon='8.784206468568 43'>
    <tag k='addr:street' v='Bahnhofstrasse' />
    <tag k='addr:housenumber' v='1' />
</node>

<node id='456' lat='47.253490102351' lon='8.784633115347 27'>
    <tag k='addr:street' v='Bahnhofstrasse' />
    <tag k='addr:housenumber' v='19' />
</node>

<way id='987654'>
    <nd ref='123' />
    ...
    <nd ref='456' />
<tag k='addr:interpolation' v='odd' />
</way>
```

Ausgangslage

Ausgangslage: Features des OSM Inspectors

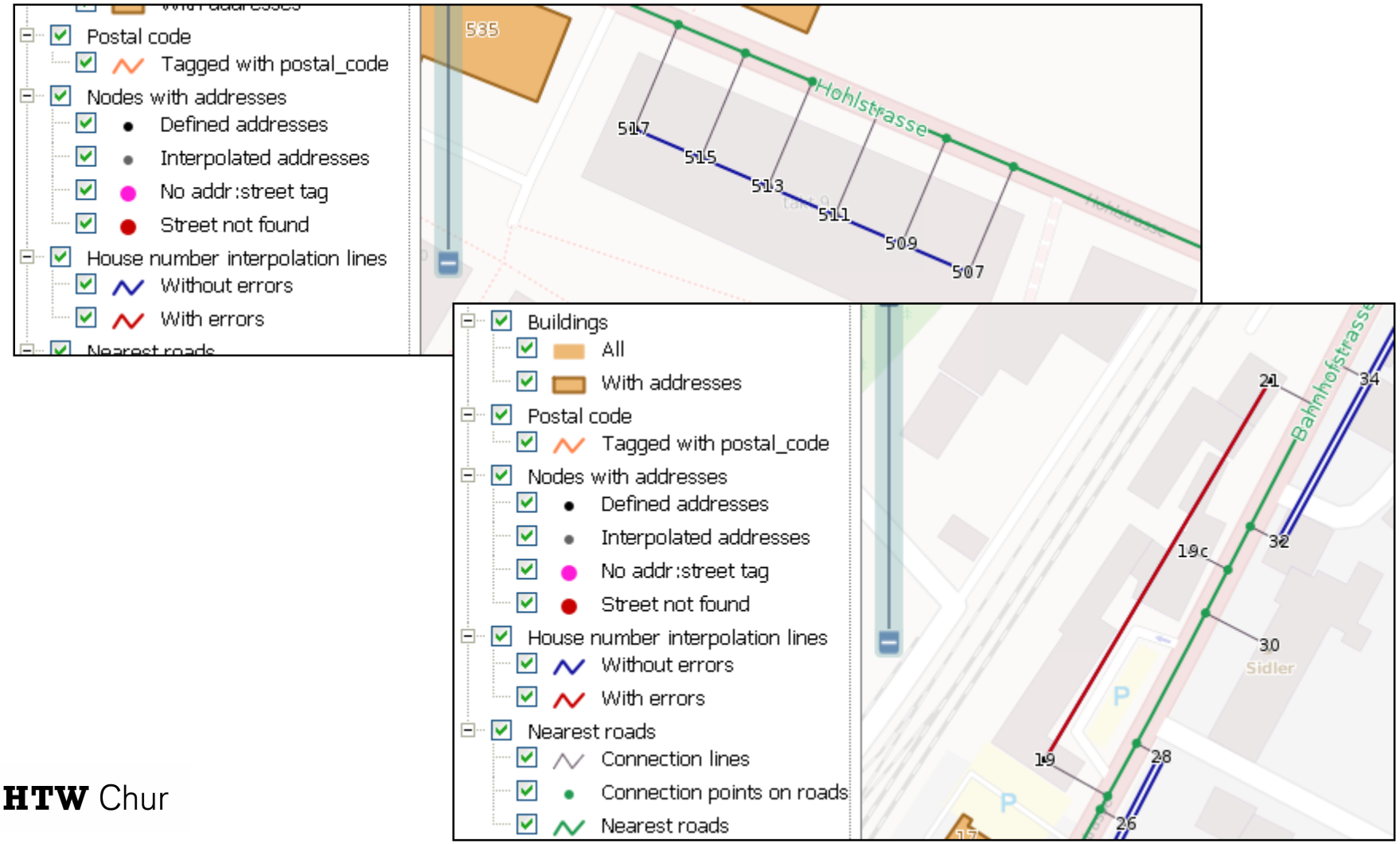
Adressierte Gebäude, Verbindungslinien, Strassen mit anliegenden Adressen, unauffindbare Strassennamen

Ausgangslage: Features des OSM Inspectors

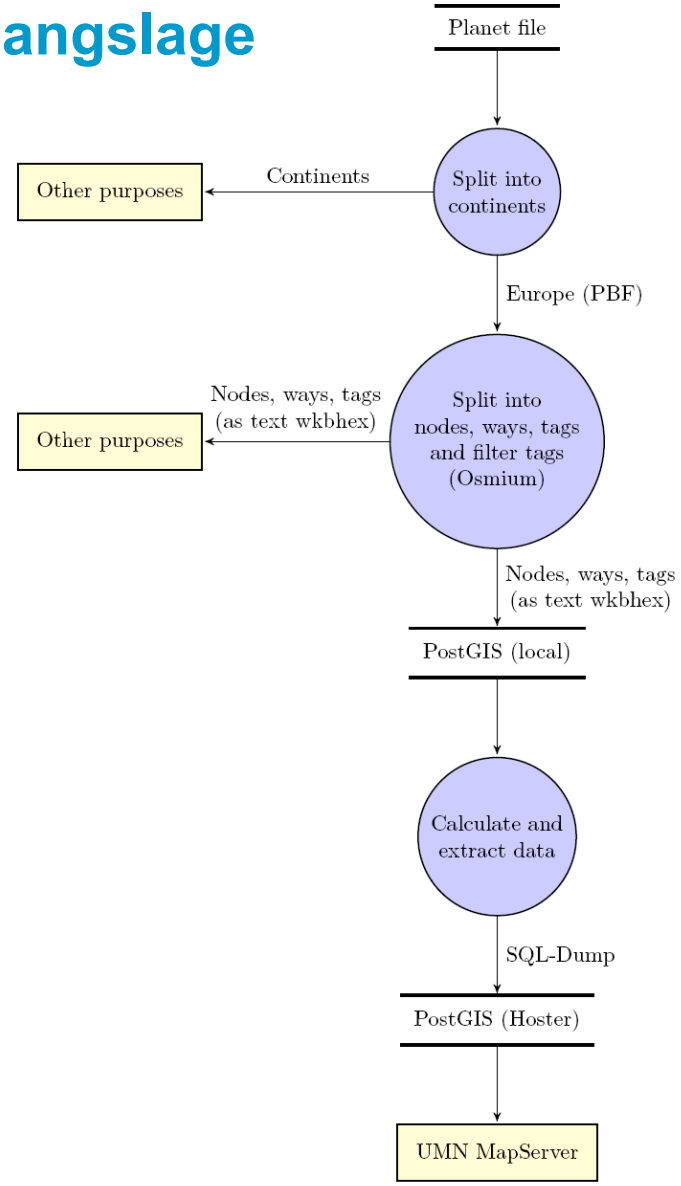
Angabe von Hausnummern ohne Strassennamen

Ausgangslage: Features des OSM Inspectors

Adress-Interpolationslinien



Ausgangslage



Generierter Datensatz wird periodisch aktualisiert.

Dauer für Europa: Über 21 Stunden!

Osmium

C++11 Bibliothek zur Verarbeitung grosser OSM-Datensätze

Wurde von der Geofabrik GmbH vorgegeben.

Für jedes Objekt (Node, Way, Relation) der Input-Datei wird eine Callback-Funktion aufgerufen.

Osmium

Input Format:

OSM-Daten in XML oder PBF Format

Interner Datentyp:

OGR (Vektorgeometrien inkl. entsprechenden Methoden: Überschneidung, Distanz, Koordinatensystemtransformationen, Centroid, konvexe Hülle, etc.)

Output Format:

Shapefile, Spatialite, … total ca. 35 Formate (mittels OGR)

Osmium - Callback Reihenfolge

Reihenfolge der Callbacks (gegeben durch Planet-Datei)

- alle Nodes
- alle Ways
- alle Relations

Osmium - Waypoint Koordinaten

Ways und Relationen enthalten nur Referenzen, aber keine Koordinaten. Weg-Geometrien können nicht direkt verarbeitet werden.

Zwei Ansätze:

1. Planet-Datei mehrmals durchlaufen: IDs von interessanten Nodes zwischenspeichern und Koordinaten im zweiten Durchlauf lesen.

2. Beim Lesen der Nodes alle Koordinaten zwischenspeichern. (Direkt in Osmium unterstützt mittels `NodeLocationForWays` Handler.)

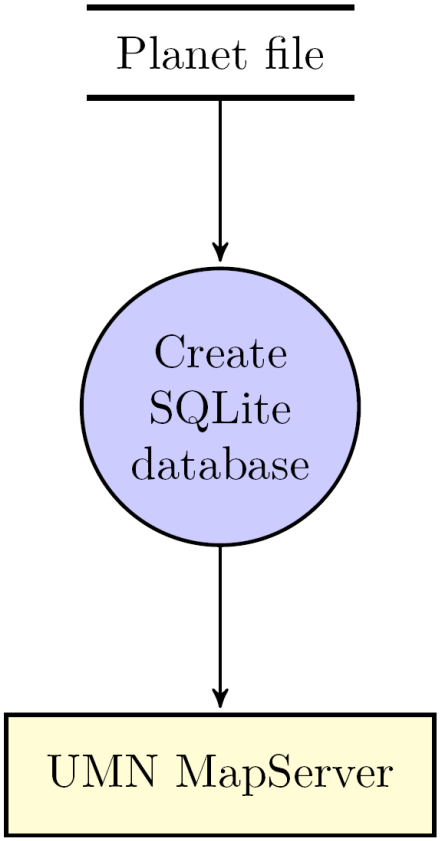
Implementierung

Implementierung - Übersicht

Input: Planet-Datei als PBF-Datei

Output: Spatialite-Datei (SQLite mit GIS-Erweiterung) für MapServer.

Bestehende Tabellenstruktur wird übernommen.



Implementierung - Übersicht

Implementierung in zwei Durchläufen:

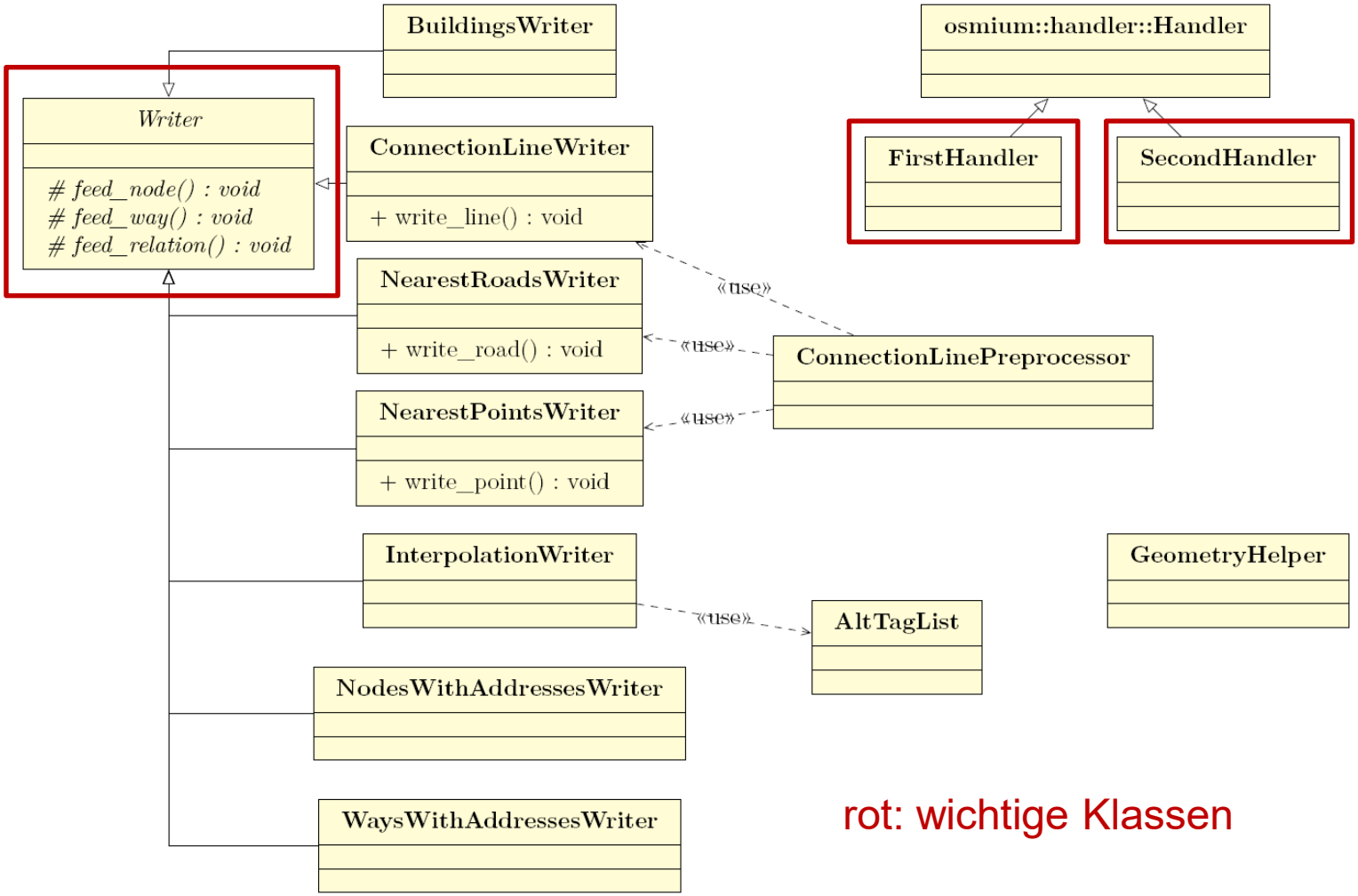
Erster Durchlauf:

- Sämtliche Node-Koordinaten speichern (`NodeLocationForWays` Handler)
- Geometrien aller Strassen mit name=… speichern
- IDs von Endpunkten von Interpolationslinien speichern

Zweiter Durchlauf:

- Zu Adressen zugehörige Strassen und nächster Punkt suchen
- Interpolationslinien auf Konsistenz prüfen
- Schreiben aller Spatialite-Tabellen

Implementierung - Klassendiagramm



rot: wichtige Klassen

Implementierung - Writer Klasse

`Writer`: Abstrakte Basisklasse

Pro SQLite-Tabelle eine Spezialisierung der `Writer` Klasse, z.B. `NodesWithAddressesWriter` und Implementation (einiger) der Methoden

- `feed_node()`
- `feed_way()`
- `feed_relation()`

In einfachen Fällen kann die Osmium-Callback Funktion direkt `feed_...()` aufrufen.

Implementierung - Nächste Strasse & Verbindungslinien

Wie findet man für eine gegebene Adresse (mit Position) die nächste Strasse und die kürzestes Verbindung?

Wichtigste Datenstruktur: `name2highways`

`std::multimap` die von einem Strassennamen auf einen oder mehrere Structs schliesst.

Der Struct enthält u.a.:

- Weg-Geometrie
- Ungefähre Position des Wegs

Implementierung - Nächste Strasse & Verbindungslinien

Übersicht:

1. Durchgang:

- Way Callbacks: Weg-Geometrien in `name2highways` ablegen

2. Durchgang:

- Node und Way Callbacks: Nächstliegende Punkte auf Strassen berechnen

Implementierung - Nächste Strasse & Verbindungslinien

Vorgehen zur Berechnung nächstliegender Punkte auf Strassen:

1. Aus der MultiMap alle Structs für den gesuchten Strassennamen ausgeben

2. Ungefähren Position verwenden: Liegt die Strasse in der Nähe? (Länge und Breite < 0.02° ? Entspricht ca. 2km am Äquator)

3. Mittels GDAL/OGR nächstliegende Strasse bestimmen (leider ohne Angabe des naheliegendsten Punktes)

4. Nächstliegenden Stützpunkt bestimmen

5. Nächstliegender Punkt auf den angehängten Segmenten bestimmen (Skalarprodukt).

Implementierung - Interpolationslinien

1. Durchgang:

- Way-Callbacks: IDs von Endpunkten von Interpolationslinien speichern

2. Durchgang:

- Node-Callbacks: Tags von Endpunkten zwischenspeichern
- Way-Callbacks: Interpolationslinien auf Konsistenz prüfen und Daten schreiben

Konsistenz heisst:

- Hausnummern sind tatsächlich Nummern
- Strassennamen, allenfalls PLZ, Ort, Land, etc. stimmen überein
- addr:interpolation=even ⇒ Hausnummern sind gerade, ebenso für addr:interplation=odd, =all
- …

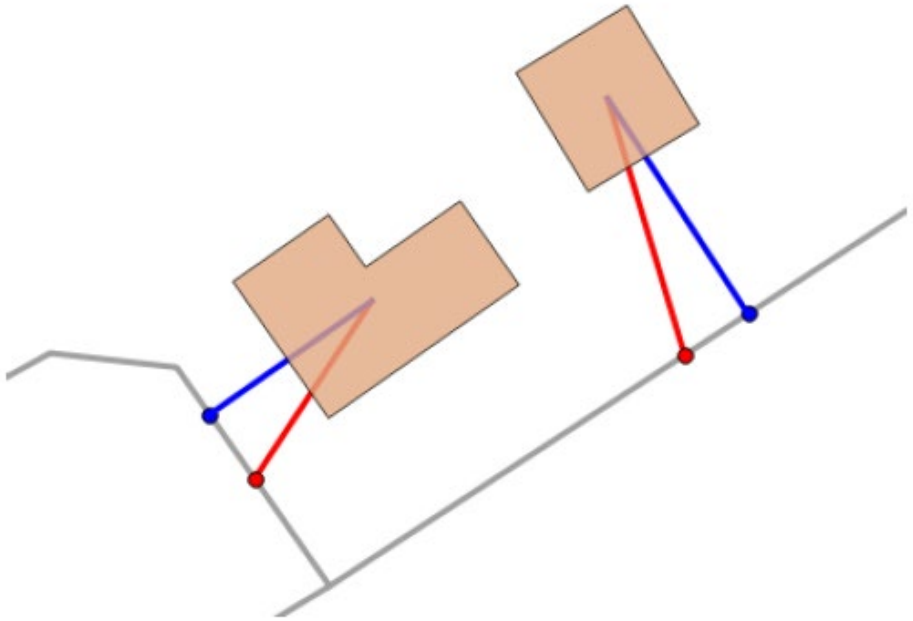
Implementierung - Fallstrick

Berechnung der nächsten Punkte auf Strassen muss in Mercator-Koordinaten (blau) erfolgen, sonst liegt die Verbindungslinie «schief» (rot).

WGS84 → Umwandlung in Mercator → Berechnung nächster Punkt → Rückumwandlung in WGS84 → Umwandlung in Mercator für Darstellung

Grundlegendes Problem:

Projektion kann nicht gleichzeitig längen- und winkeltreu sein.



Demo 1: Neuer OSM Inspector

http://tools.geofabrik.de/osmi/test.html

(Noch nicht offiziell freigeschaltet.)

Demo 2: Vergleich vorher / nachher

http://tools.geofabrik.de/wmsc/

Resultate - Ziel 1: Laufzeit verringern

	Bisherige Implementation (Europa)	Neue Implementation (Europa)	Neue Implementation (Planet)
Laufzeit	**25h 6min** + Import	**4h 51min**	**10h 35min**
Output-Grösse	3.9 GB	12.9 GB (unkompr.) 3.9 GB (kompr.)	48 GB (unkompr.) ~15 GB (kompr.)
RAM	8 GB	80 GB	148 GB

Reduktion der Laufzeit um den Faktor 5!

Resultate - Ziel 2: Objektorientierte Programmierung

Die Basis-Klasse `Writer` bietet eine einfache und modulare Möglichkeit, direkt aus der Callback-Funktion von Osmium Daten in die Spatialite-Datei zu schreiben.

Fazit

Erfolgreicher Umbau der Adress-View des OSM Inspectors.

Weltweite Ansicht kann realisiert und täglich aktualisiert werden.

Überprüfung von Adress-Daten ist nun auch für Mapper ausserhalb von Europa möglich.

Resultat ist hoffentlich eine bessere Qualität der Adress-Daten.

Vielen Dank für Ihre Aufmerksamkeit.