Interaktive benutzerdefinierte Symbolisierung von OpenStreetMap (OSM) Daten mit QGIS-Mapserver

Masterprojektarbeit
Geomatik und Planung
Frühlingssemester 2013

Autorin

Tabea Probst

probstta@student.ethz.ch

Leitung

Prof. Dr. Lorenz Hurni

Betreuung

Dr. Ionut Iosifescu-Enescu
Nadia Panchaud

Abgabedatum: 31. Mai 2013

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IKG
Institut für Kartografie
und Geoinformation

Titelbild offizielles Logo von OpenStreetMap (OpenStreetMap 2013)

I. Vorwort

Die vorliegende Masterprojektarbeit ist am *Departement für Bau, Umwelt und Geomatik (D-BAUG)* der *Eidgenössischen Technischen Hochschule Zürich (ETH)* im Rahmen des Masterstudiums *Geomatik und Planung* entstanden. Das Thema wurde am *Institut für Kartografie und Geoinformation (IKG)* ausgeschrieben und von Februar bis Mai 2013 bearbeitet. Die Arbeit steht unter der Leitung von PROF. DR. LORENZ HURNI und wurde von DR. IONUT IOSIFESCU-ENESCU und von NADIA PANCHAUD betreut.

An dieser Stelle möchte ich mich beim *IKG* für die Unterstützung und das Zurverfügungstellen der Infrastruktur, die für diese Arbeit unverzichtbar war, bedanken. Meinen Betreuern, DR. IONUT IOSIFESCU-ENESCU und NADIA PANCHAUD, danke ich herzlich für ihre kompetente Unterstützung und ihre hilfreichen Inputs. Sie sind mir bei Fragen und Unklarheiten stets zur Seite gestanden und haben mich mit Tipps und Ratschlägen unterstützt.

Ein Dankeschön richte ich auch an BARBARA BORN und RUTH PROBST für das Korrekturlesen sowie an alle anderen, hier nicht namentlich erwähnten Personen, welche durch Einbringen von guten Ideen und konstruktiver Kritik zum Gelingen dieser Arbeit beigetragen haben.

Tabea Probst

II. Zusammenfassung

Diese Arbeit befasst sich mit der interaktiven benutzerdefinierten Symbolisierung von OpenStreetMap-Daten. Als Grundlage dafür dient auf der einen Seite eine Anwendung, die den Benutzer befähigt, Swisstopo-Daten beliebig zu symbolisieren. Auf der anderen Seite stehen die Daten von OpenStreetMap. Diese von Usern erfassten Geodaten werden kostenlos zur freien Verfügung gestellt. Mit der vorliegenden Arbeit soll die Verschmelzung dieser beiden Grundlagen erreicht werden.

Um einen Eindruck von den zu verwendenden Daten zu erhalten, werden einige Informationen über deren Eigenschaften und Formate gesammelt. Es wird Software für die Verarbeitung solcher Daten gesucht und miteinander verglichen. Die Daten werden bezüglich ihrer Qualität untersucht und mit offiziellen Referenzdaten von Swisstopo verglichen. Dabei wird erkannt, dass die Daten teilweise fehlerhaft sind, das heisst über keine korrekte Topologie verfügen. Ausserdem wird festgestellt, dass das vorliegende Datenschema so nicht verwendet werden kann, sondern manipuliert werden muss.

Die Aufbereitung der Daten wird aufgrund der Erkenntnisse aus den vorangehenden Analysen durchgeführt. Die Daten müssen so angepasst werden, dass sie von der Anwendung verwendet und dargestellt werden können. Dafür werden die benötigten Informationen aus den Originaldaten extrahiert und in ein Schema gespeichert, das nötigen Voraussetzungen erfüllt. Da es sich bei der Anwendung um eine Kartenanwendung mit mehreren Zoomstufen handelt, wird ein zweiter, generalisierter Datensatz erstellt. Aufgrund der bereits erwähnten Topologie-Fehler gestaltet sich dieser Schritt jedoch eher schwierig. Im Rahmen dieser Arbeit wurde keine zufriedenstellende Lösung für dieses Problem gefunden.

Nachdem die geforderten Anpassungen für das Ändern der Datengrundlage der Anwendung gemacht wurden, werden die Daten automatisch in der Benutzeroberfläche visualisiert. Es werden verschiedene Symbolisierungssätze erstellt, um dem Benutzer eine Auswahl an vordefinierten Styles zur Verfügung zu stellen. Ein Beispiel dafür ist der beim Laden der Anwendung dargestellte Style OpenStreetMap. Er symbolisiert die Daten ähnlich wie sie auf openstreetmap.org zu sehen sind.

An der Benutzeroberfläche werden einige Veränderungen vorgenommen, um vor allem das visuelle Erscheinungsbild dieser Oberfläche an dasjenige von OpenStreetMap anzupassen. Zusätzlich werden die einzelnen Elemente so angeordnet, dass der Ablauf für das Symbolisieren der Daten klarer ersichtlich ist. Dem Benutzer soll eine Anwendung angeboten werden, mit der er die Daten einfach und nach seinem Gutdünken symbolisieren kann.

III. Inhaltsverzeichnis

IV. Abbildungsverzeichnis

1. Einleitung

1.1. Einführung und Problemsituation

Wie der Titel «Interaktive benutzerdefinierte Symbolisierung von OpenStreetMap (OSM) Daten mit QGIS-Mapserver» bereits sagt, wird mit der vorliegenden Arbeit eine Möglichkeit geschaffen, OpenStreetMap-Daten interaktiv und benutzerdefiniert zu symbolisieren. Damit die Interaktivität gewährleistet werden und der Benutzer die Symbolisierung frei wählen kann, wird eine auf einem internetbasierten Kartendienst basierende Benutzeroberfläche verwendet. Für solche Kartendienste müssen verschiedene Clients und Server miteinander kommunizieren. Dazu braucht es offene Standards, in diesem Fall die sogenannten Open Geospatial Consortium (OGC) Standards WMS und SLD. Das Institut für Kartografie und Geoinformation (IKG) der ETH Zürich entwickelt kartografische Erweiterungen zu diesen Standards. Diese werden auf der Serverseite in der Software QGIS Mapserver implementiert und auf der Clientseite mit einem OpenLayers-basierten JavaScript Client visualisiert. Damit können kartografische Projekte realisiert werden.

In früheren Arbeiten wurde bereits gezeigt, dass es möglich ist, thematische Karten mittels Webservices und SLD zu generieren. Zudem hat Christian Lorenz im Rahmen seiner Masterarbeit[1] eine Benutzeroberfläche erstellt, die es dem Benutzer ermöglicht, eine eigene Symbolisierung für Swisstopo-Daten zu erstellen oder bestehende Symbolisierungen anzupassen (→ Kapitel 2.6). Darauf aufbauend werden in dieser Arbeit weitere Möglichkeiten erarbeitet, um die benutzerdefinierte Symbolisierung zu erweitern.

Im August 2004 registrierte Steve Coast die Domain openstreetmap.org und rief damit das Projekt OpenStreetMap ins Leben. Zwei Jahre später wurde die OpenStreetMap Foundation gegründet, die dazu ermutigen soll, das Wachstum, die Entwicklung und die Verteilung von freien Geodaten zu fördern (OpenStreetMap 2013). Mit diesem Projekt wurden zahlreiche neue Möglichkeiten geschaffen. Die von Usern erfassten Daten können von jedermann gratis heruntergeladen und unter Angabe der entsprechenden Lizenz für eigene Anwendungen verwendet werden. Diese Tatsache führte zu der Idee, den Ansatz der benutzerdefinierten Symbolisierung auf OSM-Daten anzuwenden und somit zu ermöglichen, dass der Benutzer die Daten wie gewünscht symbolisieren kann.

1.2. Zielsetzung

Das Hauptziel dieser Arbeit ist es, die benutzerdefinierte Symbolisierung von OSM-Daten zu ermöglichen und verschiedene kartografische Möglichkeiten zu finden, diese darzustellen. Deshalb sollen verschiedene Styles (zum Beispiel nach dem Vorbild von OSM, Swisstopo oder der deutschen Landeskarte) erstellt werden. Zusätzlich werden folgende Zwischenziele formuliert, die das Erreichen des Hauptziels unterstützen sollen.

- Finden von Lösungsansätzen für den Import von OSM-Daten

[1] Lorenz, C. (2013): Interaktive benutzerdefinierte Symbolisierung von Vektordaten mittels erweiterten OGC-Standards. Masterarbeit, ETH Zürich, Zürich, Schweiz

- Auswertung und Verarbeitung von OSM-Daten
- Darstellung der Karten in mindestens zwei Massstäben, ähnlich Vector25 und Vector200 von Swisstopo
- Anpassung der in der Anwendung implementierten Werkzeuge an die OSM-Daten und Verbesserung der Benutzerfreundlichkeit des GUI

Um einen Vergleich mit dem Prototyp von Christian Lorenz und somit mit den Referenzdaten von Swisstopo zu ermöglichen, soll das gleiche Testgebiet (Region Andermatt) bearbeitet werden.

1.3. Inhaltübersicht

Nach der Einleitung mit einer Einführung in die Problemsituation und der Definition der Ziele wird in *Kapitel 2* auf die Grundlagen und die verwendeten Methoden eingegangen. Zuerst werden die Grundlagedaten beschrieben, dann wird eine Übersicht über die vorhandene Software für die Verarbeitung von OSM-Daten gegeben. Zusätzlich befasst sich dieses Kapitel mit den Themen «Daten-Qualität» und «Benutzerfreundlichkeit» im Allgemeinen. Abgerundet wird dieser Teil mit der Vorstellung der bestehenden Benutzeroberfläche und einer Beschreibung der wichtigsten, in dieser Arbeit verwendeten Technologien.

Kapitel 3 dient dazu, eine Übersicht über das Vorgehen zu liefern. Es fasst die zu unternehmenden Schritte zusammen und stellt sie übersichtlich dar. Der nächste Abschnitt, *Kapitel 4*, widmet sich den Daten. Es wird aufgezeigt, wie die Daten beschafft werden können, welche Erkenntnisse aus der Analyse der Daten gewonnen werden und wie die Daten schliesslich aufbereitet werden. Das darauf folgende *Kapitel 5* befasst sich mit der Anwendung. Zuerst wird gezeigt, wie verschiedene vordefinierte Styles erstellt werden können und wie die Umsetzung davon aussehen kann. Danach wird auf die Benutzeroberfläche selbst eingegangen. Es geht darum, wie diese Oberfläche verbessert und benutzerfreundlicher gemacht werden kann. Schliesslich wird deren Funktionalität erläutert und Beispiele für das benutzerdefinierte Symbolisieren der Daten geliefert. In *Kapitel 6* werden die erreichten Resultate vorgestellt und kritisch beurteilt. Zum Schluss wird in *Kapitel 7* ein Fazit gezogen und ein Ausblick auf weiterführende Arbeiten gewagt.

2. Grundlagen und Methoden

2.1. Grundlagedaten

Als Datengrundlage für diese Arbeit dienen die Daten von OpenStreetMap (OSM). Diese stehen unter der « Open Database License» (ODbL) von OpenDataCommons. Das bedeutet, die Daten sind frei und können von jedermann verwendet, bearbeitet, kopiert und weiterverbreitet werden, solange OpenStreetMap und seine Mitwirkenden als Quelle angegeben werden, zum Beispiel mit «© OpenStreetMap contributors». Die Kartografie in den Kartenkacheln sowie die Dokumentation unterstehen einer anderen Lizenz, der sogenannten «Creative Commons Attribution-ShareAlike 2.0 license» (CC-by-SA) (OpenStreetMap 2013).

2.2. Verarbeitungssoftware

Dieses Kapitel soll einen Überblick über bestehende Software für die Verarbeitung von OpenStreetMap-Daten verschaffen. Alle vorgestellten Tools sind gratis verfügbar. Sie sind, je nach Popularität, besser oder weniger gut dokumentiert und teilweise etwas kompliziert in der Anwendung.

2.2.1. Osmconvert

Osmconvert wird verwendet, um OSM-Daten zu verarbeiten und in verschiedene Formate umzuwandeln. Der Umfang an Funktionen von Osmconvert ist einiges kleiner als derjenige von Osmosis (siehe Kapitel 2.2.3). Es erledigt dafür Standardaufgaben schneller und besitzt einige nützliche Spezialfunktionen. Zum Beispiel kann ein Grenzpolygon erstellt werden, um die Daten auf ein bestimmtes Gebiet zuzuschneiden oder es können alle Linien und Relationen (siehe Kapitel 4.2.1) in Punkte umgewandelt werden. Zudem ist die Funktion «drop-author» sehr nützlich. Sie löscht alle Informationen über den User (user name, user id, changeset, timestamp), welche für viele Anwendungen, zum Beispiel für diese Arbeit, nicht benötigt werden. Mit einer weiteren Spezialfunktion werden verschiedene statistische Daten ausgegeben, was eine statistische Analyse einer ganzen Datei ermöglicht. Es gibt noch andere Funktionen, die vor allem Umformatierungen der Daten erlauben. Der Nachteil dieser Software ist, dass damit nicht direkt auf eine Datenbank zugegriffen werden kann, das heisst Osmconvert verfügt über keine Datenbank-Schnittstelle. Dieser Schritt müsste mit einem anderen Programm ausgeführt werden (OpenStreetMap 2013).

2.2.2. Osm2pgsql

Mit dem kommandozeilenbasierten Hilfsprogramm Osm2pgsql können OSM-Daten direkt in PostgreSQL Datenbanken geschrieben werden. Osm2pgsql wird von einer Community von OSM-Beteiligten unterhalten (OpenStreetMap 2013). Es unterstützt zwei verschiedene Datenbankschemas, das eine ist optimiert für das Rendering mit Mapnik[2], das andere für das

[2] Mapnik ist ein Open Source Werkzeugsatz, der das Rendering von Karten ermöglicht (OpenStreetMap 2013).

Geocoding mit Nominatim[3]. Wird das Programm mit den Defaultwerten ausgeführt, werden vier Tabellen (point, line, roads und polygon) erstellt. Um dies zu erreichen, wird ein XML-parser auf das Inputfile angewendet, welches die Knoten, Linien und Relationen prozessiert (GitHub 2013).

2.2.3. Osmosis

Osmosis ist eine Java-Applikation, die über die Kommandozeile bedient wird und mit der OSM-Daten verarbeitet werden können. Es besteht zudem die Möglichkeit, Osmosis als Bibliothek in andere Java-Applikationen zu integrieren. Die Applikation enthält Komponenten, die das Lesen und Schreiben von Daten aus und in diverse File-Formate ermöglicht. Daten können direkt in eine Datenbank geschrieben, von dort gelesen und in ein File geschrieben werden. Osmosis beinhaltet, wie Osmconvert, zusätzliche Funktionen, um die Daten zu bearbeiten. Es können Daten innerhalb einer Bounding Box oder eines Polygons extrahiert werden, es können Änderungssätze (changesets) eingespielt oder Daten sortiert werden (OpenStreetMap 2013).

Bevor die Daten in eine Datenbank geschrieben werden können, muss dort das Schema vorbereitet werden. Dies geschieht mit einem speziellen Programm, das zusammen mit Osmosis heruntergeladen wird. Es gibt zwei Versionen davon, die eine ist «Pgsimp_schema_0.6.sql», welche vor allem bei älteren Datenbanken verwendet wird, die zweite heisst «Pgsnapshot_schema_0.6.sql». Letztere bietet insbesondere bei den Tags neue Möglichkeiten, da diese in einer einzelnen Spalte mit dem Namen «hstore» gespeichert werden (OpenStreetMap 2013). Bei der in dieser Arbeit verwendeten Datenbank handelt es sich um einen etwas älteren Typ, weswegen das erste Schema verwendet wird. In Abbildung 1 ist dieses automatisch generierte Schema in Form eines UML-Diagramms abgebildet.

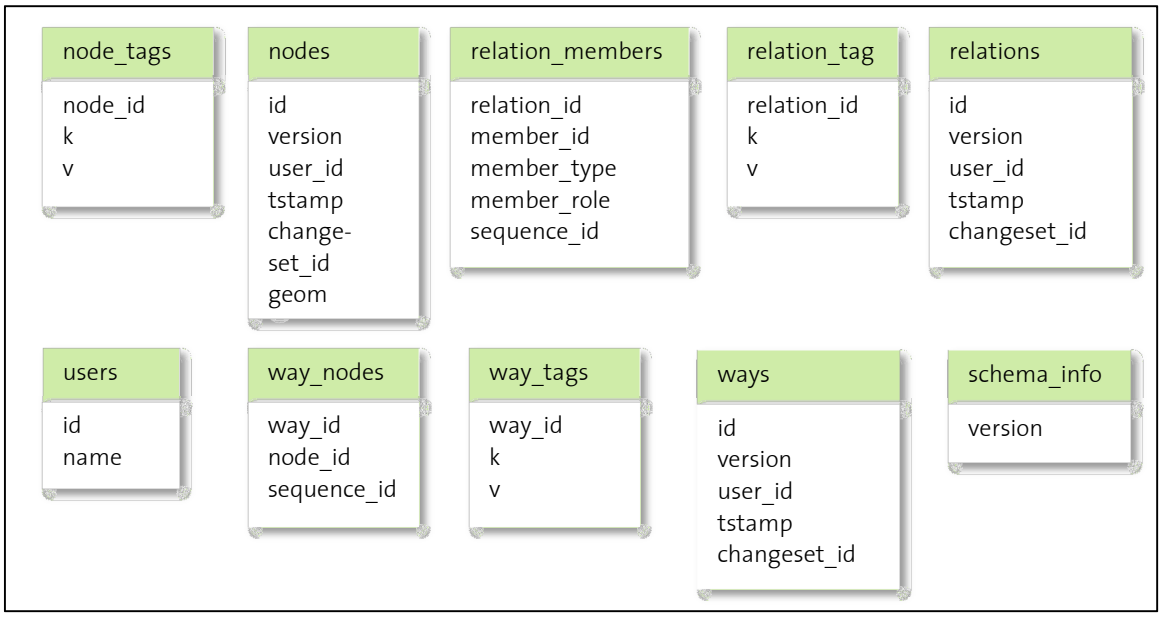


Abbildung 1: Datenbank Schema, generiert mit Osmosis

Das Spezielle an diesem Schema ist, dass sämtliche Informationen über die Geometrie in der Tabelle «nodes» gespeichert sind. Das heisst auch die Stützpunkte der Linien und Flächen

[3] Nominatim ist ein Tool, mit dem über Name und Adresse nach einem Objekt gesucht und eine künstliche Adresse dafür erzeugt werden kann (OpenStreetMap 2013).

sind in dieser Tabelle enthalten. In den restlichen Tabellen werden lediglich die Verknüpfungen und Zugehörigkeiten über die ID dieser «nodes» gespeichert. Es ist nicht ganz klar, wie ein solches Schema für die Weiterverwendung der Daten in dieser Arbeit benutzt werden kann.

2.2.4. Imposm (Mapbox)

Imposm ist ein Import-Tool, welches OSM-Daten in Form von XML- oder PBF-Files liest und sie in PostgreSQL/PostGIS Datenbanken schreibt. In der Imposm Dokumentation (2011) steht, dass dieses Tool optimiert ist für das Rendering von Daten, zum Beispiel für WMS Services. Imposm ist als Open Source Programm unter der Apache Software License 2.0 realisiert. Der Anwendungsbereich von Imposm ist begrenzt. Es werden zum Beispiel nur PostGIS Datenbanken unterstützt und differentielle Updates sind zurzeit noch nicht möglich. Dafür ist Imposm sehr effizient mit der Speicherung und einfach erweiterbar. Die Tags können gefiltert werden, sodass nur diejenigen Daten importiert werden, die später auch gerendert werden sollen. Ein Beispiel für die Verwendung von Imposm ist Mapbox. Diese Webapplikation wurde erstellt, um zu ermöglichen, dass jedermann eine schöne, individuelle Karte erstellen und publizieren kann. Basierend auf OSM-Daten, bietet Mapbox zahlreiche Möglichkeiten für eine individualisierte Darstellung von Karteninhalten. Es existiert sogar eine JavaScript API, um die Karte einfach publizieren zu können (MapBox 2013).

2.2.5. QGIS

In QGIS (→ Kapitel 2.3.5) kann ein Plugin installiert werden, mit dem die OSM-Daten einfach heruntergeladen werden können. Deshalb eignet sich QGIS sehr gut, um einen ersten Eindruck von den Daten zu erhalten. Die Daten können als Shape-Files gespeichert und bearbeitet werden. Unnötige Attribute können einfach entfernt werden. Der Nachteil ist, dass der ganze Vorgang manuell abläuft. Dies ist für eine kleine Region wie Andermatt kein Problem, wird jedoch für grössere Gebiete, wie zum Beispiel die ganze Schweiz, schnell mühsam und aufwändig.

Die GIS-Software bietet die Möglichkeit, die Daten direkt in die Datenbank zu schreiben. Dies geschieht mit einem weiteren Plugin, entweder mit «Shape to PostgreSQL» oder mit dem «PostGIS-Manager». Die beiden Varianten führen zum gleichen Ergebnis: in der Datenbank wird für jeden hinzugefügten Layer eine Tabelle erstellt.

2.3. Verwendete Technologien

Im diesem Kapitel sind die wichtigsten Technologien aufgeführt, welche für diese Arbeit verwendet werden. Zur Übersicht zeigt das folgende Diagramm, wie die verschiedenen Technologien miteinander verknüpft sind. Während die Schritte in QGIS alle nur einmal ausgeführt werden, werden die auf der rechten Seite abgebildeten Abläufe immer wieder durchlaufen. Dies geschieht jedes Mal dann, wenn der Benutzer eine entsprechende Eingabe tätigt.

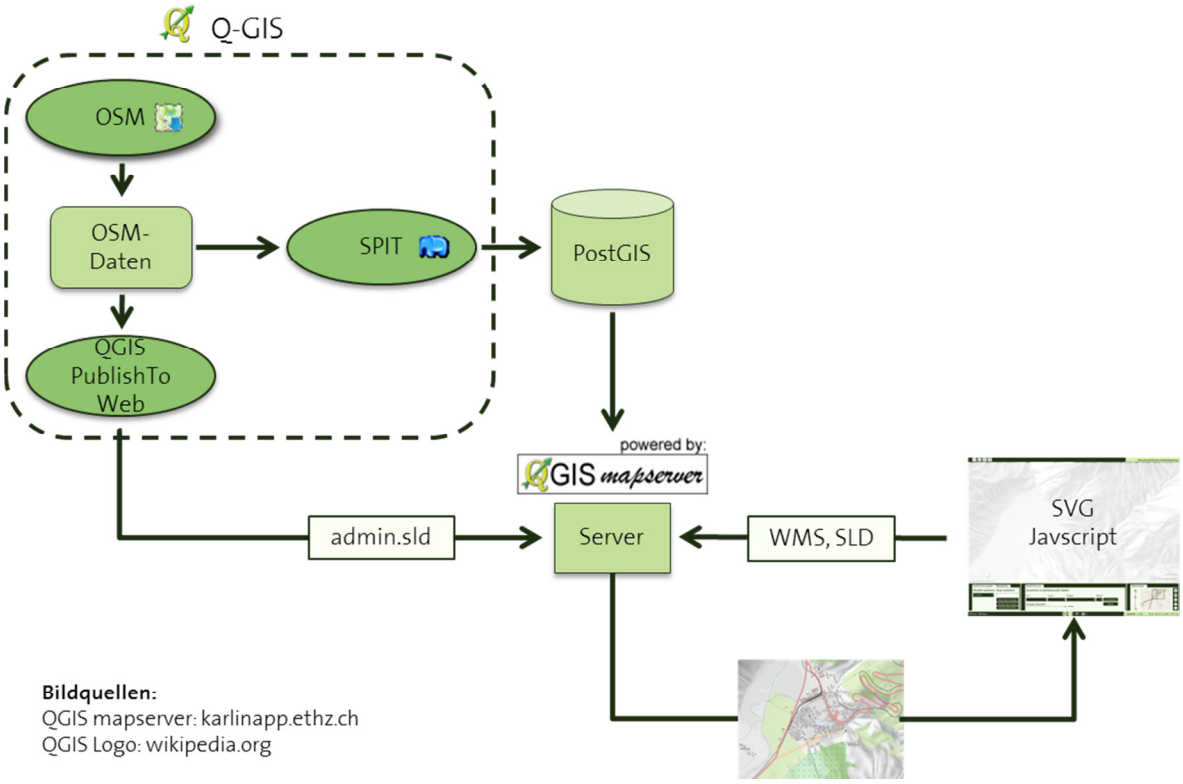


Abbildung 2: Verknüpfung der verwendeten Technologien und Programmen

2.3.1. OGC-Standards

Das Open Geospatial Consortium (OGC) ist ein internationales Konsortium, bestehend aus Firmen, Behörden und Universitäten, das zum Ziel hat, öffentlich verfügbare Schnittstellen-Standards zu entwickeln. Das OGC unterstützt kompatible Lösungen für die Nutzung von Geodaten im Netz, für «wireless and location based services» sowie für die allgemeine Informationstechnologie (OGC 2013). Für die Realisierung dieser Arbeit werden die Standards WMS (Web Map Service) und SLD (Styled Layer Descriptor) vom OGC verwendet.

WMS

Der WMS ist ein Service, der eine einfache HTTP-Schnittstelle zur Verfügung stellt, um georeferenzierte Kartenbilder von einer Datenbank mit räumlichen Daten zu beziehen. Dafür muss eine Anfrage an den Server geschickt werden, die den gewünschten Ausschnitt und die darzustellenden Layer definiert. Als Antwort schickt der Server ein oder mehrere Rasterbilder im gewünschten Format (zum Beispiel PNG oder JPEG) zurück. Es können drei verschiedene Typen von Anfragen gesendet werden. Die erste, getMap, wird benötigt, um die Rasterkarte zu erhalten. Dies ist die Anfrage, welche in dieser Arbeit mit WMS-Anfrage bezeichnet und verwendet wird. Mit getCapabilities können Metadaten abgefragt werden, das heisst Informationen über den Service oder über die vorhandenen Layer. GetFeatureInfo liefert Informationen über einzelne, in der Karte dargestellte Features (OGC 2013).

SLD

Der «Styled Layer Descriptor» ist eine Erweiterung des Standard-WMS. Er spricht das Bedürfnis der Benutzer an, die Symbolisierung der mit einem WMS abgerufenen Geodaten selbst zu bestimmen. Um solche Darstellungsregeln definieren zu können, braucht es eine

Sprache, die sowohl für den Client als auch für den Server verständlich ist. Der SE (Symbology Encoding) Standard stellt diese auf XML basierende Sprache zur Verfügung während SLD dessen Anwendung auf einen WMS ermöglicht (OGC 2013).

Der in dieser Arbeit verwendete WMS-Server des IKGs enthält erweiterte SLD/SE Standard, welche die Darstellung kartografisch sauberer Karten ermöglichen.

2.3.2. Scalable Vector Graphics (SVG)

Scalable Vector Graphics ist ein Standard des «World Wide Web Consortium» (W3C), der von der W3C SVG Arbeitsgruppe entwickelt und unterhalten wird. SVG ist eine auf XML basierende Sprache, mit welcher zweidimensionale Vektorgrafiken beschrieben und in Webbrowsern dargestellt werden können (Dahlström, et al. 2011). Für die meisten Browser braucht es dafür kein zusätzliches Plugin. Der Vorteil von XML ist, dass der Code für den Benutzer verständlich ist und somit sehr einfach von Hand verändert werden kann. Für die etwas komplizierteren Formen können jedoch auch SVG-Editoren zu Hilfe gezogen werden.

Abbildung 3: SVG

2.3.3. JavaScript

JavaScript ist eine Programmiersprache, welche es ermöglicht, interaktive und dynamische Webseiten zu erstellen. Sie kann in HTML eingebunden werden und wird vom Browser ausgeführt. JavaScript erlaubt das Zugreifen auf einzelne Objekte einer Webseite und ermöglicht damit das Ändern dieser Objekte. Da ein SVG-Element ein solches Objekt ist, eignet sich JavaScript hervorragend für die in dieser Arbeit verwendete Benutzeroberfläche.

2.3.4. PostGIS

PostGIS ist eine Erweiterung für objekt-relationale PostgreSQL Datenbanken. Sie ermöglicht das Speichern von geografischen Objekten und umfasst eine grosse Anzahl an Funktionen für das Erstellen von räumlichen SQL-Abfragen (PostGIS 2013).

2.3.5. Quatum GIS

Abbildung 4: QGIS Logo (Wikipedia 2013)

Quantum GIS (QGIS) ist eine Open Source GIS-Software, die zahlreiche Vektor-, Raster- und Datenbankformate unterstützt und Funktionen zur Analyse und Manipulation solcher Daten enthält. Es ist ein offizielles Projekt der Open Source Geospatial Foundation (OSGeo), lizenziert unter der GNU General Public Lizenz. Die aktuelle Version ist QGIS 1.8.0 (Quantum GIS 2013).

QGIS bietet die Möglichkeit, seine Funktionalität mit zahlreichen Plugins zu erweitern. Drei davon sind für diese Arbeit von Bedeutung. Das erste ist das OSM-Plugin, welches das direkte Herunterladen von OSM-Daten ermöglicht. Die nächste Erweiterung ist das Plugin SPIT (Shapefile to PostGIS Import Tool). Damit können Shapefiles direkt in eine PostGIS Datenbank importiert werden. Das letzte verwendete Plugin ist QGISPublishToWeb. Diese Erweiterung hat den Nachteil, dass es nur in einer älteren Version von QGIS verwendet werden kann. Diese wird deshalb beim Herunterladen des

Plugins gleich mitgeliefert. Mit QGISPublishToWeb wird ein admin.sld File erstellt, das alle sich im QGIS-Projekt befindenden Layer enthält und direkt auf den Server geladen werden kann. Via WMS kann dann auf die Daten zugegriffen werden.

2.3.6. QGIS Mapserver

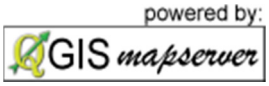


Abbildung 5: QGIS Mapserver (QGIS Mapserver 2009)

Der QGIS Mapserver ist eine Open-Source WMS-Implementierung, welche erweiterte kartografische Features enthält. Mit diesen Erweiterungen können sowohl Vektor- als auch Rasterdaten unter Berücksichtigung kartografischer Regeln visualisiert werden. Das heisst, es lassen sich Karten generieren, die eine höhere kartografische Qualität aufweisen, als die mit dem Standard-SLD erstellten Karten (QGIS Mapserver 2009).

2.4. Daten-Qualität

Die erste Frage, die sich stellt, wenn man sich mit der Qualität von Daten auseinandersetzt, ist die Frage, was Daten-Qualität bedeutet. Wie Ather (2009) in seiner Masterarbeit aufzeigt, kann die Qualität von Daten hinsichtlich subjektiver oder quantitativer Merkmale untersucht werden. Auf die subjektiven Merkmale, das heisst auf die Frage, wie nützlich bestimmte Daten für gewisse Zwecke sind, wird hier nicht näher eingegangen. Im Zusammenhang mit dieser Arbeit sind die quantitativen Merkmale viel wichtiger. Diese werden unterteilt in räumliche, zeitliche und thematische Genauigkeit sowie Vollständigkeit und logische Konsistenz. Während die räumliche Genauigkeit die Lagegenauigkeit eines Objekts im zwei- oder dreidimensionalen Raum beschreibt, befasst sich die zeitliche Genauigkeit mit der Richtigkeit der temporalen Attribute und deren Darstellung in korrekter zeitlicher Abfolge. Die thematische Genauigkeit betrachtet inhaltliche Dinge wie die korrekte Attributierung von Objekten verglichen mit der Wirklichkeit oder die richtige Zuordnung von Ortsnamen. Bei der Vollständigkeit geht es darum, zu untersuchen, ob Objekte, die in der Wirklichkeit vorhanden sind, in der Karte erfasst sind oder ob Objekte, die in der Karte dargestellt sind, auch wirklich existieren. Die letzte Kategorie beinhaltet unter anderem die topologische Konsistenz.

OpenStreetMap-Daten oder «Volunteered Geographical Information (VGI)», wie solche von Usern erfasste Daten von Goodchild (2007) genannt werden, wurden noch nicht sehr oft bezüglich dieser verschiedenen Qualitätsmerkmale untersucht. Als einer der Ersten befasste sich Haklay (2010) mit diesem Thema. In seinem Paper werden die OSM-Daten mit denjenigen des Ordnance Survey (OS) von England verglichen. Die Analyse bezieht sich vor allem auf das Gebiet von London und zeigt, dass OSM-Daten ziemlich genau sein können. Durchschnittlich liegen die Daten in einem Bereich von sechs Metern bezüglich der OS-Daten und die «Motorways» der beiden Datensätze habe eine Überlappung von etwa 80%.

Auch Ather (2009) befasst sich mit der Qualität von OSM-Daten. Zusätzlich zu den «Motorways» werden weitere Strassen-Objekte verglichen. Auch hier beträgt die Überlappung circa

80% und die Lagegenauigkeit der Daten wird, verglichen mit der Ordnance Survey MasterMap[4], als sehr gut eingestuft.

2.5. Benutzerfreundlichkeit

Wie Nielson (1993) in seinem Buch «Usability Engineering» schon vor 20 Jahren festgestellt hat, ist der Begriff «benutzerfreundlich» nicht unbedingt passend. Der Computer oder eine Maschine im Allgemeinen muss nicht freundlich zum Benutzer sein, sondern nicht im Weg sein, wenn der Benutzer seine Arbeit erledigen will. Zudem können die Bedürfnisse eines Users nicht in einer Dimension beschrieben werden, in dem Systeme «benutzerfreundlich» sind oder nicht. Die Bedürfnisse verschiedener Benutzer unterscheiden sich, das heisst ein System, das für den einen «benutzerfreundlich» ist, erscheint einem anderen unverständlich. Nielson verwendet deshalb das Wort «usability», was auf Deutsch so viel heisst wie Benutzbarkeit oder Bedienbarkeit.

Ebenfalls mit «Usability Engineering» befassen sich Richter und Flückiger (2010). Dieses Gebiet befasst sich mit der Sicht des Benutzers und deren systematische Einbindung in die Entwicklung von Software. Im engeren Sinne wird «Usability» oft als Qualitätskriterium für eine Benutzeroberfläche verwendet. Einzelne Kriterien sind zum Beispiel die Anordnung von Bedienelementen, die Verständlichkeit der Bezeichnungen und die Anzahl Klicks, die für das Erreichen eines bestimmten Ziels benötigt wird. Es ist jedoch mehr als das. Die Benutzbarkeit einer Anwendung muss immer im Zusammenhang mit seiner Verwendung betrachtet werden. Ein Benutzer muss das Produkt einfach erlernen, effizient anwenden und damit seine Aufgabe zufriedenstellend erfüllen können. Ist dies der Fall, weist das Produkt eine hohe «Usability» auf.

Eine andere Definition liefert die ISO-Norm 9241-11. Laut dieser Norm bezeichnet «Usability» das Ausmass, in dem ein Produkt von einem bestimmten Benutzer in einem bestimmten Anwendungskontext verwendet werden kann, um gewisse Ziele mit Effektivität, Effizienz und Befriedigung zu erreichen (Usability Net 2006).

Zusammenfassend kann gesagt werden, dass es bei der Benutzerfreundlichkeit nicht nur um das Produkt selber geht, sondern auch um die Benutzer. Das heisst, dass es für das Erstellen einer Software wichtig ist, die Zielgruppe zu kennen und die Benutzeroberfläche deren Fähigkeiten und Wünschen anzupassen.

2.6. Vorliegende Benutzeroberfläche

Wie in Kapitel 1.1 bereits erwähnt, liegt aus einer vorherigen Arbeit eine Benutzeroberfläche vor, welche die benutzerdefinierte Symbolisierung von Swisstopo-Daten ermöglicht. Abbildung 7 auf Seite 11 zeigt, wie diese Benutzeroberfläche aufgebaut ist.

Der Bereich mit der Nummer fünf enthält eine Übersichtskarte vom ganzen Testgebiet sowie sämtliche Werkzeuge für die Kartennavigation, das heisst Zoom- und Pan-Tools. Im Hauptkartenfenster werden die Daten dargestellt. Als Hintergrund steht einerseits ein Reli-

[4] OS MasterMap ist eine kontinuierlich aktualisierte Datenbank, welche verschiedene Layer mit unterschiedlichen Informationen enthält. Zum Beispiel gibt es einen Layer mit Adressen, einen mit dem Verkehrsnetz oder einen mit der Topographie. (Ordnance Survey 2013)

ef, andererseits eine Pixelkarte zur Verfügung. Diese können im Fenster drei aktiviert beziehungsweise deaktiviert werden. Es bietet sich zudem die Möglichkeit, deren Opazität zu ändern. Beim Eingabefenster mit der Nummer zwei können ein vordefinierter Symbolisierungssatz ausgewählt oder ein neuer Satz erstellt werden. Es steht bisher nur eine Symbolisierung zur Auswahl. Diese ist der Darstellung von Daten in Schweizer Karten nachempfunden und trägt deshalb den Namen «Swisstopo». Beim Hinzufügen eines neuen Satzes kann dessen Name vom Benutzer gewählt und im Eingabefeld («Neu») eingetippt werden. Nach dem Erstellen des neuen Satzes sind keine Daten mehr sichtbar. Diese erscheinen erst wieder, wenn ihnen eine Symbolisierung zugewiesen wird. Dafür kann im Fenster vier der zu symbolisierende Layer ausgewählt und mit den im Bereich eins vorhandenen Werkzeugen eine Darstellung zugewiesen werden.

Technisch gesehen basiert diese Anwendung auf einer 3-Schichten Architektur. Die Präsentationsschicht, welche die Daten darstellt und die Schnittstelle zum Benutzer bildet, wird repräsentiert durch das Graphical User Interface (GUI). Der QGIS-Mapserver bildet die logische oder mittlere Schicht und die Datenbank stellt die Datenschicht dar (Abbildung 6).

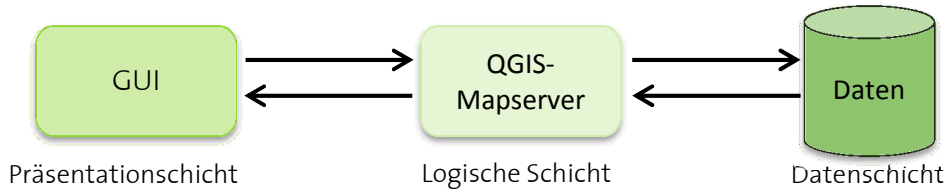


Abbildung 6: 3-Schichten Architektur

Konkret heisst das, dass sämtliche, für die Darstellung wichtigen Werte aus den Eingabefenstern abgegriffen, ein SLD erstellt und ein WMS-Request an den Server geschickt werden. Dieser verarbeitet die Anfrage, symbolisiert die Daten entsprechend des SLDs und sendet sie in Form eines PNGs zurück an die Benutzeroberfläche, wo sie dargestellt werden.

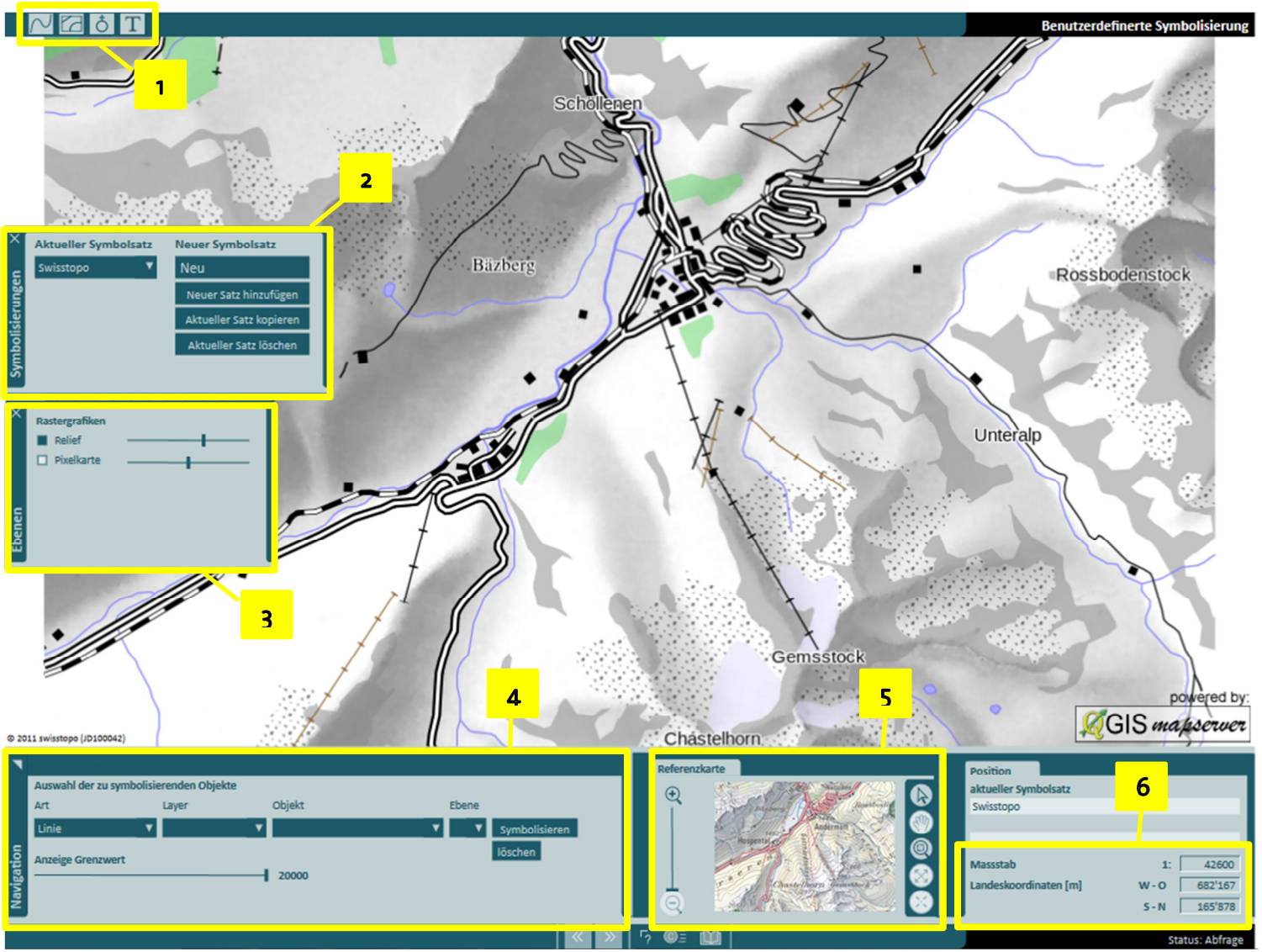


Abbildung 7: Ausgangslage für die Benutzeroberfläche (erstellt von Christian Lorenz)

3. Vorgehen

Dieses Kapitel soll einen Überblick über die verschiedenen, im Laufe dieser Arbeit unternommenen Schritte verschaffen. Auf die detaillierte Umsetzung wird in Kapitel 4 und 5 eingegangen.

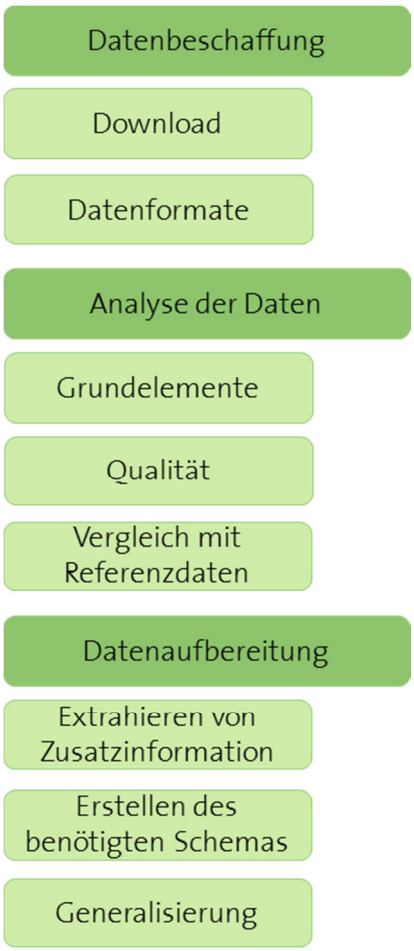


Abbildung 8: Vorgehen Datenvorbereitung

Der erste Teil dieser Arbeit widmet sich den OpenStreetMap-Daten. Damit soll ein erster Eindruck und schliesslich ein besseres Verständnis dieser freien Daten erlangt werden. Als erstes werden die Daten von OpenStreetMap über QGIS heruntergeladen. Um verschiedene Aspekte der Daten betrachten und analysieren zu können, werden Informationen über die Struktur und über die existierenden Formate der Daten zusammengetragen. Anschliessend wird, um einen besseren Eindruck von den Daten zu erhalten, deren Qualität analysiert und ein Vergleich mit Referenzdaten von Swisstopo angestellt. Basierend auf den Erkenntnissen aus dieser Analyse wird der nächste Schritt in Angriff genommen: die Aufbereitung der Daten für die Visualisierung in der Benutzeroberfläche. Zuerst werden die für die Anwendung benötigten Informationen aus den Daten extrahiert und in eine brauchbare Form gebracht. Da es sich bei der vorliegenden Anwendung um eine Kartenanwendung handelt und verschiedene Zoomstufen vorhanden sind, wird neben dem originalen Datensatz ein zusätzlicher, generalisierter Satz benötigt. Das heisst, ein weiterer Teil der Aufbereitung ist die automatische Generalisierung der Daten.

Im zweiten Teil geht es darum, die nun in der richtigen Form vorhandenen Daten zu verwenden und deren interaktive, benutzerdefinierte Symbolisierung zu ermöglichen. Dafür muss zuerst die Datengrundlage für die Benutzeroberfläche geändert werden. Anschliessend können den Daten, welche nun in der Anwendung visualisiert werden, Symbolisierungen zugewiesen werden. Dafür werden verschiedene vordefinierte Styles erstellt, die dem Benutzer zur Auswahl stehen. Am Schluss wird die Benutzeroberfläche insofern angepasst, dass sie einerseits farblich zu OpenStreetMap passt, andererseits eine klarere Anordnung der Eingabefenster besitzt.

4. Datenvorbereitung

Dieses Kapitel befasst sich genauer mit den OSM-Daten. Es werden die Möglichkeiten zum Download aufgezeigt, die Datenformate und -elemente beschrieben, die Qualität der Daten analysiert und ein Vergleich mit Referenzdaten von Swisstopo angestellt. Neben diesem eher analytischen Teil geht es im zweiten Teil um das Aufbereiten der Daten. Dabei wird gezeigt, wie die Daten vom ursprünglichen Schema in eine für die Anwendung brauchbare Form gebracht werden können.

4.1. Datenbeschaffung

4.1.1. Download

Es gibt verschiedene Möglichkeiten, die Daten zu beziehen. Um einen Ausschnitt für ein bestimmtes Gebietes zu erhalten, kann die Funktion «Export» auf der OSM-Seite verwendet werden. Dort wird ein Bereich ausgewählt, dessen Daten in verschiedenen Formaten heruntergeladen werden können. Es besteht die Auswahl zwischen einem XML basierten *.OSM-File, einer Karte im Format PNG, JPEG, SVG oder PDF und dem HTML-Code, der direkt in eine Webseite eingebunden werden kann. Falls grössere Gebiete als *.OSM-File benötigt werden, kann nicht mehr auf die Export-Funktion zurückgegriffen werden, da der Export-Bereich auf ein Viertel Grad in beide Richtungen (Nord-Süd und Ost-West) beschränkt ist. Eine weitere Möglichkeit, einen Ausschnitt aus den Daten zu erhalten, ist es, eine URL zu konstruieren, die eine API-Anfrage schickt. Wie im folgenden Beispiel zu sehen ist, wird im Request eine «bounding box» definiert.

http://api.openstreetmap.org/api/0.6/map?bbox=11.54,48.14,11.543,48.145

Die Grösse des Ausschnittes ist, wie beim direkten Export von openstreetmap.org, beschränkt, diesmal auf 0.5 mal 0.5 Grad. Werden grössere Gebiete benötigt, muss auf die erweiterte API (Xapi) zugegriffen werden. Es ist ein «read-only» Protokoll, das auf der normalen API basiert und helfen soll, diese zu entlasten. Xapi implementiert den üblichen «map request» neu, sodass die Anfrage viel schneller verarbeitet werden kann. Zusätzlich werden erweiterte Such- und Abfragefunktionen angeboten. Zum Beispiel können nur diejenigen Linien heruntergeladen werden, welche ein Tag-Element «highway=motorway» oder «highway=motorway_link» und so weiter besitzen und innerhalb einer definierten «bounding box» liegen.

http://www.informationfreeway.org/api/0.6/way[highway=motorway|motorway_link|trunk|primary]][bbox=-6,50,2,61]

Es kann eine beliebige Anzahl solcher Abfragebedingungen aneinander gereiht werden. Die auf diese Weise über die API bezogenen Daten werden im Format *.OSM geliefert. Wenn nicht die Variante mit der API verwendet werden will, kann auf Drittanbieter ausgewichen werden. (OpenStreetMap 2013)

Auf *planet.osm* können zum Beispiel sämtliche Daten in einem File, welches jede Woche aktualisiert wird, runtergeladen werden. Dieses File gibt es als komprimiertes OSM XML oder als PBF (Protocolbuffer Binary Format) File. Zudem gibt es die Möglichkeit, sogenannte «Extracts» oder «Mirrors» von Drittanbietern, zum Beispiel *GeoFabrik.de* oder *BBBike.org*, zu beziehen. Das bedeutet, die Daten von einzelnen Kontinenten oder Ländern werden in kleineren Files zur Verfügung gestellt (Planet OSM 2013).

Eine weitere Möglichkeit ist es, die Daten über das Open Source Programm Quantum GIS herunterzuladen. Dafür braucht es ein spezielles OSM-Plugin, welches einfach aktiviert werden kann. Damit kann ebenfalls ein Ausschnitt geladen werden, dessen Daten sofort in der GIS-Software visualisiert werden. Die Daten werden aufgeteilt in die drei Layer Punkte, Linien und Flächen.

4.1.1. Datenformate

Die wichtigsten und häufigsten Datenformate, welche für die OSM-Daten verwendet werden sind OSM XML, PBF, o5m und OSM JSON. Je nach Anwendung und Nutzung der Daten eignen sich diese Formate besser oder schlechter. Es muss daher abgeschätzt werden, wann welches Format am besten geeignet ist. Das erste ist ein XML-Format, welches über die API zur Verfügung gestellt wird. In diesem Format können Daten einfach heruntergeladen und weiterverwendet werden. PBF ist ein optimiertes Binärformat, das stark komprimiert ist und sehr schnell geschrieben und gelesen werden kann. Es ähnelt dem XML-Format und wurde entwickelt, um zukünftig die Erweiterbarkeit und die Flexibilität der Daten zu ermöglichen. o5m wird vor allem für die sehr schnelle Verarbeitung von Daten verwendet und ist eine Mischform der ersten beiden Formate. Es hat die Struktur des XML-Formates, verwendet aber die Kodierung von PBF. Das letzte oben erwähnte Format ist OSM JSON, welches die JSON Variante von OSM XML ist (OpenStreetMap 2013).

4.2. Analyse der Daten

4.2.1. Daten-Grundelemente

Die OSM-Daten basieren auf den drei Grundelementen Punkt (node), Linie (way) und Relation. Eine Relation ist eine sortierte Liste von Datenelementen und beschreibt die Beziehung zwischen Objekten und deren Teile (zum Beispiel zwischen einem Wald und seinen Lichtungen). Jedem von diesen drei Elementen können bestimmte Attribute (tags) zugeordnet werden, welche aus einem Schlüssel (key) und einem Wert (value) bestehen. Obwohl diese Schlüssel-Wert-Paare frei formatierbare Textfelder sind, folgen sie in der Praxis bestimmten Konventionen. Im OSM-Wiki ist eine Liste mit den sogenannten «Map Features» veröffentlicht, in der eine grosse Anzahl Möglichkeiten für solche Paare aufgeführt ist. Ein Punkt ist durch seine Position, das heisst Länge und Breite, beschrieben und kann optional ein Attribut für die Höhe enthalten. Er kann sowohl als Einzelpunkt als auch als Punkt in einer Linie verwendet werden. Eine Linie ist eine geordnete Liste von mindestens zwei Punkten. Eine solche Liste kann neben einer Linie auch eine Fläche, das heisst eine geschlossene Linie, repräsentieren. Eine geschlossene Linie hat den gleichen Anfangs- und Endpunkt und kann nicht nur eine Fläche sondern auch eine geschlossene Polylinie darstellen. Zur Unterscheidung müssen die Attribute genauer betrachtet werden. Eine Relation besteht aus einem

oder mehreren Tags und aus einer geordneten Liste mit einem oder mehreren Punkten und/oder Linien. Sie beschreibt die logische oder geografische Beziehung zwischen den Grundelementen (OpenStreetMap 2013).

4.2.2. Daten-Qualität

Wird nun die Qualität der Daten im Testgebiet Andermatt betrachtet, muss bezüglich der Vollständigkeit der Daten gesagt werden, dass diese keinesfalls gewährleistet ist, da es sich, wie schon mehrmals erwähnt, um ein Community-Projekt handelt und jedermann Daten erfassen kann. Die Menge der vorhandenen Daten ist abhängig von den Usern und deren Erfassung von Geodaten. In der Region Andermatt ist deutlich zu erkennen, dass zwar im Dorf selber sehr viele Daten vorhanden sind, ausserhalb des besiedelten Bereichs jedoch einiges fehlt. Auffällig sind Bergseen, Wanderwege oder Bodenbedeckungsflächen, welche in den Bergen sehr spärlich erfasst sind. Ein Beispiel dafür ist in Abbildung 9 zu sehen. Auf dem Satellitenbild ist in der Mitte sehr deutlich ein Wanderweg zu erkennen, bei den OSM-Daten fehlt er. Ebenfalls gut erkennbar ist die Tatsache, dass beim Gewässernetz zahlreiche, vor allem kleinere Bachläufe fehlen.

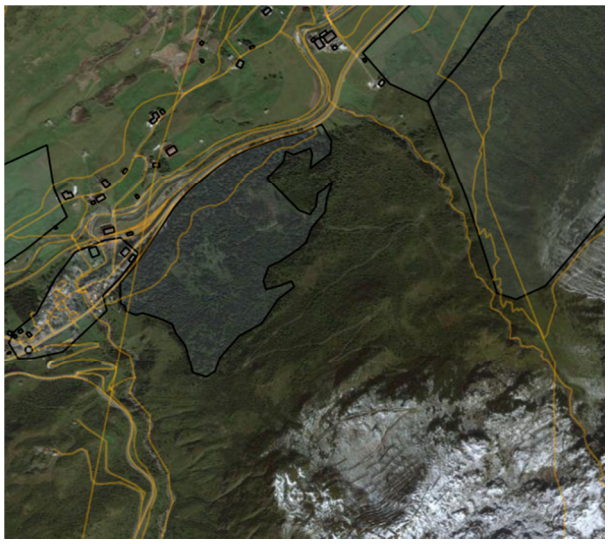


Abbildung 9: Vergleich mit Google Satellit - Wanderweg

Werden die Daten mit einem Satellitenbild verglichen, kann festgestellt werden, dass die Lage der erfassten Objekte, wie Strassen, Eisenbahnlinien, Gebäude und so weiter, ziemlich gut stimmt. Es gibt einige Abweichungen vom wahren Verlauf, zum Beispiel bei Strassen oder Flüssen (Abbildung 10), aber diese liegen für diese Arbeit im Bereich des Akzeptierbaren. Zudem sind die Abgrenzungen von erfassten Bodenbedeckungsflächen teilweise nicht ganz nachvollziehbar. Zum Beispiel gibt es Flächen, die in der Mitte eines Waldstückes aufhören.

Abbildung 10: Vergleich mit Satellitenbild (Google Satellite) – Fluss

Um die Topologie zu prüfen, wird eine GIS-Software verwendet. Mit Hilfe dieser Software können Topologie-Fehler einfach gefunden werden. Wird dies mit den OSM-Daten durchgeführt, kann festgestellt werden, dass diese sehr viele solche Fehler enthalten. Dies hat sicher damit zu tun, dass viele User, welche solche Daten erfassen, nicht so genau oder gar nicht wissen, was Topologie bedeutet und welche Regeln zu beachten sind. Einige Beispiele zu solchen Fehlern sind in den folgenden Abbildungen zu sehen.

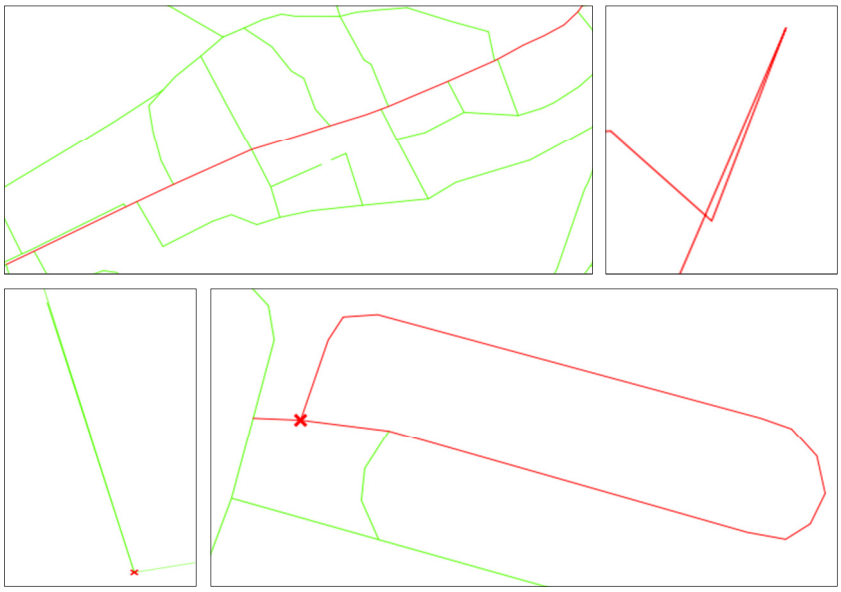


Abbildung 11: Topologiefehler in OSM-Daten

Die rote Linie im Bild oben links ist eine einzelne, zusammenhängende Linie. Das heisst die Linie ist an keiner der Kreuzungen, wo eigentlich überall ein Knotenpunkt sein sollte, unterbrochen. Rechts daneben ist eine weitere Linie sichtbar, die nicht korrekt ist. Sie überschneidet sich selbst, ebenso wie die Linie im Beispiel unten rechts. Wobei dort zusätzlich zur Selbstüberschneidung Knotenpunkte fehlen. Unten links ist zu sehen, dass eine Linie ein Stück weit doppelt vorhanden ist. Neben diesen Beispielen gibt es noch zahlreiche andere solche Fehler. Leider wurde keine zufriedenstellende Lösung gefunden, um diese Fehler automatisch zu beheben und die Topologie zu bereinigen.

4.2.3. Vergleich mit offiziellen Referenzdaten

Um abschätzen zu können, wie gut die Qualität der OSM-Daten verglichen mit offiziellen Daten abschneidet, wird ein solcher Vergleich durchgeführt. In der Schweiz stammen solche offiziellen Daten vom Bundesamt für Landestopografie Swisstopo. Die vorliegende Benutzeroberfläche verwendet den Datensatz Vector25 von Swisstopo. Da dieser in einem Massstab vorliegt, der ähnlich ist wie derjenige der OSM-Daten, bietet sich ein Vergleich mit diesem Datensatz an. Dafür werden die Referenzdaten in einem ersten Schritt genauer betrachtet. Vector25 enthält folgende Klassen (Bundesamt für Landestopografie 2007):

- Strassennetz *(str)*
- Eisenbahnnetz *(eis)*
- Übriger Verkehr *(tsp)*
- Gewässernetz *(gwn)*
- Primärflächen *(pri)*
- Gebäude *(geb)*
- Hecken und Bäume *(heb)*
- Anlagen und *(anl)*
- Einzelobjekte und *(eob)*

Für den Vergleich der beiden Datensätze wird Abbildung 12 zu Hilfe gezogen. Sie zeigt auf der linken Seite eine Visualisierung der OSM-Daten, auf der rechten Seite den entsprechenden Ausschnitt mit Vector25-Daten, heruntergeladen von GeoVITe[5]. Wie aus der Legende ersichtlich, sind die OSM-Daten in die drei verschiedenen Layer Punkte, Linien und Flächen aufgeteilt. Die Swisstopo-Daten sind entsprechend der obigen Aufzählung in logisch zusammengehörige Kategorien unterteilt. Zu erwähnen ist, dass der Layer «Übriger Verkehr» in dieser Darstellung fehlt. Das liegt daran, dass er im Datensatz Vector25 von GeoVITe nicht enthalten ist.

Werden nun die visualisierten Daten miteinander verglichen, fällt auf, dass die Bodenbedeckung auf der linken Seite im Gegensatz zur rechten Seiten nicht flächendeckend erfasst ist. Ebenfalls auffallend sind die zahlreichen Punktobjekte in den OSM-Daten. Dabei handelt es sich um alles Mögliche, zum Beispiel um Restaurants, Brunnen, Parkbänke, Parkplätze, Kirchen oder um Poststellen und Briefkästen. Diese Informationen mögen für einige Anwendungen sinnvoll sein, für diese Arbeit sind sie jedoch nutzlos. Die in den Swisstopo-Daten enthaltenen, einzelnen Punkte, zum Beispiel Kirchen und Bahnhöfe, sind dafür völlig ausreichend.

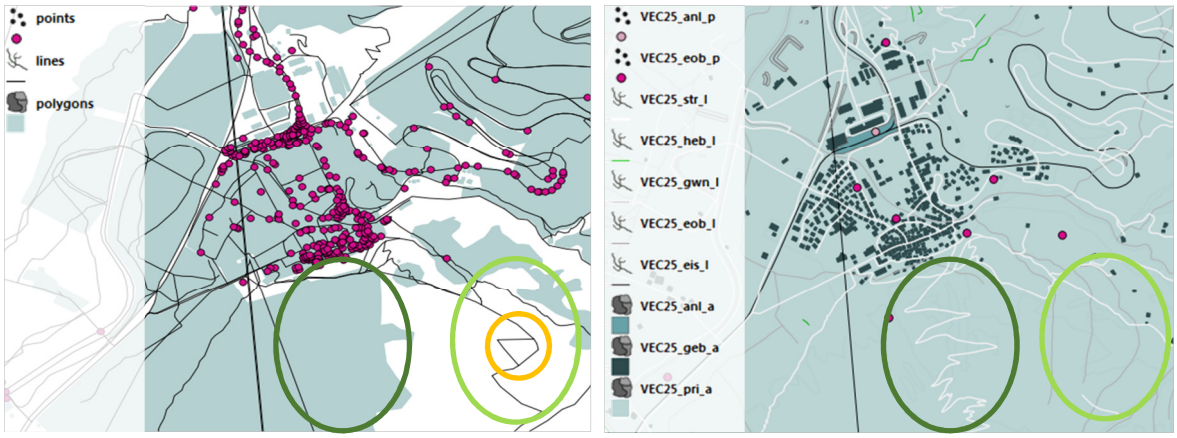


Abbildung 12: Datenvisualisierung in QGIS – links: OpenStreetMap, rechts: Vector25 © 2011 swisstopo (JA100042)

Die dunkelgrün markierte Stelle zeigt den in Abbildung 9 gezeigten Wanderweg, der bei OSM fehlt. In den Swisstopo-Daten ist er enthalten. Das Gleiche gilt für die grau dargestellten Bachläufe (hellgrün markiert) in der rechten unteren Ecke der Swisstopo-Seite. Sie sind links nicht vorhanden. Stattdessen gibt es ein anderes Linienobjekt (orange markiert), welches auf der rechten Seite nicht enthalten ist. Anhang A zeigt die beiden Datensätze ein weiteres Mal nebeneinander. Die beiden Abbildungen zeigen die Daten wie sie nach der Bearbeitung in der Benutzeroberfläche dargestellt werden. Wegen der gleichen Symbolisierung können Unterschiede besser erkannt werden. Es muss jedoch gesagt werden, dass dort nicht mehr alle Daten enthalten sind, da während der Bearbeitung einige Objekte entfernt werden.

[5] Das Ziel von GeoVITe («Geodata Visualization and Interactive Training Environment») ist es, einen klaren und intuitiven Überblick über zur Verfügung stehende Produkte von Swisstopo zu verschaffen und diese über ein einfach zu benutzendes Online-Portal anzubieten. Der Service wird vom Institut für Kartografie und Geoinformation der ETH Zürich, in Kooperation mit der ETH-Bibliothek, für alle ETH-Mitarbeiter zur Verfügung gestellt (Iosifescu, et al. 2011).

area_	perimeter	objectid	objectorig	objectval	yearofchan
33439668.7759397	819762.55187276	3863981	LK25	Z_Fels	1989
14703850.0102202	219472.87370582	18355025	GN25	Z_Wald	2006
305658.89960789	19816.53627768	3863323	LK25	Z_Fels	1989
1948.05499999	219.67037127	16193011	LK25	Z_Wald	2001
207630.72769212	4068.06861425	3863651	LK25	Z_Siedl	1989
1924.97999999	198.25814078	3864059	LK25	Z_Uebrig	1989
18728.61872541	834.35203406	18355088	GN25	Z_Uebrig	2006
2963.91703536	245.34427506	16193026	LK25	Z_Wald	2001
20415.75658396	1244.59888628	3863731	LK25	Z_Wald	1989

Abbildung 13: Datenschema Swisstopo

Werden neben den visualisierten Daten die Datenschemata (Abbildung 13 und Abbildung 14) verglichen, fallen weitere Unterschiede ins Auge. Für diese Arbeit sehr wichtig ist das Attribut «Objectval» (siehe Abbildung 13). Anhand dieses Attributs werden die Daten symbolisiert. Das heisst alle Flächen, welche zum Beispiel das Attribut Z_Wald besitzen, werden grün dargestellt. OpenStreetMap verfügt über kein Attribut, welches eine eindeutige Kategorisierung der Daten ermöglichen würde. Stattdessen gibt es das Attribut «Tags», welches zwar die benötigten Informationen enthält, jedoch in einer nicht optimalen Form. Es enthält eine, je nach Objekt kürzere oder längere Zeichenkette, bestehend aus aneinander gereihten «Key-Value-Pairs». Um die OSM-Daten für die benutzerdefinierte Symbolisierung verwenden zu können, müssen die Informationen aus den «Tags» in eine der «Objectval» ähnlichen Form gebracht werden. Wie dies erreicht wird, wird in Kapitel 4.3 erläutert.

timestamp	user	tags	name
2008-09-06T23:...	lonvia	"amenity"="parking","name"="Gemsstockbahn"	Gemsstockbahn
2008-09-06T23:...	lonvia	"area"="yes","highway"="pedestrian"	
2008-09-07T08:...	lonvia	"leisure"="pitch","sport"="tennis"	
2009-08-24T18:...	fx99	"name"="Moorgebiet","natural"="wetland","wetland"="bog"	Moorgebiet
2009-08-24T18:...	fx99	"area"="yes","highway"="pedestrian"	
2009-08-24T20:...	fx99	"amenity"="public_building","building"="yes","name"="Mehrzweckhal...	Mehrzweckhall...
2009-08-25T13:...	fx99	"layer"="1","natural"="water"	
2009-08-30T20:...	fx99	"building"="yes"	

Abbildung 14: Datenschema OpenStreetMap

4.3. Datenaufbereitung

4.3.1. Extrahieren von Zusatzinformationen

Im vorangehenden Kapitel wurde das Datenschema von OSM betrachtet und mit demjenigen von Vector25 verglichen. Wie aus den Zeichenketten ein Attribut mit jeweils nur einem Wert entsteht, wird im folgenden Abschnitt gezeigt. Als erstes müssen sämtliche Zusatzinformationen, welche im Attribut «Tags» gespeichert sind, extrahiert werden. Bei einigen Key-Value-Paaren, zum Beispiel bei «Highway», «Railway» oder «Landuse», ist dies in den heruntergeladenen Daten bereits gemacht. Das heisst es existiert jeweils ein Attribut «Highway», «Railway» und «Landuse». Viele andere wichtige Informationen sind jedoch nicht separat aufgeführt, sondern nur in der Zeichenkette enthalten. Mit Hilfe von zwei Funktionen können mit SQL («Structured Query Language») sogenannte Substrings extra-

hiert und in eine Tabelle geschrieben werden. Die erste Funktion, *substr(string, from, count),* extrahiert aus einer Zeichenkette (*string*) von der Position (*from*) an eine gewisse Anzahl Zeichen (*count*). Die zweite Funktion, *strpos(string, substring*), ermittelt die Position innerhalb einer Zeichenkette (*string*), an welcher eine zweite, kürzere Zeichenkette (*substring*) beginnt. Damit können die benötigten Informationen in die dafür vorgesehene, neu hinzugefügte Spalte «Objectval» geschrieben werden. Anhang C zeigt, wie die beiden Funktionen verwendet werden können, um das gewünschte Resultat zu erhalten.

4.3.2. Erstellen des benötigten Datenschemas

Nachdem nun geklärt ist, wie die benötigten Informationen aus den Zeichenketten extrahiert werden können, wird im nächsten Schritt das geforderte Datenschema erstellt. Das heisst, es soll ein Schema erstellt werden, welches demjenigen von Swisstopo möglichst ähnlich ist. Dafür müssen zuerst für alle bei Vector25 vorkommenden «Objectval»-Werten entsprechende Werte in den OSM-Daten gesucht werden. Die dabei entstandenen Tabellen sind in Anhang B zu finden.

Basierend auf diesen Tabellen, den Erkenntnissen aus Kapitel 4.2.3 und 4.3.1 wird das neue Datenschema erstellt. SQL bietet die Möglichkeit, sogenannte Views, das heisst virtuelle Tabellen, zu erstellen. Diese werden über gespeicherte Abfragen erstellt und können vom Datenbankbenutzer wie normale Tabellen verwendet werden. Der Zweck dieser Views ist es, den Zugriff auf Datenbankschemen zu vereinfachen. Genau diese Möglichkeit zur Vereinfachung eines Datenbankschemas wird hier benötigt. Es werden also Views erstellt, deren Schema mit dem Zielschema übereinstimmt. Die nachfolgenden Views werden erzeugt:

- Strassennetz *osm25_str*
- Eisenbahnnetz *osm25_eis*
- Übriger Verkehr *osm25_tsp*
- Gewässernetz *osm25_gwn*
- Primärflächen *osm25_pri*
- Gebäude osm25_geb
- Hecken und Bäume *osm25_heb*
- Anlagen osm25_anl
- Einzelobjekt osm25_eob und osm25_eob_l

Diese Views entsprechen weitgehend den Klassen von Vector25. Auf die Klasse mit den Flächenobjekten für Anlagen wurde aufgrund nicht in diesem Stil vorhandener Daten in OSM verzichtet. Einige Beispiele für das Erstellen solcher Datenbank-Views sind in Anhang D enthalten.

Da in der Anwendung die Möglichkeit besteht, die symbolisierten Daten zu beschriften, werden Daten benötigt, welche die entsprechenden Namen enthalten. Bei Swisstopo gibt es sogenannte SwissNames-Tabellen. Für verschiedene Massstäbe sind in diesen Tabellen sämtliche Namen, die für die Beschriftung von Karten verwendet werden, gespeichert. Bei OSM fehlt eine solche Tabelle. Die Namen der einzelnen Objekte sind im Attribut «Name» in den normalen Tabellen gespeichert. Das heisst für die Beschriftung werden folgende zusätzlichen Views erstellt:

- Beschriftung Natur *osm25_nam_p_nature*

- Beschriftung Berggipfel *osm25_nam_p_peak*
- Beschriftung Gewässernetz *osm25_nam_p_waterway*
- Beschriftung Gebäude *osm25_nam_p_building*
- Beschriftung Ortschaften *osm200_nam_p_village*

Die Geometrie aus den Ursprungstabellen wird beibehalten. Bei den Flächen- und Linienobjekten wird die Beschriftung pro Objekt einmal dargestellt und zwar im Mittelpunkt der Fläche beziehungsweise der Linie.

4.3.3. Generalisierung

Damit die Daten in verschiedenen Zoomstufen ansehnlich dargestellt werden können, muss ein zweiter, generalisierter Datensatz erstellt werden. Er soll in einer ähnlichen Form vorliegen wie die Vector200-Daten von Swisstopo. Dies ist nicht vollständig möglich, da zum Beispiel die Gebäude nicht aus einer automatischen Generalisierung aus den Vector25-Daten entstanden sind. Im Rahmen dieser Arbeit soll die gesamte Generalisierung automatisiert werden, weshalb einige Einbussen bezüglich der Qualität in Kauf genommen werden müssen.

In PostGIS gibt es für die Generalisierung die Funktion *ST_SimplifyPreserveTopology*. Sie gibt eine mit dem Douglas-Peucker Algorithmus «vereinfachte» Version der gegebenen Geometrie zurück, wobei die Topologie erhalten bleiben soll (PostGIS 2.0 Manual kein Datum). Der Douglas-Peucker Algorithmus ist ein in der Kartografie oft verwendeter Algorithmus zur Vereinfachung von Linienelementen. Er selektiert eine Teilmenge der Kanten einer Polylinie und bildet daraus eine neue, vereinfachte Linie, welche innerhalb einer vordefinierten Distanz von der Originallinie liegt (Saalfeld 1999). Die oben genannte Funktion benötigt zwei Werte, die Geometrie und die Toleranz, das heisst den maximalen Abstand zwischen vereinfachter und originaler Linie.

Für die Generalisierung der Flächenobjekte, das heisst deren Umrandung, kann diese Funktion sehr gut verwendet werden. In Abbildung 15Abbildung 14 sind einige Flächen des Layers «osm_pri_a» vor und nach der Generalisierung zu sehen. An den markierten Stellen sind die Unterschiede am besten erkennbar.

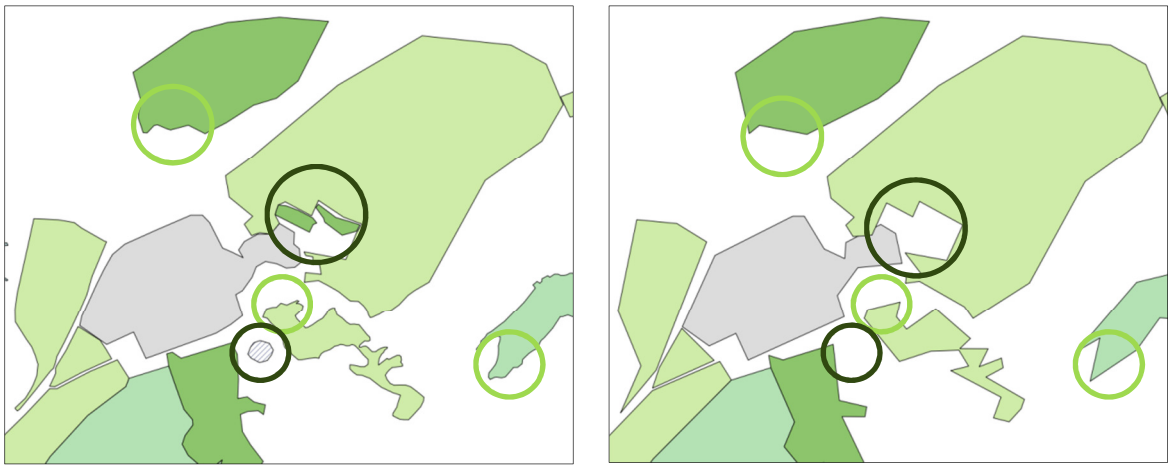


Abbildung 15: Generalisierung der Flächenelemente (links: Originaldaten, rechts: Generalisierung)

Die Toleranz für den Douglas-Peucker Algorithmus wurde auf 30 Meter gesetzt. Zusätzlich werden alle Flächen, welche kleiner sind als 15625 m^2, gelöscht (dunkelgrün markiert). Dies basierend auf der Annahme, dass eine Fläche mindestens 2.5 mm breit, beziehungsweise hoch sein muss, um in einer Karte mit Massstab 1:50'000 (ungefährer Massstab von kleinstmöglicher Zoomstufe) angezeigt werden zu können. Hierbei ist anzumerken, dass somit zwar kleine Flächen verschwinden, es aber nicht garantiert ist, dass keine Flächen mehr vorhanden sind, die eine Breite beziehungsweise Höhe von weniger als 2.5 mm haben.

Bei den Gebäuden wird in dieser Arbeit auf eine Generalisierung verzichtet. Da wäre es neben der Vereinfachung auf jeden Fall nötig, gewisse Gebäude zusammenzufassen, zu löschen oder neu zu platzieren. Dies ist automatisiert sehr schwierig zu erreichen. Deshalb wird der Layer Gebäude in kleineren Massstäben ausgeblendet.

Bei den Linienobjekten stellt sich die Generalisierung als weitaus schwieriger heraus. Es wird zwar ebenfalls die Funktion *ST_SimplifyPreserveTopology* verwendet, sie führt jedoch nicht zu zufriedenstellenden Resultaten. Der Grund dafür ist, dass hier die topologischen Fehler, die in den OSM-Daten vorhanden sind, zum Tragen kommen. Ein Beispiel für ein solches Problem wird in Abbildung 16 gezeigt. Zu sehen sind zwei Kreuzungen, welche nach der Generalisierung nicht mehr als solche existieren. Der Grund dafür ist, dass zum Beispiel die orange Linie durchgehend ist und somit an den Kreuzungen keine Knoten vorhanden sind. Das heisst der Algorithmus erkennt diese Orte nicht als Kreuzung und kann sie bei der Generalisierung nicht berücksichtigen.

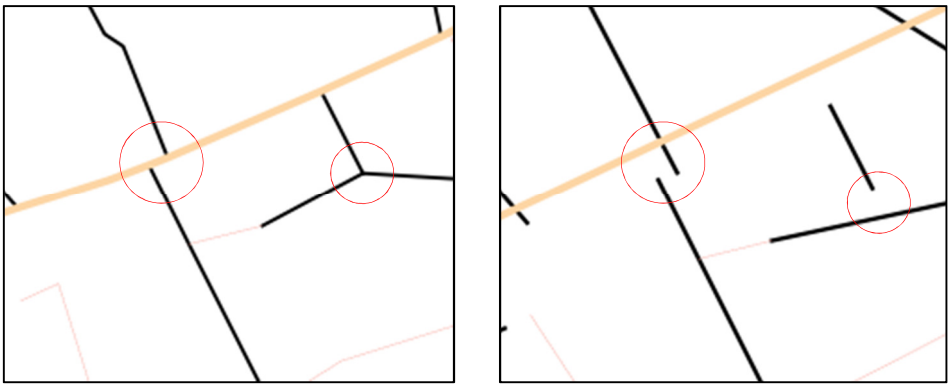


Abbildung 16: Generalisierung der Linienelemente (links: Originaldaten, rechts: Generalisierung)

Eine weitere Schwierigkeit ist das Entfernen von zu kleinen einzelnen Linien, ein Schritt, der in der automatischen Generalisierung nicht enthalten ist. Dies betrifft vor allem kurze Quartierstrassen oder Feld- und Wanderwege. Es können nicht, wie bei den Flächen, alle Linien, die kürzer sind als ein Grenzwert, gelöscht werden. Denn so würden auch kurze Verbindungslinien, die es braucht, um das Strassennetz korrekt darstellen zu können, gelöscht. Eine zufriedenstellende Lösung für dieses Problem wurde nicht gefunden.

Um solche Fehler zu vermeiden, müsste zuerst die Topologie bereinigt werden, was im Rahmen dieser Arbeit nicht möglich ist. Deshalb wird zwar eine Generalisierung der Linienobjekte durchgeführt, was die Qualität betrifft aber keine hohen Anforderungen gestellt.

Die Punktobjekte werden anhand des Attributs «Objectval» gefiltert. Das heisst, im generalisierten Datensatz sind nur noch diejenigen Punkte vorhanden, deren «Objectval» bestimmte

Werte enthalten. Die Werte werden nach dem Kriterium der Sinnhaftigkeit gewählt. Zum Beispiel macht es keinen Sinn, ein Punktobjekt für ein Wegkreuz in einem Massstab von 1:50'000 noch darzustellen.

5. Umsetzung der Anwendung

5.1. Erstellen vordefinierter Symbolisierungen

Ein Teil dieser Arbeit beinhaltet das Erstellen von verschiedenen Symbolisierungssätzen. Der Benutzer soll beim Starten des Programmes die Möglichkeit haben, mit den verschiedenen Darstellungen einen ersten Eindruck von den Daten zu gewinnen. Für das Erstellen dieser Symbolisierungen wird immer gleich vorgegangen. Zuerst wird nach vorhandenen, unterschiedlich symbolisierten Karten gesucht. Überzeugt eine Symbolisierung, wird sie genauer betrachtet und, falls umsetzbar, nachgebildet. Für jeden Symbolisierungssatz wird allen darzustellenden Layern eine Symbolisierung zugewiesen. Dies geschieht im Script WMSStart, wo alle Stilinformationen in einem Werte-Array gespeichert werden. Wie genau die Zuweisung der Darstellung zu den entsprechenden Layern geschieht, wird mit einem Code-Beispiel in Anhang E gezeigt. Folgende Symbolisierungssätze werden erstellt und im Auswahlmenü zur Verfügung gestellt: OpenStreetMap, Swisstopo, Deutschland, Schatzkarte und Rot-Grün. Zusätzlich gibt es einen neutralen Stil, der immer dann geladen wird, wenn ein neuer Satz erstellt wird.

5.1.1. OpenStreetMap

Dieser Style soll die übliche Darstellung von OpenStreetMap wiederspiegeln. Die Farben werden von openstreetmap.org abgegriffen und die Symbole nachgebildet. Bei verschiedenen Bodenbedeckungsflächen, zum Beispiel bei Wald, Gebüsch und Gletscher, wird neben einer einfachen Farbfüllung ein Pattern verwendet.

Abbildung 17: Style – OpenStreetMap

Das Resultat des im Anhang E gezeigten Codebeispiels für das Definieren eines Patterns ist in Abbildung 17 zu sehen. Es ist das für die Waldfläche definierte Pattern mit dem Symbol einer Tanne. Die Symbole für diese Muster werden teilweise mit Hilfe von Inkscape[6], teilweise direkt im Code erstellt. Ebenso verhält es sich mit Punktsymbolen. Ein Beispiel dafür ist das Kreuz für die Symbolisierung der Kirchen.

5.1.2. Swisstopo

Dieser Symbolisierungssatz wird aus der vorliegenden Anwendung übernommen. Es werden zwar kleinere Anpassungen bezüglich Darstellungsreihenfolge, Farbe, Strichdicke und Opazität vorgenommen, die grundsätzliche Symbolisierung wird jedoch beibehalten.

Abbildung 18: Style - Swisstopo

5.1.3. Deutschland

Der hier definierte Style ist der topografischen Karte Deutschlands nachempfunden. Die Style-Informationen dafür stammen einerseits von einigen Kartenbeispielen auf der Webseite vom deutschen Bundesamt für Kartographie und Geodäsie, andererseits vom Geoportal des deutschen Bundes. Das richtige Zuweisen der verschiedenen Darstellungen gestaltet sich für diesen Stil schwieriger als für die beiden vorher beschriebenen. Grund dafür ist die Tatsache, dass hier keine Daten aus der Region des Testgebietes vorhanden, sondern nur deutsche Gebiete abgedeckt sind. Daher kann die Symbolisierung nicht so einfach übertragen werden, sondern es müssen vergleichbare Gebiete mit entsprechenden Objekten gesucht werden. Ein Beispiel sind die Seilbahnen. Diese kommen in Deutschland nicht so häufig vor wie in der Schweiz und haben eine Symbolisierung (Abbildung 19), die mit dem QGIS Mapserver nicht realisiert werden kann. Dies kommt daher, dass ein

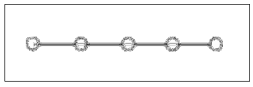


Abbildung 19: Seilbahn Deutschland

[6] Inkscape ist eine Open Source Graphik-Software mit ähnlichen Funktion wie Adobe Illustrator. Die Software verwendet das SVG-Datenformat, was sie zu einem für diese Arbeit sehr nützlichen Tool macht.

sogenannter «Graphic Stroke», der dafür benutzt wird, für diesen Server nicht implementiert ist. Auch bei der Darstellung von Gletschern gibt es einige Schwierigkeiten, da Deutschland nicht über allzu viele davon verfügt und keine Symbolisierung dafür gefunden werden kann.

Abbildung 20: Style - Deutschland

5.1.4. Schatzkarte

Diese Symbolisierung orientiert sich an den in Filmen wie Indiana Jones verwendeten Schatzkarten. Meist handelt es sich dabei um ein altes, zerknittertes Stück Pergament, das den Weg zu einem verborgenen Schatz weisen soll.

Abbildung 21: Style – Schatzkarte

Kartografisch gesehen ist eine solche Karte sicher nicht optimal. Sie soll vielmehr aufzeigen, welche Effekte mit verschiedenen Symbolisierungen erzielt werden können. Um den gewünschten Effekt zu erzielen, wird ein Rasterbild hinterlegt, das die Struktur eines Pergaments darstellen soll. Sämtliche Flächen werden nur mit einem Pattern, das heisst ohne Einfärbung der Fläche visualisiert. Speziell sind vor allem die Gebäude. Sie werden nicht als Flächen- sondern als Punktobjekt dargestellt, indem für jede Fläche der Mittelpunkt berechnet und symbolisiert wird.

5.1.5. Rot-Grün

Der Style «Rot-Grün» enthält keine spezielle Symbolisierungselemente. Er soll lediglich zeigen, dass auch mit einer sehr einfachen Symbolisierung durchaus ansprechende Karten erstellt werden können. Sämtliche Punkte, Flächen und Linien wurden nur in einer Ebene dargestellt. Das heisst es sind keine zusammengesetzten Symbolisierungen vorhanden.

Abbildung 22: Style – Rot-Grün

5.1.6. Neutral

Wie bereits erwähnt, wird diese Symbolisierung immer dann geladen, wenn vom Benutzer ein neuer Symbolisierungssatz erstellt wird. Es werden die gleichen Symbole und Muster wie für die OpenStreetMap-Symbolisierung verwendet. Die Farbsättigung hingegen wird stark reduziert. Abgesehen von den Flüssen und Bächen, die blau bleiben, werden nur verschiedene Grautöne benutzt. Der Grund dafür ist der Gedanke, dass der Benutzer eine möglichst neutrale Karte vor sich haben soll, um seiner Fantasie freien Lauf lassen zu können, ohne von vordefinierten Farben beeinflusst zu werden.

Abbildung 23: Style - Neutral

5.2. Benutzeroberfläche

5.2.1. Visualisierung der OSM-Daten

In einem weiteren Schritt geht es darum, die Benutzeroberfläche an die neuen Daten anzupassen. Dafür müssen alle Werte, welche einen direkten Zusammenhang mit den Daten haben, an die neue Datengrundlage angepasst werden. Das heisst, sämtliche Tabellennamen und «Objectval»-Werte werden geändert und an die Namen und Werte der OpenStreetMap-Daten angepasst. Grundsätzlich geht dies mit der «Suchen-Ersetzen» Funktion in einem Texteditor ziemlich schnell. Die Schwierigkeit ist, nicht nur eine eins zu eins Anpassung (gleiche Werte für «Objectval» von Vector25 und OpenStreetMap), sondern auch eine inhaltliche Anpassung zu machen. Genauer gesagt bedeutet dies, dass im Auswahlmenü auf der Benutzeroberfläche (siehe Eingabefenster vier in Abbildung 24) bestimmte Werte zur Verfügung stehen, welche ebenfalls den neuen Daten angeglichen werden müssen. Zum Beispiel sind bei Vector25 die Strassen in von eins bis sechs durchnummerierte Klassen aufgeteilt. Bei OSM gibt es nur erste, zweite und dritte Klasse sowie Feldweg, Fussweg, Pfad und Quartierstrasse. Diese neuen Bezeichnungen müssen einerseits im GUI, das heisst in der Funktion «main», andererseits in der Funktion «uebersetzer» angepasst werden. Der «uebersetzer» weist sämtlichen Bezeichnungen von der Benuzteroberfläche die entsprechenden, in «Objectval» gespeicherten Werte zu.

Für den WMS-Request muss vorläufig nur der Server angepasst werden, damit die Daten am richtigen Ort geholt werden. Die anderen Parameter, die für die Anfrage benötigt werden, sind so definiert, dass sie während der Laufzeit aus den anderen Files und aus der Benutzeroberfläche abgegriffen und übergeben werden.

Sind alle diese Änderungen erfolgt, können die OSM-Daten im Kartenfeld der Benutzeroberfläche mit dem gewünschten Style dargestellt werden.

Um für alle in der Anwendung verwendeten Daten und Rasterbilder OSM-Daten als Grundlage zu haben, müssen zusätzlich zu den dynamisch aus der Datenbank geladenen Daten die statischen Rasterbilder ersetzt werden. Dies ist zum einen die kleine Übersichtskarte unten in der Mitte, zum anderen die sogenannte «Pixelkarte», welche optional eingeblendet und hinter die symbolisierten Daten gelegt werden kann. Diese Swisstopo-Karten werden ersetzt mit OSM-Karten, welche als PNG beziehungsweise JPEG von openstreetmap.org heruntergeladen wurden.

5.2.2. Design

Aus Zeitgründen kann das Design des Prototyps nur geringfügig verändert werden. Abbildung 24 zeigt das Design der neuen Anwendung. Wird es mit demjenigen in Abbildung 7 verglichen, können einige kleine Unterschiede festgestellt werden. Am auffälligsten ist das Farbkonzept. Hat sich die Benutzeroberfläche am Anfang in blau präsentiert, erscheint sie nun in grün. Diese Anpassung wird vorgenommen, damit die Anwendung besser zur Marke OpenStreetMap passt. Zusätzlich zu dieser Farbänderung werden die einzelnen Eingabe- und Werkzeugfenster neu angeordnet. Menü eins und zwei, die sich ursprünglich auf der linken Seite der Anwendung befandenen, sind nun unten links angesiedelt. Sie teilen sich diese Position, indem sie in zwei verschiedenen Registerkarten abgelegt werden. Das Texteingabefeld, in dem der Name für einen neuen Symbolsatz eingetippt werden kann, ist jetzt als solches erkennbar. Vorher hat es sich nicht vom Erscheinungsbild der darunterliegenden Buttons unterschieden. Diese Änderung erleichtert dem Benutzer das Erstellen eines neuen Satzes. Das Menü für die Auswahl der zu symbolisierenden Layer (Nummer vier) und die Navigationstools mit der Übersichtskarte (Nummer fünf) werden lediglich nach rechts verschoben. Fenster Nummer sechs, welches bisher den aktuellen Symbolsatz, den aktuellen Massstab und die jeweilige Mausposition (in Schweizer Landeskoordinaten) enthalten hat, wird entfernt. Abgesehen vom aktuellen Symbolisierungssatz, der schon aus Fenster Nummer zwei ersichtlich wird, werden diese Informationen auf der rechten Seite des unteren Balkens eingeblendet. In der Mitte dieses Balkens besteht in der ursprünglichen Anwendung neben dem Hilfe- und dem Impressumbutton die Möglichkeit, die Tooltips ein- und auszuschalten. Da diese jedoch meistens sehr nützlich sind und dieser Button eher verwirrend als hilfreich ist, wird in der neuen Version darauf verzichtet und die Tooltips standardmässig eingeschaltet.

Alle diese Anpassungen werden in den beiden Hauptfiles, das heisst im File index.svg und im File main.js vorgenommen. Durch diese Verbesserungen ist eine Benutzeroberfläche entstanden, die zwar noch weiter optimiert werden kann, jedoch ein abgerundetes Bild abgibt.

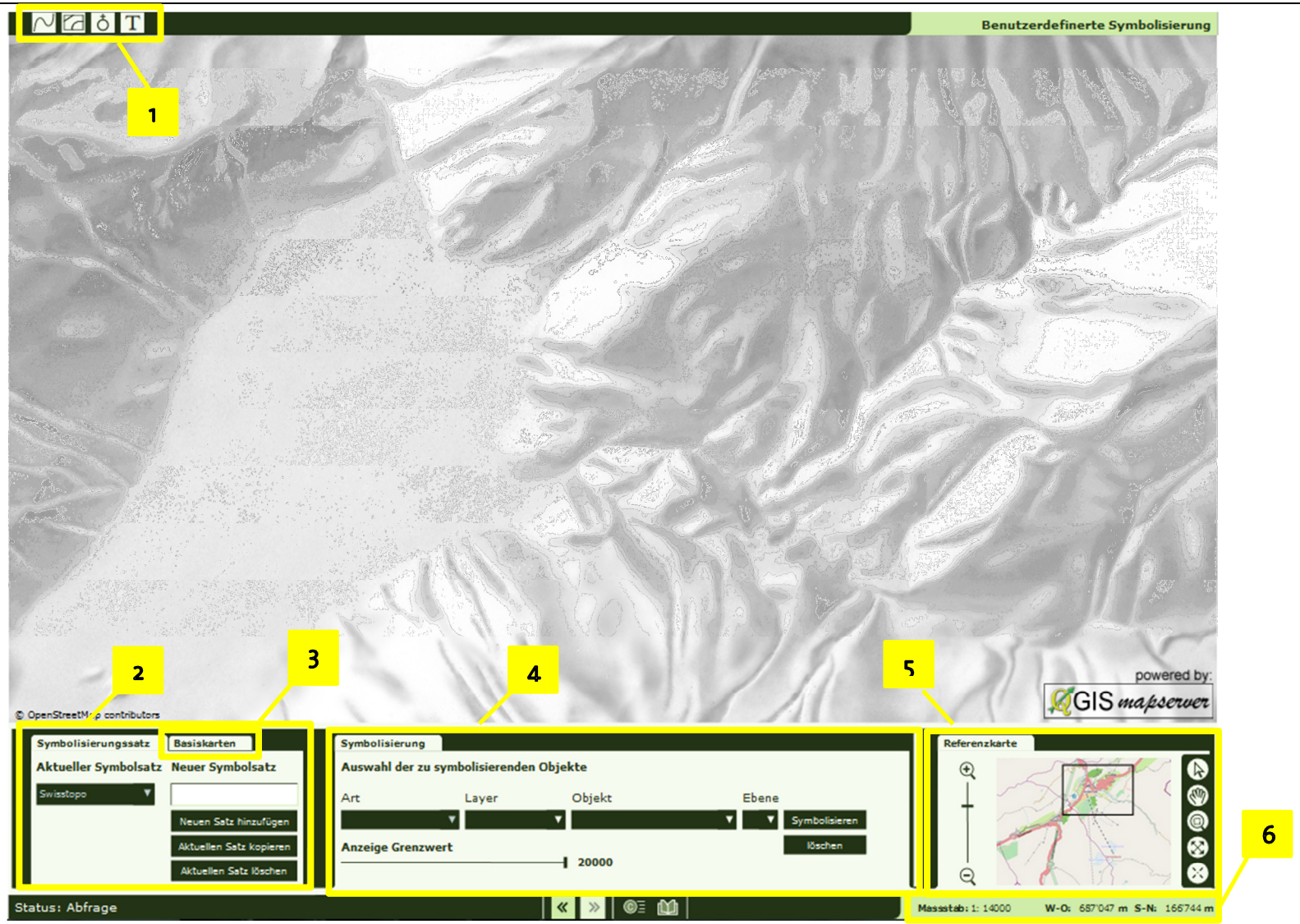


Abbildung 24: Design der neuen Anwendung

5.2.3. Funktionalität

Im Kapitel 5.1 wurde beschriebenen, wie Styles erstellt werden, welche vordefinierten Symbolisierungen vorhanden sind und wie die Benutzeroberfläche verändert wurde. In den folgenden Abschnitten wird auf die Funktionalität der Anwendung eingegangen. Es wird aufgezeigt, wie sie verwendet werden kann und was für Möglichkeiten sie bietet.

In Abbildung 25 ist das Menü mit der Übersichtskarte abgebildet. Es soll bei allen Zoomstufen zeigen, wo genau im Testgebiet der aktuelle Ausschnitt liegt. Neben dieser Karte enthält es sämtliche Navigationstools, die für das Zoomen und Panen in einer interaktiven Karte benötigt werden.

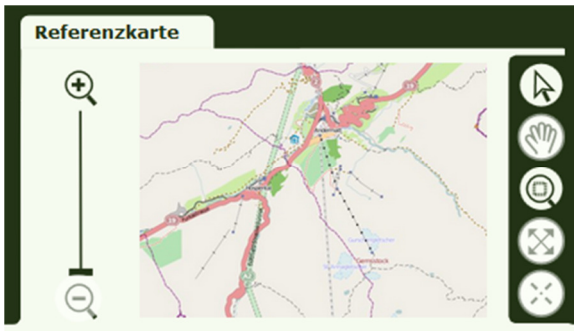


Abbildung 25: Menü für Navigation und Übersicht

Abbildung 26: Menü für Symbolisierungssatz - Wahl

Mit dem in Abbildung 26 zu sehenden Menü kann entweder ein vorhandener Symbolisierungssatz ausgewählt oder ein neuer erstellt werden. Bei letzterem werden die Daten mit der neutralen Symbolisierung geladen und können beliebig verändert werden. Dafür kann im Symbolisierungs-Menü (Abbildung 27) ein Layer ausgewählt und mit den verschiedenen Tools symbolisiert werden. Die Layer können anhand ihrer Objektklasse symbolisiert werden. Zudem bietet sich die Möglichkeit, drei verschiedene Ebenen für die Darstellung zu wählen. Das heisst, dass ein Objekt mehrere übereinanderliegende Symbolisierungen enthalten kann.

Abbildung 27: Menü für Wahl der zu symbolisierenden Layer

Abbildung 28 zeigt die für die Symbolisierung zur Verfügung stehenden Werkzeuge. Für die Punktobjekte kann gewählt werden zwischen den vordefinierten Formen Rechteck, Kreis, Dreieck, Stern, Kreuz und X und einem selbst zu erstellenden SVG-Symbol. Dieses kann mit einem zusätzlichen Editor gezeichnet werden. Im Linien-Werkzeug kann neben der Farbwahl und der Strichdicke, eine Strichlierung eingestellt werden. Wie für die Punkte gibt es für die Flächen zwei Möglichkeiten, eine einfache Einfärbung oder die Symbolisierung mit einem Pattern, das mit dem Zusatz-Editor erstellt wird. Für die Beschriftung können Schriftart, Grösse, Farbe und Stärke gewählt werden.

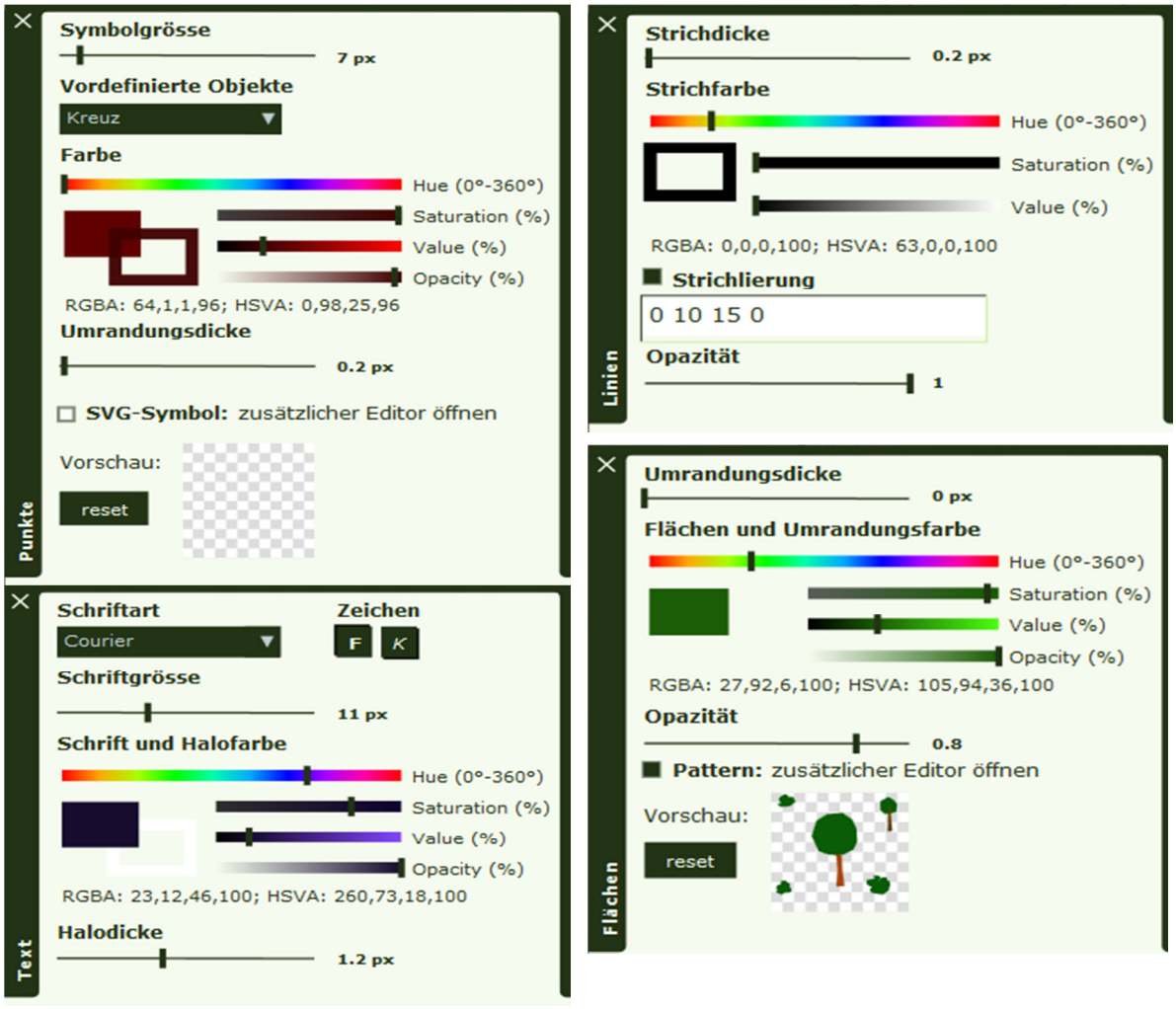


Abbildung 28: Symbolisierungstools

Mit Hilfe dieser Tools können die Daten von OpenStreetMap beliebig symbolisiert werden. Obwohl es bestimmt mögliche Symbolisierung gibt, die mit dieser Anwendung nicht erstellt werden können, bietet der vorliegende Prototyp eine grosse Auswahl an Möglichkeiten.

6. Ergebnisse

6.1. Datenimport

In Kapitel 2.2 wurden verschiedene Software-Lösungen für das Verarbeiten von OSM-Daten vorgestellt. Für das Herunterladen und Speichern der Daten in einer PostGIS Datenbank wurde in dieser Arbeit die Software Quantum GIS verwendet. Sie bietet den Vorteil, dass die Daten sogleich visualisiert werden können, was dabei hilft, einen ersten Eindruck davon zu erhalten. Die anderen Lösungen für den Import von OSM-Daten sind ebenfalls allesamt Open Source Programme. Namentlich sind dies Osmconvert, Osm2pgsql, Osmosis und Imposm. Sie werden alle über die Konsole gestartet und bedient, was sie für unerfahrene Benutzer zu eher komplizierten Tools macht.

6.2. Datenanalyse und -aufbereitung

Um für die Anwendung verwendet werden zu können, mussten die Daten vorbereitet werden. Dafür wurde zuerst eine Analyse der Daten durchgeführt. Diese Analyse ergab, dass die von Usern erstellten OpenStreetMap-Daten zum Teil sehr gut mit den offiziellen Daten mithalten können, zum Teil aber doch gewisse Mängel aufweisen. Auf der einen Seite stehen die Flächen und Linien, die noch nicht vollständig erfasst sind und einige Lücken aufweisen, auf der anderen Seite die riesige Anzahl an erfassten Punktobjekten. Diese können für eine Anwendung mit einem Standort basierten Service sehr wichtig sein, für eine Verwendung der Daten im Bereich der Kartografie sind die meisten davon überflüssig. Ein grösseres Problem als die Vollständigkeit der Daten stellen die Topologiefehler dar. Diese Fehler, die bisher mit keinem automatisierten Verfahren behoben werden können, führen spätestens bei der Generalisierung der Daten zu unschönen Effekten.

Beim Vergleich mit den Referenzdaten hat sich herausgestellt, dass das Schema der OSM-Daten im Originalzustand nicht für die Anwendung verwendet werden kann. Der Grund dafür ist das Attribut «Tags». Es besteht aus teilweise sehr langen Zeichenketten, welche die für die Symbolisierung wichtigen Informationen enthalten. Im Gegensatz dazu besitzen die Vector25-Daten das Attribut «Objectval». Es enthält jeweils einen Wert, zum Beispiel die Klasse einer Strasse, der für die Unterscheidung der Symbolisierung benötigt wird.

Mit Hilfe dieser Erkenntnisse und einiger SQL-Abfragen wurden schliesslich die folgenden Datenbank-Views erstellt:

- Eisenbahnlinien
- Einzelobjekte (Punkte und Linien)
- Gebäude
- Grenzen
- Gewässernetz
- Primärflächen
- Strassen
- Anlagen (Punkte)
- Hecken und Bäume
- Übriger Verkehr
- Beschriftung
  - Berggipfel
  - Ortschaft
  - Natur
  - Gewässer
  - Gebäude

Diese Views liegen in einem Schema vor, welches von der Anwendung verwendet werden kann. Abbildung 29 zeigt ein solches Schema am Beispiel der Primärflächen.

gid integ	name character varying(8(	objectval character varying(100)	the_geom geometry
9		residential	01030000201
39		residential	01030000201
46	Hospental	residential	01030000201
61		grass	01030000201
62		grass	01030000201
68		forest	01030000201
76		grass	01030000201
97		forest	01030000201
111		forest	01030000201
112		forest	01030000201
135		forest	01030000201
454		other	01030000201
476		other	01030000201

Abbildung 29: Daten-Schema der erstellten Views

Da in der Anwendung mehrere Zoomstufen zur Verfügung stehen, wird ein zweiter Datensatz benötigt. Dies wurde erreicht, indem der Originaldatensatz generalisiert wurde. Das Resultat ist alles andere als zufriedenstellend. Die Generalisierung wurde automatisiert, das heisst mit der Funktion ST_SimplifyPreserveTopology erstellt. Aufgrund der vielen, in den Daten enthaltenen Topologiefehler entstanden sehr viele Objekte, welche nicht korrekt dargestellt werden.

6.3. Anwendung

In Kapitel 5.2.3 wurde die Anwendung so beschrieben, wie sie vom Benutzer wahrgenommen und verwendet wird. Im Folgenden werden die im Hintergrund ablaufenden Prozesse geschildert.

Beim Laden der Anwendung wird die Hauptfunktion aufgerufen. Diese erstellt die Benutzeroberfläche und initialisiert die verschiedenen Elemente, wie zum Beispiel die Buttons oder Checkboxes. Damit die Daten mit der richtigen Symbolisierung geladen werden können und diese Layer für Layer verändert werden können, braucht es weitere Funktionen. In Abbildung 30 sind die Wichtigsten aufgeführt.

Wird vom Benutzer ein Symbolsatz ausgewählt, wird die Funktion WMSStart() aufgerufen. Darin sind sämtliche Informationen enthalten, die für die Symbolisierung der Daten benötigt werden. Die zum entsprechenden Symbolsatz gehörenden Werte werden der Funktion getWMS() übergeben. Diese verarbeitet die Werte zu einem SLD und sendet einen WMS-Request an den Server. Das von dort zurückkommende Rasterbild stellt die Daten mit der im SLD definierten Symbolisierung dar. Handelt es sich beim aktuellen Symbolsatz um einen neu erstellten, wird anstelle der Funktion WMSStart() die Funktion WMSBasis() aufgerufen. Sie enthält die Informationen für den neutralen Style und funktioniert gleich wie WMS-Start(). Da für die Symbolisierung einzelner Layer ebenfalls ein WMS-Request gesendet werden muss, verwendet auch die letzte, in Abbildung 30 aufgeführte Funktion, symbolize(), die getWMS-Funktion. Die Symbolisierungswerte stammen hier nicht aus einer Funktion mit

vordefinierten Werten, sondern werden direkt aus den Werkzeugen der Anwendung abgegriffen und temporär gespeichert. Die Funktionen WMSStart(), WMSBasis(), symbolize() und getWMS() sind diejenigen, welche hauptsächlich für die Darstellung der Daten mit einer gewünschten Symbolisierung verantwortlich sind.

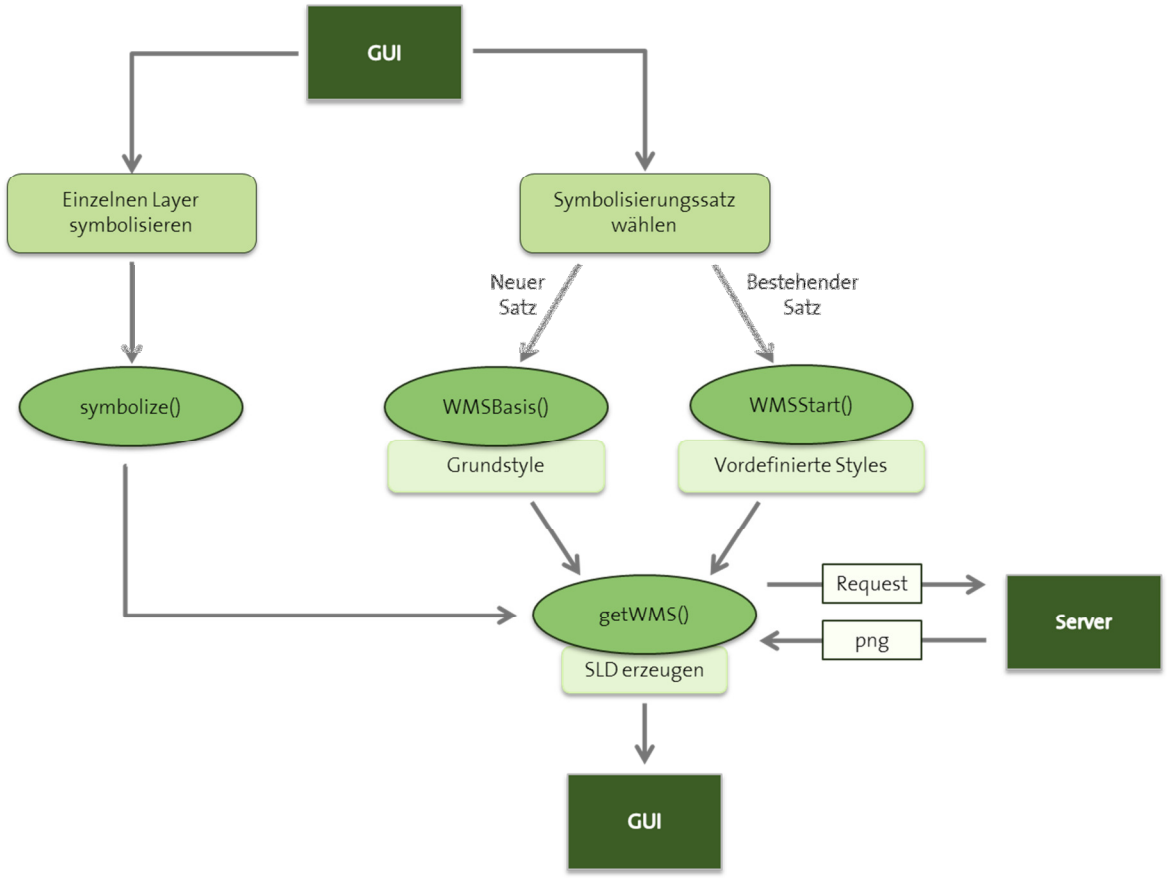


Abbildung 30: Im Hintergrund ablaufende Prozesse

Damit sämtliche Prozesse reibungslos ablaufen können, sind natürlich auch alle anderen Funktionen von Bedeutung. Eine kurze Beschreibung der einzelnen Funktionen ist in Anhang F zu finden.

7. Fazit und Ausblick

7.1. Fazit

Eine interaktive benutzerdefinierte Symbolisierung von OpenStreetMap-Daten für ein Testgebiet kann mit Hilfe verschiedener Methoden und Tools realisiert werden. Da es sich bei diesem Testgebiet um einen sehr kleinen Ausschnitt aus der riesigen Menge von verfügbaren OpenStreetMap-Daten handelt, kann diese Aussage nicht auf sämtliche Regionen ausgeweitet werden. Trotzdem können einige Schlüsse über die Daten und deren Symbolisierung in einer Benutzeroberfläche gezogen werden.

Es ist möglich, die Swisstopo-Daten durch diejenigen von OSM zu ersetzten. Weil das erstellte Datenschema demjenigen von Vector25 sehr ähnlich ist, ist in der Anwendung selbst kein Unterschied zu bemerken. Die Aufbereitung der Daten, die notwendig ist, um ein solches Schema zu erhalten, ist sehr zeitaufwändig. Zudem kann es sein, dass sich die Daten, je nach Region, stärker von den Referenzdaten unterscheiden als innerhalb des Testgebiets. Das heisst die hier verwendeten SQL-Abfragen können zwar als Grundlage benutzt werden, müssen aber höchstwahrscheinlich an die Daten in anderen Gebieten angepasst werden.

Die für den Import der Daten gefundenen Lösungen sind nicht alle ganz einfach zu verstehen und schon gar nicht einfach anzuwenden. Auch bezüglich der Dokumentation sind die verschiedenen Lösungen unterschiedlich. Die viel verwendeten und gut bekannten Tools sind besser dokumentiert als die anderen. Trotz der vorhandenen Tools besteht noch Verbesserungspotential für den Import von OSM-Daten, vor allem bezüglich der Möglichkeiten für die Weiterverarbeitung.

Für die Generalisierung der Daten konnte keine gute Lösung gefunden werden. Die Probleme mit den Topologiefehlern konnten nicht gelöst werden. Sobald dafür eine Lösung gefunden wurde, wird sich auch die Generalisierung vereinfachen. Die dafür vorgesehenen Funktionen sollten dann bedeutend weniger Probleme mit den Daten haben. Der in dieser Arbeit erstellte zweite, generalisierte Datensatz ist deshalb qualitativ schlecht. Mangels einer besseren Alternative wird er trotzdem für die Anwendung verwendet.

Obwohl die Benutzerfreundlichkeit bei diversen Anwendungen und Software immer wieder ein Thema ist und sich schon zahlreiche Leute damit befasst haben, ist dieses Thema immer wieder Gegenstand heisser Diskussionen. Dies hängt vermutlich damit zusammen, dass es keine allgemeingültige Definition von Benutzerfreundlichkeit gibt. Ob ein Benutzer ein Programm als benutzerfreundlich bezeichnet oder nicht, hängt vor allem mit seinen persönlichen Vorlieben und Erfahrungen zusammen. Trotz der vorgenommenen Anpassungen an der Benutzeroberfläche, kann diese weiterhin nicht als benutzerfreundlich bezeichnet werden. Es konnte zwar eine kleine Verbesserung erreicht werden, doch es muss noch einiges verbessert werden.

Die Symbolisierung hat eine grosse Bedeutung für die Darstellung von Geodaten. Mit völlig identischen Grundlagedaten können sehr unterschiedliche Effekte erzielt und somit auch ein unterschiedliches Zielpublikum angesprochen werden. Zum Beispiel kann mit einer

Symbolisierung wie Swisstopo der Eindruck einer seriösen Karte erweckt werden, während die gleichen Daten, symbolisiert wie eine Schatzkarte, eher ungenau und unseriös wirken.

7.2. Ausblick

Mit dieser Arbeit konnte eine Anwendung für die interaktive benutzerdefinierte Symbolisierung von OpenStreetMap-Daten erstellt werden. Obwohl sämtliche bereits für die Symbolisierung von Swisstopo-Daten erstellten Werkzeuge verfügbar sind und funktionieren, verfügt dieser Prototyp über zahlreiche potentielle Erweiterungsmöglichkeiten.

Die Verbesserung der automatischen Generalisierung ist einer der ersten Punkte, dessen Realisierung angestrebt werden sollte. Dies wird vor allem dann benötigt, wenn das abgedeckte Gebiet auf die ganze Schweiz erweitert werden soll. Dann wird nämlich eine zusätzliche Generalisierungsstufe nicht mehr ausreichen. Mit einer schlechten Generalisierung könnten daher nur für grossmassstäbige Karten zufriedenstellende Resultate erzielt werden. Deshalb macht es keinen Sinn, das Gebiet zu erweitern, solange keine ansprechende Lösung für die Generalisierung gefunden werden konnte.

Neben einer solchen Gebietserweiterung, das heisst einer Erweiterung der Datengrundlage, gibt es auch auf Seiten der Anwendung zahlreiche Verbesserungsmöglichkeiten. Einerseits kann die Benutzerfreundlichkeit verbessert, andererseits die Funktionalität erweitert werden. Zum Beispiel wäre es von Vorteil, wenn die erstellten Styles für eine spätere Wiederverwendung gespeichert werden könnten. Eine spezielle Eigenschaft von Daten, die von Benutzern erfasst werden, ist das Tempo, mit welchem sich diese Daten ändern. Im Gegensatz zu offiziellen Daten, welche alle paar Jahre nachgeführt werden, verändern sich die OSM-Daten ständig. Die Integration eines automatischen Datenupdates könnte deshalb angestrebt werden. Es gibt Anwendungen, die diese Möglichkeit bereits besitzen. Ob und wie dies für die vorliegende Anwendung erreicht werden kann, muss untersucht werden.

8. Literaturverzeichnis

Ather, Aamer. „A quality analysis of openstreetmap data.“ Master's thesis, University College London. Mai 2009. ftp://ftp.cits.rncan.gc.ca/pub/cartonat/Reference/VGI/Dissertation-OpenStreepMap-Quality-Aather-2009.pdf (Zugriff am 2013).

Bundesamt für Landestopografie. „Vector25.“ Das digitale Landschaftsmodell der Schweiz. Juni 2007. http://www.swisstopo.admin.ch/internet/swisstopo/de/home/products/landscape/vector25.html (Zugriff am 02. April 2013).

Dahlström, Erik, et al. Scalable Vector Graphics, W3C Recommendation. 2011. http://www.w3.org/TR/SVG11/ (Zugriff am 24. Februar 2013).

GitHub. 2013. https://github.com/openstreetmap/osm2pgsql#readme (Zugriff am 10. März 2013).

Goodchild, Michael F. „Citizens as sensors: the world of volunteered geography.“ GeoJournal, 2007: 211-221.

Haklay, Dr Mordechai (Muki). „How good is volunteered geographical information? A comparative study of OpenStreetMap and Ordnance Survey datasets.“ Environment and planning B, Planning & design, 2010: 682-703.

Imposm Documentation. 2011. http://imposm.org/docs/imposm/latest/ (Zugriff am 20. März 2013).

Iosifescu, Cristina, Ionut Iosifescu, Helen Jenny, und Lorenz Hurni. „GeoVITe - A Service-driven Solution for an On-demand, Userfriendly Web Access to Geodata.“ ICC2011: The 25th International Cartographic Conference. Paris, 2011.

MapBox. 2013. http://mapbox.com/ (Zugriff am 20. März 2013).

Nielsen, Jakob, und Joann T Hackos. „Usability engineering.“ Boston: Academic press, 1993.

OGC. 2013. http://www.opengeospatial.org/ (Zugriff am 25. Mai 2013).

OpenStreetMap. OpenStreetMap Wiki. 2013. http://wiki.openstreetmap.org/wiki/Main_Page (Zugriff am 09. März 2013).

Ordnance Survey. 2013. http://www.ordnancesurvey.co.uk/oswebsite/products/os-mastermap/index.html (Zugriff am 20. Mai 2013).

Planet OSM. 2013. http://planet.openstreetmap.org/ (Zugriff am 13. März 2013).

PostGIS. 2013. http://postgis.net/ (Zugriff am 24. Mai 2013).

PostGIS 2.0 Manual. kein Datum. http://postgis.refractions.net/documentation/manual-2.0/ST_SimplifyPreserveTopology.html (Zugriff am 02. April 2013).

QGIS Mapserver. 2009. http://karlinapp.ethz.ch/qgis_wms/about/index.html (Zugriff am 24. Februar 2013).

Quantum GIS. 2013. http://www.qgis.org/ (Zugriff am 24. Mai 2013).

Richter, Michael, und Markus D Flückiger. „Usability Engineering kompakt: benutzbare Software gezielt entwickeln." Spektrum Akademischer Verlag, 2010.

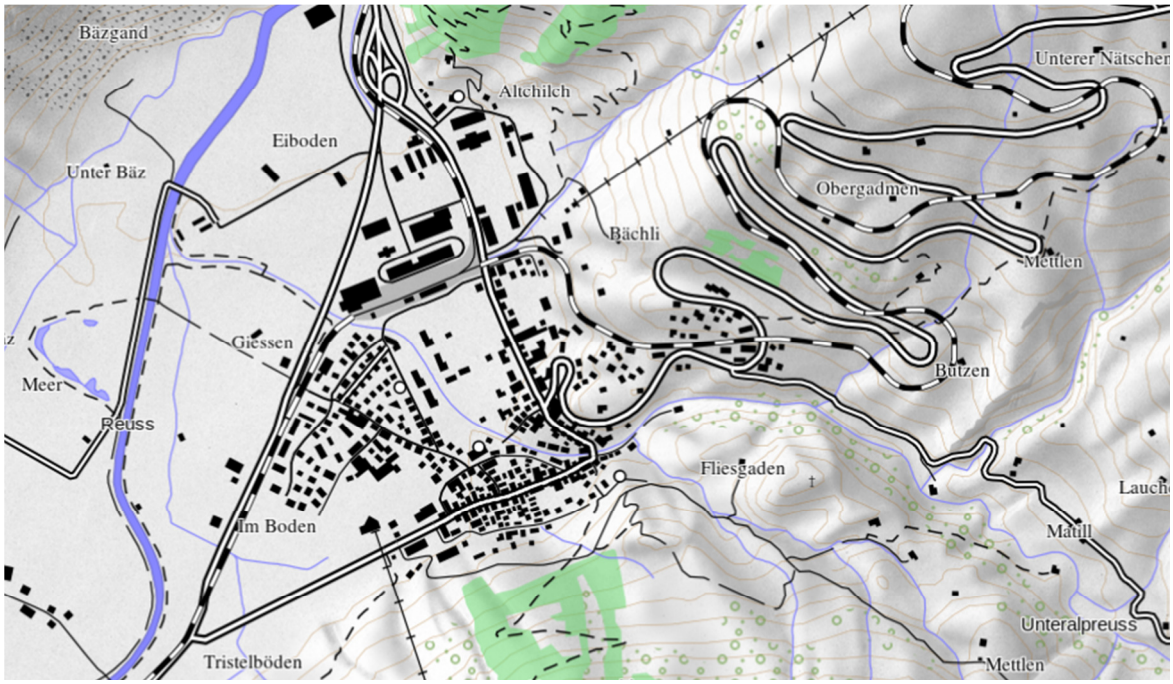
Saalfeld, Alan. „Topographically Consistent Line Simplification with the Douglas-Peucker Algorithm." Cartography and Geographic Information Science, 1999: 7-18.

Usability Net. 2006. http://www.usabilitynet.org/tools/r_international.htm (Zugriff am 20. Mai 2013).

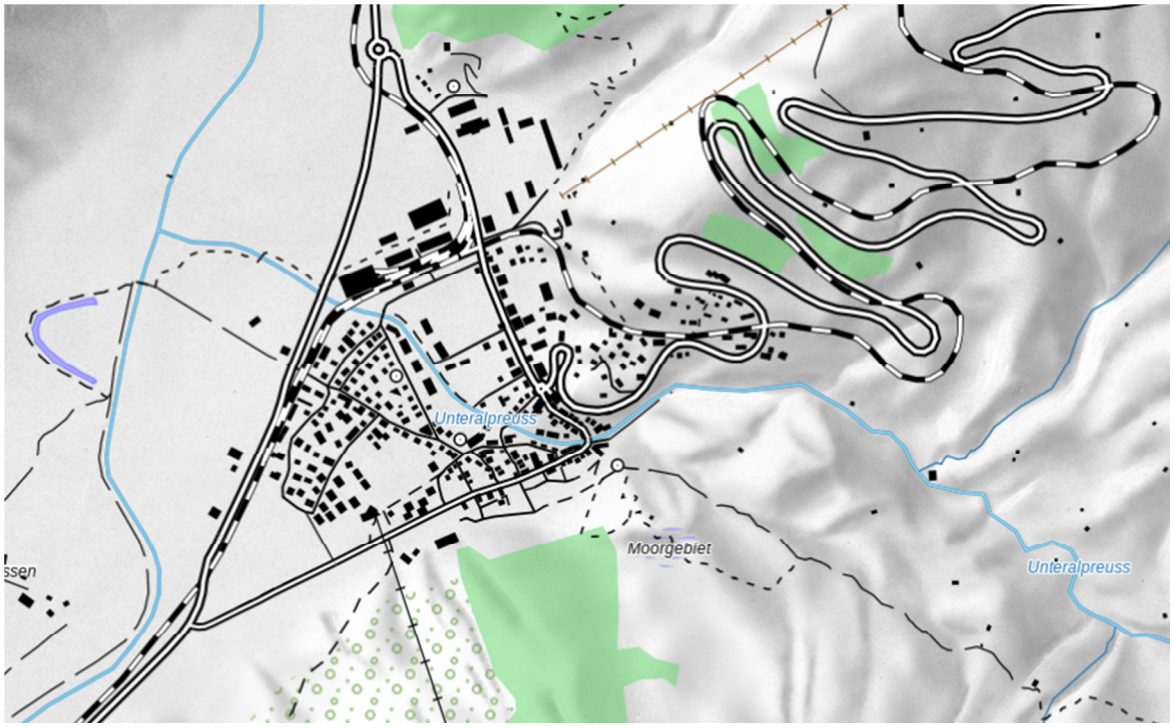
Wikipedia. 2013. https://en.wikipedia.org/wiki/File:QGis_Logo.png (Zugriff am 25. Mai 2013).

Anhang

A Vergleich OSM-Vector25



Symbolisierte Vector25- Daten © 2011 swisstopo (JA100042)



Symbolisierte OpenStreetMap-Daten

B «Objectval» Vector25 und entsprechende OSM-Werte

Die folgenden Tabellen enthalten sämtliche in den Vector25-Daten enthaltenen «Objectval» Werte[7] und die herausgesuchten, entsprechenden Werte der OSM-Daten[8].

- kein entsprechender Wert in OSM vorhanden
() könnte eventuell passender Wert sein

Einzelobjekte - Punkte	
Vector25	OSM
Antenne	-
ARA	man_made = wastewater_plant
Aussichtsturm	(man_made = tower)
Bildstock/ Wegkreuz	historic = Wayside_cross/wayside_shrine
Brunnen	man_made = water_well/ amnity = fountain
Denkmal	historic = memorial/monument
Doline	natural = sinkhole
Drehscheibe	railway = turntable
Elektrizitätswerk	-
Hafen	leisure = marina
Höhle/ Grotte	natural = cave_entrance
Hochkamin	man_made = chimney
Kapelle	amenity = place_of_worship
Kirchturm	(Kriche: amenity = place_of_worship)
Quelle	natural = spring
Reservoir	man_made = reservoit_covered/ landuse = reservoir
Schiffstation	-
Sendeanlage	-
Turm	man_made = tower
Wasserturm	man_made = water_tower
Wasserfall	-
Zisterne offen	-

Anlagen - Punkte	
Vector25	OSM
Bahnhof	railway = station
Flughafen	aeroway = terminal, aerodrome
Flughafenbahnhof	-

[7]Aus: Bundesamt für Landestopografie. (2007). *Vector25.* Abgerufen am 02. April 2013 von: http://www.swisstopo.admin.ch/internet/swisstopo/de/home/products/landscape/vector25.html
Bundesamt für Landestopografie Swisstopo. (2011). *Zeichenerklärung.* Abgerufen am 02. April 2013 von http://www.swisstopo.admin.ch/internet/swisstopo/de/home/products/accessories/brochures.html
[8] OpenStreetMap. (2013). *OpenStreetMap Wiki*. Abgerufen am 09. März 2013 von http://wiki.openstreetmap.org/wiki/DE:Map_Features

Einzelobjekte - Linien	
Vector25	OSM
Boeschungsoberkante	-
Boeschungsunterkante	-
Hochspannungsleitung	power = line
Ruine	historic = ruins
Radiosender	-
Steinboeschungoberkante	-
Steinboeschungsunterkante	-

Strassen - Linien	
Vector25	OSM
Autobahn	motorway
Autobahn richtungsgetrennt	oneway = yes
Autostrasse	trunk
Ein-/Ausfahrt (Autobahn/Strasse)	motorway_link, trunk_link, primary_link, secondary_link
Autobahnzufahrt	motorway_link
1. Klass-Strasse	primary
2. Klass-Strasse	secondary
3. Klass-Strasse	tertiary
4. Klass-Strasse	(unclassified)
5. Klass-Strasse	track
6. Klass-Strasse	path, footway
Quartierstrasse	residential/ living_street
Historischer Weg/Strasse	-
Panzerpiste	-
Parkweg	-
Alleinstehende Brücke	(bridge = yes)
Alleinstehende Brücke gedeckt	-
Alleinstehender Steg	-
Verbindungselement Wanderwege	-
BridgeType	bridge = yes
TunnelType	tunnel = yes
HikingType	sac_scale = hiking, mountain_hiking, ...

Hecken und Bäume – Linien	
Vector25	OSM
Einzelbaum (Punkt)	natural = tree
Obstbaum (Punkt)	-
Baumreihe	natural = tree_row
Hecke	barrier = hedge
Obstbaumreihe	-

Gewässernetz - Linien	
Vector25	OSM
Bach, kleiner Fluss	stream
fiktiver Bachverlauf	-
Bachverlauf unterirdisch	-
Suone (Bewässerungskanäle)	(ditch)
Druckleitung einfach	-
Druckleitung mehrfach	-
Druckstollen	-
Fluss	river
Flussverlauf unterirdisch	-
Bach ohne erkennbare Fliessrichtung	(canal)
Seeachse (fiktiv)	-
Seeinsel	-
Seeufer	-

Eisenbahn - Linien	
Vector25	OSM
Güterbahn	-
Industriegleis	service = spur/yard (yard: Rangierbhf)
Museumsbahn	preserved
Normalspurbahn eingleisig	rail
Normalspurbahn mehrgleisig	rail, tracks = Anz. Gleise
Schmalspurbahn eingleisig	narrow_gauge, funicular
Schmalspurbahn mehrgleisig	narrow_gauge, funicular, tracks = Anz. Gleise
Strassenbahn	tram, (light_rail)
Streckenverknüpfung innerhalb Bhf-Areal	-
TunnelType	tunnel = yes
BridgeType	bridge = yes

Übriger Verkehr - Linien	
Vector25	OSM
Autofähre	route = ferry (restrictions = * → welche/ob Fahrzeuge)
Luftseilbahn	cable_car, chair_lift, gondola, mixed_lift
Materialbahn	goods
Personenfähre	route = ferry (restrictions = * → welche/ob Fahrzeuge)
Skilift	drag_lift

Primärflächen - Flächen	
Vector25	OSM
Baumschule	1: plant_nursery
Fels	(2: cliff)
Fluss	waterway = riverbank
Gebüsch	(2: scrub)
Geröll mit Gebüsch	-
Geröll auf Gletscher	-
Geröll	2: scree
Geröll in Wald	-
Geröll in offenem Wald	-
Gletscher	2: glacier
Graspiste	aeroway = runway/taxiway
Piste mit Hartbelag	aeroway = runwa/taxiway
Kiesgrube	1: quarry
Lehmgrube	1: quarry
Obstanlage	1: orchard
Reben	1: vineyard
See	2: water
Siedlung	1: residential (commerical, grass, village_green)
Staudamm	waterway = dam
Staumauer	waterway = dam
Steinbruch	1: quarry
Sumpf und Gebüsch	(2: wetland UND scrub)
Sumpf	2: wetland
Sumpf in Wald	-
Sumpf in offenem Wald	-
Übriges Gebiet	1: allotsments, basin, brownfield, cemetery, farmland, 2: fell, grassland ...
Wald	1: forest
Wald offen	-

Gebäude - Flächen	
Vector25	OSM
Gebäude/ Einzelhaus	building = yes
Innenhof	-
Abgelegener Gasthof	-
Hütte	-
Kirche	amenity = place_of_worship
Kühlturm	(power = generator, generator:source = nuclear)
Lagertank	man_made = storage_tank
Perrondach	-
Schiessstand/ Schützenhaus	-
Schloss/ Burg	historic = castle
Station/ ÖV Haltestelle	railway = station, amenity = bus_station
Treibhaus	landuse = greenhouse_horticulture
Wasserbecken (Schwimmbäder, ARA)	leisure = swimming_pool, man_made = wastewater_plant

Anlagen - Flächen	
Vector25	OSM
Bahnhofareal	public_tranportation = platform, landuse = railway, railway = station
Flughafenareal	aeroway = terminal, apron, aerodrome (taxiway, runway)
Flughafenbahnhofareal	-

C Extrahieren von Substrings

Tabelle vorbereiten

```
ALTER TABLE andermatt_area ADD COLUMN Objectkey varchar(80);
ALTER TABLE andermatt_area ADD COLUMN Objectval varchar(100);

update andermatt_area set Objectkey = 'natural'
       where tags like '%natural%';

update andermatt_area set Objectval =
       substr(tags,strpos(tags, 'natural')+10,
              strpos((substr(tags,strpos(tags, 'natural')+10)), '"')-1)
       where Objectkey = 'natural';
```

Verwendete Funktionen

substr(*string, from, count*)	extrahiert aus der Zeichenkette «*string*» eine Teilzeichenkette der Länge *count* von Zeichen Nummer *from* an
strpos(*string, substring*)	sucht die Teilzeichenkette *substring* in der Zeichenkette *string* und gibt die Startposition dieser Teilzeichenkette zurück

D SQL-Code für Erstellung von Datenbank-Views

Ohne Veränderung der Geometrie

```
-- View: osm25_eis_l

-- DROP VIEW osm25_eis_l;

CREATE OR REPLACE VIEW osm25_eis_l AS
 SELECT andermatt_line.gid, andermatt_line.name, andermatt_line.objectval, andermatt_line.the_geom
   FROM andermatt_line
  WHERE andermatt_line.objectkey::text = 'railway'::text;

ALTER TABLE osm25_eis_l
  OWNER TO tabea;


-- View: osm25_pri_a

-- DROP VIEW osm25_pri_a;

CREATE OR REPLACE VIEW osm25_pri_a AS
 SELECT andermatt_area.gid, andermatt_area.name, andermatt_area.objectval, andermatt_area.the_geom
   FROM andermatt_area
  WHERE andermatt_area.objectkey::text = 'landuse'::text
        OR andermatt_area.objectkey::text = 'natural'::text
        OR andermatt_area.objectval::text = 'riverbank'::text
        OR andermatt_area.objectkey::text = 'dam'::text
        OR andermatt_area.objectkey::text = 'runway'::text
        OR andermatt_area.objectkey::text = 'taxiway'::text;

ALTER TABLE osm25_pri_a
  OWNER TO tabea;
```

Mit Veränderung der Geometrie → Generalisierung

```
-- View: osm200_eis_l

-- DROP VIEW osm200_eis_l;

CREATE OR REPLACE VIEW osm200_eis_l AS
 SELECT osm25_eis_l.gid, osm25_eis_l.name, osm25_eis_l.objectval,
        st_simplifypreservetopology(osm25_eis_l.the_geom, 30::double precision) AS the_geom
   FROM osm25_eis_l;

ALTER TABLE osm200_eis_l
  OWNER TO tabea;


-- View: osm200_pri_a

-- DROP VIEW osm200_pri_a;

CREATE OR REPLACE VIEW osm200_pri_a AS
 SELECT osm25_pri_a.gid, osm25_pri_a.name, osm25_pri_a.objectval,
        st_simplifypreservetopology(osm25_pri_a.the_geom, 30::double precision) AS the_geom
   FROM osm25_pri_a
  WHERE st_area(osm25_pri_a.the_geom) > 15625::double precision;

ALTER TABLE osm200_pri_a
  OWNER TO tabea;
```

Verwendete Funktion

ST_SimplifyPreserveTopology(geometry, tolerance)

E Beispiel für die Definition der Styles

Die folgenden Code-Beispiele sollen aufzeigen, wie die verschiedenen Styles definiert werden können. Für jeden dargestellten Layer wird pro «Objectval» ein solcher Codeblock erstellt. Er enthält sämtliche für die Symbolisierung benötigten Informationen. Dabei werden, je nach Art des Layers, nur Teile eines solchen Blocks benötigt. Zum Beispiel werden die Einstellungen für Texte nur für die Beschriftungslayer benötigt.

```
// getWMS('Areaosm25_pri_aforest';
Werte[0]['Areaosm25_pri_aforest'] = new Array();
Werte[0]['Areaosm25_pri_aforest']["ElementID"]      = 'Areaosm25_pri_aforest'
Werte[0]['Areaosm25_pri_aforest']["Art"]            = 'Area';
Werte[0]['Areaosm25_pri_aforest']["LayerName"]      = 'osm25_pri_a';
Werte[0]['Areaosm25_pri_aforest']["Objectval"]      = 'forest';
Werte[0]['Areaosm25_pri_aforest']["StrokeColor"]    = 'none';
Werte[0]['Areaosm25_pri_aforest']["StrokeWidth"]    = 0;
Werte[0]['Areaosm25_pri_aforest']["AreaColor"]      = '8dc56c';
Werte[0]['Areaosm25_pri_aforest']["Opacity"]        = 0.6;
Werte[0]['Areaosm25_pri_aforest']["DashArray"]      = "";
Werte[0]['Areaosm25_pri_aforest']["ShowThreshold"]  = 999999999;
Werte[0]['Areaosm25_pri_aforest']["Pattern"]        = null;
Werte[0]['Areaosm25_pri_aforest']["PointSymbolSize"]= 0;
Werte[0]['Areaosm25_pri_aforest']["Symbol"]         = null;
Werte[0]['Areaosm25_pri_aforest']["SvgSymbol"]      = "";
Werte[0]['Areaosm25_pri_aforest']["TextSize"]       = 12;
Werte[0]['Areaosm25_pri_aforest']["TextFamily"]     = "TESCHT";
```

Sämtliche Farben, das heisst StrokeColor, StrokeWidth und AreaColor werden mit hexadezimalen Werten angegeben. Ein Dash-Array wird mit einzelnen Werten für die Länge der Lücke und die Länge der Linie definiert.

Im folgenden Beispiel ist zu sehen, wie ein Pattern definiert werden kann.

```
var patternForest0 = '<pattern width="25" height="25" x="0" y="0">\
<g xmlns="http://www.w3.org/2000/svg">\
<path d="m 4.0158,12.24045 l -0.0539,-11.83505 0,0 0,0"
        style="fill:none;stroke:%2353ba59;stroke-width:1" />\
<path d="m 7.57235,5.89865 l -3.59291,-3.57404 -3.57404,3.59291"
        style="fill:none;stroke:%2353ba59;stroke-width:1" />\
<path d="m 7.58179,9.89696 l -3.59291,-3.57403 -3.57404,3.59291"
        style="fill:none;stroke:%2353ba59;stroke-width:1" />\
</g></pattern>';
```

Sowohl Pattern als auch Punkt-Symbole werden als SVG definiert. Ein solches SVG-Pattern kann einerseits mit einem «path» andererseits mit «WellKnownName» Formen wie zum Beispiel Kreis, Rechteck oder Linie erstellt werden. Vor allem beim Verwenden von «path»-Objekten wird die Software Inkscape zu Hilfe genommen. Dort können die Symbole gezeichnet und als SVG abgespeichert werden. Das «path»-Element kann dann aus dem erstellten File rauskopiert werden.

F Anwendung – Scripts

Im Folgenden werden die einzelnen Scripts der Anwendung mit Stichworten beschrieben.

index.svg: Hauptfile; enthält sämtliche Elemente der Benutzeroberfläche und weist ihnen eine ID zu, auf die von den anderen Scripts aus zugegriffen werden kann; definiert den Aufbau der Benutzeroberfläche

main.js enthält die init-Funktion, welche beim Laden der Anwendung ausgeführt wird; initialisiert sämtliche Elemente, zum Beispiel Buttons, Checkboxes, Sliders; ruft Funktionen auf, die für Initialisierung benötigt werden

WMSStart.js enthält alle Style-Informationen für die vordefinierten Styles; gespeichert in Codeblöcken (→ Anhang E); ruft getWMS auf

WMSBasis.js aufgebaut wie WMSStart; enthält Informationen für neutralen Symbolsatz; ruft getWMS auf

symbolizer.js greift Werte aus Eingabefenstern ab und speichert sie in gleicher Form wie Styleinfos in WMSBasis; ruft getWMS auf

getWMS.js bildet SLD aus Styleblöcken; schickt WMS-Request mit SLD an Server

uebersetzer.js weist den Bezeichnungen in den Auswahlmenüs für Art-, Layer- und Objectval-Wahl die entsprechenden Bezeichnungen aus den Daten zu (Beispiel: Strassen 1. Klasse → streets primary)

setValuesInTools.js liest die gespeicherten Symbolisierungswerte und stellt die Werkzeuge entsprechend ein (Beispiel: rote Strasse → Colorpicker wird auf Rot gesetzt)

editor.js enthält Funktionen, die für das Funktionieren der Symbolisierungstools benötigt werden; ermöglicht zum Beispiel das Zeichnen von Pattern und speichert gezeichnete Symbole

initRasterOrder.js definiert Reihenfolge der WMS-Layer, damit Rasterbilder beim Symbolisieren nur ersetzt und nicht neu erstellt werden müssen

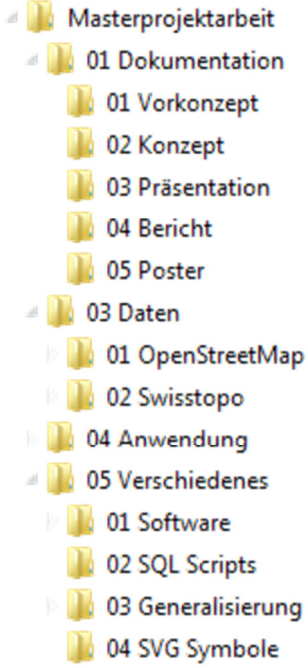
createLegend.js wird momentan nicht verwendet, soll das Erstellen einer Legende ermöglichen; Punkte, Linien und Flächen werden aufgelistet → Linien werden bis auf Dash-Array korrekt dargestellt, Flächen ohne Pattern, Punkte ohne Symbole

cartonet verschiedene Funktionen von carto.net; definiert Grundelemente wie Button, Slider, Checkbox, Fenster, ...

cron Funktionen von Juliane Cron; erstellt für Prototyp der Benutzeroberfläche

G CD

Inhalt


Masterprojektarbeit
01 Dokumentation
01 Vorkonzept
02 Konzept
03 Präsentation
04 Bericht
05 Poster
03 Daten
01 OpenStreetMap
02 Swisstopo
04 Anwendung
05 Verschiedenes
01 Software
02 SQL Scripts
03 Generalisierung
04 SVG Symbole

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

H Eigenständigkeitserklärung

Ist jeder an der ETH verfassten schriftlichen Arbeit unterzeichnet beizufügen.

Ich erkläre hiermit, dass es sich bei der von mir eingereichten schriftlichen Arbeit mit dem Titel

«Interaktive benutzerdefinierte Symbolisierung von OpenStreetMap (OSM) Daten mit QGIS Webserver»

um eine von mir selbständig und in eigenen Worten verfasste Originalarbeit handelt.

Verfasser

Name	*Vorname*
Probst	Tabea

Betreuer

Name	*Vorname*
Iosifescu-Enescu	Ionut
Panchaud	Nadia

Leitung

Name	*Vorname*
Hurni	Lorenz

Mit meiner Unterschrift bestätige ich, dass ich über fachübliche Zitierregeln unterrichtet worden bin und das Merkblatt (http://www.ethz.ch/students/exams/plagiarism_s_de.pdf) gelesen und verstanden habe. Die im betroffenen Fachgebiet üblichen Zitiervorschriften sind eingehalten worden.

Eine Überprüfung der Arbeit auf Plagiate mithilfe elektronischer Hilfsmittel darf vorgenommen werden

Zürich, 31. Mai 2013 _____

Ort, Datum Unterschrift