Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

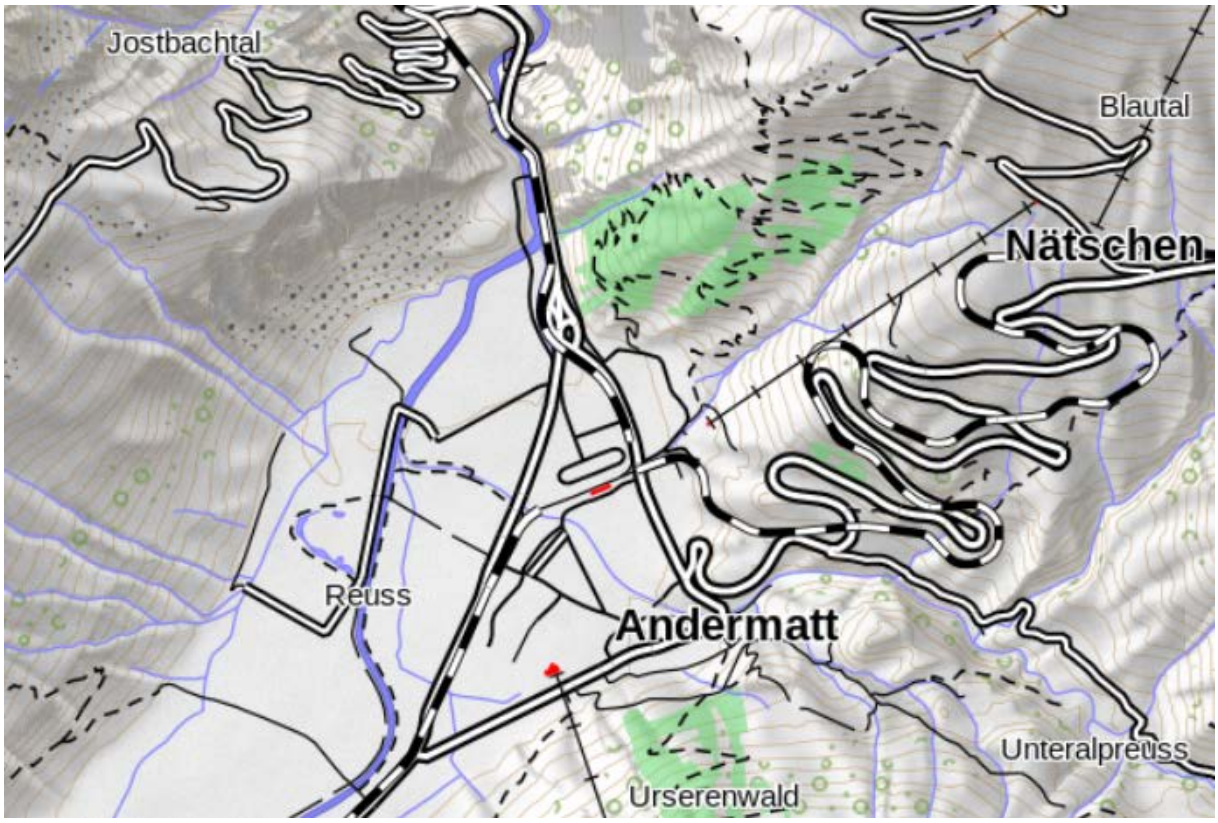
IKG
Institut für Kartografie
und Geoinformation

«Interaktive benutzerdefinierte Symbolisierung von Vektordaten mittels erweiterten OGC-Standards»

Masterarbeit

Herbstsemester 2012

Geomatik und Planung



Christian Lorenz
chlorenz@ethz.ch

Betreuung
Dr. Ionut Iosifescu-Enescu

Leitung
Dr. Prof. Lorenz Hurni

25. Januar 2013

Institut für Kartografie und Geoinformatik (IKG)
Departement Bau, Umwelt und Geomatik (D-BAUG) ETH Zürich

I. Vorwort und Dank

Die vorliegende Masterarbeit «Interaktive benutzerdefinierte Symbolisierung von Vektordaten mittels erweiterten OGC-Standards» ist der Abschluss des Studiengangs Geomatik und Planung. Sie wurde im neunten und letzten Semester am Institut für Kartografie und Geoinformation (IKG) der ETH Zürich geschrieben.

Auf der Suche nach einem Thema für die Masterarbeit schaute ich mich mir die Angebote im Bereich meiner beiden Vertiefungsrichtungen an. Das heisst am Institut für Geodäsie und Photogrammetrie (IGP) sowie am IKG. Ich fand in diesen Angeboten zwei Arbeiten welche mich interessierten wobei beide im Bereich der Kartografie angesiedelt waren.

Ich hatte also Kontakt mit Lorenzo Oleggini wegen einer Arbeit im Bereich Orientierungslauf sowie mit Ionut Iosifescu-Enescu welcher mir in einem Gespräch diese Arbeit vorgeschlagen und erklärt hat. Am Schluss entschied ich mich für das Angebot von Ionut und erstellte ein kleines Vorkonzep.

Einen Einblick in die benützte Technologie erhielt ich schon in der Vorlesung «Multimedia Cartography» welche ich im Frühlingssemester 2012 besucht hatte. Dort realisierten wir die Symbolisierung einiger Flüsse mittels SLD. Dieser Einstieg und die Tatsache, dass ein grosser Teil der Arbeit Programmierarbeit sein würde, bestärkten mich in der Entscheidung für diese Arbeit.

Bedanken möchte ich mich vor allem bei Ionut, welcher bei Fragen immer eine gute Lösung parat hatte und mir so helfen konnte. Mein allgemeiner Dank geht an das IKG unter der Leitung von Lorenz Hurni, welches mir die Arbeit ermöglichte und die Infrastruktur zur Verfügung stellte.

Ein weiterer Dank geht an meine Mitstudenten, welche immer wieder für erheiternde Momente in den unseren schon fast standardisierten Pausen sorgten.

II. Zusammenfassung

In dieser Masterarbeit werden die Schritte beschrieben, welche nötig sind um einen Prototypen herzustellen, welcher dem User ermöglichen soll Vektordaten nach seinen Wünschen zu symbolisieren und so eine Karte in seinem eigenen Design zu erstellen. Und dies ohne, dass der er selber Wissen über die verwendete Technologie haben muss.

Für diese Arbeit wird die Serviceorientierte Architektur verwendet, wobei hier Webservices gemeint sind. Es wurde der, vom «Open Geospatial Consortium» definierten, «Web Map Service» verwendet. Wobei beim benützten Mapserver [karlinapp.ethz.ch] kartografische Erweiterungen implementiert sind, welche eine bessere Darstellungsqualität erlauben. Diese Erweiterungen sind zum Beispiel Pattern für Flächensignaturen oder die Möglichkeit in SVG definierte Symbole für Punkte zu verwenden. Dass diese Erweiterungen funktionieren und qualitativ gute Karten erstellt werden könne wurde mit mehreren Projekten gezeigt [Iosifescu et al., 2011 und Ortner 2011].

Bei diesen Projekten hat aber der User nur wenige Möglichkeiten auf die Darstellung der Daten Einfluss zu nehmen. Hier setzt diese Arbeit an. Es geht darum ein webbasierten Prototypen zu erstellen, welcher benutzerdefinierte Symbolisierung von Vektordaten mittels Benutzeroberfläche ermöglicht.

Nach einer Einarbeitungsphase wurde eine vorhandene Benutzeroberfläche (GUI) übernommen und angepasst. Dabei wurde eine erste Symbolisierung erstellt welche der der swisstopo ähnlich ist. Danach wurden die Werkzeuge für die Linien, Flächen, Punkte und Texte erstellt, welche dem User die Symbolisierung ermöglichen. Dabei ist zu erwähnen, dass auch ein einfacher Editor erstellt wurde, welcher ermöglicht, dass eigene Pattern und SVG-Punktsymbole erstellt werden können.

Der fertige Prototyp ermöglicht die Interaktive benutzerdefinierte Symbolisierung von Vektordaten mittels erweiterten OGC-Standards, wobei nicht alle Möglichkeiten des Mapservers ausgenützt wurden und somit noch Potential für kartografisch besser Karten vorhanden ist.

III. Inhaltsverzeichnis

IV. Abbildungsverzeichnis

V. Tabellenverzeichnis

1 Einleitung

1.1 Einführung und Problemsituation

Einfache Karten mittels Geodaten-Webservices anzubieten funktioniert, mit den verschiedenen Webservices, welche durch Open Geospatial Consortium (OGC) definiert wurden, ohne Probleme. Wird mit diesen Services gearbeitet wird dienstbasiert gearbeitet. Eine grundsätzliche Struktur dieser Architektur (Serviceorientierte Architektur, SOA) ist in Abbildung 1.1 zu sehen.

Abbildung 1.1: Dienstbasierte Architektur

Dabei werden die Anfragen des Clients, welche je nach Typ des Services unterschiedlich sind, vom Map-Service verarbeitet und die, der Anfrage entsprechenden Daten gesammelt und zusammengestellt. Diese Zusammenstellung, wieder je nach Typ des Services sehr unterschiedlich, wird dann dem Client zurückgegeben. Dieser kann dann die Antwort nach seinen Vorgaben weiterverwenden.

Diese Architektur wird auch für den «Web Map Service» verwendet. Dieser Service liefert immer ein Rasterbild. Dieses wird aus aber aus Vektordaten erstellt und die Darstellung kann von Client definiert werden. Dies geschieht über einen XML-Dialekt, nämlich mit dem «Styled Layer Descriptor» (SLD). Mit den, in der Definition von SLD vorhandenen Möglichkeiten ist keine Kartografisch gesehen schöne Symbolisierung möglich. Daher wurden Erweiterungen zu SLD entwickelt. [Iosifescu-Enescu, 2011] Diese Erweiterungen ermöglichen kartografisch bessere Karten automatisch zu erstellen. Die Erweiterungen sind auf dem Server des QGIS-mapserver-Projektes [karlinapp.ethz.ch, 2013] implementiert und somit anwendbar für weitere Untersuchungen.

Nadia Panchaud [2012] konnte zeigen, dass die Serviceorientierte Architektur auch für 3-dimensionale Anwendungen brauchbar ist. Dabei wurden verschiedenste Web Services kombiniert und untersucht.

Auch wurde gezeigt, dass Thematische Karten mit SOA möglich sind. [Ortner, 2011] Dabei wurden die oben erwähnten Erweiterungen in SLD des QGIS-Mapserver-Projektes benützt.

GeoVITe, ein Projekte welches Kartendaten Mitglieder der ETH einfacher verfügbar macht, zeigt dass SOA auch im Umgang mit grossen Datenmengen und verschiedensten Datensätzen möglich ist. [Iosifescu et al., 2011]

Bei allen diesen Arbeiten oder Projekten hat der Benutzer keine oder nur wenige Möglichkeiten die Darstellung der Daten seinen Bedürfnissen anzupassen, ohne dass er sich in den verwendeten Technologien auskennt. In dieser Arbeit soll nun gezeigt werden, dass eine solche Interaktion möglich ist. Und zwar so, dass der User nur die Benutzeroberfläche kennen muss, und keine Kenntnisse über irgendwelche Code-Sprachen haben muss.

1.2 Ziele

Aus den Überlegungen oben ergeben sich folgenden Ziele

- Zeigen, dass die Interaktive benutzerdefinierte Symbolisierung von Vektordaten mittels erweiterten OGC-Standards funktioniert
    - Erstellung eines webbasierten Prototyps
    - möglichst hohe kartografische Qualität erreichen

1.3 Inhaltsübersicht

Nach der Einleitung im Kapitel 1 wird im Kapitel 0 «5 wird ein Ausblick gegeben, was für weitere Aufgaben oder Arbeiten in diesem Bereich und am Prototyp denkbar und nötig wären.

Grundlagen» auf die verwendeten Daten und deren Entstehung eingegangen. Auch werden die verwendeten Programme und Technologien beschrieben und je nach Bekanntheit genauer beschrieben.

Im Kapitel 3 «Methoden und Vorgehen» werden chronologisch die Arbeitsschritte beschrieben, welche nötig waren um zum fertigen Prototyp zu gelangen. Diese Schritte sind mit einigen Screenshots und Codeschnipseln erläutert. Im folgenden Kapitel 4 «Ergebnisse» wird der Prototyp und dessen Funktionsweise genauer erklärt.

Im Kapitel 5 «Fazit» wird ein Fazit über die Arbeit gezogen wobei auch die Erreichung der Ziele bewertet wird. Im abschliessenden Kapitel 5 wird ein Ausblick gegeben, was für weitere Aufgaben oder Arbeiten in diesem Bereich und am Prototyp denkbar und nötig wären.

2 Grundlagen

2.1 Technologien und Programme

In diesem Kapitel werden die in der Arbeit benützten Programme und Technologien beschrieben. In der Abbildung 2.1 ist eine Übersicht vorhanden, wie die verschiedensten Teile miteinander verknüpft sind.

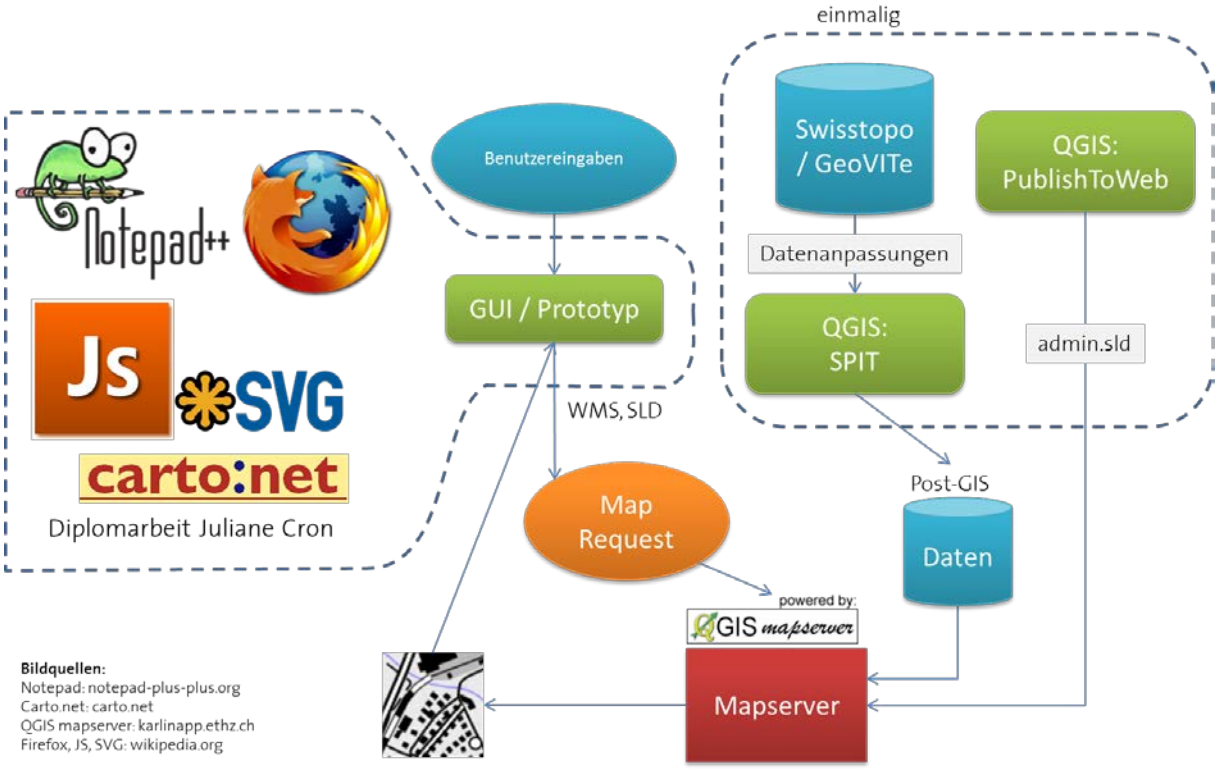


Abbildung 2.1: Zusammenspiel der verschiedenen Programme und Technologien

2.1.1 WMS

WMS ist die Abkürzung für «Web Map Service». Dies ist ein vom «Open Geospatial Consortium» (OGC) definierter Standard-Webservice. Mittels WMS lassen sich Karten oder Teile davon in eine Website einbinden. Der Server produziert nach Angaben in der Anfrage ein Rasterbild (meist PNG oder JPG), welches danach ausgeliefert wird. Der grundsätzliche Aufbau einer solchen Servicebasierten Architektur ist in der Abbildung 2.2 gezeigt.

2.1.1.1 SLD

SLD ist die Abkürzung für «Styled Layer Descriptor». Dies ist ein XML-Schema welches von OGC definiert wurde um das Aussehen von Daten eines WMS zu definieren. So kann einer WMS-Anfrage eine SLD-Definition mitgegeben werden, so dass eine auf dem Server vorhandene Geometrie nach den eigenen Wünschen dargestellt wird. Diese Definitionen sind abhängig vom Geometrietyp (Punkt, Linie, Fläche, Text)

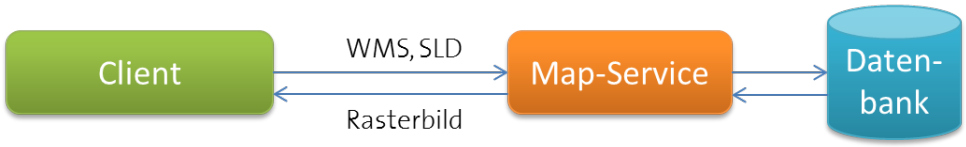


Abbildung 2.2: Dienstbasierte Architektur mit WMS und SLD

2.1.2 QGIS-Mapserver

QGIS-Mapserver ist eine Opensource WMS Implementation welche darstellerische Erweiterungen enthält. Mit diesen Erweiterungen lassen sich automatisch Karten generieren, welche eine höhere kartografische Qualität aufweisen als mit der Standard SLD-Definition. Der Server ist in der Programmiersprache C++ realisiert. [karlinapp.ethz.ch, 2013]

2.1.3 QGIS

QGIS ist ein Open Source Programm mit welchem Geodaten dargestellt und bearbeitet werden können. Mittels Plugins lassen sich die Funktionalitäten fast beliebig erweitern. Das Programm ist in der Version 1.8.0 erhältlich.

2.1.3.1 SPIT

SPIT ist ein Plugin für QGIS. Mit diesem Plugin lassen sich Daten in Form von Shape-Files in eine PostGIS-Datenbank laden. Eine solche Datenbank wurde für die Arbeit zur Verfügung gestellt.

2.1.3.2 PublishToWeb

Das Plugin «PublishToWeb» für QGIS ermöglicht einen Export von symbolisierten Vektordaten bei dem ein Projektordner erstellt wird. In diesem Projektordner ist dann ein admin.sld-File vorhanden, in welchem alle in QGIS geladenen Daten inklusive der dort ausgewählten Symbolisierung vorhanden sind.

2.1.4 SVG

«Scalable Vector Graphics» ist ein XML-Dialekt, welcher vom «World Wide Web Consortium» empfohlen wird. Da SVG ein XML-Dialekt ist, lassen sich die Dateien von Hand in einem Texteditor oder mittels spezieller SVG-Editoren bearbeiten. Mit SVG lassen sich verschiedenste Formen, wie Linien, Kreise, Rechtecke, Pfade und vieles mehr, definieren. Die meisten Browser unterstützen SVG und können die Grafiken ohne Plugin anzeigen.

Durch diese Voraussetzungen ist SVG im Web die einfachste Möglichkeit Vektorbasierte Objekte darzustellen.

2.1.5 Javascript

Javascript ist eine Scriptsprache, welche in Webseiten verschiedenster Technologie einsetzbar ist. Mittels Javascript lassen sich die Objekte von Webseiten verändern. Dies ermöglicht also eine Interaktion mit dem User. Mittels Javascript lassen sich also auch SVG-Elemente verändern oder erstellen. Diese Möglichkeit wurde in dieser Arbeit genutzt.

2.1.6 carto.net

carto.net ist eine Website welche die Veröffentlichung von verschiedensten kartografisch interessanten Projekten ermöglicht. Auf dieser Website sind verschiedenste Bedienobjekte vorhan-

den, welche für ein GUI genutzt werden können. Dies sind Objekte wie Knöpf, DropDown-Menüs, Slider und so weiter.

Diese Objekte wurden in Javascript erstellt und können sehr einfach in ein SVG/Javascript-GUI eingebunden werden.

2.1.7 Notepad++

Notepad++ ist ein Open Source Texteditor, welcher sehr viele Programmiersprachen erkennt und dementsprechend einzelne Wörter, Zeilen oder Buchstaben einfärbt. Diese Einfärbung erleichtert das Programmieren erheblich. Auch die «Suchen und Ersetzen»-Funktion ist eines der wichtigeren Tools in Notepad++ welches die Arbeit erleichtern.

2.1.8 ArcGIS

ArcGIS von ESRI ist ein Geoinformationssystem welches weit verbreitet ist und auch an der ETH verwendet und gelehrt wird.

In dieser Arbeit wurde das Programmteil ArcMap verwendet. Dies um einige Anpassungen an den Shape-Files zu realisieren, welche in ArcGIS einfacher als in QGIS waren.

2.1.9 Mozilla Firefox

Mozilla Firefox ist ein Open Source Webbrowser. Die aktuelle Version ist Firefox 17. Der Browser kann SVG ohne Plugin darstellen, auch Javascript wird ohne Probleme verarbeitet.

Für diese Arbeit wurde nur dieser Browser benützt. Dementsprechend ist die Kompatibilität nur für Firefox getestet.

2.1.9.1 Firebug

Firebug ist ein Plugin für Firefox welches bei der Entwicklung von Webseiten helfen kann, denn es zeigt Fehler in Javascript an oder man kann das aktuelle HTML- oder SVG-Dokument anschauen und auch verändern. Diese Möglichkeiten erleichtern das Debuggen des eigenen Codes. In Abbildung 2.3 ist ein Screenshot der Fehlerkonsole von Firebug zu sehen, in welcher ein Javascript- Fehler angezeigt wird.

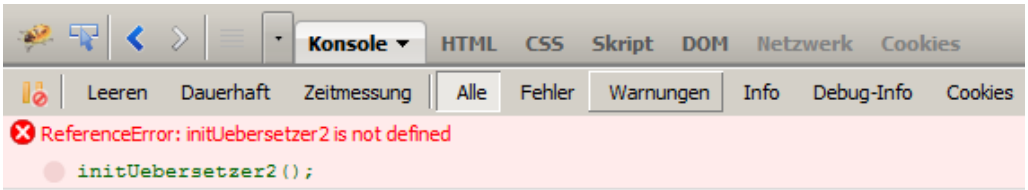


Abbildung 2.3: Firebug - Fehlerkonsole

2.1.10 GeoVITe

GeoVITe ist eine Online-Plattform (geodata.ethz.ch/geovite/) auf der man als ETH Mitarbeiter Geodaten der swisstopo beziehen kann. Auf der passwortgeschützten Website kann man sehr viele verschiedene Daten auswählen. Dies geschieht mit einigen Werkzeugen wie Layerauswahl und einem Rechteck für die Gebietsauswahl.

2.2 Daten

2.2.1 VECTOR25

Der VECTOR25 Datensatz der swisstopo ist ein digitales Landschaftsmodell welches auf der Landeskarte 1:25 000 basiert. Der Datensatz besteht aus den neun thematischen Ebenen, welche als einzelne Shape-Files ausgeliefert werden:

- Strassennetz
- Übriger Verkehr
- Primärflächen
- Hecken und Bäume
- Einzelobjekte
- Eisenbahnnetz
- Gewässernetz
- Gebäude
- Anlagen

Die einzelnen Objekte in den Shape-Files haben einige Attribute, welche weitere Informationen über die Objekte beinhalten. Vor allem ein Attribut (objectval) ist für diese Arbeit von zentraler Bedeutung. Zum Beispiel wird in diesem Attribut definiert welche Klasse eine Strasse hat oder um was für eine Primärfläche es sich handelt. Die neusten Daten für das Untersuchungsgebiet stammen aus dem Jahre 1999 und 2003. [VECTOR25]

2.2.2 VECTOR200

Der Datensatz VECTOR200 ist das Pendant zu VECTOR25, jedoch basiert dieser auf den Landeskarten 1:200 000. Die Datenstruktur von VECTOR25 und VECTOR200 ist nicht genau gleich. Auf diese Unterschiede wird im Kapitel 3.4.1 eingegangen.

Der VECTOR200-Datensatz hat folgende Layer, welche als einzelne Shape-Files ausgeliefert werden: [VECTOR200]

- Transportation — Strassen- und Eisenbahnnetz, weitere Elemente im Zusammenhang mit dem Verkehr
- Hydrography — Gewässerachsen und Seen
- Landcover — Primäre Bodenbedeckung (Wald, Siedlung etc.)
- Buildings — Stark generalisierte Einzelgebäude
- Miscellaneous — Verschiedenes
- Names — Namen
- Administrative Boundaries — Administrative Grenzen (Landesgrenzen bis Gemeindegrenzen), Nationalparks

2.2.3 swissTLM3D

swissTLM3D ist ein Landschaftsmodell der swisstopo, welches die Schweiz möglichst genau und unabhängig eines Massstabs dreidimensional abbilden soll. Das Modell befindet sich im Moment im Aufbau, wobei einige Objekte direkt aus VECTOR25 übernommen wurden. [swissTLM3D]

2.2.4 Relief

Die swisstopo benützt für ihre Landeskarten Reliefs, welche von Hand produziert wurden. Diese Reliefs sind in GeoVITe als hochwertige Scans verfügbar. Das heisst sie können blattschnittfrei bezogen werden. Die ausgelieferten Dateien haben das Rasterdateiformat TIF. [Relief]

2.2.5 Landeskarte

Die Landeskarten der swisstopo werden in den Massstäben 1:25 000 bis 1:1000 000 produziert. Scans dieser Karten sind in verschiedenen Varianten in GeoVITe blattschnittfrei verfügbar. Diese Daten werden als TIF-Dateien zur Verfügung gestellt. [Landeskarte]

2.2.6 DHM25 - Basismodell

Das digitale Höhenmodell 25 (DHM25) ist ein Höhenmodell, welches auf den Landeskarte 1:25 000 basiert. Die Höhenlinien der Karten wurden halbautomatisch digitalisiert und bilden so das DHM25 – Basismodell. Diese digitalisierten Linien sind als Shape-Files auf GeoVITe verfügbar. Leider ist kein Höhenattribut für die Linien verfügbar. So ist es nicht möglich die Linien je nach Höhe unterschiedlich darzustellen. [DHM25]

2.2.7 SwissNames

SwissNames ist ein weiterer Datensatz der swisstopo. Dieser beinhaltet alle Namen, welche auf den Landeskarten 1:25 000 bis 1:500 000 eingetragen sind. Diese Objekte sind als Punkte mit einigen Attributen gespeichert. Das wichtigste ist «name» welches dem Text aus der Karte entspricht. In dieser Arbeit wurden die Datensätze für die Massstäbe 1:25 000 und 1:200 000 benützt. [SwissNames]

3 Methoden und Vorgehen

In diesem Kapitel wird das Vorgehen während der Arbeit beschrieben. Die Reihenfolge der Unterkaptiel entspricht der Reihenfolge, in der die beschriebenen Vorgänge abgearbeitet wurden.

3.1 Gebietsauswahl

Für die Arbeit wurde ein flächenmässig relativ kleines Gebiet ausgewählt. Dies um Probleme mit der Performanz und Dateigrössen zu umgehen.

Die Wahl für das Gebiet fiel auf Andermatt und Umgebung, weil in diesem Gebiet viele Objekte, welche in den Daten vorkommen können, vorhanden sind, das heisst es ist recht repräsentativ für den Datensatz der gesamten Schweiz. Der gewählte Ausschnitt ist in Abbildung 3.1 visualisiert.

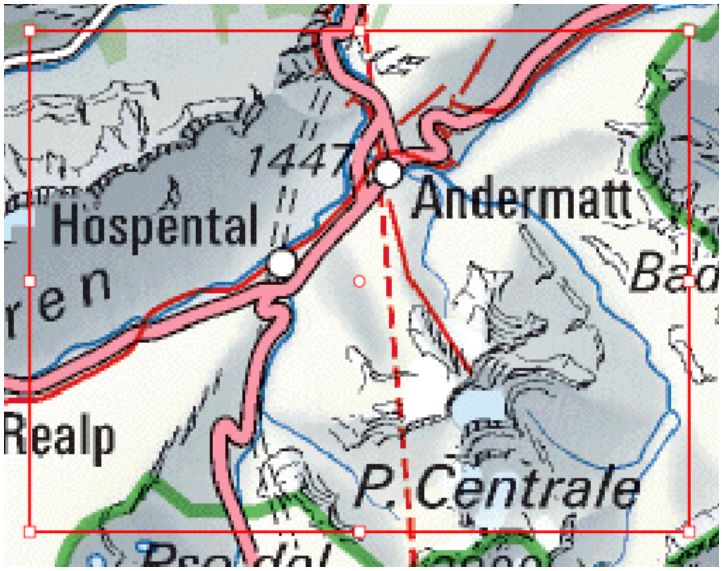


Abbildung 3.1: Ausschnitt des Testgebietes auf der Landeskarte 1:500 000
(Aus GeoVITe, Reproduziert mit Bewilligung von swisstopo (JA100120))

3.2 Datenauswahl

Die Datenauswahl geschah auf Grund der Verfügbarkeit von VECTOR25 und VECTOR200. Dies bestimmte die beiden Massstabsstufen 1:200 000 und 1:25 000, welche im Verlauf der Arbeit für die Auswahl von weiteren Daten verwendet wurden.

3.2.1 Ergänzungen mit swissTLM3D

Da die Seilbahnen (Luftseilbahnen, Skilifte, Materialbahnen) im VECTOR25 Datensatz fehlen wurden diese Daten aus dem swissTLM3D genommen. Es handelt sich um einige Linienobjekte, welche aber im gesamten Kartenbild einen grossen Unterschied machen.

3.2.2 Landeskarte

Um die aus den Vektordaten produzierte Karte vergleichen zu können wurden die Pixelkarten mit den dementsprechenden Massstäben eingefügt. Die Umschaltung erfolgt automatisch anhand eines Grenzwertes, welcher von der Zoomstufe abhängt. Diese Karten können ein und ausgeschaltet werden und deren Transparenz kann mittels eines Sliders eingestellt werden.

3.2.3 Relief

Das Relief (in den Massstäben 1:200 000 und 1:25 000) wurde integriert um die Geländeformen darzustellen. Diese beiden Rasterbilder wurden gleich wie die Landeskarten integriert und werden somit automatisch umgeschaltet. Sie sind per Benutzereingabe ein und ausschaltbar und die Transparenz ist anpassbar.

3.2.4 Höhenlinien

Da die Höhenlinien nur für die Landeskarte 1:25 000 digitalisiert wurden, sind die Höhenlinien der Landeskarte 1:200 000 nicht verfügbar. So werden im Prototyp nur auf den Zoomstufen grösseren Massstabs Höhenlinien angezeigt.

3.3 Anpassung GUI

Als Grundlage für den geplanten Prototyp diente ein GUI, welches Juliane Cron [Cron 2006] für den Atlas der Schweiz erstellt hatte. Dieses GUI hat schon viele grundlegende Funktionen (Zoom, Pan, ...) integriert und es bietet, dank der Bedienelementen von carto.net [carto.net], viele Erweiterungsmöglichkeiten. Ausserdem wurde das Farb- und Designkonzept ziemlich genau übernommen.

Das vorhandene GUI musste also an die Anforderungen von dieser Arbeit angepasst werden. Dafür wurden manche Funktionen, wie zum Beispiel die Möglichkeit zum Aufteilen des Hauptfensters in zwei Karten, herausgenommen. Die dazugehörigen Javascript-Funktionen wurden gelöscht.

3.4 Datenaufbereitung

Um die Erweiterbarkeit auf die ganzen Datensätze (ganze Schweiz) zu erhalten wurden alle Veränderungen nur global, das heisst bei allen Objekten eines Datensatzes oder Shapefiles, vorgenommen. Es wurden also wurden keine Veränderungen gemacht, welche abhängig sind von einzelnen Objekten welche im Testgebiet vorkommen.

3.4.1 Unterschiede zwischen VECTOR25 und VECTOR200

Die beiden Datensätze unterscheiden sich in einigen Punkten. Einer der wichtigsten ist, dass das wichtige Attribut in VECTOR25 «objectval» und in VECTOR200 «objval» heisst. Dabei sind die Einträge in diesem Attribut meist selbsterklärend, aber müssen in den beiden Datensätzen nicht übereinstimmen. Einige Solche Beispiele sind in der Tabelle 3.1 ersichtlich.

Tabelle 3.1: Unterschiede der Attribute zwischen VECTOR200 und VECTOR25

Layername VECTOR25	Layername VECTOR200	objectval VECTOR25	objval VECTOR200
VEC25_geb_a	VEC200_Building	Gebaeude	Z_Gebaeude
VEC25_pri_a	VEC200_LandCover	Z_Glet	Gletscher
VEC25_pri_a	VEC200_LandCover	Z_Geroel	Geroell
VEC25_eis_l	VEC200_Railway	Luftseilbahn	Seilbahn
VEC25_str_l	VEC200_Road	HauptStrAB6	1_Klass
VEC25_str_l	VEC200_Road	Fahrstraes	4_Klass
VEC25_str_l	VEC200_Road	Fussweg	6_Klass

3.4.3 Bearbeiten Höhenlinien

Um SPIT nutzen zu können und so die Daten in die PostGIS-DB zu bringen, mussten die durch das Clippen beim Download von GeoVITe zu Multilines gewordenen Höhenlinien wieder zu einfachen Linienobjekten gemacht werden. Dies wurde mit dem Tool «MultiPart to Singlepart» aus der ArcMap-Toolbox berechnet. Dieses bearbeitet Shape-File konnte nun ohne Probleme mit SPIT in QGIS auf den Server und somit in die PostGIS-DB geladen werden. In der Abbildung 3.2 ein Beispiel eines solchen Falles gezeigt.

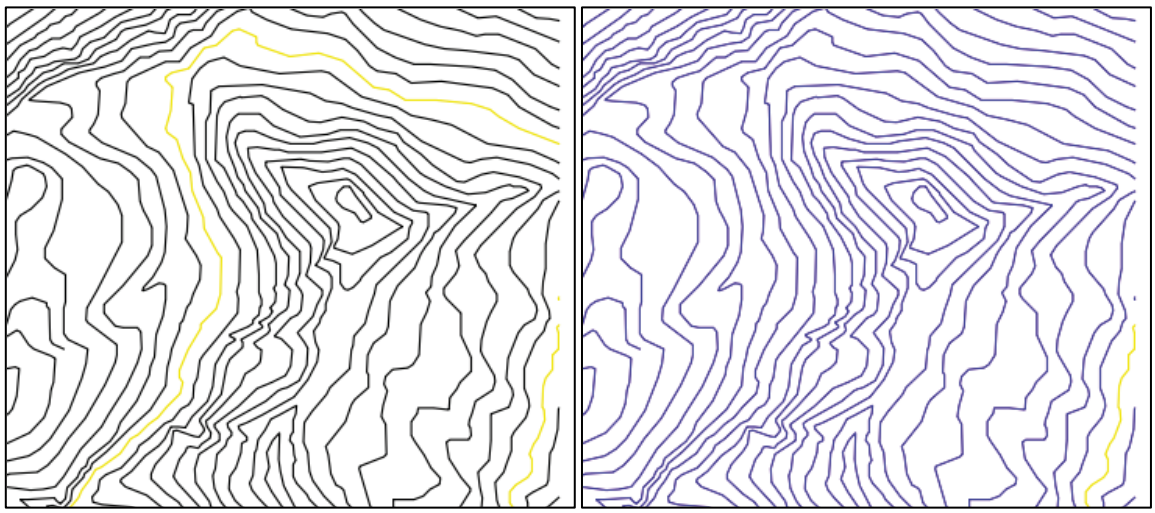


Abbildung 3.2: Höhenlinien als Multipart und Singlepart Objekte (selektiertes Objekt in Gelb) (Reproduziert mit Bewilligung von swisstopo (JA100120))

3.4.4 Upload der Shape-Files in PostGIS-DB

Alle Daten waren in Form von Shape-Files vorhanden und wurden mit SPIT (siehe 2.1.3.1) in eine vorbereitete PostGIS-Datenbank auf dem karlinapp.ethz.ch-Server geladen. Dies um danach via WMS darauf zugreifen zu können.

3.4.5 Views für Namen generieren

Da im Attribut «Objectval» der Shape-Files für die Beschriftungen sehr viele verschiedene Objekte möglich und im Testgebiet auch vorhanden sind, wurden die Objekte kategorisiert. So dass der Benutzer aus einer sinnvollen Anzahl Klassen auswählen kann. Um dies zu realisieren wurden Datenbank-Views generiert, welche danach wie eine normale Tabelle via WMS aufrufbar sind. Die in der Tabelle 3.2 aufgelistete Zusammenstellung entstand auf Basis der vorhandenen Werte in den Datensätzen für 1:200 000 und 1:25 000.

Im Anhang B ist ein Beispiel eines SQL-Codes zu finden mit welchem ein solcher View erstellt werden kann.

Tabelle 3.2: Kategorisierung der Beschriftungsobjekte

Klassenname	Werte im Attribut «Objectval»
Flurnamen	Flurname, Einzelhaus,
Gemeinde	KGemeinde, Weiler
Gipfel	KGipfel, GGipfel
Täler	Graben, Nebental
Grate	Grat
Natur	Bach, KBach, Fluss, Gletscher, KSee, Sumpf, Wald
Wege	Weg, Fusspass
Bauwerke	Tunnel, Bahnhof, OeffGeb, Ruine, Kirche, Bruecke
Felsen	Fels

Hier ist zu beachten, dass nicht alle Werte im VECTOR200 Datensatz vorkommen.

3.4.6 Umlaute in SwissNames

Da durch den Import der Namensdaten in die PostGIS-DB die Umlaute durch andere Zeichenkombinationen ersetzt wurden, mussten diese wieder zurückersetzt werden um die richtigen Namen in der Karte angezeigt zu bekommen. Diese Ersetzung wurde mit der «replace»-Operation des QGIS-«Field Calculators» in der Attributtabelle realisiert. In der Tabelle 3.3 sind die Sonderzeichen und die dazu passenden Umlaute zu finden.

Tabelle 3.3: Umlaute zu Sonderzeichen

Umlaut	Sonderzeichen	Beispielcode für Field-Calculator
ä	Ã¤	replace(name, 'Ã¤', 'ä')
ö	Ã¶	replace(name, 'Ã¶', 'ö')
ü	Ã¼	replace(name, 'Ã¼', 'ü')
Ä	Ã„	replace(name, 'Ã„', 'Ä')
Ü	Ãœ	replace(name, 'Ãœ', 'Ü')

3.4.7 Anpassung aller Vektordaten an VECTOR25

Wie in Kapitel 3.4.1 beschrieben sind die Datensätze unterschiedlich. Um diese Unterschiede auszumerzen wurden alle Datensätze auf den «Standard» der VECTOR25-Daten angeglichen. Das heisst es wurden einheitliche Dateinamen eingeführt. Einige davon sind in der Tabelle 3.4 aufgelistet. Auch wurden in allen Datensätzen, ausser VECTOR25, ein neues Attribut mit dem Namen «objectval» erstellt und da, bei den Objekten entsprechenden Werte eingefügt. Diese Werte entsprechen meist den passenden aus dem VECTOR25-Datensatz oder wurden selbst erstellt.

Tabelle 3.4: Namensstruktur

Beschreibung	Datensatz	Ursprünglicher Name	Neuer Name
Strassen	VECTOR25	VEC25_str_l	VEC25_str_l
Strassen	VECTOR200	VEC200_Road	VEC200_str_l
Höhenlinien	DHM25	Dhm25_lbasis	VEC25_hhl_l
Seilbahnen	swissTLM3D	TLM_UEBRIGE_BAHNEN	VEC25_tsp_l
Namen	SwissNames25	sn25	VEC25_nam_p

3.6 Erste Layer darstellen

Mit den vorbereiteten Daten und dem angepassten GUI konnten nun Versuche gestartet werden um erste Layer via WMS und SLD darzustellen. Dafür wurde auf eine Javascript-Funktion (getWMS()) zurückgegriffen, welche im Kurs «Multimedia Kartografie» [Multimedia Cartography, FS 2012] benutzt wurde. Das diese Funktion richtig arbeiten kann, muss ein WMS-Server bereitgestellt werden. Dies wurde in der Anfangsphase mit einem lokalen Apache-Server gemacht. Dieser Server samt allen SLD-Erweiterungen stammt von der Website des QGIS-Mapserverprojektes [karlinapp.ethz.ch]. Später wurde die admin.sld-Datei auf den Server transferiert und die Abfragen in der getWMS()-Funktion angepasst, so dass kein lokaler Server mehr nötig ist.

Mittels QGIS und dem Plugin «PublishToWeb» (siehe Kapitel 2.1.3.2) liess sich eine admin.sld Datei erstellen, welche vom oben genannten WMS-Server benützt wird um auf die Daten zuzugreifen.

3.7 Erste Symbolisierung (swisstopo ähnlich)

Nachdem erste Layer in gewünschter Symbolisierung angezeigt werde konnten, wurde eine Standardsymbolisierung aufgebaut. So dass der Benutzer des Prototyps von Anfang an eine Karte angezeigt bekommt. Für dieses Unterfangen wurde die schon bekannte Javascript-Funktion «getWMS()» weiterentwickelt. Diese Weiterentwicklungen sind unten zu sehen.

3.7.1 getWMS()

In diesem Unterkapitel wir diese wichtige Funktion anhand der Inputargumente und Codeschnipsel erklärt.

Die Liste mit den Argumenten und deren Eigenschaften ist in Tabelle 3.5 ersichtlich.

Um die Eingabe Werte der aktuellen Zoomstufe anzupassen werden der «Dasharray» und die «PointSymbolSize» dynamisch dem Massstab angepasst. Die Anpassung des «Dasharrays» ist unten als Codeschnipsel aufgezeigt.

```
// Dasharray Anpassung an den Massstab
if (DashArray != null && massstab <= ShowThreshold && ShowThreshold
      != 999999999){
  Dashs = DashArray.split(" ");
  DashArray="";
  for (i=0;i<Dashs.length;i++){
    Dashs[i] = ((-0.5/(massstabMax-massstabMin))*massstab + 1 - (-
      0.5/(massstabMax-massstabMin)) * massstabMin)*Dashs[i];
    Dashs[i] = Math.round(Dashs[i]*100)/100
    DashArray += Dashs[i] + " ";
  }
}
```

Für die Verarbeitung wurden die zu Symbolisierenden Objekte nach Geometrietyp in die Gruppen Linien, Flächen, Punkte und Text aufgeteilt. Dabei wird je nachdem was im Attribut «Art» steht ein anderer Teil der SLD Generierung gestartet. Dabei werden die weiteren Argumente benützt um die gewünschten Effekte bei der Symbolisierung zu erreichen. Jedes Argument hat einen definierten «Null»-Wert, über welchen überprüft werden kann, ob dieses Argument benützt werden soll.

Nachdem der SLD-Teil generiert wurde, wir der WMS-Maprequest zusammengestellt. Dabei wird immer der aktuelle Kartenausschnitt (Schweizer Koordinaten) und dessen Grösse in Pixel verwendet, so dass das Rasterbild, welches vom WMS-Server zurückgeliefert wird auf die Karte und alle weiteren Layer passt.

Tabelle 3.5: Argumente der Funktion getWMS()

Nr.	Argument	Beschreibung	Nullwert	Typ ()	Beispiel	Linie	Fläche	Punkt	Text
1	ElementID	Eindeutige ID zusammengesetzt aus Art+LayerName+Objectval		String	'AreaVEC25_pri_aZ_See'	x	x	x	x
2	Art	Linie, Area, Point, Text		String	'Area'	x	x	x	x
3	LayerName	Name des ShapeFiles / DB-Tabelle		String	'VEC25_pri_a'	x	x	x	x
4	Objectval	Werte für den Filter		String	'Z_See'	x	x	x	x
5	StrokeColor	Linienfarbe	'000000'	String	'11FFAA'	x	x	x	x
6	StrokeWidth	Liniendicke	5	Double	12.5	x	x	x	x
7	AreaColor	Flächenfarbe	'000000'	String	'11FFAA'		x	x	x
8	Opacity	Opazität	1	Double	0.5	x	x	x	x
9	DashArray	Strichlierung	''	String	'0 10 10 0'	x			
10	ShowThreshold	Grenzwert der Anzeige	100000	Double	15000	x	x	x	x
11	Pattern	SVG-Text des Pattern aus Editor	null	String	<pattern ...> ... </pattern>		x		
12	PointSymbolSize	Punktsymbolgrösse	0	Double	15			x	
13	Symbol	Vordefinierte Symbolnamen	null	String	'square'			x	
14	SvgSymbol	SVG-Text des Punktes aus Editor	null	String	<svg xmlns="... ..."> ... </svg>			x	
15	TextSize	Schriftgrösse	12	Double	12				x
16	TextFamily	Schriftart	'TESCHT'	String	'Arial'				x
17	TextWeight	Fett oder Normal	'normal'	String	'bold'				x
18	TextStyle	Kursiv oder Normal	'normal'	String	'italic'				x

3.8 Werkzeuge

3.8.1 Auswahl der Objekte und Grenzwert

Abbildung 3.3: Elemente zur Auswahl von Objekten

In Abbildung 3.3 ist ein Screenshot mit dem Elementen, mit welchen die zu symbolisierenden Objekte ausgewählt werden können. Hier ist wieder die allgemeine Struktur der Daten erkennbar. Zuerst wählt der User aus, welche Art von Daten er symbolisieren möchte. Danach kommt die Auswahl des Layers (hier «Primärflächen») und danach alle Objekte mit einem gewissen Wert im Attribut Objectval (hier «Gletscher»). Im vierten Dropdownmenu lassen sich die drei Symbolisierungsebenen auswählen.

Mit dem beiden Knöpfen «Symbolisieren» und «löschen» lassen sich die einzelne Symbolisierungen erstellen oder entfernen.

Unten ist ein Slider, auf welchem man einen Grenzwert einstellen kann, bis zu welchem Zoomwert die aktuell ausgewählten Objekte dargestellt werden sollen.

3.8.2 Linien

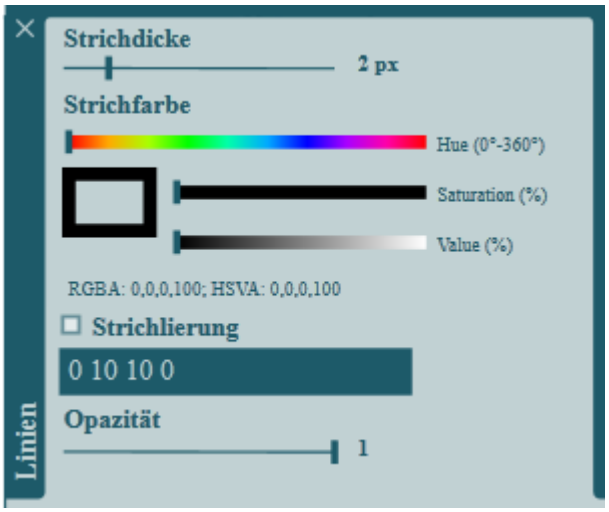


Abbildung 3.4: Werkzeug für Linien

Das Werkzeug um die Symboleinstellungen für Linien vorzunehmen beinhaltet einen Slider für die Strichdicke, einen Colorpicker für die Strichfarbe, eine Möglichkeit für die Eingabe eines einer Strichlierung sowie einen Slider für die Opazität der zu zeichnenden Linie.

Bei der Strichlierung entspricht die erste Zahl der Länge des ersten Strichs und die nächste der Länge des Unterbruchs. Die weiteren Zahlen wechseln sich weiter so ab. Um die Strichlierung benützen zu könne muss das Häkchen oberhalb des Eingabefeldes gesetzt werden.

3.8.3 Flächen

Das Werkzeug für die Flächen beinhaltet einen Slider für die Umrandungsdicke, einen Colorpicker, für die Flächen- und die Umrandungsfarbe und einen Slider für die Opazität der ganzen Ebene. Der Editor, welcher mit der Checkbox bei Pattern zu öffnen ist, wird im Kapitel 3.11 beschrieben.

Abbildung 3.5: Werkzeug für Flächen

3.8.4 Punkte

Abbildung 3.6: Werkzeug für Punkte

Die Einstellmöglichkeiten im Werkzeug für Punkte sind ein Slider für die Symbolgrösse und ein DropDown-Menü mit welchem die vordefinierten Objekte der SLD-Spezifikation ausgewählt werden können. Weiter ist ein Colorpicker vorhanden mit welchem die Flächenfarbe und die

Farbe für die Umrandung der Symbole ausgewählt werden können. Je nach ausgewähltem Typ von Punktsymbol sind wird nur die Umrandungsfarbe oder auch zusätzlich die Flächenfarbe benützt. Weiter unten ist ein Slider für die Umrandungsdicke vorhanden. Der Editor welcher via Checkbox aufrufbar ist, wird im Kapitel 3.11 beschrieben.

3.8.5 Text

Das Werkzeug für die Texte hat die üblichen Einstellmöglichkeiten für die Textformatierung. Das ist die Auswahl der Schriftart, die Schriftgrösse, Möglichkeit zur Einstellung von Fett oder Kursiv. Dazu die Auswahl der Schriftfarbe mittels Colorpicker. Zusätzlich ist die Möglichkeit gegeben die Schrift freizustellen. Dies in dem man eine Umrandungsfarbe auswählt. Der Slider für die Dicke hat keine Auswirkung, da in der aktuellen Version des WMS-Servers diese Angabe nicht unterstützt wird.

Abbildung 3.7: Werkzeug für Texte

3.9 Einbindung der Rasterdaten

Wie in Kapitel 3.2 beschrieben wurden zwei Rasterdatenlayer (Karte und Relief) in den Prototyp integriert. Die Bedienelemente für diese Rasterdaten sind in Abbildung 3.8 zu sehen. Es handelt sich dabei um je eine Checkbox zum Ein- und Ausschalten und einen Slider für die Opazität.

Abbildung 3.8: Bedienelemente für Rasterebenen

3.10 Symbolisierungsverwaltung

In der Symbolisierungsverwaltung, welche am linken Rand des Prototyps zu finden ist, besteht die Möglichkeit neue leere Symbolsätze zu erstellen oder bestehende zu kopieren. Es ist auch

möglich Symbolsätze zu löschen. Dies aber nur bei den selbst erstellten. Der Standardsymbolsatz «Swisstopo» lässt nicht verändern oder löschen.

Abbildung 3.9: Symbolisierungsverwaltung

3.11 Reihenfolge der Objekte

Da für Objektgruppe (gleicher Wert im Attribut Objectval) bis zu drei Symbolisierungsebenen möglich sind, muss die Reihenfolge dieser Ebenen definiert werden. Alle Rasterbilder erhalten also eine eindeutige «ElementID». Diese IDs sind in der Javascript-Funktion «initRasterOrder()» in einer kartografisch sinnvollen Reihenfolge fest gespeichert. Für jede ID wird beim Laden des Prototyps via WMS ein leeres Rasterbild generiert, welches beim Bearbeiten der Symbolisierung nur ersetzt wird, aber die Position behält. So wird erreicht, dass sich die Reihenfolge der einzelnen Kartenebenen nicht verändern kann.

Die festdefinierte Reihenfolge ist vereinfacht in der Tabelle 3.6 aufgeführt, wobei normalerweise alle drei Symbolisierungsebenen einer Objektgruppe einander direkt folgen. Der Spezialfall ist bei den Strassen und Eisenbahnen vorhanden, wo zuunterst, getrennt nach Verkehrsmittel, für alle Objektgruppen die Symbolisierungsebenen 0 definiert ist und darüber die beiden nächste Ebenen. So lassen saubere Doppellinien zeichnen. Ein Vergleich am Beispiel einer Strassenkreuzung welches dieses Vorgehen zeigt ist in Abbildung 3.10 illustriert.

Tabelle 3.6: Reihenfolge der Kartenlayer

Layer	Symbolisierungsebenen
Höhenlinien	0-2
Primärflächen (inkl. Flusslinien)	0-2
Gewässernetzlinien (Bäche, Druckstollen, ...)	0-2
Bahnhofsflächen	0-2
Strassen (alle)	0
Strassen (alle)	1-2
Eisenbahnen (alle)	0
Eisenbahnen (alle)	1-2
Seilbahnen	0-2
Weitere Linien (Hecken, Hochspannungsleitungen, ...)	0-2
Gebäude	0-2
Punktobjekte (Wegkreuze, Reservoir, ...)	0-2
Beschriftungen	0-2

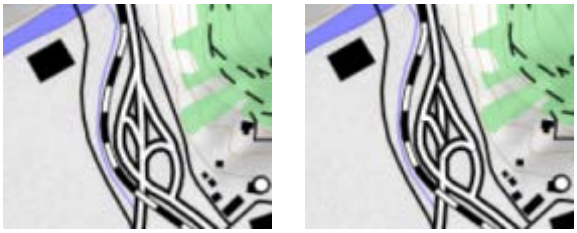


Abbildung 3.10: Vergleich zwischen optimierter (l.) und einfacher (r.) Reihenfolge der Kartenlayer (Reproduziert mit Bewilligung von swisstopo (JA100120))

3.12 Editor

Um Pattern bei Flächensignaturen und SVG-Symbole bei Punkten zu ermöglichen wurde ein Editor erstellt, welcher für beide Funktionen taugt. Der Editor besteht aus einem zusätzlichen Fenster zu den eigentlichen Werkzeugen und wird über eine Checkbox eingeschaltet. Dabei wird er auch automatisch geöffnet. In Abbildung 3.11 sind die beiden Editoren abgebildet.

Der Editor erstellt SVG Pfade welche von der Funktion «symbolize()» bearbeitet und dann als String der «getWMS()» weitergegeben wird.

Als Hilfe beim Zeichen der Pfade, ist eine rote Hilfslinie vorhanden, welche entlang dem Pfad um vom letzten Punkt zum Mauszeiger geht. Dies erleichtert vor allem das Zeichen von Flächen ohne Rand.

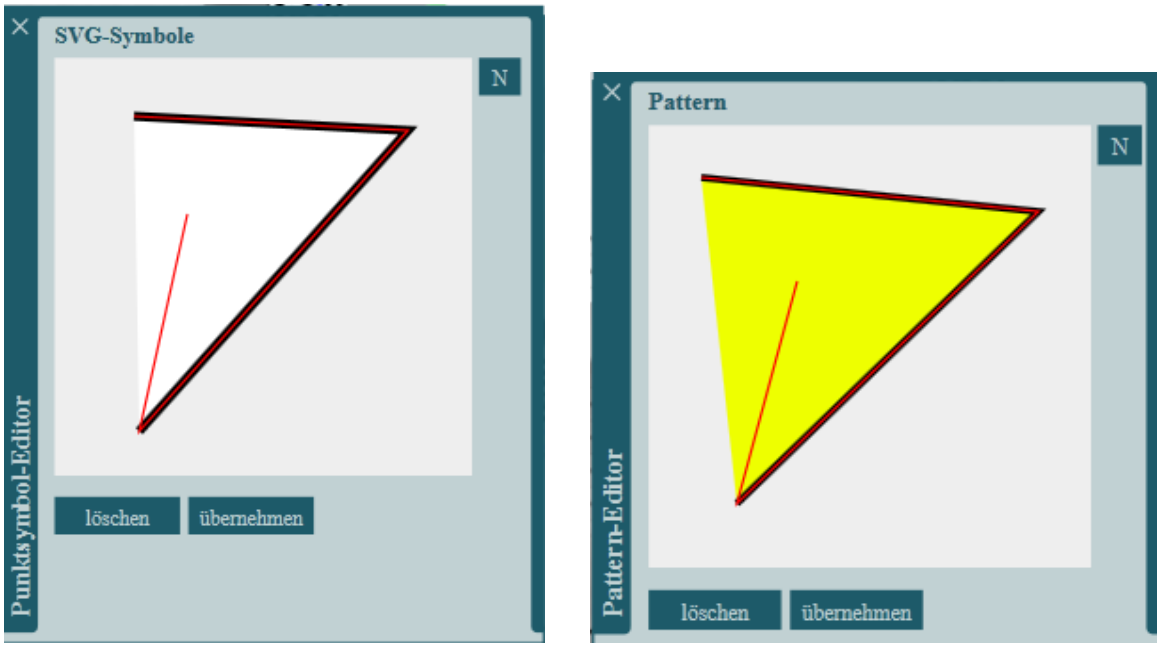


Abbildung 3.11: Editoren für Punkte (l.) und Flächen (r.)

3.12.1 Vorbereitungen Erweiterung

Im Quellcode des Prototyp wurden schon Vorbereitungen getroffen, um den Editor mit weiteren Zeichnungswerkzeugen (Kreise) auszustatten. Es wurden Knöpfe erstellt und auch die Verarbeitung von Benutzereingaben vorbereite. Diese Knöpfe sind aber ausgeblendet und für den User unsichtbar.

3.13 Legende

Die automatische Erstellung einer Legende wurde begonnen, konnte aber aus Zeitgründen nicht abgeschlossen werden. Die Auflistung der Legendenelemente für Linien, Flächen und Punkte funktioniert.

Die Darstellung der Linien ist ziemlich zufriedenstellend. Wobei der Dasharray nicht bei allen Objekten sauber funktioniert. Die Erstellung der Flächenobjekte funktioniert, aber ohne die Darstellung von Pattern. Die Liste der Punktsymbole wird zwar dargestellt, da werde aber einfach die Werte für die Flächen und Umrandungsfarbe genommen und als Rechteck dargestellt. Dies entspricht natürlich nicht den eigentlichen Symbolen.

Abbildung 3.12: Legende

3.14 Impressum und Hilfe

Es wurde ein Impressum mit den Informationen über diese Masterarbeit sowie die Daten und deren Quellen erstellt. Zudem wurde eine Kurz-Anleitung erstellt, so dass ein neuer Benutzer des Prototyps einen Einstieg in die Webseite finden kann. Diese Hilfe wird beim Laden des Prototyps automatisch angezeigt. Diese beiden Fenster sind in der Abbildung 3.13 abgebildet.

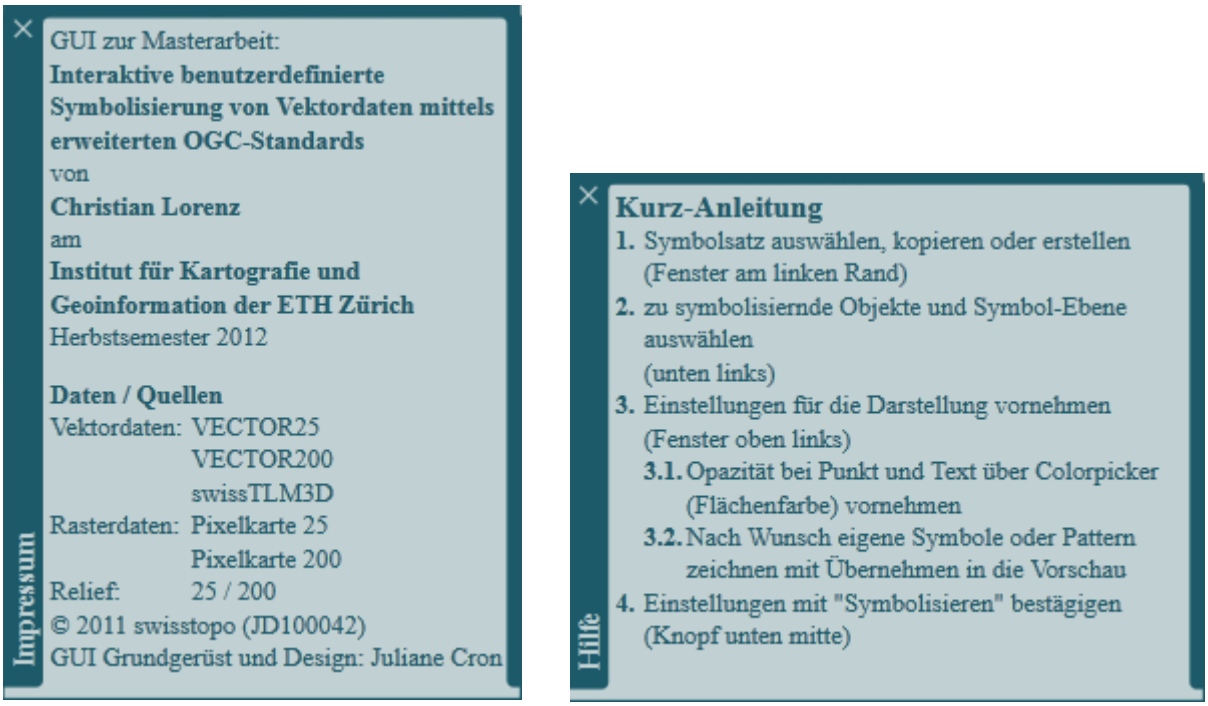


Abbildung 3.13: Impressum und Hilfe

3.15 Bereinigen

Nach allen diesen Punkten funktionierte der Prototyp soweit, aber es waren viele Codeschnipsel übrig geblieben, welche nicht oder nicht mehr nötige waren. Dazu gehören auch Funktionen von Juliane Cron, welche in dieser Arbeit ausgeschaltet und nicht benötigt wurden.

Alle diese Überbleibsel mussten entfernt und der bestehende Code bereinigt und für eine spätere Bearbeitung kommentiert werden.

Alle Javascript-Dateien welche von carto.net stammen wurden nicht bearbeitet.

4 Ergebnisse

4.1 Prototyp

Als Ergebnis dieser Arbeit ist der Prototyp zu betrachten. In diesen wurde alle Arbeit gesteckt und er beinhaltet auch alle Resultate, welche erarbeitet wurden. Im Folgenden werden einzelne Aspekte des Prototyps aufgenommen und näher erläutert.

Mit dem bearbeiteten Screenshot unten wird der allgemeine Aufbau des Prototyps gezeigt. Im Anhang A sind einige grössere Screenshots zu finden.

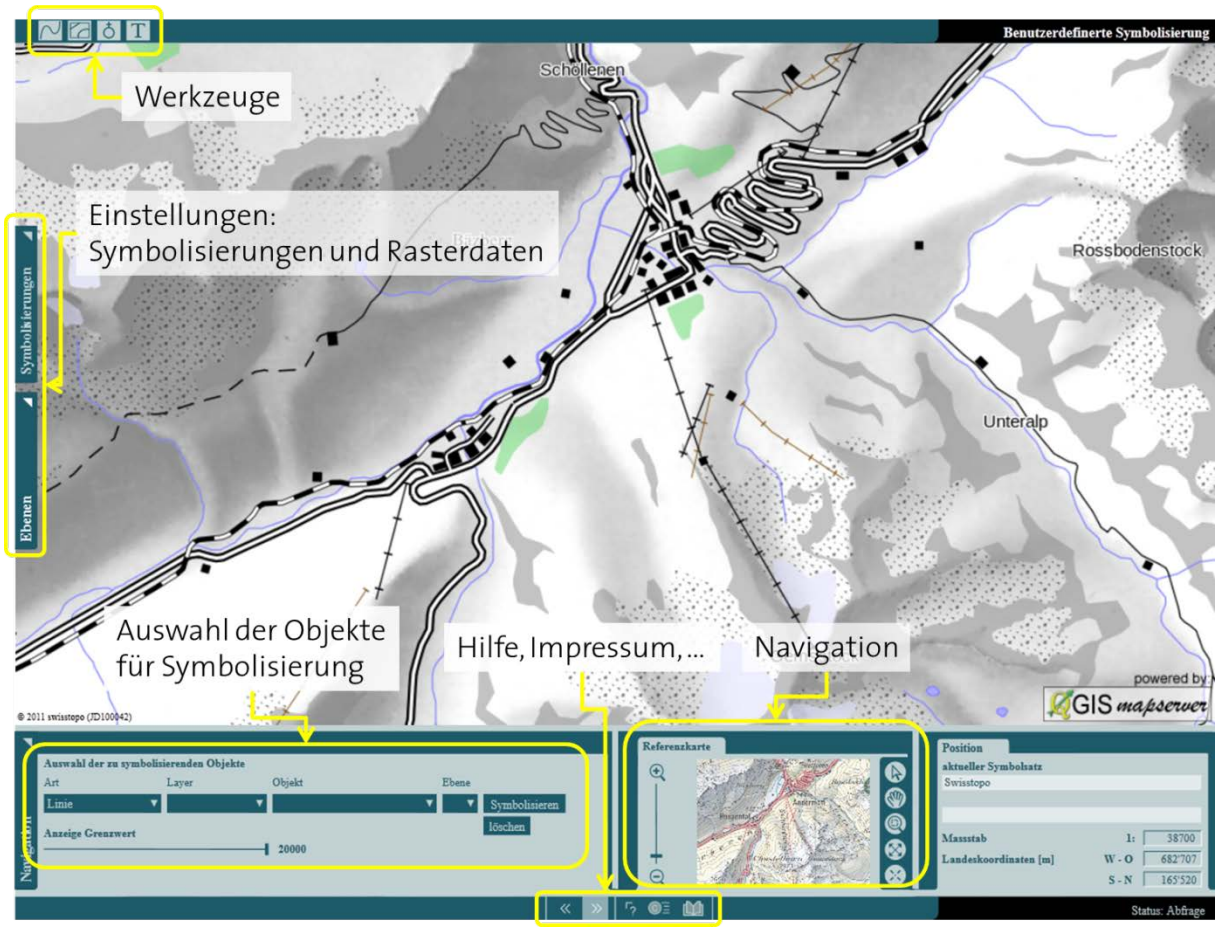


Abbildung 4.1: Screenshot des Prototyps mit Erklärungen

4.1.2 Ablauf wichtiger Benutzereingaben

Um den Aufbau des Prototyps aufzuzeigen sind in der Abbildung 4.2 die drei wichtigsten Benutzereingaben und deren Verarbeitung zu sehen.

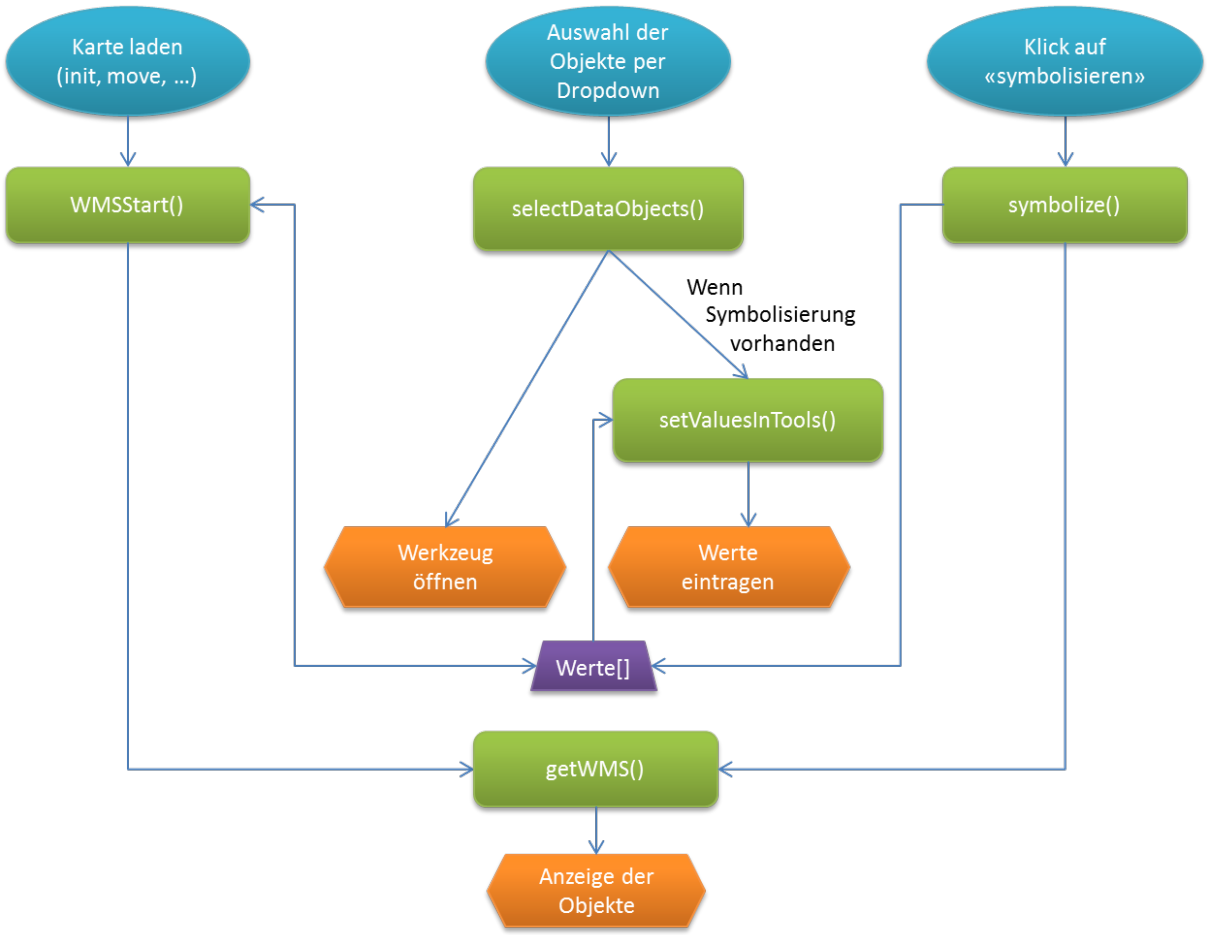


Abbildung 4.2: Ablauf einiger Benutzerinteraktionen

Das violette Trapez («Werte[]») ist dabei der Speicherort für alle einzelnen Werte, welche für alle möglichen Symbolisierungen abgelegt sind oder werden.

«Werte[]» ist ein dreifach verschachtelter Array. Ein Aufruf von Daten sieht so aus:

```
Werte[0]['LineVEC25_eob_lHSP_Ltg1']["StrokeWidth"] = 0.6;
```

Dabei ist die Zahl in der ersten Klammer (hier `0`) die Nummer des gewünschten Symbolsatzes. In der zweiten Klammer (hier `'LineVEC25_eob_lHSP_Ltg'`), also der zweiten Stufe von Arrays ist die «ElementID» zu finden, mit der jede mögliche Kartenebene eindeutig identifizierbar ist. Hier sind auch die drei möglichen Symbolebene pro Objektgruppe aufrufbar. Die unterste Ebene hat keine Nummer die beiden oberen haben eine zusätzliche 1 oder eine 2 am Schluss der «ElementID». Mit dem Wert in der dritten Klammer ist der Parameter der «getWMS()» Funktion aufrufbar. Mit der Zuweisung `= 0.6;` wird also der die Strichdicke auf 0.6 geändert.

4.1.2.1 Karte laden (init, move, zoom, ...)

Die Aktion «Karte laden» tritt immer dann auf, wenn das Kartenbild aktualisiert wird. Dies ist der Fall beim ersten Öffnen des Prototyps sowie bei Änderungen des Kartenausschnittes durch zoomen oder verschieben der Karte.

Diese Aktion löst die Javascript-Funktion «WMSStart()» aus. Diese Funktion beinhaltet die «swisstopo-ähnliche»-Symbolisierung welche beim ersten Laden des Prototyps in den Datenarray «Werte[]» abgelegt. Danach wird die Funktion «getWMS()» in einer Schleife aufgerufen um alle Layer, zu welchen Daten in «Werte[]» liegen, darzustellen. Dies geschieht für den aktuell ausgewählten Symbolsatz.

4.1.2.2 Auswahl der Objekte per Dropdown

Immer wenn ein Benutzer Objekte zum symbolisieren auswählt tritt die Funktion «selectDataObjects()» in Aktion. Wenn die Art der Objekte ausgewählt wird, wird das entsprechende Werkzeug (Linie, Fläche, ...) geöffnet. Bei der Auswahl der Objekte (z. B. Gletscher, 1. Klass,...) werden, falls für diese Objekte und die aktuelle Symbolebene (Standardauswahl " ") Werte vorhanden sind, diese in das passende Werkzeug eingesetzt. Dieses Einsetzten macht die Javascript-Funktion «setValuesInTools()». Dabei werden die Werte aus dem Datenarray «Werte[]» ausgelesen.

4.1.2.3 Klick auf «symbolisieren»

Ein Klick auf die Schaltfläche «symbolisieren» ruft die Javascript-Funktion «symbolize()» auf. Falls mit den Dropdown-Menüs eine Objektgruppe (inkl. Symbolebene) ausgewählt wurde, welche symbolisiert werden kann, werden die Werte aus dem entsprechenden Werkzeug übernommen und so aufbereitet, dass sie in «Werte[]» richtig abgelegt werden können. Diese Aufbereitung beinhaltet zum Beispiel die Umrechnung der Zahlenwerten aus dem Colorpicker in ein String mit RGB-Werten, oder das Auslesen und umformatieren des SVG-Codes aus dem Vorschaufeld des Editors.

Danach wird die getWMS()-Funktion einmal mit den neuen Werten aufgerufen, so dass die eingestellte Symbolisierung visualisiert wird.

5 Fazit

5.1 «Proof of Concept» erbracht

Mit dieser Arbeit konnte gezeigt, werden, dass die in der Problemsituation (Kapitel1.1) geforderte Interaktion grundsätzlich möglich ist. Mit dem dienstbasierten Ansatz können Karten dargestellt und von den Benützern des Prototyps verändert oder von Grund auf frisch symbolisiert werden.

5.2 Aufwand zum «perfekten» kartografischen Produkt

Die Kartografische Qualität, welche mit dem erstellten Prototyp möglich ist, ist sicher nicht perfekt. Da würden die Erweiterungen in SLD welche der QGIS-Mapserver auf karlinapp.ethz.ch zur Verfügung stellt eine höhere Qualität erlauben.

5.2.1 Komplexität

Mit der möglichen Einbindung von weiterer Optionen, welche der Mapserver zur Verfügung stellt, steigen zwar die Möglichkeiten für den Nutzer, aber es steigt auch die Komplexität der Benutzeroberfläche sowie des SLD-Teiles des Maprequest an den Server. Vor allem das Problem mit dem Maprequest ist technisch lösbar aber je nach Grad der Komplexität sehr aufwändig.

Auch die Einbindung zusätzlicher Optionen in der Benutzeroberfläche ist möglich, aber dabei muss auf die Benutzerfreundlichkeit geachtet werden. So dass ein Nutzer, der das erste Mal die Oberfläche ausprobiert, nicht von den Einstellmöglichkeiten überfordert wird.

5.2.3 Daten

In Abbildung 5.1 ist ein Beispiel abgebildet, in welchem die mangelnde Qualität ersichtlich ist. Dabei geht es um die Strichlierung, bei welcher der Unterbruch der Linie nicht auf einen Ecken des Weges fallen sollte, so dass der Verlauf des Weges klar ersichtlich bleibt. Dies könnte in der automatischen Symbolisierung erreicht werden, in dem der Weg nicht als Objekt mit vielen Stützpunkten sondern als viele Linien mit jeweils 2 Stützpunkten behandelt würde. Dazu müssten die Daten umformatiert werden.

Weiter sind die nicht beschrifteten Höhenlinien sichtbar. Dies da wie in Kapitel 2.2.6 beschrieben, das Höhenattribut für die Linien fehlt. Es ist also nicht möglich die Linien der Höhe entsprechend zu Symbolisieren oder zu Beschriften.

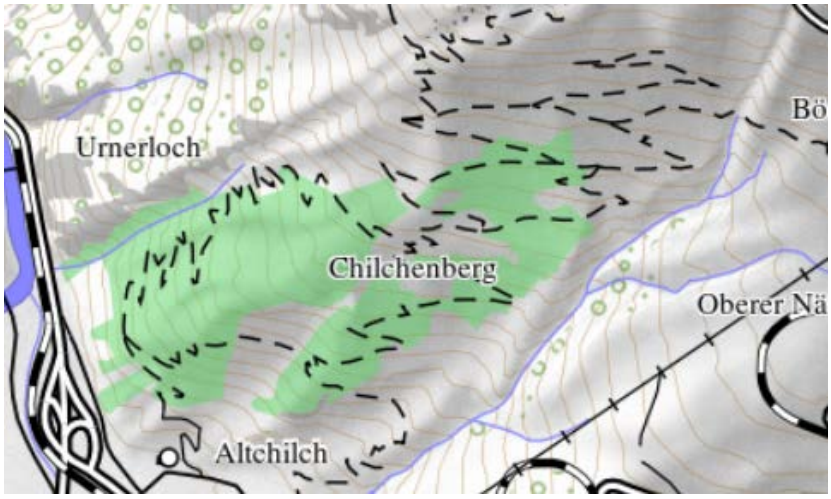


Abbildung 5.1: Beispiel mangelnder kartografischer Qualität

6 Ausblick

In diesem Kapitel sind Möglichkeiten beschreiben, wie in diesem Bereich weitergearbeitet werden kann und welche Verbesserungen oder Änderungen im Prototyp wünschenswert und sinnvoll sind.

6.1 Weitere Symbolisierungen

6.1.1 Galerie Symbolisierungen

Aktuell ist nur eine Symbolisierung als Vorlage im Prototyp integriert. Um die Möglichkeit zu haben, Schweizer Daten in anderen Symbolisierungen darzustellen, wäre eine Galerie von verschiedenen Symbolisierungen wünschenswert. So, dass man zum Beispiel die Schweizer Landeskarte mit der deutschen Symbolisierung anschauen könnte.

6.1.2 Speicherung von Symbolisierungen

Um die Galerie von Symbolisierungen zu ergänzen könnte eine Möglichkeit geschaffen werden, mit der, die von Benutzer der Website erstellten Symbolisierungen, serverseitig abgespeichert werden können. Das würde dem Benutzer auch die Arbeit mit dem Prototyp erleichtern, da eine Symbolisierung gespeichert und zu einem späteren Zeitpunkt wieder geladen und weiter bearbeitet werden könnte.

6.2 Ganze Schweiz

Da, wie in der Einleitung zu Kapitel 3.4 beschrieben, keine gebietsspezifischen Änderungen an den Daten vorgenommen wurden, sollte es möglich sein, die beschriebenen Anpassungen auf die ganzen Datensätze, das heisst auf die ganze Schweiz, auszudehnen. Dabei sollte der Prototyp im Grundsatz ohne Probleme weiter funktionieren.

Einige Probleme würden aber sicher auftauchen. Die Einbindung der Rasterdaten (Landeskarte und Relief) müsste via Server realisiert werden, da eine Karte oder Relief über die ganze Schweiz als eine Datei nicht möglich ist.

Diese Erweiterung würde dann auch bedeuten, dass eine Einbindung in GeoVITe möglich würde. Im Stile von «Design your own maps in GeoVITe»

6.3 Massstab und Zoomstufen

Durch das, dass nur Daten auf Grundlage von zwei Massstäben vorhanden sind, wurde das Projekt mit nur einem Wechsel zwischen verschiedenen Generalisierungsstufen realisiert. Eine Erweiterung auf weitere Massstäbe würde die Möglichkeit eröffnen kleinere Unterschiede zwischen verschiedenen Zoomstufen zu realisieren.

Dies ist aber in der Schweiz nur mit Generalisierung bestehender Vektordaten oder mit der Digitalisierung weiterer Landeskarten möglich. Diese beiden Möglichkeiten sind entweder mit viel manueller Arbeit verbunden, oder aber die entsprechenden Automatisierungen sind (noch) nicht ausgereift genug. Das Automatisieren wird bei beiden Möglichkeiten wohl nie die gleiche Qualität haben, wie die manuelle Bearbeitung.

6.4 Verbesserungen bestehender Prototyp

6.4.1 Editor

Im Editor sind bis jetzt nur einfache Pfade möglich. Das heisst es werden einfach die geklickten Orte mit einer Linie verbunden. Es sind also keine Objekte wie Kreise, Rechtecke oder Freihandzeichnungen möglich. Diese Erweiterungen würden die Erstellung besserer und komplexerer Pattern oder SVG-Symbole erlauben.

Ausserdem wäre eine Möglichkeit zum sauberen Abschliessen des neu gezeichneten Pfades sinnvoll. Dies könnte über einen Mouseover-Effekt beim Startpunkt des Pfades realisiert werden.

6.4.2 direkte Symbolisierung bei Einstellung in Werkzeug

Eine Anwendung der Einstellungen aus den Werkzeugen direkt auf die Karte wäre wohl möglich und für die Benutzerfreundlichkeit wünschenswert, aber mit der jetzigen Performanz des Prototyps nicht sinnvoll, da es immer einen kurze Zeitspanne braucht, bis die eingestellten Werte aus den Werkzeugen übernommen und in der Karte dargestellt werden.

Durch diese Zeitdifferenz wäre der Benutzer dann auch nicht sicher, ob die aktuelle Karte wirklich seinen neusten Einstellungen entspricht.

6.4.3 Legende

Die begonnene Legende sollte fertiggestellt werden, da wenn möglich auch Webkarten eine Legende beinhalten sollten. Die Einbindung der Pattern und SVG-Symbole, welche als zwar in SVG aber als ein einzelner String abgelegt sind, müssen ausgelesen und wieder in SVG-Objekte überführt werden um verarbeitet zu werden.

Der noch nicht erstellte Textteil der Legende sollte mit ähnlichem Code wie die anderen drei Klassen realisiert werden können.

6.4.4 Mehr Optionen der Werkzeuge

Wie im Fazit (Kapitel 5.2.1) erläutert sind noch einige Möglichkeiten der Darstellung offen, welche in diesem Prototyp nicht verwendet wurden. Die Einbindung weiterer Darstellungsoptionen ist also ein Gebiet welches noch Potential hat.

7 Referenzen und Quellen

carto.net, 2013 Neumann, A. (http://www.carto.net/papers/svg/samples/index.shtml#iact, aufgerufen am 8.1.2013)

Cron, J., 2006. Graphische Benutzeroberflächen interaktiver Atlanten, Diplomarbeit, ETH Zürich,

DHM25,
http://www.swisstopo.admin.ch/internet/swisstopo/de/home/products/height/dhm25.html (aufgerufen am 8.1.2013)

Iosifescu, C., I. Iosifescu, H. Jenny und Lorenz Hurni, 2011, GeoVITe – A Service-driven Solution for an On-demand, Userfriendly Web Access to Geodata, ICC2011: The 25th International Cartographic Conference, Paris, 11S

Iosifescu-Enescu, I. 2011, Cartographic Web Services, Dissertation, ETH Zürich, Schweiz

karlinapp.ethz.ch, 2013 Website der Entwicklungsgruppe von QGIS-Mapserver (aufgerufen am 8.1.2013)

Landeskarte,
http://www.swisstopo.admin.ch/internet/swisstopo/de/home/products/maps/national/digital.html (aufgerufen am 8.1.2013)

Multimedia Cartography, FS 2012, Vorlesung am Institut für Kartografie und Geoinformation der ETH Zürich. Vorlesungsnummer: 103-0228-00L,

Notepad++
http://notepad-plus-plus.org/ (aufgerufen am 8.1.2013)

Ortner, F. 2011, Dienstbasierte Kartengenerierung für Web-Atlanten unter Anwendung erweiterter OGC-Standards, Masterarbeit, Universität und ETH Zürich, Schweiz

Panchaud, N., 2012, Service-Driven 3D Atlas Cartography, Masterarbeit, IKG ETH Zürich, Schweiz

Relief,
http://www.swisstopo.admin.ch/internet/swisstopo/de/home/products/maps/national/digital/details.html (aufgerufen am 8.1.2013)

SwissNames,
http://www.swisstopo.admin.ch/internet/swisstopo/de/home/products/landscape/toponymy.html (aufgerufen am 8.1.2013)

swissTLM3D,
http://www.swisstopo.admin.ch/internet/swisstopo/de/home/products/landscape/swissTLM3D.html (aufgerufen am 8.1.2013)

Alle Swisstopo-Daten, Bundesamt für Landestopografie swisstopo
(Art. 30 GeoIV): 5704 000 000

VECTOR25,
http://www.swisstopo.admin.ch/internet/swisstopo/de/home/products/landscape/vector25.html (aufgerufen am 8.1.2013)

VECTOR200,
www.swisstopo.admin.ch/internet/swisstopo/de/home/products/landscape/vector200.html (aufgerufen am 8.1.2013)

A. Screenshots Prototyp

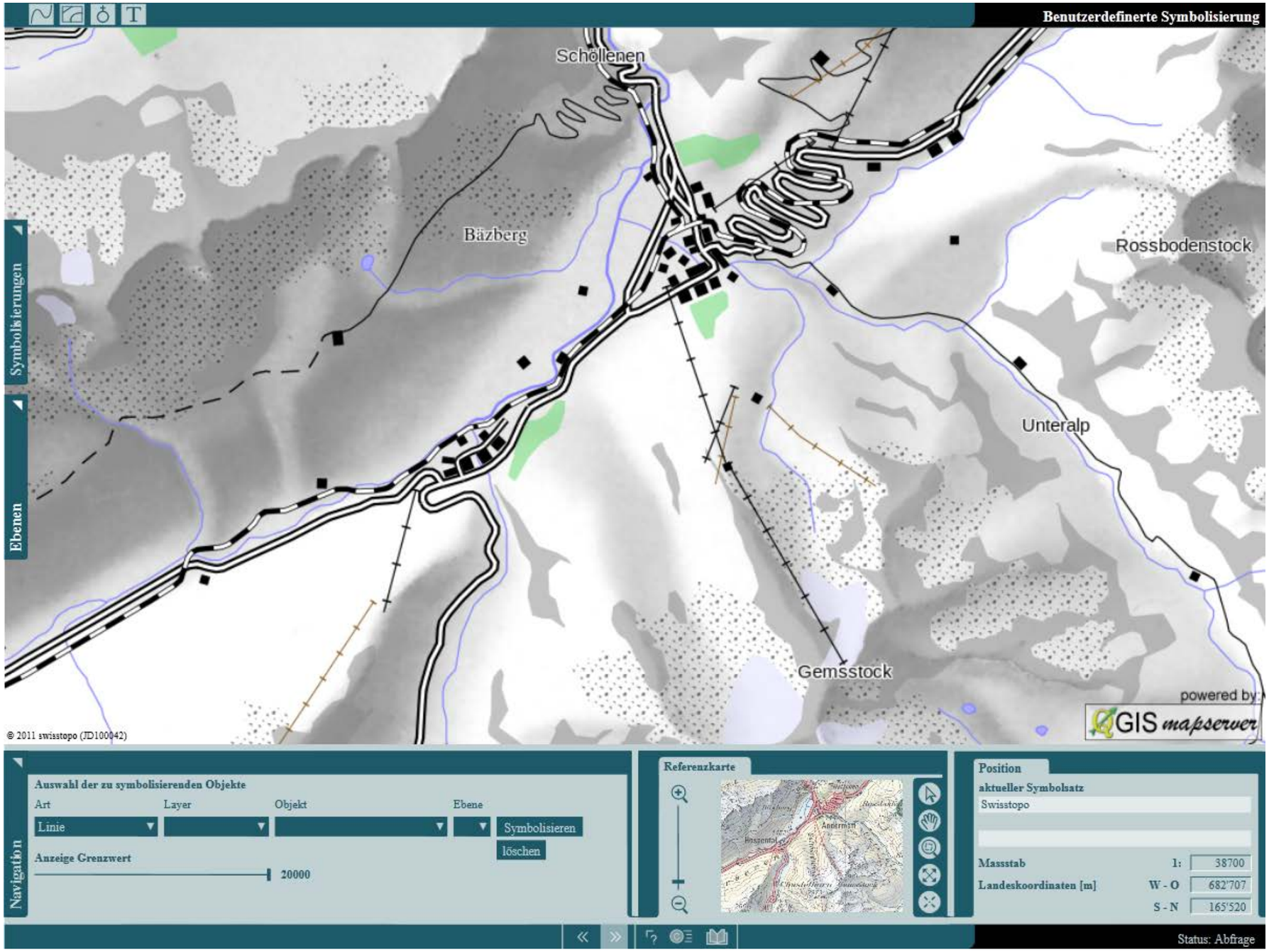


Abbildung A.1: Screenshot des Prototyps (Reproduziert mit Bewilligung von swisstopo (JA100120))

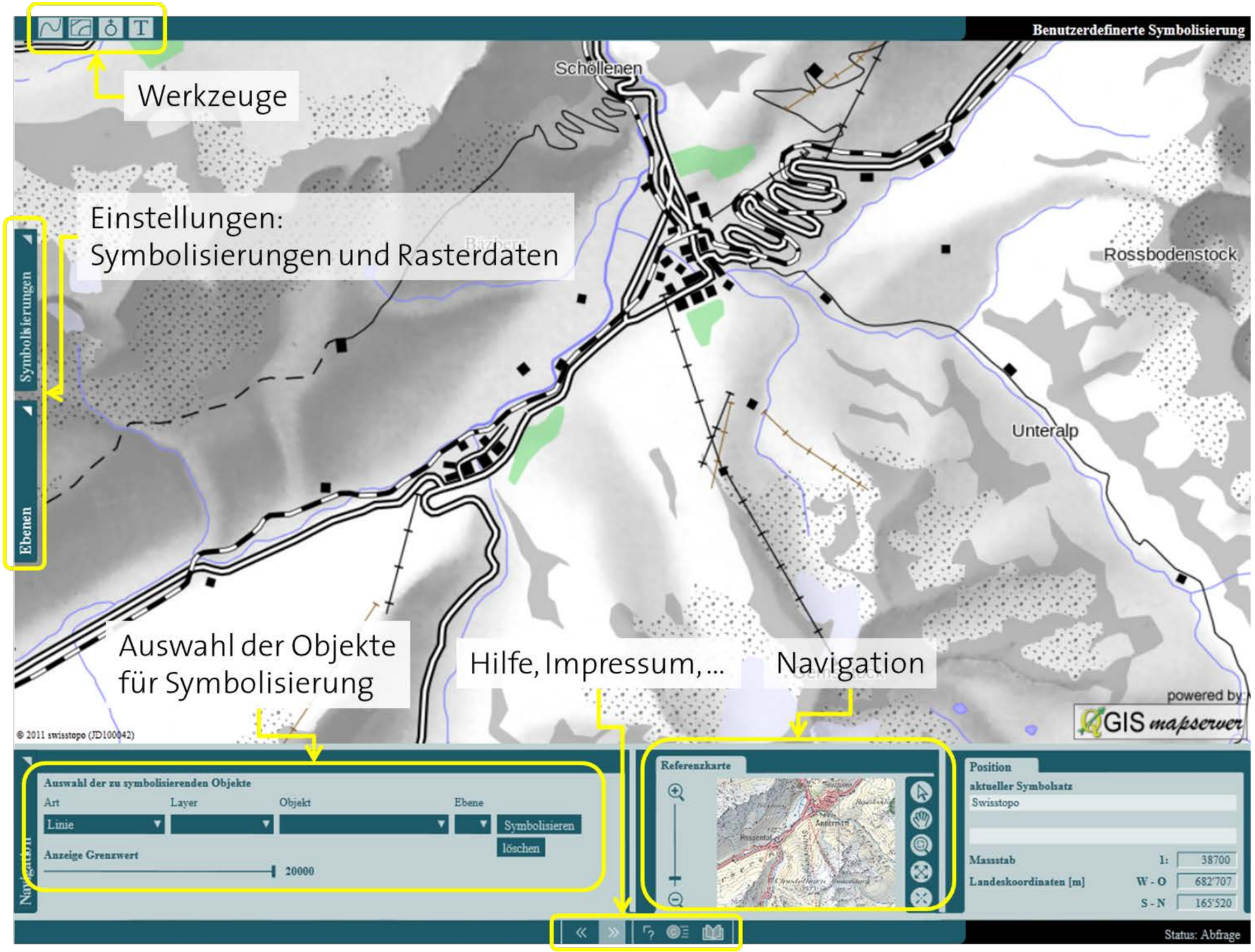


Abbildung A.2:Screenshot mit Erklärungen der Werkzeuge (Reproduziert mit Bewilligung von swisstopo (JA100120))

B. Beispiel SQL-Code für DB-View

```
CREATE OR REPLACE VIEW vec200_nam_p_weg AS
 SELECT "VEC200_nam_p".the_geom, "VEC200_nam_p".gid, "VEC200_nam_p".name,
    "VEC200_nam_p".gemname, "VEC200_nam_p".altitude,
    "VEC200_nam_p".objectval
   FROM "VEC200_nam_p"
  WHERE "VEC200_nam_p".objectval::text = 'Weg'::text OR
    "VEC200_nam_p".objectval::text = 'Fusspass'::text;

ALTER TABLE vec200_nam_p_weg
  OWNER TO christian;
```

```
CREATE OR REPLACE VIEW vec25_nam_p_flur AS
 SELECT "VEC25_nam_p".the_geom, "VEC25_nam_p".gid, "VEC25_nam_p".name,
    "VEC25_nam_p".gemname, "VEC25_nam_p".altitude, "VEC25_nam_p".objectval
   FROM "VEC25_nam_p"
  WHERE "VEC25_nam_p".objectval::text = 'Flurname'::text OR
    "VEC25_nam_p".objectval::text = 'Einzelhaus'::text;

ALTER TABLE vec25_nam_p_flur
  OWNER TO christian;
```

C. Beispiel WMS-Anfrage

```
http://karlinapp.ethz.ch/cgi-bin/qgis_map_server/ch/qgis_mapserv.fcgi?
    SERVICE=WMS&VERSION=1.3.0&REQUEST=GetMap&CRS=EPSG:21781
    &TRANSPARENT=TRUE&BBOX=687417,163563,690783,165956&width=1024
    &height=728&format=image/png&SLD_BODY=
<StyledLayerDescriptor xmlns="http://www.opengis.net/sld"
    xmlns:ogc="http://www.opengis.net/ogc" units="mm">
  <NamedLayer>
    <Name>VEC25_eob_p</Name>
    <Title></Title>
    <UserStyle>
      <Name>red</Name>
      <FeatureTypeStyle>
        <Rule>
          <Filter>
            <PropertyIsEqualTo>
              <PropertyName>objectval</PropertyName>
              <Literal>BiStock</Literal>
            </PropertyIsEqualTo>
          </Filter>
          <PointSymbolizer>
            <Graphic>
              <Mark>
                <SvgSymbol>
                  <svg  width="30" height="30"
                       xmlns="http://www.w3.org/2000/svg" >
                    <g>
                      <line x1="15" x2="15" y1="0" y2="29" stroke="black"
                         stroke-width="2"/>
                      <line x1="7" x2="23" y1="8" y2="8" stroke="black"
                         stroke-width="2"/>
                      <line x1="11" x2="19" y1="29" y2="29" stroke="black"
                         stroke-width="2"/>
                    </g>
                  </svg>
                </SvgSymbol>
              </Mark>
              <Size>3</Size>
              <Rotation>
                <ogc:PropertyName>objectazim</ogc:PropertyName>
              </Rotation>
            </Graphic>
          </PointSymbolizer>
        </Rule>
      </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

D. CD

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

E. 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 Vektordaten mittels erweiterten OGC-Standards»

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

Verfasser

Name	*Vorname*
Lorenz	Christian

Betreuer

Name	*Vorname*
Iosifescu-Enescu	Ionut

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

Ort, Datum Unterschrift

Durch die Unterschrift bürgen Sie für den vollumfänglichen Inhalt der Endversion dieser schriftlichen Arbeit.