Interactive Mapping of Distance Modeling

Development of a Visualized, Fully Web Based Hike Planing Tool with SVG

Urs Sieber
Mühlheim 8a
CH-9323 Steinach

Master Thesis 2008
Supervised by Prof. Claude Collet

Steinach, May 13, 2008

University of Fribourg
Faculty of Science
Departement of Geoscience
Unit of Geography

Dedicated to Steve Biko, the leading power of the Black Consciousness movement, which allowed a soft revolution in South Africa after decades of Apartheid.

He personifies “the man to be” together with his companions who have struggled for the same honorable goal.

Abstract

This thesis demonstrates how Internet based mapping and distance modeling of hiking trails can be achieved. From raw data to the finished hike planning tool, a possible strategy is developed, which is comprehensible and fully considers the following key criteria: cost factor analysis on independent topology, totally dynamic generation of maps and diagrams, use of the open and progressive SVG standard, barrier-free application, free availability via Internet without the need for installation of a software on the client computer.

In a first phase global theory about spaces and distances as well as about graph theory and distance modeling is introduced. The focus lies on the development of Dijkstra's algorithm, which calculates minimal costs based on an independent topology. With the help of a self-dependently developed pseudo-heuristic we try to improve the algorithm's bad runtime within broad topologies and to adapt it for Internet use. Thus geographic reflections and concepts from Computer Science are combined. Extensive comparisons between algorithm with and without heuristic testify a runtime improvement up to 35% due to the heuristic.
We can apply distance modeling only on topologies. Step-by-step the reader is explained how to proceed to build up a forceful topology out of the available raw data, comparing the two dimensional hiking network and three dimensional signpost locations. Among other information the topology stores data about way time, difference in altitude, way distance, gradient and trail covering. The digital raw data for the perimeter of the canton of St.Gall exclusive the Toggenburg was made available by the cantonal Institute for Geoinformation. Because height marking is only available for signpost locations, the heights for secondary nodes have to be approximated. An audit of the approximation quality with the help of the digital terrain model of Switzerland shows an acceptable deviance of statistical approximation and reality of slightly more than 2%. The concrete realization of the topology within a MySQL database is carefully discussed. Special focus lies on waytime calculation for hiking trails, which is based on the calculation methods given by the Swiss Hiking Association. The final topology links 1'846 nodes with 5'142 edges.

In a second phase the visualization of data resulting from distance modeling on the topology takes place. Goal is the dynamic generation of digital maps and the route profile out of the antecedent modeling. To secure the key criteria of a barrier-free, web based application, we agree on the modern, XML standardized and open SVG programming standard. The work with the new and modern standard shows its advantages and disadvantages.
To weigh the advantages of digital maps by comparison with analog maps, the following interactive functions are introduced into the digital map: dynamic coordinate display, zoom and pan functionalities, overview map with sensitive area to drag and drop the map focus, asynchronous load of data in separate layers. As a frequent cause of problems, the different browser implementations of the SVG support is discussed. With the help of layers a differentiated level of detail is made available to the user. The asynchronous load of data into the map with AJAX technologies is discussed. Further we deal with the introduction of georeferenced aerial images. The aerial images need to be converted from high resolution quality images to adequate Internet quality raster images without losing their reference. Ultimately we present how to build an adequate user interface, which guides the diverse implementations. Various tests on as many platforms as possible allow statements about the system's forces and weaknesses.

Key questions about distance modeling, interactive mapping with SVG and usefulness of an online hike planning tool can finally be answered: distance modeling on a topology is basically realizable, even through web components. SVG will only succeed, if not only the programming language is standardized but also the SVG support in different browsers is unified. Interactive Cartography and GIS applications via Internet will totally change the geographer's oldest domain, which is measuring and mapping. We guess interactive maps will replace analog maps in the future.
We see a possible use of the hiking planer mostly for tourist regions or more specific for hiking regions that distinguish themselves with hiking trails or similar networks like ski slopes or cross country sky tracks for example. We think of restricted regions which like to offer an attractive service for their guests. There is a high development potential in the database's extension with additional information to secondary services like restaurant locations, timetable and public transport, points of interest, GPS export or predefined hiking proposals. These features provide possibilities to increase the economic value and could make a hiking planer additionally attractive for these regions.

* * * * * * *

Zusammenfassung

Die vorliegende These zeichnet einen Weg auf, der interaktive Kartierung und Distanzmodellierung von Wanderwegen verfolgt. Ausgehend von Rohdaten bis zum fertigen Wanderplanungsinstrument, wird eine mögliche Vorgehensweise entwickelt, die nachvollziehbar ist und die folgenden Ausgangskriterien vollständig erfüllt: Kostenfaktor-Analysen auf eigener Topologie, vollständig dynamische Generierung der Karten und Diagramme, Verwendung des offenen und progressiven SVG-Standards, barrierefreie Anwendung, frei zugänglich via Internet ohne Installation von Software.

In einer ersten Phase wird die allgemeine Theorie zu Räumen und Distanzen sowie zur Graphentheorie und Distanzmodellierung verarbeitet. Hauptgegenstand bildet dabei die Erarbeitung des Dijkstra Algorithmus, der auf einer Topologie basierend kleinste Kosten berechnen kann. Mittels einer selbständig entwickelten Pseudo-Heuristik wird versucht, die schlechte Laufzeit des Algorithmus bei grossen Topologien zu reduzieren und den Algorithmus für die Verwendung über das Internet fit zu trimmen. Dabei werden geografische Überlegungen und Konzepte aus der Informatik miteinander kombiniert. Verschiedene Kostenfaktoren verlangen nach unterschiedlichen Ausprägungen der Heuristik. Ausgiebige Vergleiche des Algorithmus mit und ohne Heuristik belegen eine Verbesserung der Laufzeit von bis zu 35%.
Distanzmodellierung ist nur auf Topologien anwendbar. Schritt für Schritt wird dem Leser erklärt, wie aus den vorhandenen Rohdaten, bestehend aus 2-dimensionalem Wanderwegnetzwerk und 3-dimensionalen Wegweiserstandorten eine mächtige Topologie aufgebaut wird, die unter anderem Daten zu Wegzeit, Höhendifferenz, Wegdistanz, Gradient und Belag speichert. Die digitalen Rohdaten des Kantons St.Gallen wurden für den Perimeter des gesamten Kantons exklusive der Region Toggenburg ausgehändigt. Da Höhenangaben nur für Wegweiserstandorte verfügbar sind, müssen Höhen für sekundäre Knoten angenähert werden. Eine Überprüfung der Annäherungsqualität mit dem digitalen Höhenmodell der Schweiz ergibt eine akzeptable Abweichung der statistischen Annäherung zur Realität von etwas mehr als 2%. Die konkrete Realisierung einer Topologie in einer MySQL Datenbank wird genau erläutert. Im Speziellen wird auf die Wegzeitberechnung von Wanderungen eingegangen, die der offiziellen Berechnung der Schweizer Wanderwege zu Grunde liegt. Die finale Topologie verknüpft 1'846 Knoten mit 5'142 Pfaden.

In einer zweiten Phase werden die Daten aus den Distanzmodellierungen auf der Topologie visualisiert. Es sollen digitale Karten und das Profil der Modellierung dynamisch generiert werden. Damit das Ausgangskriterium einer barrierefreien Implementierung und einer frei zugänglichen, web basierten Anwendung erreicht wird, wird der moderne, XML standardisierte und offene SVG Programmierungsstandard gewählt. Die Arbeit mit dem neuen und modernen Standard zeigt zugleich dessen Stärken und Schwächen auf.
Damit die Vorteile digitaler Karten gegenüber gedruckten Karten zum Tragen kommen, werden folgende interaktive Funktionen in die digitale Karte eingebaut: dynamische Koordinatenanzeige, Zoomfunktion, Übersichtskarte mit sensitivem Bereich, um den Kartenausschnitt zu verschieben, asynchrones Datennachladen in separaten Layern. Als häufige Problemursache wird die unterschiedliche SVG Unterstützung der unterschiedlichen Browser thematisiert. Dem Benutzer wird eine differenzierte Detailtiefe mittels Layern angeboten. Das asynchrone Datennachladen in die Karte mittels AJAX und die basierenden Techniken werden angesprochen. Weiter wird der Einbezug georeferenzierter Luftbilder, ebenfalls vom Kanton zur Verfügung gestellt, behandelt. Die Luftbilder müssen von hoch aufgelöster Qualität in internettaugliche Rasterbilder komprimiert werden, ohne ihre Referenzierung zu verlieren. Schliesslich wird aufgezeigt, wie eine ansprechende Benutzerschnittstelle die Benutzerinputs und die diversen Implementierungen steuert. Vielfältige Tests auf möglichst vielen unterschiedlichen Plattformen erlauben Aussagen zu Stärken und Schwächen des Produktes.

Leitfragen betreffend Distanzmodellierung, interaktiver Kartographie mit SVG und Nutzen eines online-Wanderplaners können abschliessend beantwortet werden: Distanzmodellierung auf einer passenden Topologie ist grundsätzlich gut realisierbar, auch über Webkomponenten. SVG wird sich nur durchsetzten, wenn nicht nur die Programmiersprache standardisiert ist, sondern auch die SVG Unterstützung in den Browsern vereinheitlicht wird. Interaktive Kartographie und GIS Anwendungen via Internet werden aber die älteste Domäne der Geografen - das Vermessen und Kartieren - gänzlich verändern und in Zukunft wohl gedruckte Karten verdrängen.
Der Einsatz des Produktes in Realität wird vor allem für Tourismusregionen und Wandergebiete gesehen: definierte Regionen, die sich mit Wanderwegen oder ähnlichen Streckennetzen wie Skipisten oder Langlaufloipen profilieren und dem Kunden einen attraktiven Service bieten möchten. Entwicklungspotential steckt in der Erweiterung der Datenbasis mit zusätzlichen Informationen zu sekundären Diensten wie Restaurantstandorten, Fahrplananbindungen, Sehenswürdigkeiten, GPS Export oder vorgefertigten Wandervorschlägen. Diese Ansatzpunkte bieten Möglichkeiten, den wirtschaftlichen Nutzen des Produktes über den Eigenwert hinaus zu erhöhen und für Regionen zusätzlich interessant zu machen.

* * * * * * *

Resumé

La présente thèse se penche sur la question de la cartographie et de la modélisation des distances, sur une base interactive, pour des sentiers de randonnée. De la base de données brute jusqu'au produit final, nous avons développé une procédure qui soit compréhensible et qui prenne en compte les critères clé suivants: analyse des coûts sur une topologie indépendante, génération dynamique de cartes et diagrammes, utilisation du SVG-standard progressif et ouvert, application possible sur tous les navigateurs, librement accessible par Internet et sans installation de l'application par le client.

Dans une première phase, la théorie des espaces et des distances dans un sens géomatique, la théorie des graphes et de la modélisation des distances sont introduites. Le centre d'intérêt est lié à la révision de l'algorithme de Dijkstra, qui calcule les coûts minimaux basés sur une topologie indépendante. Avec une pseudo-heuristique développée indépendamment, on essaie de réduire la durée d'exécution caractéristique des topologies denses et d'optimiser l'algorithme pour une utilisation sur Internet. Pour cela, des notions géographiques et des concepts informatiques sont combinés. Un ensemble de comparaisons de l'algorithme avec et sans l'heuristique montre une progression de rapidité allant jusqu'à 35%.
La modélisation des distances n'est applicable que sur une topologie solide. Le lecteur est guidé pas à pas dans la production de cette topologie à partir des données brutes contenant le réseau des chemins de randonnée (en deux dimensions) et la position des indications Suisse Rando (en trois dimensions). La topologie contient de nombreuses informations dont la durée de marche, la différence d'altitude, la distance, le gradient et le type de surface. Les données brutes numériques du canton de St.Gall ont été fournies pour le périmètre du canton, à l'exclusion du Toggenburg. Comme les indications altitudinales ne sont disponibles qu'à l'emplacement des poteaux indicatifs, l'altitude des carrefours secondaires a été déduite par approximation. Un test qualitatif de l'approximation à l'aide du modèle d'altitude numérique de la région indique un écart d'environ 2% par rapport à la réalité. La réalisation complète de la topologie dans une base de données MySQL est décrite en détail. Les calculs pour la durée de marche sont basés sur la segmentation officielle de Suisse Rando. La topologie finale lie 1'846 noeuds avec 5'142 sentiers.

Afin de tirer parti des avantages des cartes interactives par rapport aux cartes analogiques, les fonctions interactives suivantes sont implémentées dans la carte numérique: affichage direct des coordonnées, fonction de zoom, vue d'ensemble sensible sur laquelle on peut modifier la fenêtre de la carte, chargement asynchrone des données supplémentaires dans des niveaux séparés. Une cause fréquente de problèmes est le fait que les navigateurs ne supportent pas le standard SVG de la même façon. Cet aspect est traité en détail dans ce travail. L'utilisateur peut choisir son niveau de détail individuellement. Le chargement asynchrone de données dans la carte à l'aide de AJAX et les techniques nécessaires à la réalisation sont traités. De plus, l'introduction des photos aériennes géoréférencées est discutée. Les photos aériennes doivent êtres compressées et réduites, passant d'une excellente résolution à une qualité moindre convenant à Internet, sans perdre leurs références. Enfin, il est montré comment une interface interactive dirige les implémentations diverses. Des tests variés réalisés sur une large palette de navigateurs permettent de témoigner des forces et des faiblesses du produit.

La conclusion permet de répondre aux questions clés, concernant la modélisation des distances, la cartographie interactive avec SVG et l'avantage d'un planificateur interactif de sentiers de randonnées. La modélisation des distances sur une topologie adaptée est, en principe, réalisable, même avec des composants Web. SVG va pouvoir s'appliquer si, non seulement le langage de programmation est standardisé, mais aussi si les navigateurs supportent uniformément SVG. La cartographie interactive et les applications de SIG par Internet sont en train de modifier le domaine le plus ancien des géographes - les mesures et la cartographie - et ont la capacité de remplacer les cartes analogiques dans un futur proche. L'utilisation du produit dans la réalité a surtout lieu dans des régions touristiques et de randonnées. Ce sont des régions bien définies qui se singularisent par la présence de sentiers ou de réseaux similaires tels des pistes de ski ou des installations pour le ski de randonnée et qui veulent offrir un service attractif à leurs clients. Le système a un potentiel de développement qui est lié à une extension de la base de données avec des informations additionnelles pour des service secondaires comme la localisation de restaurants, la connexion à des horaires de transports publics, la localisation de centres d'intérêt, des propositions de randonnées particulières, ou encore l'exportation des résultats sur GPS. Ces thèmes de développement proposent des pistes en vue d'augmenter le bénéfice économique du système et de le rendre plus attractif pour les régions concernées.

* * * * * * *

Contents

List of Figures

List of Tables

1 Preface

I really like hiking. There is no better way to relax than hiking on foreign tracks from signpost to signpost. You can learn much about the region, you are mostly away from traffic and noise and you always discover new things in nature. Hiking as leisure is engrained in Swiss mentality and in my opinion it is not only a sporting amusement but also a cultural one.

Nowadays, applications with route planners spring up like mushrooms and rapidly diffuse. Due to Google, Search or Endoxon, the companies have learned that within this sector economic benefit can be earned. Their function is always quite similar: you can plan a trip from A to B over all possible road categories. Then you get the result showing either the fastest or the shortest way to the destination. You can print the details, and plot the profile. With their help you will certainly find your route without any problems. Thus a printed route map is not necessary anymore. The digital representation is able to substitute the printed map.
We often have a map in front of us, when we plan a hike, for example on holiday or simply for a weekend trip. For such cases, there exists no counterpart to the road planers mentioned above. If you don't want to stay for weeks it is not worth buying your own printed map just to plan one hike. Disposing a hike planing tool available free on Internet that provides analogical functionalities as the known systems and goes even further, would be a great invention. This reflection guided me to the thesis' topic. I would like to understand all that lies behind a route planer, which concepts are it's backbone, how paths can be calculated and of course, how the results can be visualized. The adaption of the concepts for hiking purposes, where probably different cost factors are important, and the possibility to go a step further and dispose even an interface to interchange routes and hiking ideas, is my motivation. My goal during this thesis is to bend the bow from theory to a final application with all steps of analysis, conceptualization, generation, realization and finally visualization and presentation.

Personally it is most important to identify with a project lasting for such a long period. Three major facts guarantee this in my case: with hiking routs, I have chosen a topic which really interests me and has a reference to my every day life. The combination of Geography and Computer Science as my university studies are both integrated into the thesis. These two subjects equally represent my interests. I have been kept motivated by the fact of finally having something that can be applied and developed not only theoretically but allows finding realization.

Without the help and support of a lot of my dear friends and without the good contacts that could be arranged during the last year, the thesis could not have been realized. It is here the place to thank each person having helped to finish this thesis successfully:

- Prof. Claude Collet, Fribourg
  UniFR, supervision
- Viktor Styger, Neu St. Johann
  Responsibility "St.Galler Wanderwege"
- Tobias Reber, Kopenhagen
  Technical support
- Prof. Martin Lehmann, Bern
  UNIBE, ideas and support
- Dr. Regula Rohner, Zürich
  English corrections
- Margrit Sieber, Steinach
  German corrections
- Valentine Duhem, Zürich
  French corrections
- Michael Götte, Tübach
  Mayor, access communal homepage
- Ernst Locher, Tübach
  Raffeisenbank Steinach, printing

Last but not least my thanks go to my family and my cohabitant who have financially supported the project and have motivated me during the whole process. Without them, the realization would have been impossible.

2 Introduction

Interactive applications visualizing geographic features are getting more and more dispersed. With progress in Computer Science and Remote Sensing during the last decades, the digital data basis and the techniques for its manipulation have dramatically developed. In a time where services are more often disposed over Internet the need for jumping on the bandwagon increases in the geographic domain, too. Cartographic applications combined with interactive functionality and guided through user input are the main geographic excesses in this domain.

The thesis should approach each necessary step to go for Internet based cartography and distance modeling of hiking trails.
Our goal is to develop an interactive hiking planing application, which is the result of distance modeling on an individual topology and interactive mapping. Based on results computed by distance modeling techniques and their visualization with interactive mapping through Scalable Vector Graphics (SVG), we aim to invent a system restricted by the following main criteria: cost factor analysis on own topology, totally dynamical generation of maps and diagrams in SVG, barrier-free application and free availability on the Internet without the need for software installation. Within the hiking sector, we want to go through every detail, from theoretical background to the final application. Knowing the digitally rehashed data for the hiking network in the canton of St.Gall, the hiking sector qualifies for this thesis.

How can we deduce a conform hiking network topology out of the raw data? Is it possible to characterize the topology in a usual database? Which thoughts have to be made to guarantee a functioning use on the Internet? These are questions that have to be answered to come from raw geographic information to a data basis which is suitable for interactive application.

How do we have to proceed if we like to apply calculations on the built data representation to a hiking topology? What's behind a calculation finding the shortest, fastest or optimal way? Is it possible to implement such a calculation working with our data basis? Which adaption has to be attached to the computation to make it compatible to Internet use? The answers to these questions let us develop an individually distinct approach to distance modeling. Within this part we have to empathize with a hiker's behavior anytime. We have to think about his intentions and his wishes for an interactive application in the hiking sector.

For interactive mapping we want to use the new and modern SVG standard technology. Here the knowledge base won't be as much extended. We really want to deal with a new technology which is said to be the future technique for interactive mapping. Will the work with SVG during this thesis confirm it's potential for the future? Which are advantages and where are possible disadvantages? Is it possible to really generate SVG maps out of our calculated data? What cartographic aspects have to be considered when interactively mapping a perimeter? Which experiences can help descendant developers to deal with SVG?

More generally we want to discuss new possibilities given by interactive mapping. As cartography is the Geographer's oldest trade, it is important to know about chances in this domain. We assume interactive geographic application to have great economic potential, wherefore we want to discuss a possible use of the hiking planer, too.

2.1 Thesis structure

The thesis is structured into five parts:

Part one introduces into the thematic and focuses on differences between existing products and the one to be developed. The digital revolution does not stop for Cartography. The revolution from printed maps to their digitized counterparts is discussed, too.

In the second part the process from raw data to the final topology is investigated. How to build up and design the future topology? From a geographical point of view, how does

it have to be constructed to support all future needs? How to transform the raw data into the corresponding form that matches the topological requirements? Questions like these are accurately answered. The underlying theory of graphs and the used approach to calculate on these graphs are introduced, too. Dijkstra's algorithm as a core theme is intensively described. The development of a supporting heuristic based on geographical ideas is discussed. Finally the procedure from theory to practice can be directly followed when writing how to apply calculations on our topology.

A third part is dedicated to the implementation of the web framework. Visualization techniques for digitized Cartography and dynamical processing of all kinds of output are dealt with. The disposition of different data on the map within delimitated layers is part of the section. The implementation and manipulation of georeferenced ortho-photographs and the combination of these raster files with vector data is a main content, too.

Finally part four is reserved for tests, conclusion and look into the future. An evaluation of the thesis from the writer's point of view is annexed.

The appendix as fifth part contains the most important source codes referred in the thesis. Further it lists test results and calculation tables.

2.2 time dimension

The preparatory work to this thesis started in December 2006. First, contacts to responsible persons and institutes had to be realized. The question had to be raised if they could accompany a thesis and were willing to provide help and time if needed. Concerning the institutions, the demand for data base for an university study has to be handed in long time before starting with a thesis because passing through the institutional bureaucracy simply needs time.

During the spring 2007, the theme and the ideas were zeroed in feasibility and size of the thesis always in mind. In late spring first contacts followed with Prof. Claude Collet, the supervising responsible of University of Fribourg. In the same time data collection with the help of the cantonal institute for geoinformation in St.Gall started and the necessary data base was delivered in May 2007. Several meetings with the cantonal administration for the St.Gall hike routes have helped defining the final disposition.

After the supervisor had accepted the disposition the development of the thesis began. Building up the topology lasted into September 2007. Visualization and programming followed until mid December. Parallel to the development of the idea of the thesis, the written work took place. Step by step was systematically developed and directly commented and described in the paper.

Final tests and lineups as well as corrections and read through of the thesis lasted until the end of 2007.

Part I
Theoretical Background

3 Digitized Maps on the Internet

This chapter should give a brief overview of the evolution of digital maps focussing on Internet applications. It mentions historical milestones as well as today's techniques and concepts in use. An outlook into the future of digital mapping is given, too. To get closer to the topic, tools for planning hiking tours or similar applications already existing on the Internet are presented briefly. The differences between existing tools and the one that will be developed in the course of this thesis are pointed out.

3.1 Techniques for digitized map presentation

Along with the giant progress in Computer Science during the last decades, the techniques for map representation on screen have widely developed, too. The development in digital mapping has and will be deeply linked with the improvements in Computer Sciences (MacEachren et al. 1994).

3.1.1 Digital mapping in the past

Before the invention of the Internet and the huge progress in Computer Science during the last decades, digital maps have been used primarily in specialized applications on single machines. We remember the first digitized maps in black and white with poor resolution and cornered borders that barley matched our imagination of classic maps. Olbrich et al. mention the quantitative revolution in geography in the 50ies as the birth of computer based cartography. They also cite the first article concerning digitized mapping in 1959 (Olbrich et al. 1996). Digitizing maps has a direct purpose in rapid presentation and illustration of statistical data. MacEachren calls the development from maps as pure visualizing tools to a more elaborate presentation tool the birth of digitized mapping. The accumulation of data sets has forced a more productive way to represent spatial features (MacEachren et al. 1994).
During the 60ies and 70ies digitized mapping was limited for lack of screens, insufficient printers and computer capacities. Screens for graphical representation were largely absent and not yet developed, so poor prints were the only way to visualize the produced maps. Only huge mainframe computers had the capacity to calculate and visualize the input data. So the technique was highly reserved for researchers and potent companies (Olbrich et al. 1996).
The development of the Personal Computer in the mid 80ies brought the desired resources for a wider use of digitized mapping. New software and more flexible programs were innovated. New possibilities with better printers (plotters) and screens enhanced the usergroup of this technology. More and more digitized maps were adjusted to common painted maps and thus better satisfied the users' need, which was the production of a digitized map with the quality of the painted map (Olbrich et al. 1996). In Monmonier's book published in 1982 you can see how the new possibilities were applied and how people and institutions used the tools for digital mapping (Monmonier 1982).

The next innovations basically concerned improvements in software and implementations of GIS.[1] More powerful PC's and better screens allowed to digitize a huge mass of geographical data and now partly replace of classic painted maps. Better printers allowed to get plans and maps in a quality that almost outdo the old maps. With the increasing amount of users, the demand for interactive, flexible maps and software products has advanced. The development of the World Wide

[1] Geographical Information System (GIS): A digital information system allowing to deal with spatial data.

Web 1991 pushed this development ahead, too (Living Internet 2007). In their outlook for the 90ies, Olbirch et al. discuss the notion of desktop-mapping. What is state-of-the-art today was predicted by saying that automation in digital mapping would increase to the point that users could just input the data and get the desired results visualized at once. Multimedia concepts and interactivity as further improvements of printers and screens and the advancing capacity of PC's would allow to produce all our spatial knowledge in digital databases and digital atlases accessible to anyone (Olbrich et al. 1996).

3.1.2 State-of-the-art - Today's techniques

Today more and more spatial data are directly memorized digitally on servers and these data won't be printed nor painted anymore. Even subjects of public law as for e.g. land-register are already digitally recorded in many communities and the digital maps have their legal validity (GeoMedia Technology 2007). So you can say digital maps are getting more and more important and they are slowly replacing traditional maps. The advantages are obvious: (i) accessibility from every computer, (ii) capabilities of different visualizing methods and processing of huge spatial data, (iii) interactivity and multimedia possibilities, (iv) efficient update without the need to redraw the whole map area, (v) fast and productive - so cheaper - way of cartography at all (MacEachren et al. 1994, Muller 1991, Dransch 1997).
Due to the wider range of the Internet digital maps and applied software are accessible for every normal computer user. Products as `www.earth.google.com` or `www.map24.com` have familiarized the Internet community with digital maps since the beginning of this millennium. Surveyors' offices nowadays keep their spatial data digitally and their archives are getting digitized by and by. The shift towards computer applications is now taking place in the most traditional branches of cartography. The quality of maps reaches - and in some cases already exceeds - traditional maps. Their resolution is increasing with the better capacities in Computer Science and Remote Sensing so that resolution is already outdoing painted maps.
If we focus on technologies used for visualizing maps on the World Wide Web we have to mention the following concepts that mostly affect today's methods:

Static maps There are still many badly resolved maps available. These are maps that were compressed to small gif's or jpg's and they are used as normal images. Often their origin lies in a traditional painted map that was scanned and processed after digitizing. These raster graphics are fast but on the other hand multimedia application is not possible and the static form makes them passive. These kinds of digital maps never meet the requirements of digital maps as alternative for traditional maps (Reichenbacher 2005).

Animated Maps Macromedia's programming language "Flash" is now widely used to process digital maps and its applications. Flash allows the programmer to design maps on the basis of geographical databases and has routines to programme interactive possibilities into a map, too. Animated maps are being produced easily. The vector based methods allow more efficient visualization on different screens and better zooms into a feature. Resolution is linked with the geographical data in the database, so it is directly related to measurements made in terrain (Goad 2002, OpenSwf 2007). These vector-based maps already draw even traditional maps, which partly explains Flash's success in this domain. To display Flash files the freely available Macromedia-Flash-Player is necessary. Some good examples for the use of Flash-maps are `www.mapbureau.com/mapgallery` or `www.internationalmapping.com`.

Individual technologies Potent offerers as "ESRI - ArcGIS", "Google" or in Switzerland "Endoxon" have their own

specialized and developed technologies to produce digital maps. They can afford the reasonable expenses for their own programming methods. Their individual way helps establish the leadership. Concerning "Google" we have to mention that many updates for "Google Earth" are OpenSource standard and point to free interfaces to known standards.[2]

SVG Some progressive institutions or bureaus already use SVG (Scalable Vector Graphics) to process their maps. SVG is a programming language purely designed for the purpose of digital map processing. As an XML-Standard it is a standardized language, highly flexible and adaptable. SVG is foresight to be the language for digital maps in the near future. More and more similar products are flooding the market and the Internet (Eisenberg 2002).

3.1.3 Techniques used in the future - SVG

The author is convinced that with SVG a new area in digital cartography is ringing in. SVG unites all the advantages of preceding techniques and was specially developed for cartographic purposes. Dana Tomlin in coherence to SVG puts the main aspect in a nutshell (Tomlin 1990):

> Raster is faster, but raster is vaster, and vector just seems more correcter.

With the vast development in Computer Science the last years, "faster" is no more a problem of digital cartography. More qualitative aspects gain people's interest. Quality in digital cartography goes with vectorization. Vector graphics allow to use interactive concepts. This is far more than just viewing maps as was done before. The broad use of the World Wide Web and the expansion in economic web services that are for sale ask for a compatible, standardized, vectorizing, open and cartographic aspect supporting programming language (Neumann 2003). SVG provides all this. What is more, SVG domiciles many special features meeting the claims of cartographers. Just to state some examples: coordinate support, fast path displaying methods with little memory requirement, sophisticated transformation techniques but also tiny details especially for mapping as special line caps, line joins or adapted typeset (Eisenberg 2002).

Pushed through several prestigious graphic companies as Adobe, Corel, Apple, IBM, HP or Macromedia the first release of SVG 1.0 took place in 2000. The broad support grants a high compatibility in the future, which is absolutely necessary to avoid the rapid disappearance of many of the products in this domain. The relative new SVG format is based on XML and inherits all its advantages. It is published as an open standard and is accessible by everyone (Neumann 2003). The standardization guarantees a longer lifetime since variations are defended. The open standard and the easy going notations and concepts favor SVG over the complex and difficult Flash language.
To broadcast a new language or format, it is totally inevitable to come around with additional features guaranteeing border free compatibility with products still in use. The best language concept does not make sense, if nobody is able to run it. Therefore several free plug-ins exist, some originally offered by the above mentioned supporters and upgrading existing products and thus making them compatible with the SVG standard. Every new release of famous graphic software now supports natively SVG, as do products of manufacturers like Adobe, Corel, ESRI etc. (Adobe 2007a, Corel 2007, ESRI 2007). To let SVG demonstrate its forces in Internet publishing of maps, support of every common web browser is absolutely coercive. In the domain of web browsers SVG also becomes more and more popular and in most new releases SVG is supported natively. Microsoft's popular browser "Internet Explorer" still doesn't support SVG natively,

[2] OpenSource Software is free available and the source code is accessible for everyone. That makes OpenSource software adaptable and flexible for everyone (OpenSource 2007).

but works very well with the freely available Adobe SVG Viewer plug-in. Whereas Microsoft's concurrence has still reacted and progressively implemented native SVG support to its browsers (as do all versions upwards Firefox 1.5, Opera 8.5, Konqueror 3.5.5, Safari 3, Netscape 9), world's browser leader is still resting on its laurels (Code Dread 2007). But as "Internet Explorer" is apprehending aspiring new free browsers, Microsoft's developers promise to implement native SVG support into the next version (SVG Wiki 2007).

This information confirms that the support for SVG in every common browser will be given. Over 98% of worldwide Internet users are able to display SVG graphics (Janco 2007). The difference lies in the kind of support: Either natively or with a plug-in. Adobe's SVG Viewer plug-in, which is still available in the third version and worldwide leader, was specially developed to let SVG fully play off its advantages (Adobe 2007b). This is the reason, why once the plug-in installed, there are no display troubles anymore, for complex SVG applications, too.
One disadvantage is that a user who accesses a site containing SVG contents and has not yet installed the plug-in or uses a browser without native SVG support, has to attain, download and install (this is mainly an automatic process) the plug-in before he can see anything on the screen. This extra procedure may detain certain users from watching your site. Within the following years, however, this small "bug" should disappear, too.

3.2 Overview of existing solutions concerning interactive hike planning

The development in making geographical data accessible for everyone via the Internet and other distribution lines is making great progress. Leica Geosystems, a world leader in geographical information, looks at the GIS future with a meaningful header: "The clickfuture in GIS". This means that GIS more and more penetrate the information world which is accessible to all of us thanks to Internet. GIS usable through the Internet face a prosperous future. In addition, interactivity will become an important use of this online GIS (Leica 2007).
Individual use, free availability, border-free access as global concepts are nowadays in fashion. Nevertheless the following over all view on the most similar existing projects to the one in question shows in the hiking sector, or more generally in more specific leisure topics that no product accomplishes the three mentioned concepts and the designated functionalities. A list with existing variants of similar solutions for hiking planing tools is here given with short explanations and delimitations:

- Swiss Map 25: This product from Swisstopo is available for the whole country and gives the user a full overview on Swiss 1:25'000 maps. On the maps hiking routes are listed and there are also tools allowing waytime calculations and routing. The product needs to be installed on a proper machine and uses quite sophisticated system requirements. It costs 180 CHF per sector as single license. The country is divided into six sectors. The range of multimedia functions and the quality of maps is high, but for rapid searches, either at a public Internet access point, or in a tourist bureau, it is too expensive, needs some introduction time and is bound to a private computer (Swisstopo 2007).

- Requests for prepackaged tours: There exist many applications where, with user defined criteria, the search for readymade routes can be conditioned. The request conveys a list of matching possible routes that are already well described. Unfortunately you are limited by the prepackaged routes and real searches for fastest or shortest ways are not supported. The tours are indeed well documented and visualized on a map. But the storage of predefined tours are limited (Regiun Surselva 2007a, St.Galler Wanderwege 2007). Information on the tours are freely downloadable and don't need any special system equipment to be

watched. An Internet access is sufficient.

- Making your own route: There is a project in the Regiun Surselva, too, based on Macromedia Flash, allowing the user to input his preferred cultural points of interest. The application calculates the way to go there and to meet all the given points. The system only comprises roads down to communal order (Reguin Surselva 2007b). So it is not comparable and usable for hiking purposes and the nodes are restricted by pure cultural points of interest. A free search over every node in the routing graph is not possible in this tool, either.

- GPS Tours: Many tourist information offer prepackaged GPS-Tours nowadays. The tours are easily downloadable on a common GPS receiver and are documented similarly to the offerings discussed before. The corresponding map is mostly available as Flash map, too (Toggenburg 2007, St.Galler Wanderwege 2007).

- Overviews on the route network: The coalition Appenzeller Wanderwege offers an overview network map with indications of waytime for the most important edges. The waytime is only given for climbing direction. With this sort of chart the hiker is able to calculate his own way more or less by hand. The search for the fastest way or alike needs more sophisticated analysis and takes time. Neither the profile nor an exact route planning is given. The map is of low resolution and doesn't accept zooming or panning (Appenzeller Wanderwege 2007).

- Route planer for roads: Not matching for hikers but nevertheless brought up are the well known tools to calculate road trips. Products with quite sophisticated functionalities still exist on the www and are available free. As for example `maps.google.de` or `map24.ch` the tools support different cost factors, too, and let the user input its preferences. The products are mentioned because they give a good indication in which direction the project for hiking trails should go (Google Maps 2007, Map24 2007).

When we analyze the already existing solutions we recognize limits, where the planned system intends to go further and introduces new functionalities. We would like to design a tool combining more or less all the different advantages of the above descriptions. Disposing a tool

- free availability on the Internet
- without the need for installation or special software components
- giving full freedom in the search for hiking routes
- visualizing maps, diagrams and further details in an open standard (SVG)
- disposing an exchange interface to communicate about hiking and searched results via email
- working totally dynamically on an extern server
- can be included in and adapted to every existing web framework
- is fully multilingual
- can be easily transferred to other hiking data and topologies

equals a new concept not available until now. Of course the system cannot compete with the big players on the market like Google or Search. But it can fill a niche which has been neglected until now and is therefore justified.

4 Overview over the Sources of Raw Data

Early made contacts to the cantonal institute for geoinformation of St.Gall proved itself a good strategy. With the cooperation of the St.Gall section of the Schweizer Wanderwege, which is an attached institute to the cantonal geoinformation bureau, additional weight could be given to the demands for specific geodata. The first informative request to the responsible person in the unit took place in January 2007. Institutional clarification always takes time and the canton being the owner of the geodata copyright couldn't rush the permission for accessibility of the desired data. Nevertheless the responsible bureau accepted the request for data with only one constraint: Not the whole data amount for the whole canton was handed over, only about 75% of the data were committed. We agreed on leaving out a great part of the Toggenburg. The cantonal parts of the City of St.Gall, Rorschach, Unterrheintal, Oberrheintal and Sarganserland are fully masked with the obtained data. As a consequence the available data has a touch of artificiality, because paths on data borders do not end on nodes but on artificial intersections.
For the described geographical area the canton has delivered the data on hiking routes containing the data on pathway of trails, surface of trails and locations of signposts added with much metainformation (height, district, municipality, location name, internal features). There is a software, running on ESRI's ArcGIS that allows the responsible staff to digitally manage hiking routes of the canton. In this software, a digital terrain model is cutting the pathways and thus allows so to extract elevation data, too. Due to the high data amount for the digital terrain model and the heavy computer charge, the bureau makes an external company calculate the blend between model and digital pathway data every year. This is the purpose why no elevation data was made available. To get the important relation to elevation, all locations with signpost are stored with its exact height. All this digital information is present in `.shp` file structure and is best presentable with the corresponding ESRI ArcGIS Software.
Further the orthophotographs for the area, already georeferenced, are accessible in a very high resolution. The photographs are divided in areas of 1 km^2, are in RGB color scheme and coded in `.tif` format. The filenames are defined by the left upper image corners in ordinary country coordinates. So the file `726_262.tif` for example illustrates the square kilometer defined by the coordinates (726'000, 262'000), (727'000, 262'000), (727'000, 261'000) and (726'000, 261'000). Resolution is given by 1 pixel corresponding to 2 meters in reality. It is obvious that this highly resolving images with an average memory requirement of over 60MB are not compatible with Internet use. At this point improvements and strong data reduction has to be set on.

There is, unfortunately, no digital data available on further interesting details for hikers. Neither data on points of interest nor data for access points to public transport or restaurants are registered digitally. This is also confirmed by the contact of the St.Galler Wanderwege. For the future, planing looks ahead to list such metadata on hiking trails and implement it to the existing GIS (Styger 2007a).

From the Geographic Institute UniFR we additionally have vector data in a low resolution for main roads, railway traffic, borders, rivers and lakes on hand.

Summarizing data on-hand, the important data for developing the foresight project is available thanks to the co-operation with the canton's bureau and the help of the St.Galler Wanderwege. This comprises the basegeometry, added information needed for calculations of trails and different cost factors and furthermore visualization with orthophotographs.

5 Distance Modeling

5.1 Basic concepts of distance modeling in GIS

We don't like to plunge directly into the theory of graphs before briefly discussing the fundamental concepts and notions in the field of distance modeling.

If we assume a hiker walking from location a to location b, we must know what factors influence the time needed for the hike. The two parameters length and velocity obviously play an important role. But when we differentiate more, velocity depends on other external factors like coverage, altitude difference or gradient. Walking on a hard ground goes easier than hiking on a natural ground covered with high grass. The "friction", which brakes, is higher due to the natural ground. Climbing a steep slope highly reduces velocity whereas a little descent increases velocity. In such a case the topography is responsible for the friction. Thus walking the same length and probably the same hike route takes different times depending on the direction the hiker takes.
We can clearly see that the space for the hiker is not homogenous at all. The hiker moves in a topographic space where the surface is not plane. We speak of an "anisotropic skewed surface" that models our hiker's reality best (GITTA 2007, Collet 2007). Now to model the distance on this surface seems to be easy. It is easy as long as we speak about the popular meaning of distance that is defined by the euclidian metric and characterizes the spatial geometric distance. The notion of "distance" implies certainly the euclidian meaning but others, too, depending on the metric used: Besides the euclidian distance there are other distances our hiker is interested in. The time needed for example is of a high interest potential. But also other distances with non-euclidian metrics are interesting: "altitude difference" which describes the distance in dependence of the meters in altitude which have to be climbed from a to b or "gradient", which gives a relation between gradients to climb during the hike. When we compare for example the euclidian distance with the "time distance", we recognize a fundamental difference in their metrics: whereas the euclidian metric is non-directional, the time metric is directional. There is a distance difference when hiking from a to b or hiking from b to a due to the topographical friction that is directional.

Basically "distance" describes a quantitative spatial relation. Another fundamental concept used in geography is the "topology", which describes more a qualitative relation in space. The topology gives answers about the contiguity and connectivity of features. Due to topological expressions we can qualitatively locate objects in relation to other objects lying in their neighborhood. The proximity of objects can be described with topological notions (GITTA 2007). Going back to our hiker, the topology of his real world would describe for example which hike route connects two locations. Is there a railway station accessible from the actual location? Which way sections do I have to pass by to get to the desired destination? Are there multiple possibilities to come from A to B? All these questions deal with topological information about the hiker's reality.

For our project, we spring from a skewed anisotropic surface to model. The following chapters deal with the process of building up the corresponding topology, definition and working with multiple distances. Finally the coupling of topology with the surface is in focus when trying to visualize hiking paths on the map.

5.2 Theory of graphs

We trace back the theoretical history of the theory of graphs until the 18^{th} century when Leonhardt Euler solved the problem known as "the bridges of Königsberg" (see fig. 1). In 1736 he solved the problem of crossing just once each of the seven bridges linking the borders and an island in the river Pregel by mapping it to a graph-problem (illustrated in fig. 2). He thus solved the problem for every ar-

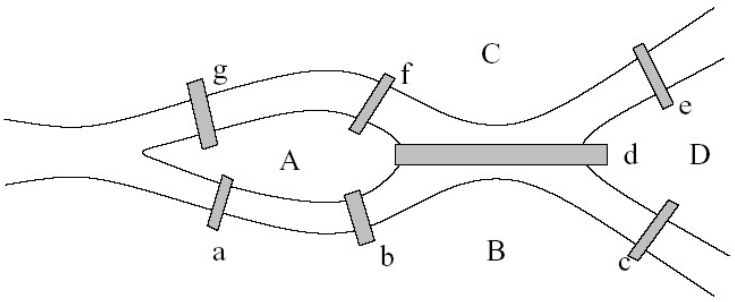

Figure 1: *The bridges of Königsberg as illustrated in Hartmann et al 2006. Is it possible to walk through the town crossing every bridge once and get back to the beginning? This problem guided Euler to the conception of the graph theory.*

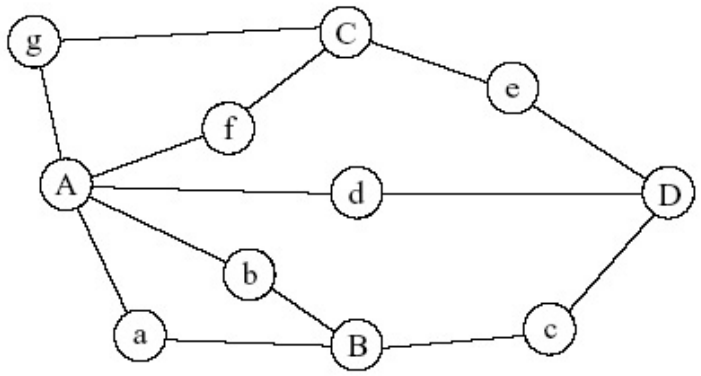

Figure 2: *The bridges of Königsberg translated to a graph with edges and nodes as illustrated in Hartmann et al 2006.*

bitrary graph (Hartmann et al 2006). Euler further developed more fields of the graph theory.

In our everyday life many problems are abstracted with representations as graphs: Relational problems are preferred for the theory because their representation logically refers to graphs that illustrate well the existing connections. Some applications are computer networking, travel-way-problems, chip-design, transportation, sociology with actors relations, in chemistry's molecular-science or in physics when calculating atomic structures. The field of application covers almost every part of our lives. During the last decades the theory became even more famous because the additional computing power of modern computers allow to solve rich and huge optimization problems on wide graphs (Gross et al 1999).

5.2.1 Notions and basic concept

As a mathematical concept the graph theory establishes clear notions and definitions to the single components and methods. During this chapter the most important definitions for distance modeling are introduced. The knowledge of these basic notions will help to define clearly our approach to model a hiking network.

As illustrated in fig. 2 Euler painted fixed points (bridges, borders, island) and edges (connections to this fixed points) to model the graph. These are the basic components of an arbitrary graph: nodes (vertices, points) and edges (lines) linking nodes. Mathematically spoken an undirected graph G is a pair of sets containing vertices V and edges E such that

$$G = (V, E) \quad (1)$$

$$E \subseteq [V]^2 \quad (2)$$

Equation 2 declares that every edge is a 2-element subset of V. This means every edge contains of two nodes that are connected by the edge. The elements of E are its endpoints (Diestel 2005). The *order* N of a graph is defined as $N = |V|$ whereas the *size* M is $M = |E|$ and counts the number of edges in the graph. The relation between size and order characterizes the graph's *density*. The smaller $\frac{|V|}{|E|}$, the denser the graph, the higher the node's degrees (Sedgewick 2002).

Important for the graph's topology are the descriptions of linkage and hierarchy in the network. Two nodes are adjacent when they are connected with an edge. So two vertices $\{i, j\} \in V$ are neighboring when $\{i, j\} \in E$. The *degree* of a node is defined as the number of adjacent nodes. Vertices with degree zero are called *isolated* (Hartmann et al 2006). One can equally say that the degree of a node i is the number of edges at i.

We also have to look at paths and its correlations with topology. A path P is a non-empty subgraph of G such that

$$\begin{aligned} P \in G &= (V, E) \\ V &= \{x_0, x_1, \ldots, x_k\} \\ E &= \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\} \end{aligned} \tag{3}$$

where all nodes x_i are distinct. The vertices x_1 and x_k are linked by P and are its endpoints (we often say a path P from x_1 to x_k). The nodes x_2 to x_{k-1} are the *inner vertices* of P. The number k of edges in a path P is called its *length* P^k (Diestel 2005).

A graph is called *directed* or *digraph* when its edges $(i, j) \subset V \times V$ are ordered pairs of nodes. One can imagine a digraph as a graph with directed edges as "arrows" or "arcs". Now it is possible to characterize the *degree* more specifically: The *outdegree* of a node is the number of edges outgoing, whereas the *indegree* is the number of incoming edges.
Finally it is necessary to introduce the notion *weight* . A *weighted graph* $G = (V, E, \omega)$ is a graph where its edges are weighted with a rational number:

$$\omega : E \to \mathbb{R} \tag{4}$$

Weighted graphs occur frequently in graph theory. For example weights characterize the travel time corresponding to the edge, transportation costs, the strength of a force acting with the edge or in a social graph representing friendships give information to the intensity of the friendship between the actors. There are also other terms used for weight as *cost* or *distance*. In reality many optimization problems have the goal to find a path with minimal or maximal weight (Bondy et al 1976).

Now that all the necessary definitions and notions have been introduced fig. 3 repeats and illustrates again the basic concepts of graphs. One can generally characterize the shown graph G as follows:

- G is a directed, weighted graph
- Its size is 11 and its order is 6
- $V = \{1, 2, 3, 4, 5, 6\}$
- $E = \{e_{(1,2)}, e_{(2,3)}, e_{(2,6)}, e_{(6,1)}, e_{(6,3)}, e_{(3,4)}, e_{(3,5)}, e_{(4,5)}, e_{(5,6)}, e_{(6,5)}, e_{(2,4)}\}$

If we look more precisely at node 3:

- Its degree is 4
- There are 3 incoming edges, so its indegree is 3 and its outdegree is 1
- Adjacent nodes are: 2,4,5,6

When we want to define the shown path:

- $P \in G = (V, E)$
- $V = \{1, 2, 3, 5, 4\}$
- $E = \{e_{(1,2)}, e_{(2,3)}, e_{(3,5)}, e_{(5,4)}\}$
- Weights are $\omega(E) = \{e_{(1,2)} = 3, e_{(2,3)} = 1.5, e_{(3,5)} = 0.7, e_{(5,4)} = 2.0\}$
- Its length is 4 (P^4)

5.2.2 Dijkstra's algorithm calculating minimal weights

As mentioned before, we often look for minimizing or maximizing an optimization problem within graphs. One of these problems is the well known application that calculate one's timetable or give out the best way to travel

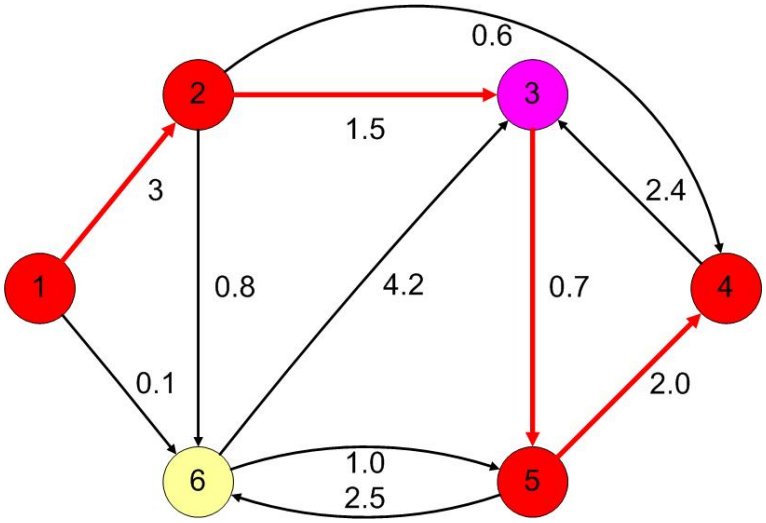

Figure 3: *This figure shows a sample graph G with a path P represented in red.*

from x to y. On the Internet many types of such applications are available for free. `sbb.ch` searches the fastest way on their track system to go from x to y, `map24.ch` looks for the best route to drive from a to b or `expedia.com` looks for the cheapest flight from the departure to the destination.

How is it possible to calculate the wished results within a short time? If we look at a simple weighted graph and overview it, we are able to draw a minimal weighted path from a to b by hand. But graphs with sizes greater than approximately 20 are almost impossible to overview and solve by hand. If you work on bigger graphs you have to implement an algorithm that automates the requests.

In 1959 Edsger Dijkstra published his famous paper "A Note on two Problems in Connexion with Graphs" wherein he presented a possible algorithm to solve a so-called shortest path problem for directed graphs with non negative edge weights (Dijkstra 1959). If we imagine a weighted digraph where nodes represent localities, edges represent roads connecting these localities and edge weights show the distances of the roads; Dijkstra's algorithm is used to calculate the shortest path from a vertex a to vertex b.

For the understanding of the way Dijkstra proceeds, a qualitative explication of the algorithm's functioning is given. The proceeding is quite intuitive, we unconsciously imitate Dijkstra's idea when we are proceeding in simple digraphs (after Bernreuther 2000).

Step 1 initialization The initial vertex is looked for (the startnode) and the distance is initialized with the start value. All its neighbors are recursively checked.

Step 2 neighbors The weights to the neighbors are compared and the smallest overwrites the actual distance.

Step 3 priority The neighbor connected with the smallest weight gets the priority to the further search. All its neighbors are checked again and the edge's weight is added to the actual weight. The smallest summed weight overwrites again the actual distance. If there exists a neighbor a hierarchy higher with a smaller summed up weight, it is the one that gets the priority and its summed weight is stored.

Step 4 recursion The recursive search goes on from the startnode's neighbors to its neighbors until the final node is found. The stored distance is the shortest path from start- to endnode.

If we apply these steps to the example graph in Figure 3 the proceeding of the algorithm becomes clearer. As example query the shortest path from vertex 1 to vertex 4 should be found

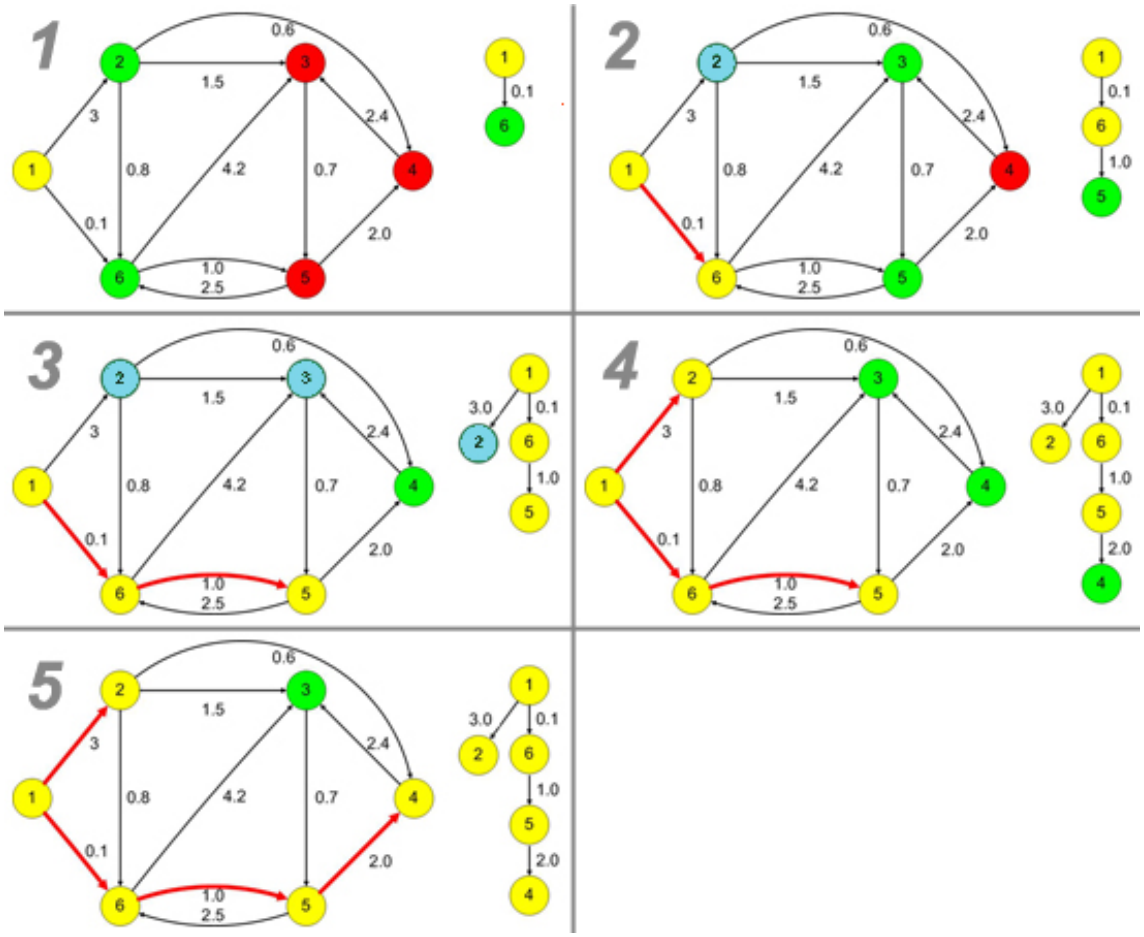

Figure 4: *The same graph as in Figure 3. Stepwise visualization of Dijkstra's algorithm looking for the shortest path from vertex 1 to 4. Yellow nodes are already treated and introduced in memory. Green nodes are neighbors to be looked at. Blue nodes are neighbors still considered but not directly in priority. Red nodes are in the shown step not in consideration. On the right hand a stepwise built spanning tree with all shortest paths to the matched nodes is drawn.*

applying Dijkstra's algorithm. Each step of the approach is formulated in the following and directly visualized stepwise in Figure 4:

1. The graph illustrated matches the requirements: It is directed and there are no negative weights. So it is possible to apply Dijkstra's algorithm.

2. Initialize vertex 1 and set distance $d = 0$.

3. From its neighbors 2 and 6 the weight to vertex 6 is smaller $\omega(e_{1,6}) = 0.1$. So $d = 0.1$ and priority is lying at node 6. (Step 1 and 2 in fig. 4).

4. The outdegree of node 6 is 2, so one has to look at the next two nodes accessible from node 6: node 5 and node 3. Cumulative distance to 3 would be $d = 1.3$ and to node 5 $d = 1.1$. So node 5 gets the priority and the actual distance is renewed to $d = 1.1$. There is no path a hierarchy higher that is matching different nodes with less distance. (Step 3 in fig. 4).

5. From node 5 with outdegree 2 the new node 4 gets into relation. It has to be mentioned that already prioritized nodes won't get in touch again by the algorithm (as would get node 6 again). The summed weight to node 4 would be $d = 3.1$ $(0.1 + 1.0 + 2.0 = 3.1)$. Now another "shorter" node at higher hierarchies is coming into the game: node 2 has a summed weight from node 1 $\omega(e_{1,2}) = 3.0$. So now priority is changing to node 2 and actual distance is set

to $d = 3.0$. (Step 4 in fig. 4).

6. The outgoing edges to be looked at are connecting now node 2 with node 3 and 4. Cumulated weight to node 4 would be $d = 3.6$ $(3.0 + 0.6 = 3.6)$ and would be considered, because summed weight to node 3 would be higher with $d = 4.5$. But because node 5 is still considered and its cumulated weight to node 4 is the smallest $d = 3.1$ it gets into touch and node 4 becomes priority. The distance has to be actualized with the value 3.1. (Step 5 in fig. 4).

7. The query has finished, because the endnode 4 has been found. The recursive query has stopped now. The problem has been solved with Dijkstra's method and its result presents itself as follows: The minimal weighted path from node 1 to 4 is $V = \{1, 6, 5, 4\}$ with edges $E = \{e_{(1,6)}, e_{(6,5)}, e_{(5,4)}\}$ and weights $\omega(E) = \{e_{(1,6)} = 0.1, e_{(6,5)} = 1.0, e_{(5,4)} = 2.0\}$. It's shortest path has a cumulated weight of 3.1 or in other words its distance is 3.1. The path's length is 3.

It has to be added that during the searching steps within the algorithm for all treated nodes we have integrally found their shortest paths to the startnode. In the case above the shortest paths from node 1 to nodes 2,4,5,6 have been found. All these nodes were once in priority and it does not exist a shorter way to access them. In other words it can be said the algorithm spans a "tree" of nodes where the branches characterize the shortest ways to access these nodes (Sedgewick 2002). Figure 4 illustrates the stepwise growing of the tree of shortest distances on its right hand, too. This algorithm's advantage is also used in computer network routing, where Dijkstra's algorithm is used in the famous OSPF-routing-method[3], therefore Dijkstra's algorithm has its eminence (Tanenbaum 2003).

The above pseudo code is implemented as copied in appendix C.3.1. Because one is working fully server-based and the inputs are processed directly into the calculations, the code is directly executable in PHP. One is making shifts of object oriented PHP - meaning the full functionality lies in methods of one and the same class. The succession of steps is implemented in several methods and in the end the full class "trailDijkstra" is present.

An important characteristic of an algorithm is its complexity or runtime. The used time to compute the result is of big importance especially in graph theory, because there are mostly huge amounts of data to be treated. Graphs with order greater than 1000 are often used. The fact that as late as 1959 Dijkstra invented the first method good enough to calculate on wide graphs already implies that it is a quite complex problem. In literature we can find following information to the algorithm's complexity: In the worst of all cases the algorithm has to touch every edge and every node until it reaches the endnode. So the complexity depends directly on the graph's order and size. Regardless which technique to store the data is used, we have to read and process data and in a next step we have to write data down (Bernreuther 2000). This is relevant for each vertex in the graph. In Computer Science such a complexity is known as

$$O(|V^2|) \tag{5}$$

(Sedgewick 2002). It means that in a worst case scenario V^2 basic computing steps are needed to solve the problem. The notation is independent of computer power, it gives just a relative limit. An exponential complexity (eq. 5) is quite bad for calculating large problems. Such a complexity leads up to a higher and higher exponential ascending runtime. With the progress in computer power exponential complexities become more and more manageable, even for a large amount of data.
There is, depending on the graph's density, possible improvement in complexity. With more sophisticated ways to store datastructures the complexity can reach

$$O(|E| \times \log |V|) \tag{6}$$

[3]OSPF (Open Shortest Path First): Routing protocol used in routing techniques to calculate the best way, data packets are routed the most effectively through the network.

(Bernreuther 2000). The implementation of the required data structures for ontime calculation via extern servers won't be an alternative for the hike planning tool developed within this thesis.

There also exist other possible algorithms to calculate almost similar "shortest-path-problems". For the sake of completeness they are briefly brought up here.

Kruskal's algorithm Invented in 1956, Kruskal's algorithm finds a path in a graph so that all nodes are somehow connected with an edge (no isolated nodes) and the total edge-weight is a minimum (Kruskal 1956).

Prim's algorithm As Kruskal's algorithm Prim's algorithm calculates a minimal subgraph that includes every nodes and its total edge-weight is a minimum (Gross et al 1999). Prim's and Kruskal's algorithm nevertheless are not appropriate for finding shortest paths from a to b.

A* algorithm This algorithm, invented in 1968 by Hart, Nilsson and Raphael, is probably the most sophisticated variant treating minimal weight problems. The new is a heuristic that limits the recursive search. It guides the prioritized search in the direction of the endnode. So possibilities decrease and computing time does so, too. Large parties of the examined graph are immediately setted aside and the search is concentrated towards the endnode. The implementation of such a heuristic is difficult and depends on the graph's density (Gross et al 1999).

For the hike planning tool in this thesis, the A* algorithm is not appropriate, because a needed heuristic in the sense of the algorithm could hardly be coded within the limits of hard- and software techniques used. Instead of the A* algorithm, a pseudo-heuristic that as well reduces the graph's size and order to be scanned will be invented.[4]

5.3 Different cost factors

5.3.1 walking time

In reality a hiker is mostly interested in the time needed for a hike. The times are usually printed on the signposts and are giving the hiker a clue how long it takes to get to the destination.

Within rough calculation we expect a normal hiker to cover 4.2km an hour when the trail has no slope. For every ± 100 meters in altitude one should add another 1000m to the length of the trail to get the very roughly approximated time to walk. This e.g. a trail of length 6.4km and 200 meters in altitude would result in a straight hypothetic trail of 8.4 km and would be walked within 2 hours. This data is approved by Mr. Styger's of the St.Galler Wanderwege. He says that this rough approximation helps to get a first result rapidly and is widely used. But for more precise results one has to adapt a much more complicated calculation (Styger 2007b).
It is obvious that the time for a hike is especially determined by the parameters height difference or slope and length of the track. There are other components (surface, temperature, weather conditions, etc.), too, but without much effects on the time. This secondary parameters are left out because no qualitative data is available to this components and because they do not influence the resulting time significantly. According to the Swiss Hiking Association only the two parameters height difference and length are applied for time calculations (Schweizer Wanderwege 2007a). The Swiss Hiking Association is also defining the function to calculate very precisely the transit time for a specific trail taking the height difference and the length into consideration. A highly complex polynomial of degree 15 is approximating best the waytime for each possible slope. Of course for our purposes for an exact calculation for the fastest way, we use the complex and detailed polynome that gives

[4] Look up chapters 5.4.1 and 5.4.2 on page 18f

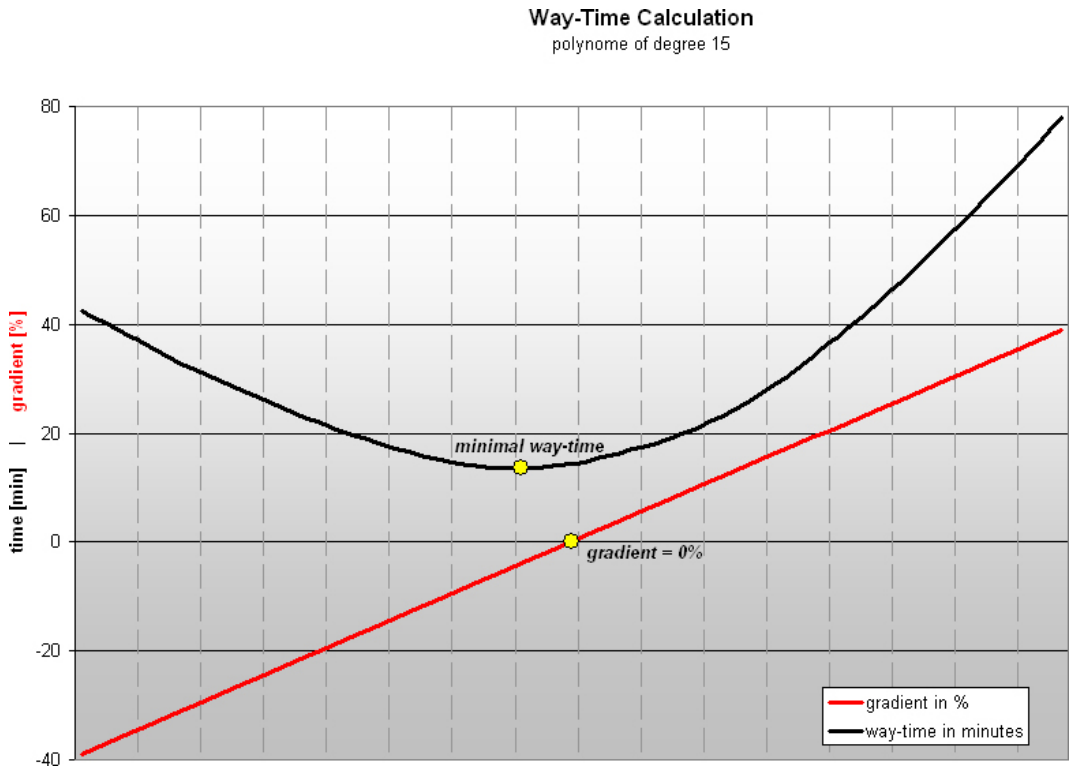

Figure 5: *The graph shows the waytime in relation to the slope gradient of a trail. Data is taken out of the official calculation table for hiking times of the swiss hiking association "Schweizer Wanderwege" (Schweizer Wanderwege 2007a).*

the most detailed results. In appendix C.1.8, lines 9 to 25 and 52 to 57, the whole polynome is copied as it is calculating waytime for the track segments in our topology. Figure 5 visualizes the resulting graph that is showing the development of waytime depending on slope gradient. If one wants to draw a brief conclusion of the graph in Figure 5, the following things should be mentioned: (i) the more a trail ascends, the more time is needed for the same length and this develops exponentially, (ii) if the trail descends between 0 and -8%, less time is needed than if you walk straight on; the more the path descends towards -8%, even the less time is needed. (iii) The hiker keeps the highest velocity by a gradient of -8% and therefore is the fastest. What surprises us is the fact that (iv) when the trail gradient is less than -8%, the hiking time needed increases again. The more the trail is descending, the more time is used to cover the length of the track. It must be considered that the exponential evolution of waytime with descending trails is not as much marked as with climbing trails.

It is important to envision that the polynomial would also calculate higher gradients than ± 40 percent as it is shown in Figure 5. In such cases (normally no such gradients appear in reality) the calculations would give abnormal and absurd values because of the exponential development of the polynomial. When developing the tool we have to bear in mind that the waytime is constraint to the abnormal gradients.

5.3.2 Access user preferences with different cost factors

As time is probably the most important key feature for a hiker's choice of trail, there are several other dimensions that influence he's preferences for a specific trail. With the available data (as described in section 4) other weighting factors are implementable. Aim is, to let the hiker join his private preferences as much as possible when looking for a trail; as it is common in already existing projects, where

e.g. the interested party can choose the route on the street from a to b regarding the fastest way possible (time), the shortest way (length), the way with the smallest difference in meters in altitude etc.
The data given allows to work with the following cost factors, too, going further than time. This allows the user to set his preferences when looking for a specific trail:

Shortest path Regarding the length of the edges, a search for the geographically shortest path is possible to implement. There is no regard to differences in height, just geometric length is looked up, when going through the algorithm.

Slope gradients Someone likes to walk the flattest path possible avoiding heavy mountings and prefers to look for a trail with minimal slope gradient. Because gradients still are used for time calculations a wider use for this weighting factor is well possible.

Meters in altitude If a hiker likes to avoid extra meters in altitude and wants to access his destination on the most direct way possible, he would certainly look for the way with a minimum of cumulated meters in altitude. Height differences between the nodes are offer such a search, too.

Surface The given surface parameter for every edge finally lets us implement an almost qualitative preference. The personal weighting of natural or hard underground gives the user a component in which he is capable to preference the kind of walk. As natural ground cover assures a more natural landscape than hard ground (not explicit), this factor can drive a more qualitative search.

It is very important to remember that Dijkstra's algorithm cannot deal with negative cost factors (Dijkstra 1959). In cases where negative weight values are possible, one has to adapt the weight to transform it to only positive values.

It has to be mentioned that it makes sense to accumulate this factors and set more than just one single weighting factor. The possibility to introduce a constraint to the search as e.g. 'looking for the shortest way possible, preferably on natural ground' and giving the force to the components of the constraint, opens a wide field of further secondary weighting factors. For this reason, a good concept has to be invented to realize this. Handling the described factors and secondary factors needs more thinking on implementing a weighting factor to the searching algorithm and on melting cost factors.

5.4 Algorithmic processing

5.4.1 Dijkstra's algorithm applied on Geodata

Remembering the functioning of the shortest-path-algorithm and its runtime, we clearly recognize limits when applying it to a wide set of geodata. The higher the size of the graph the exponentially more possibilities are treated when looking recursively at every edge for the minimal weighted path. If we consider that our graph with the hiking route network comprises thousands of nodes and even more edges, the computing capacity needs to be taken into consideration. When assuming a multiuser service over the Internet, the possibility of several queries at the same time becomes real. This loads the server, too, and a fast and non-problematic use is becoming less possible. Therefore improving methods have to be developed. Because the number of users cannot be influenced, the improvement has to access by the size of our graph.
How do hikers normally walk, what are their specific attitudes in choosing a route for a hike? Answering these questions will probably introduce an idea of how to cut the graph down to a smaller size for each personal query.

If we think about the attitudes, the best way is to play a hypothetic example. Hiker X wants to find a route from A to B. If he wants to get to B (i) the shortest way possible, he studies the map visually, localizes A and B and then focuses on the area in between for the

route that connects the two points directly. It is possible that a constraining obstacle lies in between A and B and so he would have to plan an extra way to get around the obstacle. And this is the situation in fig. 6 (2), demonstrating the area lying in between A and B that has to be widened to open the space for non directional devious paths. The worst case would be an obstacle lying directly on the line from A to B and near one of the localities.

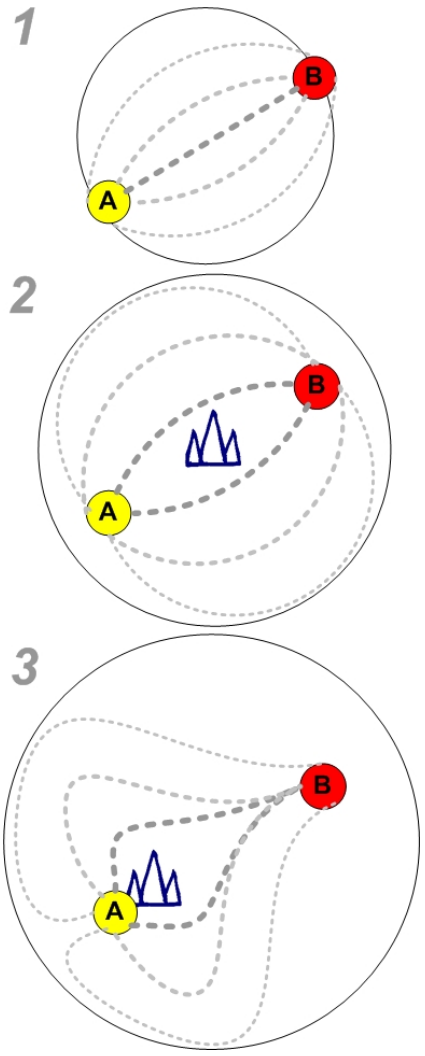

Figure 6: *Three possible situations when looking for a shortest hiking path. Situation 1 without obstacle and a straight path possible, 2 with a handicap in the middle of the way and 3 with the obstacle near the origin.*

Look at fig. 6 (3) to get an idea of the situation. Here a vast circuitous way in the direct line from A to B has to be taken into consideration. The area not only in between A and B must be widened, it must also be widened much in the orthogonal direction and even in the direction away from the connection into the hinterland. In this case, the surrounding searching area is opening much, as visualized in fig. 6 with a much expanded circle. An individual search for the shortest way becomes more difficult because overview is decreasing. Having discussed every possibility for the case when looking for the shortest path, we recognize that the focus always lies in the area between the origin and the destination. The search is widening recursively from the origin to the middle of the imaginary conduit to the destination. From this perpendicular bisectors of the sides the search is tapering towards the destination. The widening is dependent of the density of the graph and to possible geographical obstacles. The less dense the hiking network is and the more an obstacle is close to the origin or destination, the wider the focussed area has to be regarded.

We have now looked at the case where the goal is to find the shortest path to B. When we assume the users to introduce their personal preferences, things have to be thought of even more carefully. With the constraint for the shortest way, which is a pure geographical parameter, the consequences are satisfying. Now, looking at each possible cost factor, consequences has to be accommodated. When looking for (ii) the fastest path, it's not necessarily the shortest or straightest route from A to B. If there are steep slopes our hiker reaches point B faster if he accepts small diversions and walks on a regular moderate slope from A to B. The fact that a diversion means a longer walk, obviously results in more time consumed. Cost factor waytime and distance both depend on length and must therefore be treated similarly. Anyway the area in focus (in fig. 6 adumbrated by a surrounding circle) must be opened more but not too much compared with the case for shortest paths.
We assume that our hiker X is looking for a trail with (iii) minimal meters in altitude to get from A to B. He wants to avert extra climbing. In this situation every additional topographic barrier in the direct geographical line to B must be bypassed. Even bypass-

ing a whole massif or a valley is possible. A widening of the focus area to much more than in cases (i) and (ii) is necessary. If A and B are near each other, the possible diversion can include more perpendicular movements along the connection line and so the surrounding perimeter must have a minimal diameter.
The same as for (iii) can be said when regarding the factor of (iv) smallest slope gradients from A to B. Long extra ways can't be avoided and even trails leading away from start- and endpoint are taken into account.
If hiker X wants to walk preferably on a (v) specific surface, he has to allow for long diversions, too. Looking at the surface attribute for every edge and knowing that there exist many nodes accessible by just one kind of surface, the search must be highly extended. The use of this parameter as a constraint factor is probably not convenient and has to be rethought. The parameter as secondary cost factor influencing another factor would be better.

Knowing every factor's consequences to the focus area of the algorithm, one can conclude: (i) a spatial area is delimiting the focus area. (ii) The best approximation is a circular perimeter with its center in the middle of the connection line between origin and destination. A circle clasping the widening to and the closing of possibilities from the perpendicular bisectors of the sides is the best. (iii) The perimeter's diameter depends on the distance between start- and endpoint and considerably on the cost factor and has to be adapted to each case. There is a minimal diameter for close start- and endpoint.

If we want to reduce the graph's size for a calculation, a heuristic can be developed based on the conclusions (i) to (iii). When building the graph for a path calculation we have to compare the chosen cost factor, define a diameter for the focus area, build up the graph with nodes lying in this diameter and thus limit calculation and recursive steps significantly. In addition we help the algorithm to find its path faster because we also define a clear guidance to the destination. With such a heuristic we improve the search sustainably and reduce computation time to an acceptable waiting time. Multiuser action will be better supported by this application. Another advantage is the fact that once the heuristic applied, the system is getting more independent of the size of the complete network. Regardless how big the size of the whole graph is in its total expansion, for the calculations of routes just a small subgraph is considered.
With the heuristic we improve Dijkstra's algorithm towards the A* algorithm. The A* algorithm already works with sophisticated heuristics guiding the algorithm in its searches (Gross et al. 1999). With a small adaption a much better result can be achieved, almost similar to the result of the much more expensive techniques used within the A* algorithm.

5.4.2 Implementation in PHP and MySQL

Analyzing the "trailDijkstra" class as implemented in app. C.3.1 no methods must be adapted for the heuristic because they only work with the graph's data information. The heuristic just reduces the graph's nodes and does not change data structures. Therefore only the part constructing the graph out of our topology table has to be adapted to the heuristic approach.
The "trailDijkstra" class works with a variable characterizing the limit for infinite distances. Such distances cause the algorithm to exit the computations, which means no result has been found. This value limits the search to a quasi-infinite cost where the algorithm should break and outputs that there is no solution. This is the value where we can make the heuristic play as follows; (i) looking for all nodes lying in a surrounded area to the geographical midpoint of origin and destination, (ii) addressing all edges connecting a node outside the area of the infinite weighted distance. With this approach Dijkstra's algorithm works recursively until it touches a border where the surrounded area with infinite distances lies. Then the search is guided in the lines of the limited area and all the outlying nodes are not treated anymore.

The size of the calculated graph decreases dramatically, especially for long paths.
An important step is, of course, to elaborate the constants for the infinite weighted distances for each cost factor as well as the optimal diameter limiting the treated area for the search in relation to the used cost factor. An externalized file consists of all these constants and defines the routines distributing the correct values for the actual case. The values have to be tested carefully and must be adapted by experience to a well reasoned constant.

The extract in app. C.3.4 shows the implementation in PHP and MySQL. This is the first time when the topology becomes linked with the algorithm. Function `fctQuery` as quoted in C.3.2 line 26f returns the query made for implementing the heuristic. Because aggregate functions are rapidly computable on the server side through MySQL (Strassenprogrammierer 2007), faster than through PHP, the distance for every node is directly computed when asking for the table with the topology. Further we see the initiation of the diameter and the infinite distance on behalf of the constant file (app. C.3.2) defining all the constants and the routines to extract them.

5.4.3 Modifications on the algorithm to calculate cost factors

The implemented algorithm builds up the graph with edges stored each in arrays in the form `array(from, to, cost)`. Therefore it is not necessary to change the main "trailDijkstra" class to adapt the algorithm to different cost factors. The selection in the database has to be adapted to the preference the user inputs. The easiest way to do so is to switch the query made on the topology table for each possible weighting factor. With this procedure the query table always conforms to further treatment.
The realization is achieved with two help functions stored in the constant file (C.3.2), too. The one returns the query string adapted to the used cost factor and the other computes the absolute weight value that is processed in the algorithm. Careful treatment has to be applied to compound factors and to the more qualitative factor "surface". Where compound factors have to be melted together deliberately, the factor "surface" depends on which kind of surface is preferred: natural or artificial. The following list gives an overview of the different implemented cost factors:

primary factors

- minimal waytime for a path
- minimal meters in altitude
- minimal slope gradient
- minimal distance

secondary factors

- preferred natural surface combined with waytime: the time factor is once more weighted by the factor surface, whereas the unwanted totally artificial surface doubles the time for an edge.
- preferred artificial surface combined with waytime: the time factor is once more weighted by the factor surface, whereas the unwanted totally natural surface doubles the time for an edge.
- minimal waytime in combination with minimal gradient: this factor gives a result meaning the fastest way with regard to small slope gradients and a low possibility of steep gradients.
- preferred natural surface combined with distance: the distance factor is once more weighted by the factor surface, whereas the unwanted totally artificial surface doubles the distance for an edge.
- preferred artificial surface combined with distance: the distance factor is once more weighted by the factor surface, whereas the unwanted totally natural surface doubles the distance for an edge.

Of course other combinations for secondary weighting factors are possible, too. But the ones implemented are the most useful for hikers whereas other combinations do not make much sense.

With this step the system is now so far developed that it is able to calculate a path from an origin to a destination with regard to the users preferences in cost factors. The system uses a sophisticated heuristic that allows a broader user service and limits computation time in an essential way. The result is prepared for further disposition and is rehashed into the following array form, containing every important information about the resulting path specially coded:

```
[costFactor,
origin ID, 1. node, 2. node, ...,
destination ID via1, xxx,
origin ID via1, 1.node, 2. node, ...,
destination ID via2, xxx,
origin ID via2, 1. node, 2. node, ...,
destination ID]
```

Each way section is encoded in the same structure: the starting node ID is followed by the chronologically traversed node ID's until the destination node is reached. If there are multiple sections due to via's, the sections are associated with the "xxx" marker in between. The first element in the result array contains the ID for the underlying cost factor.

5.5 Server and webinfrastructure

A common client-server-architecture works in the background of the future system. As the system should be accessible from the Internet, an Internet address is reserved for the project. It has been installed for a long time, for test purposes and first usage from exterior clients. The URL named `http://www.virtours.ch` links the project homepage. To get directly to the site with the hiking planning tool, an URL-redirect is installed, called `http://www.virtours.ch/hiking`.

The following minimal basic configurations and materials are necessary to run the hiking system in a web interface: (i) database connection to a relational database system (MySQL recommended because of compiler programming), (ii) a powerful web server infrastructure allowing heavy calculation queries, (iii) the execution of PHP (version 5 recommended).

5.5.1 Serverhosting

As server, a hosting at `webland.ch` is established. Microsoft Server 2003 runs on the server and all common script languages are interpretable: PHP5 and JavaScript are relevant for our system and are fully supported. Webland operates a very safe and powerful hosting center in Switzerland, which is capable to come through heavy data load and congestive client calls (Webland 2007).

5.5.2 Database

The database, also running at `webland.ch`, is based on MySQL. MySQL databases are licensed under OpenSource software and are freely available over the net. The MySQL technique defines the most popular way to drive databases in the World Wide Web (MySQL 2007). As MySQL perfectly matches PHP, the two basic used programming languages for the project; the combination suits well. PHP supports a highly sophisticated language package to treat and query MySQL databases.

Because of security reasons the full access data to the database won't be given here. But if a look into the database would interest you, don't hesitate and ask the author for the compatible access data.

6 From Raw Data to Geoinformation

This section describes the process and the needed steps to be followed to transform our raw data material to substantial geoinformation.

6.1 Data preparation

If we want to model distances as described in the theoretical chapter 5, we have to build up an appropriate data structure matching the needs of distance modeling and calculating distances. The data structure has to rebuild a graph with nodes and vertices digitally. In geography and especially in GIS one falls back on geographical databases (GDB) that model data structures to the need of a geographer. Spatial elements, in our case topological elements, are well describable within this structures (Bartelme 1995). Only within a normalized database set the further calculations become possible at all. So this is the reason why we have to transform initial shapefiles (not legible for databases) into data load that is readable for databases. The database permits a fast and treatable heavy query on our data in acceptable time ranges. If raw shapefiles are accepted many work and heavy compilations will be obsolete.

6.1.1 Geographical database GDB

In our case the Geographical Database (GDB) is constructed the way the database scheme in Figure 7 is represented. The showed entity relationship model focuses on the used tables (entities) and the relationships between the tables. There are the two tables BASENODES and BASEEDGES wherein the initial data is stored. Out of the table BASEEDGES the whole graph is reproduced. The nodes are stored in the table N_NODES and the graphs edges are introduced into the table TOPOLOGY. The original node data from BASENODES is synchronized with N_NODES. This leads to a graph representation where the table TOPOLOGY represents all the topological relationships of our graph. In its records all necessary information for an edge are stored: linkage to starting- and endpoint in N_NODES, linkage to the edge representation in BASEEDGES and all kinds of attributes for the selected edge as different cost factors for example. These linkages are visuialized in Figure 7 for a better overview.

Tables 1, 2, 3 and 4 indicate the inner structure of every database-table. When the system is in use and heavy queries are running, data types for faster computation have to be chosen thoughtfully. Names of attributes are all in minuscules and should remind us of their purpose. The primary keys are printed in bold.

6.1.2 Dataintegration into the database

For reasons of received data amount compressed in shapefiles (`.shp`) that are not directly transformable into a database, data transformation and data introduction into the database needs to be discussed carefully and in detail. This step of converting initial data from simple readable shapefiles for ArcGIS into data load (ASCII) readable for a used MySQL database is not simple to go through (ESRI Support 2007).
In the following paragraphs the succession from raw data into the built digital graph representation is declared.

Converting shape- to textfiles: First of all one needs to convert the raw coordinate sequences out of the shapefiles. ESRI does not well support the export for `.shp` files storing edges into text based coordinate files. For nodefiles ESRI supports the export into textfiles as well as it exports corresponding attributes of nodes, too (ESRI Support 2007). Converting the initial nodefile does not need extra treatment, but it is difficult to convert shapefiles storing edges into sequences of coordinates. Therefore we rely on help from a freeware software called “Shape2Text” by Zonum Solutions (Zonum 2007).

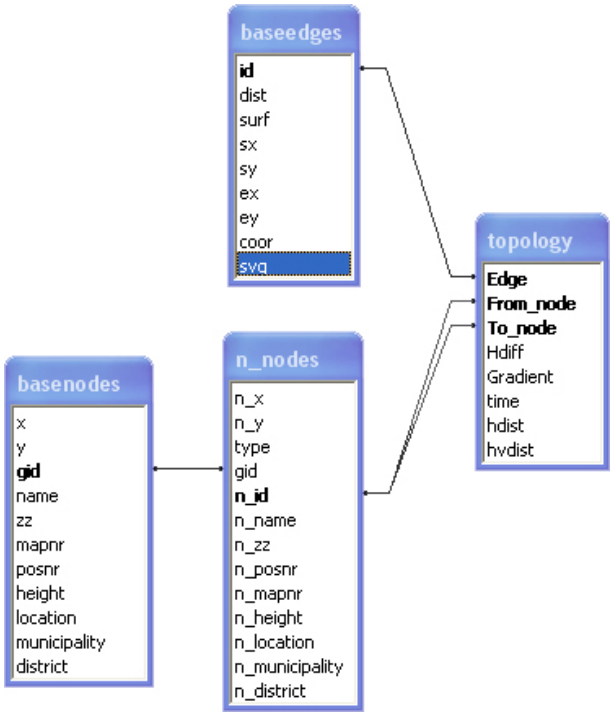

Figure 7: *Database layout with foreign key structure and relations between the tables. The tables are explained more detailed in tables 1, 2, 3 and 4.*

BASENODES

NAME	TYPE	DESCRIPTION
x	double	x coordiante
y	double	y coordinate
GID	**int**	**geographical ID**
name	float	node name
zz	int	internal ID
mapnr	int	internal ID of corresponding map
posnr	int	internal position number of signpost
height	int	altitude of node above sealevel
location	text	name of location in municipality
municipality	text	political municipality
district	text	political district

Table 1: *structure of table Basenodes. Initial table with raw data of nodes, which are equipped by a signpost.*

BASEEDGES

Name	Type	Description
id	**int**	**edge ID**
dist	double	horizontal distance of edge in meters
surf	float	surface type: 0 = neutral, 1 = nature, 2 = hard
sx	double	x coordinate of starting point
sy	double	y coordinate of starting point
ex	double	x coordinate of endpoint
ey	double	y coordinate of endpoint
coor	longtext	string of coordinates building the edge, separated by comma
svg	longtext	string of SVG presentation of edge as path

Table 2: *structure of table Baseedges. It is built by compilation of the initial datafile baseedges.shp.*

N_NODES

Name	Type	Description
n_x	double	x coordinate of node
n_y	double	y coordinate of node
type	int	1 = pseudonode, 2 = node signpostep, 3 = calculated node out of basegeometry
gid	int	geograohic ID of node from signpostep
n_id	**int**	**internal ID of graph node**
n_name	float	name of node if type = 2
n_zz	int	ZZ of node if type = 2
n_posnr	int	POSNR of node if type = 2
n_mapnr	int	MAPNR of node if type = 2
n_height	int	HEIGHT of node if type = 2, else approximated calculated height above sealevel in [m]
n_location	text	political location of node if type = 2
n_municipality	text	political municipality of node if type = 2
n_district	text	political district of node if type = 2

Table 3: *structure of table N_nodes. It is built by compilation of tables 1 (basenodes) and 2 (baseedges).*

TOPOLOGY

Name	Type	Description
edge	**int**	**edge ID from baseedges**
from_node	**int**	**ID from n_nodes, startnode**
to_node	**int**	**ID from n_nodes, endnode**
hdiff	double	difference in height in [m]
gradient	double	gradient in [%]
time	double	time in [min]
hdist	double	horizontal distance in [m]
hvdist	double	diagonal distance in [m]

Table 4: *structure of table Topology. It is built by compilation of tables 2 (baseedges) and 3 (n_nodes). It finally describes the graph and also stores the specific data to every edge in the graph as walking time or diagonal distance.*

We just have to specify the input shapefile and a ASCII conform textfile that records all coordinate chains of every edge is output. The output looks like the following cleavage. The clip shows a first converted polyline with its coordinate sequences. Every edge is introduced by the number of nodes the polyline contains.

```
Entity type:Polyline

XCoo YCoo
5  Nodes
741877.6043366370 265125.3558531549
741898.3124973658 265037.0625187650
741934.6875133737 264881.7812356526
741934.6875133737 264435.8124913849
741928.1898684654 264424.7897903221
```

It needs to be added that this output format for polylines is well known in GIS. Data storage highly increases after conversion because we quit the border of elaborated compression techniques used by ESRI in its proper file format `.shp`. The output textfiles are huge because of the high amount of coordinate chains listed. For further treatment it is totally coercive to process automatically.

6.1.3 Step-by-step topology buildup into GDB

It has been agreed to build up the GDB in a stepwise approach. This procedure forces us to really think about what is being done in every step every time the topology has to be updated or renewed. If these steps are not executed carefully enough, the future system won't work correctly and will display wrong results.
Of course, a simple script could automatize the steps, and thus could totally build up the topology without any necessary user input.

a) Compilation of base nodes: In a first step, all data (coordinates, heights, locations) of every basic node has to be introduced into the table BASENODES. To do this, the raw data stored in the source ESRI `.shp` file has to be transformed into a textfile either with ArcGIS's export function (ESRI Support 2007) or with the described software by Zonum Solutions (Zonum 2007). The compiler copied in appendix C.1.1 reads the data and inserts it in the GDB into the corresponding table. No attribute is transformed during this compilation.

b) Compilation of base edges: To get the edges' data into the GDB, we have to compile the raw coordinate textfile. Allowing for further treatment with SVG the compilation is built to directly convert the existent polylines into SVG compatible line representation. In the same step, every edge's length is computed. There is a good approximation of the real length of an edge due to very small edge segments (in range of several meters) and computation of euclidian distance in between them. It has to be mentioned that no vertical distances (difference in height between start and endpoint) are taken into consideration because as long as our topology isn't finished, there exist many edges without any declaration of height. With future use of the graph in mind we also store the starting and end point coordinates of each edge in the GDB. Lines 32-55 in appendix C.1.2 shows above automatic compilations and transformations.

c) Computation of surfaces: The received data about trail surfaces allows us to create this edge attribute, too. It's important to think about how to represent this data: There are edges, which are totally capped by only one specific surface. But there are also trails, with surface changes along the way. Because we can not divide these edges into several parts (it is not possible, because there is no information about edge sequences and frontier points where surface is changing) it is necessary to describe the surface by just one exclusive value. Further, there are edges for which we do not have any information about their surface. To adhere these restrictions we manage with following concept: Edges with no data about surface get the value 0. All other edges get a surface value that corresponds to the arithmetic average of their total surfaces. Because there are only to possible surface values ('1' correlates to natural covering and '2' to hard covering) all values range from 1 to 2. It is important to remember that a possible

surface value of '1.75' for an edge does not imply an edge with basically hard cover. It does just imply that there are more edge sequences with hard cover than with natural covering. It is not allowed to make a relation between distance and surface value. Only integers are the proof for fully single covered edges.
This compilation step has to transform raw surface data available in the form [Edge ID, Cover] as shown below

```
1,2
1,1
2,1
3,1
4,1
4,2
4,1
```

into one single surface value and to store this value into the corresponding edge tuple in the GDB. Concerning the values above we would receive seminatural coverage for edge 1, natural surface for edge 2 and 3 and a mixed surface represented with the value 1.33 for edge 4. Appendix C.1.3 represents the compiler's source code implementing this step of topology buildup.

d) compilation of graph's nodes:

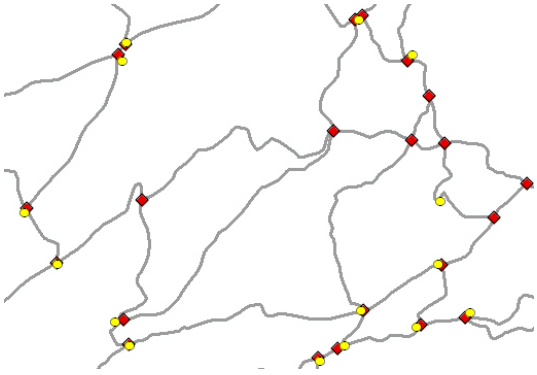

Figure 8: *A snapshot near Sargans. Red rhomboids represent nodes that are end- or starting points of edges in our grid. Yellow circles show locations of signposts. Not every end- or starting node is locked to a signpost. A careful look at the snapshot shows another sensitive singularity: Signposts are localized in their exact position. This position is obviously not in the middle of a crossing as the nodes are. They are situated aside the trails. Or even some meters away in the field.*

Before the basegeometry can be built up, all nodes where edges meet have to be introduced into the GDB. Before the nodes with signpost in BASENODES are heeded, we first have to bother with all the nodes stretched by the edge's grid. There are many nodes which are not equipped with a signpost. Figure 8 shows a capture in the region of Sarganserland. It's clearly identifiable that not every node is locked to a node with signpost. To fill in each node which is start or end of an edge, we consider the start- and endpoints entries in table BASEEDGES. To avoid a double entry, which would completely destroy our further topology, we firstly have to test whether a node still exist in the table or not. If we imagine reality, there is normally more than one edge starting at a crossroads. All these points have to be arranged at one and the same node in our topology to guarantee regular connections between edges.
With this ideas, the approach for the needed compilation is given (see appendix C.1.4): Take all the edges' starting points and check if they are already introduced in the nodes table N_NODES. If not, store the node into the table. Continue the same way with all the endpoints. With great foresight we have also set a marker standing for the class of node. In this case, the nodes are originally from the baseedge's geometry, marked with number '3'. Table 3 shows the table's framework and all the descriptions for the attributes.

e) Synchronization of graph's nodes: Remembering that all nodes equipped with signpost are locked to a start- or endpoint of an edge in our topology, we have to synchronize now the nodes equipped with signpost with all their implied attributive absolute data to the before introduced nodes in N_NODES. The obvious problem that signposts are not directly localized in the center of crossroads on the tracks but are always situated a little to the left or to the right, causes this necessary extra step. In Figure 8 this situation is directly visualized. If we want to connect the attributive data that is available just for nodes with signposts to the topology, this is an important transformation to be made. Conceptually we have to implement a compiler search-

ing for every node with signpost the geographically nearest node in the topology, and when it is found, attach all the attributive data to this node. There are cases where three nodes are placed directly aside each other forming a triangular crossing. In such a case the compiler should attach the data to each of these nodes, because they each represent the same location of the corresponding signpost. To do this, a range checks every comparison of distances between node with signpost and node already in the topology. If this distance lies in the range, the data is attached to each node matching the range limit. Section C.1.5 deals with the source code and in line 15 the range is stored. In our case we agreed to fix the range to 50 meters. Within this distance two localizations are still in sight and near. A signpost matching this condition in reality is valid for each of the two nodes.
Here it has to be mentioned that this is a fairly sensitive value that must be established carefully. This value depends on the topic the topology is representing. Every modeling needs its appropriate value representing the best of the modeled reality. Many tests, beginning with smaller ranges have not shown a satisfying result: Many crossings with more than one node aren't fully connected to one single signpost. A good approximation representing reality best has been found with 50 meters.
There exists GIS-Software as Idrisi or ArcGIS that support quite similar functionalities. The user defines a "snapping area" in which the edges are automatically connected to nodes. Unfortunately for our case the sophisticated self programmed solution is necessary to have the best results.

f) Computing heights: There are nodes, not connected with attribute data. If there are crossings of two hike trails without a signpost there isn't any attributive data for this node. So the node exists just because of the crossing and our needed topological relations and not because it is a real origin for new hike trails. Therefore it is necessary to approximate attributive data for such nodes that is necessary for computations. To calculate waytime it is necessary to know the height of each node. We have constructed a compiler looking up each node without attributive data and adding approximated height (look up the whole compilation code in appendix C.1.6).
As calculation technique we follow global triangulation techniques used in GIS to compute accessibility: Voronoi-Polygons or Thiessen-Polygons (Gitta 2007). Figure 9 visualizes a described situation. For the best possible approximation, we have to consider the distances between the nodes. The distance weights the corresponding height inverse proportional. This means qualitatively that the further away a node is, the less its height influences the calculated height. There is an inverse relation between distance and influence. The more surrounding nodes are be considered the better approximation results. In our case, we take the three nearest nodes with signpost and absolute height into calculations, inspired by the polygon methods used in GIS. This approach guarantees a good relation between computation time and accuracy.

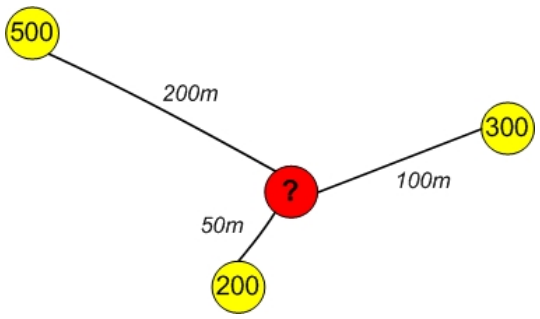

Figure 9: *A generic situation showing a non-starting-point of a hike trail without signpost surrounded by the three nearest starting-points with correct heights as attributive data. Distances to the surrounding nodes are mentioned in meters and heights are written into the corresponding nodes. Equations 8 - 9 show the calculation process to compute an approximative height for the node in question.*

For a better understanding an example referring to Figure 9 is printed here: Goal is the approximation of the node's height painted in red. It's three nearest neighbors are visualized in yellow with distances and absolute heights. To get each influence to the resulting height

(eq. 8) the inverse distance has to be divided by the summed up total inverse distance (eq. 7):

$$(1/200) + (1/100) + (1/50) = 0.035 \quad (7)$$

$$\frac{1/200}{0.035} = 0.1429 = 14.3\%$$

$$\frac{1/100}{0.035} = 0.2857 = 28.6\%$$

$$\frac{1/50}{0.035} = 0.5714 = 57.1\% \quad (8)$$

Finally it is an easy step to compute the height (eq. 9). The absolute heights have to be weighted with its corresponding influence values and summed up for the result:

$$\begin{aligned} 500 * 14.3\% &= 71.45 \\ 300 * 28.6\% &= 85.71 \\ 200 * 57.1\% &= 114.28 \\ \sum &= 271.44 \end{aligned} \quad (9)$$

In this example the node would get the calculated height of 271.44m. These calculations are automatically executed for every single node that is not starting-point within the programmed compiler in C.1.6 (look up lines 29-36).

g) Compilation of topology: Finally the topology has to be established with our edge data and node data. The two components have to be connected - each edge needs to be connected to its corresponding start- and endnode. Therefore we need another compiler (see appendix C.1.7 fills the table TOPOLOGY which connects the N_NODES and BASEEDGES tables. It takes every edge, looks for the matching nodes for its two endpoints and builds up the topological information. The important information that each edge could be paced off in one and the opposite direction leads to a double entry for each edge - one with node a as start- and node b as endpoint and in the opposite direction with node b as start- and node a as endnode. With the direction also the height difference will inverse.
Further the compiler calculates the difference in height for each edge, because this information is needed to calculate the friction, the transit time for each edge, in the next step.

h) Calculation of transit time: To completely build up the topology, the friction of slope gradient in relation to time has to be adapted to the edges. The official polynomial from the Schweizer Wanderwege as described in 5.3.1 gets in use. The last compilation step includes the calculation of the transit time in its specific direction for each edge. To avoid absurd gradients of plus or minus 40%, gradients are smoothed to max +40% and min -40%. This is the case when an edge is connecting two very close nodes, which are approximated in their height (look at situations indicated in fig. 8). This filter guarantees correct time values , even for very short edges with approximated heights. Annex C.1.8 is the source code for the last step of compilation and building up the GDB and topology.

6.1.4 Final topological key data

As the topology is now finished, a look at its components may give a good idea of the volume and the data load on which our algorithmic calculations are based. Therefore the following list characterizes our hiking topology:

- The topology is building a multi-dimensional weighted digraph
- Its size is 5'142
- It consists of 1'846 nodes, which is also its order
- The minimal node's degree is 2
- For each node its minimal outdegree is 1 and its minimal indegree is 1, too. There aren't any separated nodes
- 0.36 equals to the topology's density.

6.2 Quality of approximated heights

Because the node's height is the most sensitive data that get in the various calculations for our track, it is necessary to verify if the used approximation technique is suitable. Therefore the heights of all approximated nodes (nodes

Height comparison: all approx. nodes

NODE ID	APPROX	DHM25	DIFF	ABSOLUTE DEVIANCE [%]
12	397	397	0	0
15	396	396	0	0
24	397	397	0	0
96	398	398	0	0
101	592	592	0	0
180	399	399	0	0
198	601	601	0	0
225	618	618	0	0
301	410	410	0	0
320	401	401	0	0
...				
1274	1439	1950	511	36
1822	1759	1115	-644	37
1486	1416	1965	549	39
1452	502	774	272	54
974	1676	545	-1131	67
1828	470	910	440	94
1818	536	1093	557	104
1451	522	1097	575	110
1824	522	1097	575	110
1816	515	1100	585	114
Average error: 4.52%				

Table 5: *Testing the quality of approximated heights. The best and the worst approximations are shown. In this comparison the artificial nodes due to lack of data are included.*

of type 3 cf. appendix 6.2) have been compared with the height given by the Digital Height Model DHM25 (Swisstopo 1996). The fine resolution of 25 meters allows a good comparison of calculated heights and heights out of the model.

The results, shown more detailed in tables 5 and 6, are satisfying. The average failure lies below 2.1% which is totally acceptable for our purposes. There are many nodes lying at the artificial border, where our network ends due to lack of data. On these locations divergences are much higher. Concerning the cantonal border between St.Gall and Appenzell Ausserrhoden in the St.Galler Rheintal, we have to consider that the huge deviances exist because the border lies on the Alpstein ridge and our network abruptly ends there. Taking these "artifical nodes" in consideration too, the error increases to slightly above 4,5%.

Because our tool does not support calculations that start or end on secondary nodes, it is allowed to assume the error is less than 2.1%, the "artificial nodes" not considered. As conclusion the approximation step for heights of secondary nodes in our topography is suitable.

6.3 Testing the implemented algorithm

As with this chapter the system's background is finished, ample tests have to be applied. The system's calculations must resist heavy number of queries and different collocations of graphs. It is also necessary to test the heuristic for improving calculation time and adequate used constants. The following step of mapping the results shouldn't be studies before this important milestone is approved.

The test phase consists of two different test situations: The one is visualized in Figure 10. The example digraph with size 24 and order 8

Height comparison: without "artificial nodes"

NODE ID	APPROX	DHM25	DIFF	ABSOLUTE DEVIANCE [%]
12	397	397	0	0
15	396	396	0	0
24	397	397	0	0
96	398	398	0	0
101	592	592	0	0
180	399	399	0	0
198	601	601	0	0
225	618	618	0	0
301	410	410	0	0
320	401	401	0	0
...				
272	568	612	44	8
1267	733	674	-59	8
1749	832	899	67	8
619	608	559	-49	8
944	1336	1444	108	8
387	568	522	-46	8
631	628	577	-51	8
17	517	559	42	8
1762	1033	949	-84	8
1710	549	594	45	8
Average error: 2.07%				

Table 6: *Testing the quality of approximated heights. The best and the worst approximations are shown. In this comparison the artificial nodes on cantonal borders are not included.*

is weighted with random values. Adumbrated in green and black color is the covering material "hard" or "natural" underground. To apply the heuristic, too, the numbers at each edge correspond to the walking time and to the distance equally. This simulation helps to control the correctness of the computed results. Searches can easily be made manually within the painting which let us clearly verify the result.
The other situation implies a direct algorithm test on the real topology. Here real situations can be simulated and challenging tests are possible. Due to the graph's size, meaningful test results should appear. Along this situation the heuristic's constants will be checked.

Test configuration To verify a robust result, the test configuration is implemented specially. To reduce side effects when the server is not busy or heavily charged, a common trick is applied; the time measuring from the beginning to the end of calculation is repeated several times. A loop works through the algorithm for a defined time and at the end the average time is computed. Time is measured in milliseconds on behalf of the PHP function `microtime()` (PHP 2007).

To verify the constant values used for the heuristic, experience needs to be introduced. It is necessary to check the digital result on cases that can be controlled manually. For this scenario an analog hiking map is brought in (Kantonal st.gallische Wanderwege 2002). The tests checking the constants are applied on paths that are well controllable manually. Just like this, qualitative declarations and reasoned adaption is possible.

A situation with near start- and endpoint and another where the nodes are set wide apart are applied always 3 times to the real

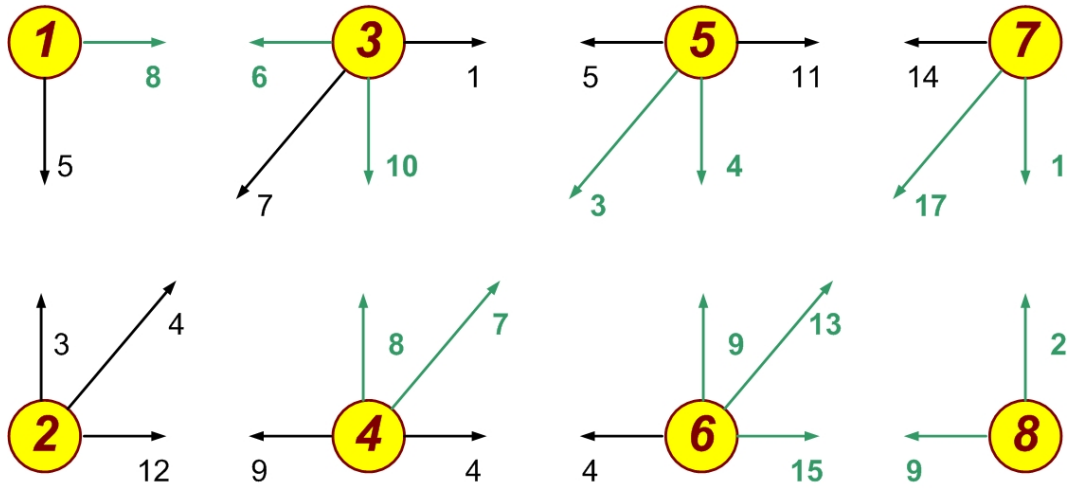

Figure 10: *An example graph on which the implementation has been tested. The walking time for every specific edge is given by the number corresponding to the directed arrow. There are always the two values for time, one for each direction. Further the edges are colored in black meaning hard underground and in green meaning natural ground surface. Therefore weighting factors in this example are waytime and surface.*

topology. The search for the long path is chosen for a distance of about 20 km by air. The loop is running through the algorithm for each case 20 times. After this procedure, the heuristic will be set off and all tests are repeated without heuristic.

Results and conclusions Interpreting the different results for the tests on the example graph, shows a correct running of the algorithm and the implementation in every domain. Checking the use of different cost factors and comparing the result with the manually computed data always matches. The example graph is of such a small size that runtime is not really perceptible. Important within this step is the fact that all outputs are correct and it seems the implementation runs at least in qualitative terms perfectly well.

On the quantitative side the runtime tests display pleasant consequences. All results are fully listed in the calculation table's appendix (cf. A.1). We detect different statements: (i) For very short paths the heuristic's application retards runtime by about 10%. This is comprehensible if we remember what the heuristic is doing. Before the graph is built up, all nodes in the topology are compared to the mid point and their distance is calculated. After this the search starts. For such small distances this extra step needs more computer power than a directly introduced search without limitation. When we look at the results, the needed 10% are almost negligible and longer runtime is limited to a split second. (ii) For long paths the conclusion looks different. It is clearly obvious that the geographical constraint on the graph reduces runtime dramatically by about 35%. This result was not expected in such a clear message. For longer searches full recursive search needs much more time than computation of geographical distances for every node to define the limiting area. The algorithm's guidance reduces wrong sided searches and the implemented "intelligence" is purpose for faster goal attainment.
Within normal use of the system, cases will be more frequent where longer paths are requested, as mentioned in (ii). Hikers do not consult the system for distances lying almost within eyesight. The queries to the system will mostly consist of demands for longer paths. This is where the heuristic improves the implementation significantly and disburdens the server.

Through diverse qualitative comparisons and reflection with the help of the analog map (cf. Kantonal St.Gallische Wanderwege 2002)

the constants are developed as they are fixed in C.3.2 lines 5 to 23. The following careful considerations lead to the final constant definition:

- Generally the constants are chosen not too tightly knowing that computation is quite fast and correct results are weighted more than some milliseconds of a faster output.

- Regarding the network density of hiking routes on the analog map, we see quite a dense net of routes in any possible area. Therefore constants do not need to widen the focus area too much in case of cost factors minimal distance and minimal time. Started with a widening for factor distance's radius of 2'000 meters, this constant is to be adapted after some tests have really shown different result. Increasing the radius by 5'000 meters always delivers correct results to cases in test. A comparison on the analog map with this increasing amplifies the adaption. With a focus area widened by 10'000 meters aggregated (each side an increasing of 5'000m radius), the shortest and fastest ways should always be matched by the algorithm. The topological density undermines this impression.

- For cost factors that are not forcefully linked to the geographic orientation from start- to endnode, as do distance and waytime, much higher constant values are needed. To circumvent a mountain because one is looking for a track with minimal meters in altitude, larger diversions have to be treated. There is a hill in the middle of our sample route on the analog map. A hike from Berneck SG to Rorschach covers a linear distance of about 5 kilometers. But in between, there is a hill called the Rorschacherberg. To really bypass the hill and find the way with the smallest difference in altitude, a focus area widened by only 10'000 meters would be insufficient. Therefore, constants for attributes "minimal meters in altitude" and "minimal gradient" are fixed with a radius increase of 15'000 meters. In consequence calculation time will increase minimally. Thinking of hikers' behavior, the focussed factors will not be as popular as time or distance.
  Within these limits all tests passed off correctly and the verification manually confirmed the results, too.

- The qualitative factor "surface" is not controllable on the analog map. Therefore other reflections have to be made. As this factor does not depend on the local geography, the one and only possibility is to set the constant so that it is a good interim arrangement between correct result and runtime. As the factor "surface" is always coupled with another factor, we agreed to set the constant to 20'000 meters, meaning, the focus area spanned by the diameter of start- and endnode is widened for a search query including the attribute "surface" by 40'000 meters. Think about mightier topology in a future system, the defined area thus guarantees a proper running of the server as well as satisfying results.

- Secondary factors are always connected with the bigger constant value of the two attributes.

Part II
Application - A Visualized Trail Planing Tool With SVG

As a second part of this thesis, the now following pages deals with the implementation of the results calculated on behalf of the built topological concepts. The visualization process of individual researches on topology needs special techniques. The goal of processing every part of the system server-sided, without any resources used from the client so that the resulting interface can be accessed through internet barrier-free, is probably the biggest hurdle to deal with. The publishing of dynamically built images over the Internet is a domain not well developed.
SVG slowly fills in this sector but there are still little systems generating SVG code dynamically on the net. Problems are obvious: Since SVG is a W3C standardized XML language not all browsers support the full SVG concepts. There are several dialects supported by the different deliverers as Microsoft or Mozilla for example (Code Dread 2007). If we want to connect even more user functionality by using JavaScript and AJAX the chaotic status becomes even worse. These are probably reasons why SVG is still developing slowly in the online domain. SVG nowadays is often used for local applications that are running on the client. On local systems SVG support can be defined on the application itself and the SVG visualization does not depend on external sources.
Nevertheless the supposed approach deals with the more complicated way: a dynamically processed, from different browsers with different SVG implementation supported, freely available over the Internet and fully functional visualization tool is planned. The following chapters will guide you through the most important concepts to implement. Often there are small code snippets described to reach a certain goal or to implement a complex functionality. The whole codings are available in the appendix and enables the reader to explore the used implementations as far as he likes.

It is necessary to include the future system to a functioning web interface. Real tests and statements to usability and application are just accomplishable in a real and seeming interface that resembles the real Internet world. As the goal should be an encapsulated independent system that allows an integration into any possible web system regardless of the technique used and framework utilized, parameters containing language and graphic should be adjustable easily.

In our case the web framework consists of a content management system (CMS). The programmed project for this master thesis, called `virTours.ch - virtual Tours`, is based on the underlying OpenSource CMS Typo3. Typo3 is a well known, freely available, powerful content management tool (Typo3 2007). As Typo3 is one of the most sophisticated CMS on-hand on the market, it is a good specimen for testing the systems configurations: When in the complex boundary of Typo3 the hiking system works well, integration into other CMS and into individual homepages should not meet any problems.

The CMS configuration allows a multi-language activity. The multi-language configuration is foresight for future usage probably. Tourist interests ask for a barrier free conversation which implies a multi-language interface. It is important to think about this concept before starting programming. A multi-language interface needs the planning from the beginning of a project to avert huge additional effort.

7 Visualization with dynamically processed SVG

In this chapter, the whole process from raw topological data to graphical representation of the results is described. The core functions dynamically generate SVG graphics. To implement all additional features and interactivity to belie the expectations on interactive mapping over the Internet, a wide range of applications and programming techniques are needed. Figure 11 shows a flow chart listing the programming concepts and -languages used to realize all the foresight applications.

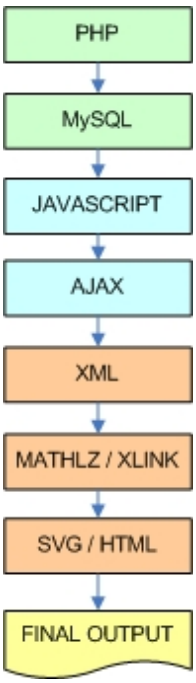

Figure 11: *A flow chart visualizing the programming techniques and -languages used. The deeper the chart, the more specialized the concepts.*

7.1 Generating an interactive zoom map in SVG

As SVG is an ordinary XML standard, access to other interfaces working with XML are well possible. This allows to think about interactivity of an SVG file with the user through JavaScript and other paradigms. To make full use of Internet based maps and to really get advantages in comparison with printed maps, interactivity has to be introduced into our map.

A printed map requires arduous folding, a computer screen does not. However, the whole map has to be visible, too, even on the screen. Therefore zoom and pan functionality has to be implemented for sure. The difficulties with a printed map when it has to be folded or when first looking for a specific location and every folded side has to be looked at can be bypassed on computer side with an overview map always guaranteeing the user's orientation. An overview map showing the whole map and coevally displaying the actual extract in the main map on the overview guarantees easy orientation when the user has zoomed in. Further the area marking the extract on the overview could be sensitive to mouse events so that the extract could be relocated to a new position just by drag-and-drop.

These ideas were fundamental goals for our map. The following subsections give the description of how we realized all this interactivity and functionality.

7.1.1 Different viewports

SVG supplies the concept of viewports. A viewport can be described as a proper window with a predefined intern broadening implicated. So for every viewport, a separate coordinate system is supported. We could define a viewport measuring 300×400 pixels on the screen showing a perimeter 300km times 400km. In this example a pixel would value 1km in the intern coordinate system. The concept is very useful for mapping features: the own coordinate system with individual origin can be defined in SVG. The coordinate system used can be adapted to the exact perimeter treated and difficult coordinate transformations from original coordinates to screen coordinates are dropped. It is the best way to deal directly with original coordinates when painting features into a map.

For our case, we need two viewports: one for the main map and the other one for the overview map, which is much smaller on the screen than the main map. For the main map we need to know the map's extension we would like to deal with before defining the viewport. The perimeter is given by the dimensions of

the hiking network. As our data is all in the official Swiss coordinate system, which is metric, the introducing tag for our main map

```
<svg id='myMap'
    viewBox='725000 -268000 45000 55466'
    x='172' y='5' width='382' height='471'>
```

defines a new viewport (attribute viewBox) expansions from (725'000,268'000) in its upper left corner 45'000 meters in width and 55'466 meters in height. The extension on the screen is given in pixel: with offsets x and y our map will measure 382 times 471 pixels exactly. To avoid contortion the relation between height and width is the same for the viewport expansion on the screen and the expansion of it's internal coordinate system. On closer inspection of the above the careful reader recognizes the minus sign in the second parameter for the viewBox attribute. Unfortunately viewports implement a coordinate system upside down, meaning the ordinate ascending downwards and not upwards as usual in printed maps. A small trick deals with the problematic: when defining the origin in the negative y-coordinate, no difficult transformations are needed. The only thing necessary to adapt is to invert all y-coordinate values in every feature showed in the active viewport.
Now the generation of the overview map is an easy step.

```
<svg id='myOverviewMap'
    viewBox='725000 -268000 45000 55466'
    x='25' y='238' width='120' height='148'>
```

We see that the difference lies just in the screen expansion of the map. Where the main map is much bigger, the overview map is quite small and placed in a different location (to avoid overlapping) on the framework (compare the x and y offset).

7.1.2 Zoom functionality

To realize a zooming function we now have a clue in our hands. We just need to change the viewBox parameters to change the focussing size of the window (the viewport). As the screen extension always remains the same, the scale between pixel and coordinate value modifies with changing viewBox values, which is equal to a zoom.
For the user's interacting we need two functions: one for zooming in and the other for zooming out. It has to be defined, which factor to use from one zoom level to the other and that the zooming remains centered, the zooming has to be done around the central point of the actual viewport. Both functions work similarly: (i) readout the actual viewbox values, (ii) transform the values for width and height with the predefined factor, (iii) calculate the new viewbox origin regarding the central point and (iv) assign the new values to the viewbox of the main map to apply the zoom. Further, on the overview map the focus area should be displayed to keep orientation. Therefore the function `mySetOverviewRect()` is executed in the main zoom function. It draws a semi-transparent rectangle over the overview map with the same extension as the actual viewport of the main map.
JavaScript provides the necessary methods to access the viewport values in the map document. So the zooming functions are realized with JavaScript. More precise the two functions are mustered to one single function with an easy if / else condition handling the zoom in or the zoom out.

The results from several tests directly on the map have showed that a zoom factor of 1.5 fits best for our purposes. It implements a good compromise so that the user does not have to apply too many zooms to get to a level where features are visualized big enough and that the user has enough different zoom levels focussing the extension wished for. The zoom function is executed when the user clicks on the corresponding symbol on the map interface. For zoom in a big "+" is allocatable and for zoom out a big "-" sign is available. Many interactive maps use these symbols to characterize zooming functionalities.

One important error in reasoning is still existent: what when a user wants to zoom out so many times that the viewport becomes too big and overlaps the full main map extension? In such a case, the map clashes over their borders and shows areas without any data. To avert this, a check function (referring to Ueberschär

2006) is edged in the zoom function. The check function takes the changed viewport values as parameters, compares them with the full map extension possible and if overlapping, pans the viewport to the outlying borders. The values are checked in the way that the window keeps the zoom level but it is panned to match the foresight perimeter for sure. In case of a zoom out over the maximal values, the viewbox is cut down to these values.

So the zoom function is enhanced in the following way: in a first step the new values with the zoom applied are calculated. Then the check function tests the new values and if matching, the values are allocated to the viewbox. If not, the values are panned or cut down to a matching level and further allocated to the viewbox. We refer to the full coding in appendix C.4.1 to have a look at the functions `myZoom()` and `myCheckVB()`.

Further two other useful zoom interactions are programmed executable through the user interface. One is a reset function, resetting the viewbox to the full map extension. This is useful in the case the user wants to restart it's looking up process on the map. The other one goes in the same direction and sets the focus on the calculated path. In other words it sets the viewport to the exact extension of the resulting path so that it is in the optimal focus from it's starting node to its destination. As the user probably wants to analyze his path more carefully and follows it zoomed in, he would like to have the possibility to reset his zooms and pans and return to the optimal focus. As this focus depends on the calculated path this reset function has to be dynamically computed for every new result. Further it is necessary to consider the ratio between width and height to avoid distortions. Complicated PHP code produces the dynamic JavaScript function:

```
// Get the viewport focussed on the
// calculated Path, if there is one.
if($_SESSION['totalPath'] != ''){
//Get coordinates from start and end
$PathArray = explode(',',$_SESSION['totalPath']);
$qLoc = "SELECT n_x, n_y
    FROM ".cStrTABLENodes."
    WHERE ".$_SESSION['FromID']." = n_id
    OR ".$_SESSION['ToID']." = n_id";
$res = mysql_query($qLoc);
$n1 = mysql_fetch_row($res);
$n2 = mysql_fetch_row($res);
$offset = $cMap['offset'];  //Widening in meters
$xDia = abs($n1[0]-$n2[0]);
$yDia = abs($n1[1]-$n2[1]);
//Checking for proportions
if($yDia*$cMAP['xDia']/$cMAP['yDia'] > $xDia){
    $yDia = $yDia+$offset;
    $xDia = $yDia*($cMAP['xDia']/$cMAP['yDia']);
    if($n1[1]<$n2[1]){$cornery = $n2[1]*(-1)-
       ($offset/2);}
    else{
       $cornery = $n1[1]*(-1)-($offset/2);}
       $cornerx = (($n1[0]+$n2[0])/2)-($xDia/2);
      }
    else{
       $xDia = $xDia + $offset;
       $yDia = $xDia/($cMAP['xDia']/
        $cMAP['yDia']);
       if($n1[0]<$n2[0]){$cornerx = $n1[0]-
         ($offset/2);}
       else{$cornerx = $n2[0]-($offset/2);}
         $cornery = ((($n1[1]+$n2[1])/2)+
         ($yDia/2))*(-1);
        }
    }
else{
    $xDia = $cMAP['xDia'];
    $yDia = $cMAP['yDia'];
    $cornerx = $cMAP['xC'];
    $cornery = $cMAP['yC'];
} //endElse

//Produce the JavaScript function
function myResetCalc(){
    document.getElementById('myMap').setAttributeNS
    (null,'viewBox','".$cornerx." ".$cornery."
    ".$xDia." ".$yDia."');
    mySetOverviewRect(["".$cornerx.", ".$cornery.",
    ".$xDia.", ".$yDia."]);}
```

First the coordinates of the point of departure and destination are looked for. With this coordinates an area is spanned. Further the area has to be transformed to an area in the same proportions as the given ratio between width and height. If the resulting path extends more north-south, the height is the defining value and upside down the width is determining in case of extending more west-east. Finally the JavaScript function is dynamically constructed.

The two reset zoom functions are applicable for the user when clicking a big "R" symbol right next to the "+" and "-" signs for resetting to full extension and when clicking on the symbol in the middle of the panning rose where a centering or focussing is adumbrated (cf. fig. 12).

7.1.3 Panning functionality

Panning is introduced in our map with an interface showing a kind of compass rose (cf. fig. 12) that controls the function lying behind. As we already know the technique used for zooming, the explication for panning is easy: if the user clicks a panning symbol, (i) the actual viewport is selected, (ii) values are transformed taking account of the direction wished for panning, (iii) the new values are checked with the `myCheckVB()` function and then (iv) applied to the viewport to reset. Finally (v) the overview rectangle on the overview map has to be actualized, too, executing the `mySetOverviewRect()` function. The parameter defining the step width applied for every pan event is just added to or subtracted from the actual viewport values. As a good constant 750 meters per step excels. `myPan()` in appendix C.4.1 is the function's print.

7.1.4 Sensitive overview map

With the panning interface described above, just continuous stepwise panning is possible. To change the viewport directly to a distant focus, this principle is not suitable. Often users like to freehandedly discover a map and to change location focus quickly. To realize a concept implementing this wish, our overview map gets into the game. Because the overview map all the time shows the whole map extent with the actual clip showed in the main map, it is a good tool for navigation. Our approach is to make the transparent rectangle covering the actual area in the overview map sensitive for mouse events and drag-and-drop. Assuming the possibility to drag the rectangle on the overview map to the desired location, no matter which area is on the full extent, and getting the new clip focussed in the main map when dropping, would give much freedom to navigation.
Reaching this interactivity needs much more thinking than zoom or stepwise pan. In the following, the underlying ideas for implementation are mentioned whereas a detailed description of the code is abandoned. With references to our code in C.4.1, the used text sources (Ueberschär 2006, Eisenberg 2002) and internet sources (SVG Wiki 2007, SVG Developers 2007) for this section, the interested reader can inform himself in detail.

Figure 12: *The full navigation panel as implemented into the map framework. Compass rose for panning functions, resets and zooms followed and overview map with transparent green sensitive area for panning by drag-and-drop.*

What has to be considered to develop a sensitive overview map? The core module has to select mouse coordinates on the screen when starting dragging and when dropping. Out of this offset in screen coordinates, the transformation into our used coordinate system in the viewport has to be calculated. This is the first time when we need to calculate the scale between screen and viewport which is a long procedure. Here again the most hindering fact is that different browsers support different methods to get the screen variables and to transform coordinates. In a conditional programm part all possible methods are checked and if available, are executed. Basically the transformation works as follows: (i) a transformation matrix is built up defining the transformation from screen into map coordinates, (ii) a fictive coordinate pair is initialized and (iii)

gets the values from the point to be transformed. Further (iv) the transformation is applied to the fictive point. Finally (v) the transformed point is returned (cf. function `myCoordTransform()`).
Now having a function that links screen and map, the user's mouse events can be handled. When clicking, the actual position is stored and the offset to the actual viewport is memorized. When moving the sensitive area, the actual position is continuously tracked. The moved distance is directly transformed into map coordinates and added to the actual viewport. Like this, the viewport is up to date for every displaced position anytime. The offset (accounts the position where the user picks the sensitive rectangle) has to be considered, too and has to be calculated into the transformation. With the drop, the actualized values are checked if not overlapping the full map extension (`myCheckVB()`) and then applied to the main map's viewport. All this is implemented in the so called function `myDrag()`. To get an idea how the final result looks like, Figure 12 shows the whole navigation panel with sensitive overview map to be dragged.

7.2 Disposing extra layers with the trail

Every modern portal for interactive mapping provides further geographical information and data in extra layers lying above the map background. This data can be loaded into the map without reloading the map each time. It is the user who decides the level of detail he wants to see and how his map should look like. The decision lies in his hands and a big part of individuality is guaranteed by using this feature. A comparison with `maps.google.ch` or `map.search.ch` shows the perfect realization of the layer technique in this portals.

7.2.1 Compilation of passive data into GDB

As a first step, passive data allocated by the layers has to be introduced into the GDB. The data received is all in `.shp` format and needs a transformation to match the needs for the database. To prepare the database for the passive data, two new tables are introduced into the GDB: one to host polygonal shapes called "ENVPOLYGON" and the other containing all the linear features as rivers, streets etc. called "ENVVECTOR". The table just store the feature class (as river, lake, canton, highway etc.) and the SVG representation for this feature. Like this, future use in layers can be quite modular: every kind of feature is selectable by an independent class out of the tables.
The compilation process happens quite similar to the compilation techniques in the topology's build up. After transformation `.shp` files into ASCII files containing the raw coordinate pairs, a compilation script read in the text file, transforms the coordinate chains into SVG code and introduces it into the GDB. For every feature, small adaption has to be applied to the compiler script, but we do not want to go into detail here for these steps.

Figure 13: *A comparison between original aerial image and compressed and resized image should give an idea of the level of downsizing. Both images are exactly the same extract. They are both not zoomed, therefore the different sizes result from reducing image size. We can still recognize a lot of details in the processed image at the bottom in comparison with the original one. Nevertheless, a massive compression is adumbrated by the smaller size and the difference in sharpness and contrast.*

Concerning the georeferenced aerial images from the cantonal bureau of geoinformation, the image transformation is looked at more carefully. As in chapter 4 described, each tile measuring 8 times 8 km with a resolution of 2 meters needs about 60 MB of memory. Of

course this filesize is inappropriate for Internet use. A resizing and compression is necessary to decrease filesize to a justifiable limit. To reach optimal compression values allowing a small filesize and nevertheless a good quality, several tests have been made. Using Adobe Photoshop 7.0, the image compression was adapted as long as the optimal values were reached. To reduce filesize even more, a downsizing of the image by 45% is inevitable. The result, visible in a comparison in Figure 13, reduces filesize to less than 300 KB for a tile fully covered with the aerial image. The compromise between loading time and quality seems optimal for this compression level. It was a goal to keep the possibility of a zoom into the image and see more details anyhow.
In a next step all original images were compressed and downsized with the atomization function in Adobe's Photoshop. The original filenames were kept so that a concatenation between coordinates and filename is still possible. The high compression was also exercised in ArcGIS but with dissatisfying results.

7.2.2 Asynchronous load of details

To implement an equal service, the technique behind has to be understood first of all. With reference to the book of Ueberschär (Ueberschär 2006), a very good article in the "XML Magazin" (XML Praxis 2006) and several good internet sources (Mozilla 2007, Carto:Net 2007) the used programming language "AJAX" was learned. AJAX stands for "Asynchronous JavaScript and XML" (Mozilla 2007). "Asynchronous" means that data can be loaded into an already built webpage over the http protocol. The data must not be loaded synchronously with the full page at the beginning. "JavaScript" indicates the programming language over which AJAX is directed. And finally XML defines the standard transmitting format for data that are loaded via AJAX. Just with this definition we already know how we have to proceed to load layers dynamically into our map: (i) dispose a JavaScript function addressable by user input that (ii) calls the AJAX precedures fetching (iii) the XML data dynamically processed through a script on the server side and finally (iv) input the XML data to the correct position of the existing file on the client side.

The AJAX core functions deal with the exchange of data over the Internet. There are predefined methods implemented in browsers that understand the use of these functions. Here a really big handicap gets again into the game: different browsers are based on different standards. Whereas Microsoft and Adobe SVG Viewer understand the old AJAX function `getURL()`, Mozilla and newer browsers deal with the `XMLHttpRequest()` method. The two functions produce more or less the same result but they work totally differently. They still use different parameters. With regard to all browsers (we want to develop a tool working barrier-free on every common browser that is able to display SVG) a diversion has to be implemented. With relation to the suggestion given in Ueberschär, we deal with a function combining both possibilities. The function tries to apply the `getURL()` method first and when failing it applies the `XMLHttpRequest()` method to load external data. The function called `myAddXML()` in the file in appendix C.4.2 corresponds to the implementation. The function demands for parameters "source file name" on the server disposing the data to load in, "destination id" corresponding to the destination in the main file where the new data has to be placed in and the "encoding" defining which format the added data is written in. The encoding is only necessary for the older `getURL()` method (Mozilla 2007, Ueberschär 2007).

That AJAX is able to access additional data on the server, it has to be in the same XML dialect as the destination file is. Therefore it is important to never forget the definition of the header-type and the XML namespace used. This tiny but limitating fact caused hours of tests and abortive attempts.

```
//Definition XML dialect
header("Content-Type: image/svg+xml");
//XML namespaces in use
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
```

The allegation above is just necessary for one of the two methods described, which caused

even more problems: once the implementation worked without this data but when testing the same code with another browser it suddenly failed. As for our project the XML namespace for SVG is applied, the document header in the files containing the data to load in has to be set to `image/svg+xml` as in the main file and the namespace for SVG (`http://www.w3.org/2000/svg`) has to be defined in every file. When using other namespaces, too in the fetched file, they have to be defined additionally (e.g. xlink). To be sure to apply the correct XML namespaces a view on W3C is worth doing (W3C 2007).

A difficult step comprises the introduction of the asynchronously loaded data into the still existing document on the client side. Therefore another principle has to be introduced: the DOM. The "Document Object Model" defines the hierarchical composition of an XML file. Within the DOM, every element of the document is clearly identifiable and addressable. The document tree that is spanned with the elements (e.g. in SVG the root element `<svg>...</svg>` encapsulates all other document elements) is parsed in the XML document and after parsing every node or element or even whole branches of the document tree are addressable. To use the DOM optimally it is recommended to identify every element in a XML file by the `id=''...''` attribute. This attribute additionally rules navigation in the document. For further complements to this topic we refer on the W3C standards and documentations (W3C DOM 2007).
Now when the DOM is captured on loading of the document, the navigation in the element hierarchy becomes possible. We can choose a node of the document element tree and associate new branches or change the addressed node as we like. Via JavaScript and DOM-methods the access to the elements is done easily. Concluding the use of the DOM, we now have the technical framework to apply the additional data, loaded through AJAX, into our main document to the exact place we have foresight. In our map or SVG file, the DOM consists of several branches where one comprises all elements belonging to the main map. To reserve a place in the DOM for every possible additional layer data that could be introduced into the DOM via AJAX, every layer gets an empty container element. The element is a `<g>`-element, meaning a group element with no content. Addressing the ID of such an element, data can be added to this node position in the DOM and can be deleted again form this container node, too. Below, the containers as they are used in our SVG-DOM are copied:

```
<g id='LayerGroup1'>
   <g id='LayerEnvpoly'/>
   <g id='LayerEnvcomm'/>
   <g id='LayerEnvvec'/>
   <g id='LayerEdges'/>
   <g id='LayerNodesSignpost'/>
   <g id='LayerNodesSecondary'/>
   <g id='LayerNodeNames'/>
   <g id='LayerRes'/>
</g>
```

Lets play an example for better understanding: a user activates the layer "Edges" in our map. Via AJAX the script is addressed that computes the edge's network dynamically in XML. The XML data is transmitted from the server to the client. The DOM method `document.getElementById('LayerEdges').appendChild.[TheXMLCodeTranmitted]` advises the AJAX components to introduce the additional data as child node in the DOM to the `<g>`-element identified by `LayerEdges`. Another aspect presents itself when understanding the example and the container elements: the hierarchy of how the containers are introduced into the DOM defines at the same time the layer hierarchy in the produced map. As the code in the main file is synchronously interpreted by the browser, the lower a container is located in the document, the higher it is placed on the result. This implies that the container listing has to be thought of carefully. Which layer has to overlay another and at the same time covers possible data from the underlying layers?
Our approach defines the layer hierarchy as follows (see code snippet above): the bottommost layer shows the landmarks as rivers and lakes. It is followed by the layer containing the communal frontiers and names of the municipalities. The regional traffic lines

(`LayerEnvvec`) are not as important to our purpose as the hike routes. This is why the hiking network (`LayerEdges`) is listed after the regional traffic. As the most important layer, overlaying all other layers, the resulting path is defined. It should always be visible, doesn't matter which other layers are activated at the same time.

To implement the inverse functionality that deletes a layer from the main file again, we can simply delete the corresponding child nodes in the container by use of the DOM method `[containernode].removeChild`. We implemented a method that takes the container node as parameter and deletes all its children when executed (function `myDelEle()`).

To get the full functionality the two main functions for loading data and deleting data from the main file are brought together into one single function (cf. `myLayer()` in the file extract C.4.2). The function itself deals with activating a layer or deleting parts of the DOM where the layer data is collected. So first of all a check is accomplished that tests if the layer is already loaded (it simply looks for child nodes in the container). If it is loaded, the execution deletes the container content and with this the layer. Else the `myAddXML()` method is utilized to load the layer data asynchronously into the DOM. With this implementation an interface can be developed where the user clicks on the same symbols to activate or deactivate a layer.

7.2.3 Interface to activate layers

A common interface is programmed that should offer a good usability. The layers are listed in the corresponding space under the overview map. The rectangles anteceding the labels characterize the status of the corresponding layer. As it is filled and a cross marks the rectangle, the layer is activated. When hoovering a deactivated layer with the mouse, the rectangle fills and shows the user that the layer can be loaded into the map. The labels are sensitive, too. A click on a label equals in the same procedure as when clicking into the rectangle.
When clicking the client browser notices it and makes the `myLayer()` function via JavaScript work with the correct parameters which induces the asynchronous load of the new layer data.

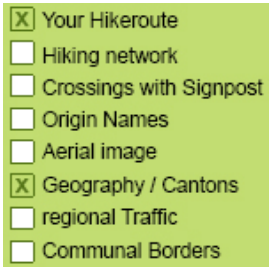

Figure 14: *The layer interface as shown in the map. English as language is active and the layers "Your Hikeroute" and "Geography / Cantons" are loaded. These are the two layers loaded per default as well.*

Figure 14 illustrates the interface. The layer activating the aerial images is just camouflaged as a layer. In reality a normal link lies behind the visualization, forwarding the user to the new map in the mode for aerial images. The user does not recognize this and the interface keeps easy to use with just one kind of interactivity.

7.2.4 Display geographical features

A very important layer that helps the user to navigate and orientate shows the main geographic features in the focussed area. How do we proceed when studying a map? We first get an overview over the area when looking for the remarkable landmarks. Landmarks as lakes, rivers or mountain ridges are helpful markers in a map to get a fast overview. Especially for Switzerland cantonal borders are a main feature in regional maps (cf. Wanderkarte Ostschweiz) too. The goal of fast orientation help doesn't imply a need for extremly detailed data, because data is just used for overview purposes. Therefore, the delivered vector data from the geographical institute UniFR (IGUF 1996) is suitable.
Not only fast orientation but also little memory requirements are provided for this layer: as it is a key layer, it is often displayed and, therefore, little data load for a fast transmission is needed. For this purpose the less detailed vector data is suitable, too.

SVG provides good techniques for representing map signatures as they are especially developed for geographic map representation. Looking at the official signatures used by Swisstopo (Swisstopo 2003) almost every important signature is possible to reproduce with SVG. Therefore it is obvious to use the known and usual signatures for the features represented by this layer.
When a client asks through AJAX for this layer, the script in appendix C.5.1 is run. It firstly gets the SVG paths from the GDB for lake-, river-, and cantonal border features. After data storing in arrays the XML code is generated for every feature individually. For every different signature we use a new surrounding `<g>-Element` (group element) that defines the look for the embedded data. Each group element links to the common constants file defining colors, strokes or fillings centrally.
As it is usual for rivers, we use a linear signature showing a light blue riverbed with thin dark blue borders. To reach this representation a common "trick" is used (leaned on Ueberschär 2006): painting two corresponding paths, the one in the background in dark blue with a stroke-width slightly bigger than the overlying light blue path. To avoid unnecessary data load for such a representation SVG supports the `xlink` method. Encapsulated group-elements are chosen by their ID and are overlaid automatically without displaying the same code several times. The following code snippet shows such a procedure in a simplified way (the code is a cut from the file in app. C.5.1):

```
//Main Group for River vectors
<g id="myRiver" stroke="darkblue"
    stroke-width="15" fill="none"
    stroke-linecap="butt"
    stroke-linejoin="round">
    //The encapsulated group
    //The main vector path
    <g id="innerMyRiver">
       <path d="xxx"/> //endPath
    </g>
    //Copy the encapsulated group
    //with different attributes for
    //the middle line in the main path
    <use id="myRiverMiddle"
        xlink:href="#innerMyRiver"
        stroke="blue" stroke-width="10"/>
</g>
```

The two lakes in focus are Walensee und Bodensee (Lake Constance). For lakes, polygons are used with a light blue filling, adapted on real maps, with a marine border. The lakes are named in their center of gravity with different font-sizes matching the area covered by the lake. Fonts are in italic and serif as it is common in printed maps.
In the same way, the cantonal borders are represented with a wider, transparent path and a fix middle line. Green, as it is the used color, helps the user to orientate rapidly in the politico-geographical context.

This important layer is loaded per default with the resulting map. It can be easily deactivated by the layer navigation and in case of reuse again asynchronously loaded into the map.
When aerial images are activated this layer becomes automatically resized. Because quality of vector data is not as precise as the used aerial raster images, an overlay would not match. Therefore in aerial image mode just the lakes and the cantonal borders are shown within this layer. The two different approaches can be followed in the source codes for both layers in the appendix (C.5.1 and C.5.2).

7.2.5 Display terrain

Another characteristic feature in a map is it's terrain. Normally it is visualized by isohypes and a shading background image giving the area a light / dark vision. The hillshading effect lets the viewer getting an idea of the terrain and possible hills or valleys. As isohypes are not available in vector format, the terrain for our tool has to be represented only by the hillshading background image. The image is adapted to the purposes: it has to match the need for little memory requirements as it will be loaded through a public Internet connection. Generated with `ArcGIS` out of the DHM75[5] (Swisstopo 1996) a terrain image

[5]DEM75 is the abbreviation for digital elevation model or "digitales Höhenmodell". The "75" gives it's resolution. So for the DEM75 for each area covering 75 x 75 meters a generalized elevation value is available.

with a clear hillshading effect results. The angle of sunlight radiation to the terrain is just little adapted to the values normally used. To get the best results, an angle of 35 degrees vertically and 145 degrees horizontally was used. In the result, the conquering mountain range Alpstein with its foothills are well recognizable.
As the terrain is a raster image, it's input to SVG needs to be done by the `<image>` tag which loads an image file into the map. This leads to another problem, coming from different implementations of SVG in the different browsers: as Internet Explorer supports asynchronous load of files, Mozilla for example does not. As the terrain background is an important characteristic of a map, we defined to load the terrain as a default layer directly into the map. The layer management is implemented with a simple trick: if the user wants to disable the terrain background, a white rectangle will be drawn over the terrain image. This lets this layer "load" in asynchronous mode, too.
The layer is defined as the layer lying the most in the background of the generated map. All other layers are defined to lay over this layer.

7.2.6 Display signpost locations

Within this layer the locations where signposts are situated are visualized. These locations are important because all route calculations start or end on such nodes. Normally located on hike route crossings, these nodes are an important feature the user likes to know, when regarding the route network. To avoid heavy data load, subnodes are delivered to a separate layer and just the nodes described are introduced to the signpost-locations-layer. The locations are pointed out as circles with a yellow filling in adaption to signpost's coloring, and a black stroke around.
When demanding for this data layer by a user input, the corresponding script (see app. C.5.8) firstly queries the nodes out of the N_NODES table with attribute type = 2 standing for nodes in the topology with signpost. After querying, the results are converted through PHP into the needed XML correspondence. The `<circle>`-Tags are filled with the circle's center where the x-coordinate corresponds to the attribute n_x and y-values equal the negative value of the n_y attribute. The negative y-value is needed, because the origin in SVG viewports lies in the upper left corner. The advantages of Scalable Vector Graphics become obvious when applying the values for circle sizes: as in a CAD-software, feature sizes are applied directly in the defined coordinate system. So definition of circle diameter is directly in meters. This gives a direct idea of the covered area by the visualized node.

7.2.7 Display names of locations

Location names are directly linked to the nodes where signposts are located. To let the user identify the locations, their naming is necessary. The layer can be activated in the layer management and is not coupled with the layer displaying the nodes with signpost. Two reasons lead to their separation: the data load for the two layers together would increase the waiting time unnecessarily and we want the user to keep the best possible individuation. Probably he just likes to analyze the hiking route network and is not interested in each node's name.
There is no directly linked data where to display the naming. Therefore the link between node and name has to be done. The following procedure shows the best result: (i) get the names and the x- and y-coordinates out of the GDB, (ii) reproduce for every node the location name with the `<text>`-Tag and position it on the x- and y-location and finally (iii) embed the XML code with the names into a group element so that the text-anchoring can be set for each entry at once and, because the circle is probably already covering the location, translate all entries so that they lie some meters above the matching node. Like all codes, this script is copied to the appendix, too (cf. C.5.9).

It is important to know that with this procedure, it is possible to get some labels covering each other. But guaranteeing the goal of full dynamic processing, this is not avoidable. When creating static maps, the labeling pro-

cess is done as described but has to be adapted manually to avoid overlapping and to set the label to the best local position. This procedure would disturb the dynamic processing.
To let the names be readable when zooming, the font-size is predefined to 50 meters.

7.2.8 Display hiking network

To let the user get an idea of the neighboring hiking routes, a layer showing the hiking route network is adequate. When enabling this layer, the whole routing network is overlaid in a simplified manner: the best visualization for this layer would be a display of the routes in their precise paths. Therefore we use the SVG path already calculated when building up the topology. The amount of data transmitted for this visualization would reach a threshold where waiting time for users with slow connections get inacceptable. As solution to this problem, the layer just displays the lines, where connections between nodes exist. It is a good compromise between data load and overview to proceed like this: The user rapidly gets an idea of what ways are possible to take without seeing the real path. But as the real path following the terrain, still interests the user much, we decided to fully visualize the real path for the calculated path: the data load is minimized, the user still gets the concrete information for his desired path and he is able to get an overview of further existing connections. Because the density of the graph is quite high, the abbreviation with direct lines is still not too bad. The short linear intersections between nodes do not much diverge from concrete paths.
The layer script which is loaded asynchronously when activating the layer first reads the starting- and endnnodes of every edge in our topology out of the GDB. In a next step the result is transformed to a XML document containing one single path visualizing all connections between nodes.
In the vector map the paths are illustrated in red in the same way as the hiking network in the printed map. For better visualization when aerial images cover the background, the same layer is available with the path displayed even sharper and with black thin borders (cf. files LayerEdges.php in C.5.5 and LayerEdgesSat.php in the appendix C.5.6).

7.2.9 Display subnodes

In the final map, this layer of subnodes is no more available. During the testings it has helped the developer to get a better idea of the network. When primary- and subnodes are printed, all existing nodes are visualized on the map. Adding the hiking network layer, the developer can check the built topology in its total perimeter, which is very helpful in the developing phase. The subnodes are produced exactly the same way as the primary nodes described above. To distinguish the nodes on the map, color and size are chosen differently. The file "layerSubnodes.php" copies the source in the appendix C.5.7.

7.2.10 Display municipality borders and names

For some users, who already well know the region where they want to look for a hiking route, municipality borders are helpful for orientation. Another layer provides this information. When activated, the municipality borders are shown with dashed borders as in Swisstopo's official signatures (Swisstopo 2003). In the municipality's gravity center, the commune's name is displayed with a serif font. When aerial images are activated in the background, the same layer is available with different signatures better contrasting with the images in the background.
The coordinates for the gravity centers are provided by Swisstopo openly (Swisstopo 2007). All reliable communes in the canton, where data is available, are named in this way: the names are copied directly to the layer script and are placed centered to the given gravity center.
Another aspect results from quality of received data: where the sources for traffic lines and landmarks as lakes and rivers are of bad resolution and more or less approximated, the source containing communal borders is much more precise. This causes a conflict in visualization

of both kinds of data together. The cantonal borders that should follow the communal frontiers or the lake borders don't match and disturb each other. The solution to this problem gives the simple idea that two qualitative different data layers can not be shown together at the same time and overlap. The communal layer lies in the layer hierarchy over the layers with less precise data. When the communal layer is activated, a white rectangle is drawn over the whole viewport first, covering all layers in deeper hierarchy with more approximated data. So it is assured that interfering between different data becomes impossible.

7.2.11 Display resulting hikeroute as dynamic layer

As it is an important milestone, the dynamic representation of the resulting path calculated through Dijkstra's modified algorithm in a separate own layer is here discussed more carefully. To keep the most possible flexibility after visualizing the calculated result and to be free in the layer handling, the resulting path should be represented as single layer always visible, overlapping all other layers and being the root of the layer hierarchy. The implementation caused difficult hurdles to take: (i) extraction of location names, (ii) signpost locations and (iii) SVG edges connecting them out of the calculation results, (iv) querying this data just for the nodes and edges covered by the track to reduce data amount and finally (v) think about compatibility with the display of underlaying layers.
For the querying a dynamic SQL statement is created demanding for the necessary data out of the different tables for every node visited in our track. The SVG paths are also requested dynamically. A key function returns an array containing the used data for the starting node and the data for the edge connecting itself with the next node in the resulting path. To avoid double entries when treating via points, the resulting string array out of Dijkstra's algorithm has to be filtered first of all. The double entries for via nodes, marked with an "xxx", must be deleted and the first entry defining the used cost factor has to be put aside. In a next step, the data stored in the array is transformed into XML step-by-step. To avoid complications in visualization with other layers, the naming has to be done with the same size and in the same position as the naming layer provides. To overlap the signpost layers nodes, the size defining the radius for a node is slightly bigger in the result layer. To emphasize the resulting path from the rest of the hiking network, another coloring and a broader path width is chosen. Finally the result layer is loaded and activated "onload", because it is the main purpose of the map. The user has full management over the layer as it is introduced to the layer menu in the map. The generating script is available in C.5.10.

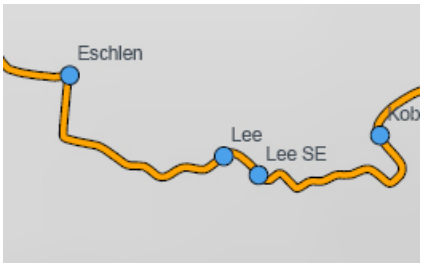

Figure 15: *SVG provides good methods to visualize paths that are a collection of several small lines chained.*

In the construction of this layer, SVG's advantages for digital map representation become visible. The resulting path showing the track as it follows the terrain in reality is curvy and non linear. As the digitized path consist of hundreds of coordinate pairs connected with a line, the result should look cornered and not natural. With the two attributes "stroke-linejoin" and "stroke-linecap" SVG is able to display this chain of lines rounded and this way approximates paths representation in printed maps well. As already described in the geographical features layer, the path is bordered with thin black lines using the "xlink" technique in encapsulated group elements. Fig. 15 clearly focuses this visualizing techniques.

7.2.12 Display regional traffic

Regional traffic lines traverse the terrain and give good orientation marks, too. The available vector data allows to display regional traffic in a single layer. High streets, highways and railway lines can be loaded within this layer. Computation differs from other layers just in the way the signatures are chosen: referencing to Swisstopo (Swisstopo 2003), the signatures reach official signatures as well as possible. For the railway lines, the differentiation between narrow-gauge railway and standard gauge railway is not done.

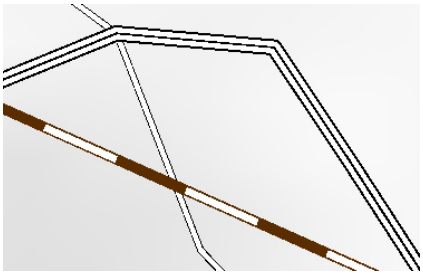

Figure 16: *The traffic layer provides the main regional traffic lines with its signatures approximated to the federal standard.*

Instead of using the signatures for standard gauge railway for all lines (a filled black line), we use the signature for narrow-gauge railways for all railway tracks, because it contrasts better with the background. For highways and main streets the official signatures can be achieved through the technique with encapsulated groups and the "xlink" method as shown in Figure 16 by an extract.

7.3 Underlaying ortho-photographs

The distributed ortho-photographs covering the dealt perimeter give the possibility to offer a totally different mode for our map. As the photographs are georeferenced, paths should perfectly match the ortho-images.
Normally this feature would be offered in the map as additional layer, too. During the implementation we have considered that AJAX and asynchronous load of data in XML works well for pure XML files but not for XML data containing references to extern files. Adobe SVG Viewer and Internet Explorer, using the old `getURL()` method, do not support load of external files, whereas Mozilla and many other Browsers do support this because they use the newer method `XMLHttpRequest()` (cf. Ueberschär 2006). Regarding this complication, we decided to implement the feature as totally new mode to our map. As the mode should be available for every user, indifferent which Browser in use, the aerial images have to be loaded with the map synchronously. The images have to be part of the whole map and can not be offered as optional layer. To keep handling easy, the new mode is camouflaged as a normal layer in the layer management module on the map. But when clicking on the alleged layer to load the images into the background, the whole map is newly loaded. To avoid long waiting times, in a first step the images are only loaded for the perimeter covered by the resulting path. This allows us to provide still good quality raster images where zooming in brings more details into the focus and limits loading time to a reasonable value. In a next step the user is able to activate again a camouflaged layer which is loading a new map with all aerial images loaded into the map. This feature of course uses much more data load, because every available image is loaded in a quite high quality.

7.3.1 Showing images for the resulting path

To load just the relevant aerial images for the resulting path, a function has to calculate which files have to be attached and it has to output the correlated XML code. Because the images are already named regularly (compare section 4) and the filename implies the coordinates, it is possible to process the code dynamically for every resulting path individually. The function proceeds in first memorizing every coordinate pair for every node in the resulting path. The coordinates are then transformed into the code needed for the image filename: every image tile is named by it's north-west corner in kilometers and comprises of 8 x 8 km^2. So we have to see on which tile the actual node is lying and then reduce its coordinate to a km value and floor it to the

next possible corner value of a tile. The following code is a simplified copy from the original script showing the coordinate transformation for every node in the result path to get the filename of the underlying aerial image tile:

```
//transformations for file names
//offset for x and y = 6km
//inner floor to floor original coor to km
//outer floor to leave the rest of division by 8
$tempx = (floor((floor($xcoor/1000)-6)/8)*8)+6;
//because inverse coordinate system finally 8km
//have to be added to the result to match the
//correct tile
$tempy = (floor((floor($ycoor/1000)-6)/8)*8)+6+8;
```

This converted coordinates are stored in an associative array. To delete double entries that result from several nodes lying on the same tile, the array has to be filtered first. The resulting array contains once every coordinate pair in km defining the north-west corner from the corresponding tile. Out of the array the XML-code can be computed loading the needed images into the background of the map. The whole function and its output is visible in section C.5.11.

Figure 17: *Underlaying aerial photographs give the user a direct idea how his path looks in reality and which covering is traversed. The same extract as in Fig. 15 is chosen.*

The result is satisfying: the vector layer for the resulting path matches the georeferenced images totally. Even the smallest direction changes are visible in both of the data: vector layer and aerial photograph. The overlaying gives the perfect conclusive image. The accented path directs the user where to follow the way on the background image.

7.3.2 Loading all images into the map

If every allocatable tiles should be visible on the map, the complex function gets obsolete. Because of the claim to keep all processes dynamical, the selection is done as follows: every node in the topology is queried, transformed and adapted to match the image's filenames. Because double entries are filtered, it is possible to proceed like this. Apart from this, everything for this mode is equivalent as for the mode where just the underlying tiles of the resulting path are shown.

7.4 Further features and visualization of additional information

Additionally to the navigation and layer interactivity, our map provides further information and concepts. The following subsections describe the additional features and give an overview on what is to be discovered in addition to the navigation and layers.

7.4.1 Get the coordinates displayed real time

In the section describing the navigation functions, we introduced a transformation method calculating map coordinates out of screen coordinates. The same function can be used again if we want to display the map coordinates of the actual mouse pointer. It is very useful to get the map coordinates displayed real time when discovering the map with the mouse. In combination with a printed map, locations can be looked for just by placing the mouse on the wanted position in the map, looking at the coordinates and then searching for the coordinates on the printed map. When following a path with the mouse, the coordinate series are visible at anytime.
As the transformation method generates the map coordinates without having consideration for zoom level, this additional feature can be developed without big effort. The small function `myMouseCoords()` (cf. app. C.4.1) which is executed with every mouse move per default, generates the x- and y-coordinate and disposes them for a set into the map. As the real time changing display for the mouse position belongs to the map itself, the coordinates are directly set into a container element on the main map in the bottom right corner. The container itself is localized on top of every layer, so it is always visible. To enhance contrast

in aerial image mode, the display is underlaid by a semi-transparent rectangle matching the extension of the values.

7.4.2 Scale Bar and coordinate grid

As on a printed map, the coordinate grid and a scale bar are indispensable. These are passive features, never changing and completing our map's look. When realizing the coordinate grid, SVG's possibilities for mapping become again candid. It is totally easy to draw the grid: just use the original coordinates within a `<path>` element to span the grid. On every map we draw the 10 km grid and the 5 km sub-grid which has a slightly narrower stroke-width. To finish the map on the total extension, a small border is drawn, showing the coordinates on the ending of the 10 km grid. The scale bar is drawn in the bottom-left corner and gives the relation between the map and 5 km in reality.

7.4.3 Hike route profile

The disposition of the route profile amends the services available for the user. If his path is calculated, a path profile should be disposed to get a direct idea of the vertical progress of his hike path. As the map just delivers information in the two horizontal dimensions, a profile is a needed complement. Diverse information should be visible at a glance: altitude, progressing distance and time as well as location names. We want to additionally dispose the information about gradient value. This can be reached by allocating different colors to different gradients. To chase our pursued path, all the graphic should be dynamically computed SVG code. Figure 18 displays an example profile where all specialities are well visible.

The implementation starts with gathering the data out of Dijkstra's result. We need all the necessary information for every single node met on the resulting path. Therefore each node is serially taken and queried on the GDB. Because we need data about the connecting edges, too, this data has to be queried additionally for every location on the route. The data for every node and outgoing edge is stored in an associative array which is the data container for visualizing the profile. Lines 37 to 77 in the code in appendix C.6 echoes the data array's build up.

Before setting the graphical parts, we have to calculate the scaling of altitude and length so that the resulted path perfectly fits the foresight diagram extension. The data array is run-through whereas minimal and maximal height is searched and total length and time is computed. Out of this data the scaling factors for height and distance are extracted.

Next all the information has to be transformed into SVG code visualizing the profile's details. Therefore a main viewport with the coordinate system for the diagram is initialized seizing 12000×8000 and origins at (-1000, -7000). With this extension it is possible to draw the diagram's origin at point (0,0) which makes the use of complicated offsets obsolete. The used method to draw the diagram works as follows: a node is taken into account, it's total distance from the departure is scaled, it's height is scaled, too, and offset to the minimal scaled height. Further it's distance to the next following node is scaled and the scaled height of the following node is computed, too. With this procedure we now have all the four corner points in the viewport coordinates to draw a polygon defining the following path section from the node focussed. In a loop, all nodes on the path are approached the same way and all polygons are drawn. To introduce another interesting information into our diagram, the polygons are filled with a more or less dark color marking the steepness of the corresponding section. The steeper a gradient, the darker the polygon. In the background the color is computed in the way that the gradient (available for every edge in the GDB) is accounted with the `fill-opacity` attribute of the polygon. To reach good results the following transformation is used.

```
//Adapt opacity in dependence of gradient
//$opGr = Offset of 0.2 to get clear results
$fOp = $opGr + (3 * abs($r[$q]['gradient']) / 100);
$poly .= "<polygon fill-opacity='".$fOp."' ...
```

To get a fill coefficient ($0 < \text{coeff} < 1$) allowing good contrasts, the absolute gradient (in per-

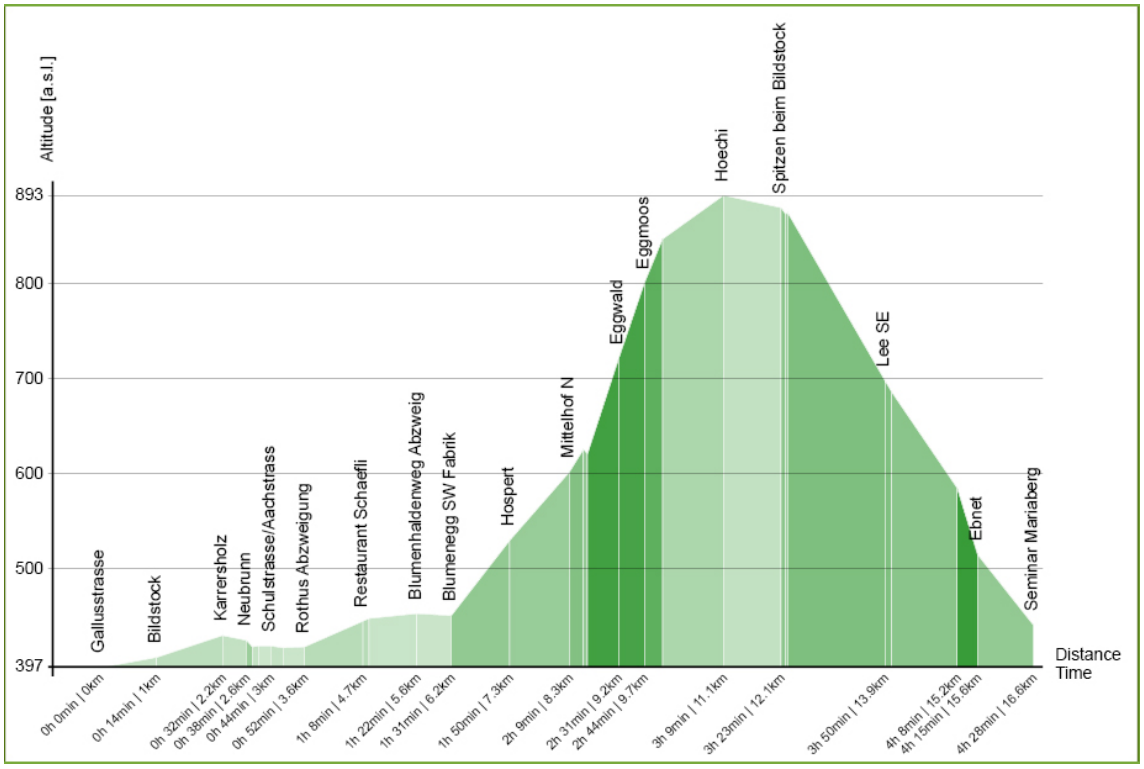

Figure 18: *A screen print gives an idea of the profile calculated and points out the following specialities: (i) The darker the color, the higher the gradient, (ii) the profile is scaled to the diagram's extension, (iii) altitude grid is dynamically adapted a labeled with the maximal and minimal values (if extre values near a hundred limits no disordering possible) (iv) course of distance and time is displayed (disordering impossible because of minimal gap between neighboring nodes is defined), (v) location names are displayed whenever possible but without disorder (minimal gap limiting distance to neighboring labels).*

cent) is taken, divided by 100 to get a value below 1. Next this result is forced by trippling its value and enforce contrasts. Finally a small offset is added which avoids clear (white) polygons.

The labels in the diagram need special treatment because many spheres have to be thought of carefully. We need a diagram that is suitable for very short paths with only little waysections but also fits very long ways with many sections where polygons become quite thin. In the latter case a caption for every location won't be possible anymore without overlapped and unreadable labels. A minimal space between each label has to be defined and considered when computing the labelling. The same counts for the text information for distance and time. A loop goes through every node and actualizes an implied buffer. If the buffer is below the minimal space, the location won't be labelled and the empty space is added to the buffer. If the buffer reaches the limit, the location will be named and the buffer is reset. The font rotation is chosed to optimally use the existent space in the diagram.

As a last resort, the elevation grid must be dynamically laid over the diagram. As we already have computed the minimal and maximal height in our diagram, we first draw this extreme values. A small function than computes the elevations divisible by 100 lying between the extremal values. The inlying limits are drawn with a thin black line and labelled on the ordinate. As well as for the location names, we introduced a buffer to avoid two lines and labels almost overlapping each other in the case where extremal values are near hundred limits. Lines 189 to 214 in the code copy in C.6 echoes the computation described.

7.4.4 Detailed information about the track

Leaned on the big players like `google.ch` or `search.ch` we would like to dispose the user also schedular detailed information about his track. This is the simplest way to loom the whole route information. A table disposing every possible information that can be queried out of our data gives the user the final information if he is interested in. Of course this schedules are often used in printed forms on the journey itself. Therefore also this feature has its importance and is implemented carefully for the sake of completeness.
The strategy to get the data into a descriptive table is easy: sequentially work through every node in the resulting path querying the necessary information to be disposed. As we have already introduced a similar function to get all the information out of Dijkstra's result for our diagram, we can take the same code and change the necessary lines to adapt it to the purpose (listed in app. C.7). The final look of the table shows the history of the linear path with information about nodes reached and edges passed. The history visualizes for every node it's special information about elevation, location name, municipality, walked distance to the node, time needed to reach the node and total meters in altitude overcome. Between data for two neighboring nodes the inlying edge section is described with its absolute length, gradient, meters in altitude and the time supposed to walk through the section. The via points are specially colored and marked to keep the orientation if long tables out of long paths result. At the history's end, the path is summarized with total distance, length, time and meters in altitude climbed.
With the table the user gets full access to all necessary information about his track in one single view. Printing the schedule for the real hike is no problem as well as getting the full idea of the personal hike calculated.

7.4.5 Invitation for a hike via email

In a more qualitative framework this thesis would like to vitalize hiking as an activity as well as demonstrate a possible way to advertise for regional hiking possibilities. Disposing modern ways of planning and organizing will help to strengthen hiking in a world getting more and more stressful and booked up. Still more tourism is interested in an appreciation and advertising for local hiking networks. A wide, maintained, accessible and advertised hiking region gets more and more important for a sustainable and soft tourism.
Interactive maps could contribute to a realization. What about planning a hiking trip with friends in the local region the next weekend? What about discovering locations we have never been before and having a predefined and already discussed route at hand? Assuming this situation, it would be nice to have our system coupled with the possibility to send our hiking route to a friend. If you look at your personal route in all it's facets you might feel like discussing it with your friends, adapt it and finally send it as an invitation to your friends. This would give totally new possibilities.
Having this objective in mind, we implemented an interface where the user can generate a personalized email for his friends containing a link guiding directly to the personal route. The email can be sent as direct invitation for a hike or simply as advice or idea for a future hiking tour. More over it could be the source for a discussion of the best way to choose for the next hike or gives subsequently access to track the route hiked the previous day.
What we have to do is to think about an interface that lets another user view the same information like the user sending the email. To avoid extra traffic on the server, an extra calculation for the already computed way is not desired. Therefore, the first step is to save every calculation on the server with the possibility to allocate it to a user. We store each output of Dijkstra's calculations with the user's IP and the actual time stamp into a table in the database. Like this we implement a small history and log for our system at the same time, which can be used for user statistics etc. In a next step we need to allocate to

the link sent by the email an explicit identification allowing a script to look for the calculated path in the history database. We define a request variable in the link containing the unique ID number identifying the corresponding data tuple in the table (cf. lines 61 to 71 in C.8.1).
When the addressee clicks on the link he would like to be directly guided to the map with the proposed hike route. To realize this, a new script (see app. C.8.2) intercepts queries on the server which contain a marker defining the email link as source. It requests the ID out of the URL, looks for the data in the history table, sets up the session in exactly the same way as it would be done when asked directly for this route and finally redirects to the script generating the map. The user who clicked the link does not detect any of these computation steps because he is quickly redirected to the map with the visualization looked for.
Of course the email is generated in the language in use and is personalized as well as possible.

For the sake of completeness it has to be added that the mail class, responsible for automatic html mails via PHP, is leaned on a work of Rainer Feike (Feike 2007). A wonderful mail class disposing every possible methods for a professional dynamic email generation. The mail class is available under the GNU General Public License and is OpenSource software.

7.4.6 Multi language use

As our project should be as modular as possible, a multilingual organization is a must. To meet tourism information needs, a service in several languages has to be possible. To realize a multilingual project, we decided to implement a separate language interface from the beginning. All labels and captions need to be language variables outsourced in a language script containing all the labels in every available language. Even if this procedure gives a lot of nebulosity into the codes and makes them much more difficult to read, it is worth to implement a multilingual interface from the first codings. So you clearly think about variable names and labeling. An implementation afterwards is arduous and more aggravating sometimes the whole code must be changed to realize the multilingual possibilities.
Concrete, our approach is leaned on the methods used within Typo3. All the labels are stored in an multidimensional associative array. The array is divided into subarrays for every language. A default language is defined which is applied everytime the language selection over user input is not active. Every subarray contains the labels for the defined language associated with exactly the same denomination. When including the language script into a page, the subarray for the correct language is stored always into the same variable named `$lang` (Typo3 2007). This allows to select the correct subarray with the desired language at the very beginning of each file and then using just the standard associations to get the correct labels echoed. For example in the code below the English subarray is chosen and stored into the `$lang` variable. In the next line, the use of the labels with it's standardized associative pointers is echoed.

```
if ($langID == 1){
    $langcode = "en";}
$lang = $LOCAL_LANG[$langcode]; ...
'.$lang['mapH1'].' <p>'.$lang['mapIntro'].'</p>
```

Because the tool is prepared for the use in eastern Switzerland, default language is German. For purposes of demonstration the tool is fully implemented for English speaking users and German speaking users.

7.4.7 Outsourcing of constants

It is necessary to have the most important constants defining colors, DB identifiers or layout externalized in one single constants file if we want to implement the tool totally barrierfree. The system should be adaptable without much effort to different website layouts. If we didn't externalize all possible constants, we would lose much time finding the constants in all files. Further more we have several files containing related contents (think about maps, prints, aerial images etc.) where many constants are used several times. Changing them to adjust a website would cause much redundant work.

We store all important constants in a separate file which serves as interface between appearance and implementation. The file is similarly built as the language file. A multidimensional array stores the constants. With an inclusion at the very beginning of each file, the constants are prepared to be allocatable during the whole execution. To print an example for the language and constant file, the constant file is fully listed in the appendix in C.9.7. The script heading in C.9.6 for example shows the selection of the constants and the language allocation. On the print, the file's structure with the multidimensional array is well observable.

8 Implementation of the Web-Interface

8.1 Connection between topology and visualized output

The system's modular build up demands for a good strategy to communicate between single parts. PHP offers good possibilities to exchange data from one file to another either by it's concept of sessions or directly through request variables annexed to URL's. Both concepts are utilized within our project.

Generally all data with coherence to the user's path are stored in a session on the client side. With every new visit on our site, PHP installs a session on the client side. During the application of the hike planning tool, data to origin, destination, vias and cost factors are stored into the session variable. Because it is memorized on the client, the server location can change and through newly querying the session, all data is still available. An arduous buffering in a database disappears and saves a lot of resources. The session variables are active as long as the session isn't renewed or deleted or the session is automatically reset due to time lapse.
After path calculation via Dijkstra's algorithm the session is updated with the resulting array. This allows to shift the result from the calculation scripts to the visualizing scripts without any difficulties and long buffering procedures. If we can not fall back on an active section and still need information to a user information, data is directly transmitted by attachment on the URL. With request variables this data can be shifted without active session.[6] We come across an ideal data transmission, if we for example look at the case where we want to transmit the history ID with the information about the path by a link in an invitation email.
Tests have shown that within the use of asynchronous techniques, data transmission directly via URL's is safer. Where the session query failed sometimes, the shift via the called URL is 100 percent safe.

8.2 User form

The core content which rules user input and transfers it to manipulation is a graphically simple user form on the welcome page. In the flow chart fig. 19 the whole first part is managed by the user form. All data interactions between user and system has to be guided through the script in the background of the form.
There are three possibilities foresight how the user can input his preferred hike parameters: (i) he writes an existing location with signpost into the form, (ii) he gives in a search word like a municipality's name or a location name and the script assumes a set of possible matches, (iii) the user can look up the existing locations directly on a map allowing a choice by mouse click.

It is supposed that not many users know exact location names, as they are introduced into our topology. Therefore it is necessary to help the user find it's diesired location and input it to the form. The user normally knows the municipality or a key word for a location where the search can be initialized. When he gives in such a search word and clicks on "matching propositions", the user form executes a fuzzy query on the database to search best matching locations. The search computes all possible results by a fuzzy match on municipalities or existing locations. MySQL supports techniques for fuzzy logic. With the `MATCH() AGAINST()` construct, a query becomes fuzzy and results are not only found within hard search borders anymore. To apply the fuzzy search, a table index over the columns `n_location` and `n_municiplaity` in the table `N_NODES` has to be initialized first which allows a fuzzy comparison between search word and table entries and further speedup the queries (MySQL 2007). The complex query to find the best propositions is quoted in lines 169 to 177 in appendix C.9.1.

[6] A request variable can be attached to every URL. In the form http://www.abc.ch?REQVARname = REQVARvalue&REQVAR2name = REQVAR2value as many variables can be directly transmitted to a new site defined with the URL. In the new location, a PHP script can demand for the values stored in the request variables.

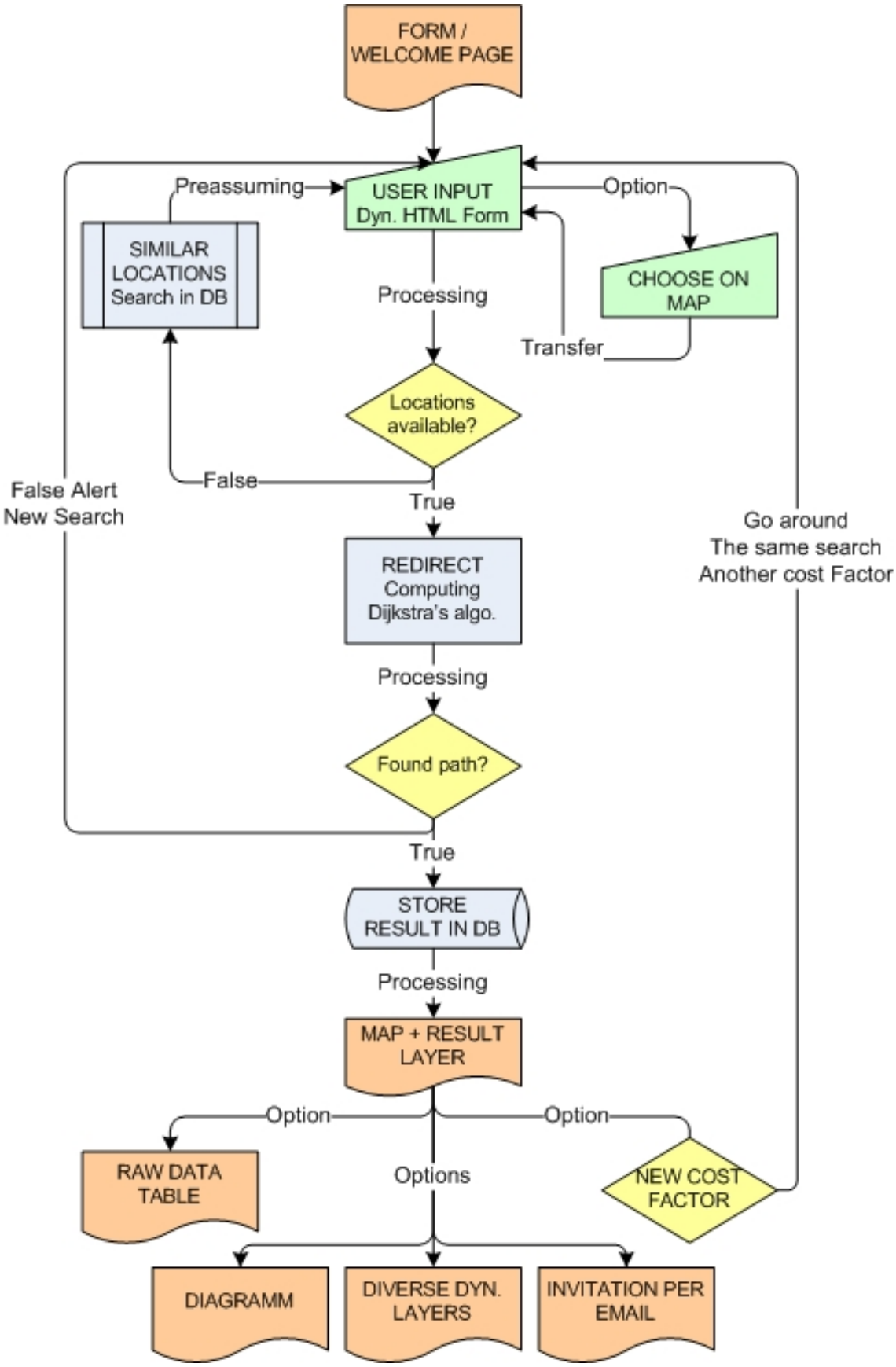

Figure 19: *The flow chart gives an overview over the processing in the user interface and visualizes the possible steps during an user session. Two major components are data input and data output possibilities.*

The propositions are then presented in a dropdown menu from which the user can choose his favorite.

Sometimes we know a location not by name but by spatial placement. To implement a user interface that allows direct location search on a map, a subscript builds up the corresponding map to choose from. When the user fills in a municipality name or another search word and then pushes the button "search on map", the subscript is executed. To keep rapid build up without heavy layer data, the database is looked up for the search string in a first step. When it is found (a municipality or an existing location) the layer with the interactive node names is limited to a surrounding area of 6000 meters to the center of gravity. Assuming the user has given in the name of a municipality, he would like to choose the point of his departure from an existing location within the given community. It is not necessary to print all location names reaching far behind the area defined by the commune so that 6000 meters is suitable and surely contains all the desired possible locations. A new layer script provides all these calculations and prints exactly the location names and placements lying in the described perimeter (script quoted in C.9.2). The layer computes the data in the way that features become interactive and linked with the main user input script. When clicking on a name or location node, the ID is transmitted via URL to the user input script and there automatically filled in the input form. The focus in the map is initialized by default to the center of gravity of the searching results. For example for a municipality, all matching nodes are taken into account, the average x- and y-coordinate are computed and then the map's focus is set to this center of gravity (cf. processing line 51-67 app. C.9.4).
If the search string doesn't match the topology's entries, the search on the map is widened to the whole existing map perimeter. In this case another layer is loaded by default, computing every existing node name and placement with the same linkage as described above. Now there is a choice of all locations and the navigation to the interesting area has to be done manually (cf. layer script in C.9.3).

Of course, for the search on the map, additional layers for better orientation can be loaded asynchronously into the map. Supported layers can add the hiking network and communal borders and labels.

Finally, when at least locations for origin and destination are found in the database, the calculate functions becomes active. A "calculate route" button is automatically activated above and below the form. As soon as the button is clicked, the calculation process is triggered by the form script and user input has finished for now.

8.3 Manoeuvring between the single parts

Figure 19 symbolizes all possible output options at the bottom of the chart. A navigation possibility between the different output options has to be given to the user. Providing a menu that allows to switch from any possible output directly to another gives the upmost freedom to the borrower. We reach this functionality by placing a small link-menu in each header of a single output option. Through the menu, all other options can be directly loaded.

As the output is calibrated to fetch any homepage layout, a function opening the output in a new independent window with full enlargement should also be given to the user.

A print function must be implemented, too. If someone likes to print the map, diagram or detailed information table, he would like to have an adapted output that matches the needs for good quality printing without unwanted information.

8.3.1 Embedding SVG graphics

Several methods to embed SVG graphics into html pages are proposed in literature. Where some sources swear on the application of the `<embed>`-tag in html other propose the use of the `<object>`-tag (Ueberschär 2006, Meinike 2006, Eisenberg 2002, Carto:Net 2007). Nevertheless, both of them failed when using Internet Explorer 7.0. For us, the best way to embed SVG graphics into html is to *not* embed

them at all. Any complicated and fragile interaction can be avoided this way. We recommend a strict division of html pages and SVG pages. This can be realized through the use of frames or better inline-frames. Every common browser nowadays supports inline frames which allow the use of this concept (Selfhtml 2007).
Our web interface is built up with the same strategy for every output option: a main frame with header menu and footer includes an inline frame which loads the output option file, unimportant wether if SVG graphic or simple html output. With this implementation the SVG files can be loaded as proper SVG files without any embedding and interaction in a surrounding html file. To get an idea, how we realize the web interface, appendix C.9.5 quotes the frameset to present the page with the SVG map included in an inline frame.

8.3.2 Adaption for prints

Basically the adaption for prints includes some simple decolorization. When the printer icon is clicked, the actual output option is loaded into a new window. If the output contains an SVG graphic, the background colors are set to white which doesn't disturb the printing output. In the new window, the graphic is automatically enlarged to the full window extension meaning there is no absolute width value for the graphic. This allows the printers to fit the graphic automatically to 100% of the possible paper width. For html outputs, a JavaScript function `print()` is initialized and loaded per default. The function automatically opens the print dialog. In contrast to the html output, the JavaScript function does not work within SVG files. To print the graphics, the user still has to ask for printing manually.

8.4 A first example

To adhere the results until now, the following print screens give an idea how the tool looks like and how the described interactions between system and user are implemented. The data output, diagram and map in both modes, aerial image and vector graphic, are printed as examples.

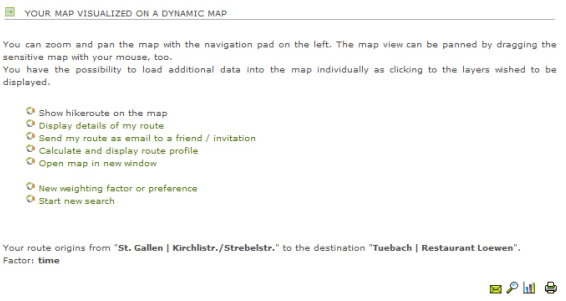

Figure 20: *The header menu allowing page navigation from one to another output. The header includes the general way description and the possibilities to print and enlarge the actual output.*

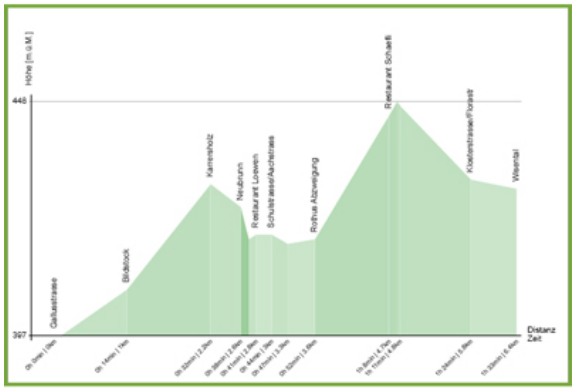

Figure 21: *The diagram for the calculated path.*

Figure 22: *Listing of the detailed data for the resulting route.*

Figure 23: *Vector map output with parts of the navigation and layer menu.*

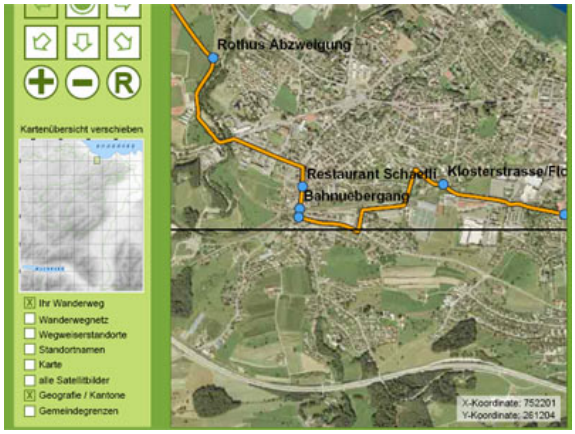

Figure 24: *The map with navigation parts in aerial image mode.*

9 Tests and Improvements

The third part is reserved for tests, improvements, conclusion and further ideas.

9.1 Description of the test phase

Within the test phase the system is analyzed regarding disfunctions and possible improvements. The tests are divided into server side testing and testings on the client side, which includes usability and user interactions.

9.1.1 Server infrastructure

To look at the server when it is really loaded with data, calculations and several requests reaching the domain per second, a test script simulates the test run. It implements via PHP a sample request that is sent to the server every 200 milliseconds. The request includes a query as it is realized by user input via the user form on the welcome page. So every request triggers a calculation on the topography and results in a dynamically built map.
Parallel to the script's execution, the server is manually requested with user input. The duration until the server responses is stored in a table to be available for further manipulation.

The comparison between the response time when the server is additionally loaded with the requests sent by the script and a single manual request give an idea, how the server infrastructure reacts on heavy traffic. We have to consider the fact that we can not change server installation because we are just requesting to leased server space. Therefore, we do not have the possibilities to adapt the server to our needs. What we can test is the functioning, when the server deals with heavy traffic, as it would appear in real use.

When comparing the results, we consider a slower manipulation when the script is running of around 35%. Assuming that 5 requests are sent to the server per second, the response time and the map's generation retards of one third. Now discussing the results that are completely printed in table 11 in annex A.2, we can point out the following facts:

- The server can deal with five heavy requests per second. These are the most loaded requests because Dijkstra's algorithm is applied and the accesses to the database are heavy. If a user enters the tool, he requests such a calculation not more frequently than about every 30 seconds. Normally he analyzes the result carefully and looks up other information as diagram or route details. These requests are not comparable with the calculation step, because generation and data transfer need much less time.

- The server response and waiting time for the user widen with server load. This is a common server reaction. Several parallel queries slow down the execution velocity. With five requests per second server velocity slows down by 35%.

- As long as the system isn't used for interregional purposes the server capacity should be able to deal with the possible traffic.

9.1.2 Tests on client side

We have tested all functions and interactions on as many systems as possible. Through these large tests we have recognized the system's demerits and interfaces where improvement is necessary. The most aggravating factors are the different browser implementations of the SVG and AJAX standard, as we have seen during the development. Therefore we have deemed as very important to test the hiking tool on every system available without too much effort. The different client systems where tests have been made, are illustrated as follows. The listings contain the accurate client description followed by the most distinctive observations and an attempt at an explanation.

Opera 9.24, Win Vista There are no bugs with Opera. Opera is known to natively support SVG very well (Code Dread 2007).

IE 7.0, Win Vista, ASVGV [7] There are no disfunctions recognizable.

IE 7.0, Win XP, ASVGV There are no disfunctions recognizable.

IE 6.0, Win XP, ASVGV There are no disfunctions recognizable.

IE all Versions, without ASVGV SVG data is not printed at all.
⇒ No native SVG support in all versions of Internet Explorer.

Safari 3.0.4, Win Vista Apple opened it's Safari browser lately to all Windows Operating Systems. It is not possible to activate layers through AJAX. Therefore all maps are impractical. The SVG diagram is perfectly visualized. Otherwise Safari implements SVG nearly as well as Adobe's SVG Viewer. All calculations are really fluent and fast. The navigation reacts rapidly.
⇒ Unfortunately there is no AJAX support with Safari. We believe a conflict lying in the use of the AJAX methods `getURL()` and `XMLHttpRequest()`. Tests with external SVG files (taken from Ueberschär 2006) showed the same attitude.

Safari 3.0.4, Mac OS X The layers are not loaded at all. Diagram and remaining SVG data are correctly echoed. All mutations are not printed correctly.
⇒ Safari does not support AJAX. There are conflicts with mutations (ä,ö,ü) which can not be echoed by Safari.

Firefox 2.0.0.9, Mac OS X All layers can be loaded but without fonts and text contents. Paths, circles and polygons are correctly represented. Otherwise the font is slightly distorted. The profile is also echoed without labels. Navigation and coordinate prints are all right.
⇒ Probably the browser's standard font is different from the usual standard font as used in our SVG files.

[7] ASVGV: Adobe SVG Viewer

Firefox 2.0.0.10, Win Vista There are no disfunctions recognizable.

Firefox 2.0.0.1, Win XP There are no disfunctions recognizable.

Firefox 2.0.0.10, Win XP, ASVGV :
The coordinate display, all layers and navigation is jammed. It is not possible to load data into the map. The profile is correctly visualized.
⇒ ASVGV and native SVG implementation by Mozilla conflict. It is possible to disable native SVG support in Firefox. But the disfunction remains because the AJAX machine from ASVGV arguably uses a different AJAX implementation from standard Firefox.

Firefox 1.1, Win 98 There are no disfunctions recognizable.

Firefox 1.0.5, Win XP Neither map nor diagram can be printed like this. Detailed data and email function are available.
⇒ There is no native SVG support in this old version of Mozilla's Firefox. Firefox supports SVG natively starting from version 1.1 (Golem 2007).

Firefox 2.0.0.1, Win 2000 There are no disfunctions recognizable.

Concluding the results above, we can generally say that all browsers support the implementation. There are problems due to different AJAX implementations and different operating systems (Macintosh, PC). The worst conclusion is that with Mac's standard browser Safari the implementation fails and the hiking tool is almost not usable. Many potential clients are therefore excluded from the hiking tool's use which has to be changed. Searching for the causes we viewed different external SVG files containing AJAX functionalities in the Safari browser and all of them failed, too. Unfortunately the Internet community is not as highly interested in Macintosh Software so that there are no plausible explanations found online.

During the test phase on the client side we ran more than 430 queries on the topology. Many external test person checked the tool. Observations of people's handling and application have given interesting interfaces where the system still has to be improved. Usability, functionality and content as well as comprehension have been checked intensively. Several small bugs appeared during these tests:

- Mutation letters are not always printed in the SVG files
- When returning to the user form and asking for a calculation with same locations but different cost factor the user input is not updating the session variable transferring the cost factor into Dijkstra's calculation
- Font sizes that are defined by points are variably printed in different browsers. There is the possibility to enlarge browser fonts in the browser itself. When a client has activated an enlargement, the fonts in points are distorted and do not match into the panels anymore.
- If Dijkstra's calculation fails, the error message is not computed. Instead of it the user is redirected to an empty map
- Font sizes are chosen too small. Several test persons bemoan the illegible labels in the panel.
- There is a transfer problem from the user form to the map from which the user can choose it's location. To set the focus, the script generating the map needs the user's entry transmitted. The session variable containing this input sometimes does not actualize.

In the following chapter the improvements solving all these problems are discussed. The test phase on client side has given a good idea of the system's use in real application.

9.2 Improvements

To improve the hiking tool and solve all existing problems we have to invest some time into different adaption. The most important adaption brings the solution for Safari browser. As we do not want to exclude Macintosh clients, we have to think about avoiding AJAX components in Safari browsers. When avoiding AJAX we clearly have to remove all data load into the map after the initial loading process. What we can do is to load all necessary data directly on load, meaning all the layers that we want to make disposable have to be included into the main map generating script. Further a fall differentiation has to direct the different script according to the browser in use.
We have decided to implement another map generation script specially for Safari browsers. It's differences lie in the layer management: There is no layer menu available except the link changing to mode with underlying aerial images. With the map, the essential layers are included in the main map script: resulting path and environmental vector data are loaded per default. In aerial image mode, the data for the resulted path and the environmental vector data is included, too. Further the user form is limited to textual interaction only. For Safari clients the map to choose the locations from are not accessible anymore. We have to update the existing files with the fall differentiation. It is included in the user form to limit the form functionality in case of Safari and in Dijkstra's calculation file because this redirects to the map file and further in all files where we can access the map (from header menus).
Knowing that Safari users are restricted by this reaction, we believe that this solution is the best improvement so far. Safari clients are no more excluded and can access all data with just small quality loss for the SVG maps. There is no better advice for solutions available. Further more we do not have proofs for where an existing dynamically generating SVG tool implements AJAX for Safari browsers. Starting points to solve the variable transmission problems through session variables are at first sight not available. Several tests where the session variables are output and checked during computation failed. Not before reading an interesting advice written by an Inter-

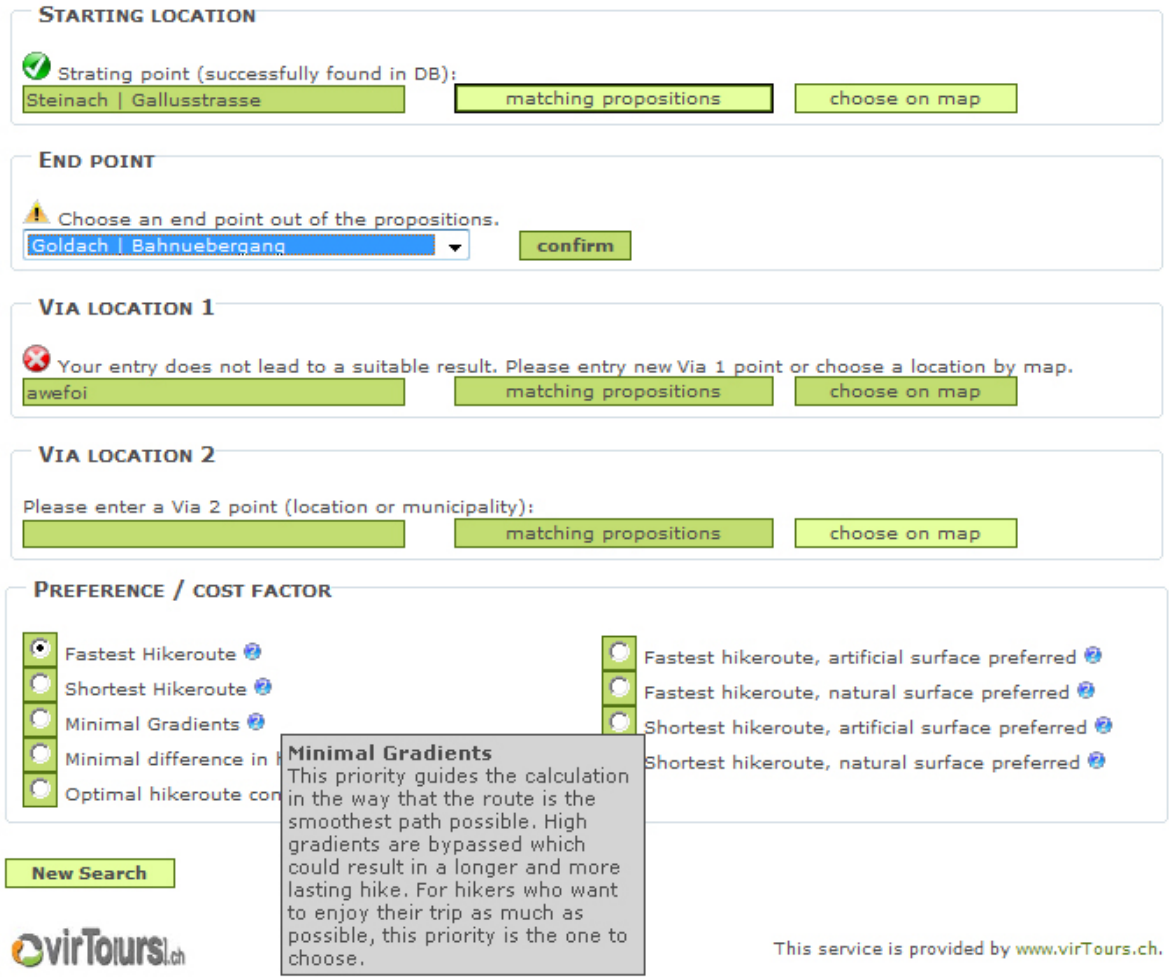

Figure 25: *Improved user form. Supporting icons to the form labels and excluding of non-used buttons during the input procedure and more exact labels are alleviating the use. Major improvement are the further descriptions for each cost factor in a mouse-sensitive tooltip (in this figure active for the cost factor "gradient").*

net community (Entwickler Blog 2007). Referring to the article, session variables specially behave with the use of the `header`-command for redirections. If we redirect in a script to another, the session variables are not actualized per default. The session variables are first written, when the script has totally finished. This is not the case when redirected to other scripts. The solution to the problem is the explicit command `session_write_close()` to write down the session variables. With this command included in all scripts using the `header` function for redirection the data transmission works correctly.

Further all fonts are adapted to absolute font sizes given in viewport units. Like this browser influences to font sizes can be reduced. Generally the labels are enlarged and more space is built to give enough possibilities to adapt font size to a minimal limit readable by everyone. The language file containing all labels is filtered for mutational letters which are directly replaced to correct html encodings. An improved exception handling is introduced to the calculation file . If a calculation fails, the user gets an error message containing possible solutions to its request and a links directing to the user form for his input adaption.

Tests from the responsible person of the hiking section and his team are resulting in several additional improvements (Styger 2008). The user input form is now more intuitively applicable. The comments to the form fields are supported with small icons, labels on buttons are clarified and more conspicuous colors are used for the most important mouse interactions. Not active buttons are excluded during the input procedure. A major improvement for the user is the inclusion of detailed explanations to each of the cost factors in the form. With tooltips for each factor every preference can be studied more carefully. The tooltips are dynamically processed with regard to the active language. So even this detail is implemented for a multi-language use. Fig. 25 shows the input form in it's definitive look. As the pick of a location for the hike from the map is the most important way for hikers to choose their locations, the map itself is improved. For better orientation the terrain is included per default. Standard zoom level is higher than before to guarantee readable labels from the beginning. Also the traffic layer is loaded per default for better orientation on the map.

After all these improvements we finish the work on the tool. There are no more possible improvements to be done lying within our capabilities. One last step remains: the official lineup.

9.3 Lineup

By now, tests have always referred to the same website and URL `virtours.ch/hiking`. To widen tests we want to apply the tool to another homepage with different layout and different users. Further more, we would like to test the system in real use. Having the possibility to advertise the tool on an administration site would be the ideal concept to enlarge tests.
Therefore, the municipality administration Tübach was asked to get the right to publish the hiking tool on their official website, too. With the single restriction that publishing has to be free and without any cost for them, the mayor accepted the request.
In a next step all files had to be transmitted to the municipality server. We left all the files as they are and did not modify them before transmitting. Like this, we could directly control the internal links and references which should all be coded with relative paths and still be correctly working within a different URL, too. For the first test, we did not adapt any constant, either.

A first visit shows pleasant results: all the links are still working, navigation and layer management is correctly supported, queries and calculations are executed without problems. We can conclude that the full functionality can easily be transferred to another server. With deeper testings a bug in the mail functionality is uncovered. It is the only file, where we have not totally considered relative paths. The dynamic linkage to the preferred path is

not adapted to the new URL of the municipality server. The problem is erased by placing the URL in the constant file, too. Where all adaptable constants are listed.

Figure 26: *The main map adapted to the new environment.*

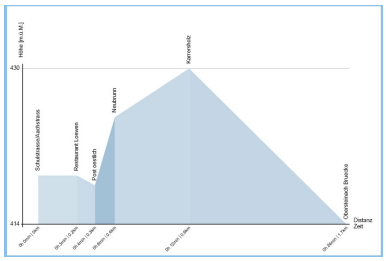

Figure 27: *The diagram's color adapted to communal website's design.*

Finally the tool has to demonstrate its capacities to adapt to different layouts. Figures 26 and 27 give an idea of the new layout. The color constants and layout definitions are being adapted to the new surrounding website components and then refreshed on the server. We have to mention that with the first tests, we have recognized that several small features in our tool are not involved into the color and layout adaption. To get the full control over color and layout in the constant's file, all script passages not involved yet, are inserted and linked with the constant's file. After some small changing procedures in the scripts, the final output fully matches the new website (cf. overview fig. 28 with main map). It can be directly accessed by the link `http://tuebach.ch/t3/index.php?id=357`.

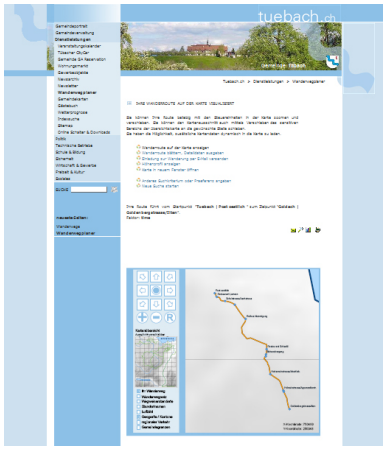

Figure 28: *A cut displaying the whole website with the main Map included and fully adapted to the corporate design.*

To adapt the tool even more to the needs of the community the user form can be changed in the manner that in the user input field where departure locations are input, the default entry corresponds to the community. This is implemented to the user form by allocating the community name into the input field in the cases where no other departure is not yet introduced.
All the transferring and adaption work took less than 30 minutes. For us, this time approves a sophisticated implementation in the borders of our qualification.

In agreement with the municipality administration the population's attention to the hiking planer is called by an advertisement in the municipality bulletin. This measure should attract people to visit the tool and experiment with it. We hope for hits on the tool to simulate real adaptation for a longer term. Perhaps people help to spread the tool by linking it to friends via the email function. There might be improvement suggestions by the inhabitants when they individually use the tool.

Now, as the hiking tool is running on two different servers and is included into two totally different frameworks, we can conclude the testing phase. The system would be now matured and ready for use in every web based context.

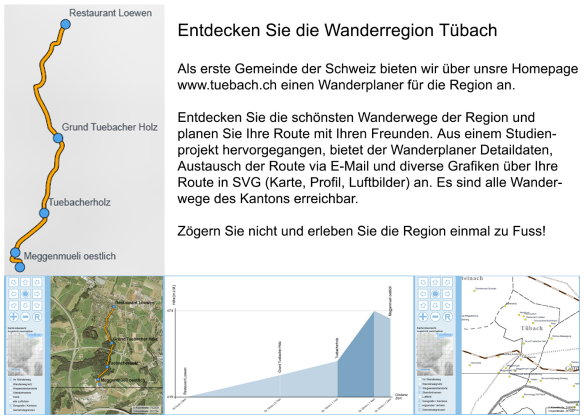

Figure 29: *Advertisement in a communal bulletin to attract people and call attention to the hiking planer. The insertion is published 12-12-2007 and reaches all inhabitants of Tübach.*

10 Evaluation, Conclusion and Outlook

This last chapter summarizes the conclusions concerning the thesis' project. Whereas the evaluation is more personal, the conclusion should give a most objective appraisal. An outlook into the future should finally complete the chapter giving ideas of possible further developments of the hiking system and, more general, trends in interactive mapping technologies or cartography and SVG for the future.

10.1 Evaluation of the whole project

During the long period of working on this thesis and the intensive addiction to the topic it was possible to learn a totally new trade. The progressive theme with GIS technologies, SVG and interactive mapping was challenging and fascinating at the same time.

Challenging because official resources or literature and professional help are rare due to the stage of development and the quite new technology. But it was also challenging because different developers follow their own strategies to reach the same goal of implementing an open standard differentially. Personally I can not understand why an open standard as SVG is implemented and supported in as many ways as there are developers. Each browser differentially supports SVG visualization. Heeding all possible singularities during each visualization step debilitated and strongly aggravated the development work. It is an ambivalent task to use an open standard to implement a GIS tool and bear in mind all singularities for all possible browsers. I am sure that SVG will get it's future soon and alternates older languages as for example `Flash`. SVG is perfectly designed to support the Geographer's wishes. The work with SVG as an interactive mapping language showed all the advantages (cf. cartography and signatures, vector based visualization, AJAX, dynamic content generation, different viewports and coordinate systems) and the huge number of possible applications and interactive exchange. GIS applications and the ongoing trend towards web based systems will help SVG to a breakthrough. This will happen only with one restriction: the standard has to be implemented and supported similarly on each possible system. Only when developers are not restricted by diverging SVG support new implementations based on the open standard are affordable by the companies.
Another challenging part, too, was settled in the domain where Geography meets Computer Science. All the good ideas from a geographic point of view had to be translated into a formal informatics language. It is not sufficient to think only theoretically of a theme - it is essential to never forget to think about possible realization and consider eventual obstacles barring the way to go. Especially for Geography as a wide ranging domain, the interfaces between other disciplines and Geography are, respecting the experiences made during this project, key features to learn more about.
Another challenge was of course the more administrative part of such a project: advertising for the personal idea and concept, looking for possible contacts and acquiring data from official administrative organizations, projecting the own progress and of course hanging on to achieve the set milestones even in times when things are not going as fluently as desired.
The writing in English has caused more difficulties than expected. Explaining complicated issues not in one's mother tongue is a difficult task and sometimes it would be easier to describe procedures in one's mother tongue. But there is also an advantage when writing in a foreign language: we have to think more about formulating, which enables us to describe facts more precisely. Generally the challenge with the language was obvious but has helped to brush up the English language.

The project was fascinating and motivating in several ways. Choosing the topic of the thesis I joined two major personal interests. The Geographic sphere as well as the more abstract Computer Science sphere were united. Even if the Geographic aspects do not come to the fore, they were omnipresent and comprise

the backbone of the whole thesis as GIS and topology, distance modeling and Cartography, thinking about advertising hiking routs, developing the heuristic based on qualitative ideas how a hiker is bearing, hundreds of coordination transformations, different coordinate systems, calculation on maps etc. The Computer Science's domain plays it's role in the final realization. It is more a means to an end rather than the main topic for the thesis. A complete realization from raw data to final output would not have been possible without a Geographers wide view. Normally a Computer Scientist just accomplishes missions that are already totally planned. Managing a whole project with all its aspects from looking for contacts, planning and thinking about ideas, working through qualitative theories and not stopping at this point motivated me much. Reaching further than a plain theorist does opened new dimensions. Ideas have to be realizable within certain resources if we want them to come real.

When I above mentioned the lack of literature and professional help as challenge I may have to mention the informal help that is given by Internet community. There are so many really good portals and forums where you can post your question or find existing solutions. Even for SVG and all the used new technologies there are plenty of volunteers that have made experiences with the topic and share them with other developers. They are even ready to help an anonymous person with a very individual problem for free. This kind of information transfer could substitute professional literature in the near future especially for progressive themes. Thanks to the Internet community the realization got to what it is actually.

Internet technologies provide a huge range of functionalities and possible interactive interaction between client and server. Almost every wish can be programmed anyhow. It fascinates me to implement useful functions for interactive mapping going far beyond the possibilities with printed maps. For Geography the new technologies are a big chance to present the domain in a proper light and advertise for it. As Geography functions as an interface between natural sciences and human sciences it is predestinated to work with the interactive level, which aims to integrate qualitative user input into quantitative digital processing. If qualitative data is made visible on the Internet with quantitative technologies it establishes and will establish a new business for future Geographers.

Finally the concrete goal of implementing an interactive hike planning tool, which clearly is not yet realized in Switzerland, motivated me to keep on working on the theme. Developing something that is not yet developed and that can be finally presented in a way everyone can understand additionally motivated me. Keep on working to build something you finally have in your hands is much easier for me than working on a topic without concrete practical output. During the whole project motivation could be held because a concrete goal was in view. Therefore, I can affirm that the work fascinated me from the beginning to the end and my personal attitude towards the project has always been very positive.

10.2 Conclusions

With the end of the thesis we can now come back to the initial questions. The experiences during the thesis allow us to give importance to our answers.

Concerning distance modeling we would like to bring up several statements. We can clearly confirm that with the spartan raw data available it is possible to create a functioning topology on an external server within a usual MySQL database. But to build up the topology many compilation steps and reconsidered transformations are necessary. The topology needs to be created manually in a way where the different compilers that fill the database tables and connect the topological features have to be coded. To really make the topology usable for hiking purposes we need to complement nonexisting data within approximative calculations. Anyhow, we are surprised at the variety of possible cost factors that have been made available with the meagre data basis. Dijkstra's algorithm suits well for our purpose

and correctly calculate results on our topology. It also suits in the view of graph theory because the hiking network fits in the restrictions for Dijkstra's algorithm that dictate a non-negative weighted, directed graph. We implemented the algorithm in object-oriented PHP. The resulting runtime with the raw implementation is too weak to be applied to the Internet. Therefore this domain had to be manually adapted, too. With the introduction of a pseudo-heuristic based on geographical thoughts, the runtime could be advanced by about 35%. The final implementation delivers results in acceptable time but we are aware of producing faster results with an implementation in another, more machine based programming language as C for example. The constants defining limitations and calculation ranges on our topology are to be carefully deliberated with respect to geographical facts.

Concerning interactive mapping we have to conclude in a more ambivalent way. The interactive mapping with SVG provides new and progressive possibilities for Geographers on the one hand. Especially in cartography SVG plays off its advantages. Signatures, vectorial features, interactive functions as zoom, pan, coordinate display or dynamic layer management in combination with AJAX technologies open totally new perspectives in digital cartography. In this area SVG beats today's standards. On the other hand, we have to deal with a strong disadvantage. Although SVG is an open XML standard, its implementation and support in the various browsers extremely differs. This fact is fatal for the future development and progression of SVG because programming gets difficult and arduous. Considering all possible exceptions and programming for every individual SVG support a matching implementation finally reduces economic benefit and so reduces the interest to enforce the standard. To defend SVG's potential it has to be added that SVG was not only developed for Internet use but simply as a new standard language for digital mapping. If we don't depend on browser support SVG perfectly matches the need of cartographers and programmers. We suppose the SVG future to be just bloomy if the browser developers agree on a predefined, standardized SVG support in their software, too. If not, obstacles will remain too big and other, individual and not open standards will be favored.
Whatever, the thesis clearly showed the possibilities that are open to cartographers within interactive mapping and new technologies. The advantages of interactive maps in comparison with printed maps are obvious: no more page turning, no folding, easy search, individual level of detail, interactive functionality and therefore personalized applications. Printing just the map cleavage that contains our perimeter in focus still allows to physically have a map in hand. We are confident that interactive mapping will alternate printed maps in the near future. This won't be the end of the cartographer's guild - on the contrary - the oldest trade of a geographer, the measurement and mapping, will get a revival that has already started with booming applications Google and others provide.

To answer the question, if we are able to develop something not existing yet and how this thesis contributes to understand modern techniques for interactive mapping, we suggest the following statements. Of course we cannot reach the big players in interactive mapping like Google or Search.ch. With a huge budget and a team containing hundreds of professionals, they can provide faster searches on their own servers and infrastructure, they can fall back on a more detailed data basis and are able to consider small trifles more carefully. What we reached is a niche-product providing nevertheless new and unique functionalities: the hiking network is not available in Google nor in Search.ch. The hiking network differs from a route network which can be categorized into highways, arterial roads and so gives approaches for faster searches. The hiking network consists of one single, equal category. We provide an interface allowing the user to broadcast his route to friends and invite them for a hike. A social component that advertises for hiking and indirectly advertises for the hiking region is implemented. Our system does not contain any advertisement nor is it

financed or pushed by secondary parties. The system is built barrier free and can be adapted easily to every individual homepage or corporate design. An introduction into an existing web portal which is not for the big players is still possible. Additionally we display coordinates real time and finally, we us a progressive, new standard. Even new releases and projects announced in newspapers and magazines do not enter the niche of the hiking tool (Tagblatt 2007, Tages-Anzeiger 2007).
With the developed approach we have entered new territory. We assume the biggest benefit for future developers lies in the domains where Geography meets Computer Science. Adjusting distance modeling to the needs of hikers, generating SVG dynamically out of PHP, combining topological theories, geographical thoughts and concepts with programming procedures need a lot of interfacial thinking. Above all in the hiking domain, where we assume to find much economic potential for virtual geographical systems, we also meet the biggest benefits of this thesis.

If a use in reality is regarded as possible alternative more improvements could be done. Mainly runtime and content attributes depending on zoom level (larger font size when zooming out etc.) could be enforced to ameliorate the system. Further there are huge possibilities to expand the tool and make it economically more interesting. The next section tries to give an outlook into future developments and possible extensions.

10.3 Outlook into the future

We recognize several interfaces in our project, where extensions could be added. As available data basis is limited to geometric data, a widening to qualitative data could be reached. Providing layers showing restaurants, points of interest or timetables of aerial cableways would enlarge services to a great extend. Thinking about the possibility to accept advertising on the map could increase economic benefits. For example a tourist attraction could advertise for it's programme and could dispose additional information directly on the map for a small due. This service could be programmed in a way that the client is able to introduce his advertisement totally autonomous. Especially for touristic regions the service could be enlarged to other subjects apart from hiking trails. It is cogitable to enlarge the topology to subjects like ski slopes, cross country ski tracks, jogging tracks. The concepts would not change for these subjects. The network and topology could also be upgraded with public transport being included in the calculations.

The concept how the zoom level is introduced into the dynamic generation of content could be broadened. Making layers containing labels and fonts depending on zoom level would help to structure the maps in a more sophisticated way. There is the possibility to decrease runtime with a new implementation of the algorithms in a faster programming language. Probably there is still potential in our pseudo-heuristic to imitate the A*-Algorithm even more carefully.

At this point we have to refer to an extension that is still implemented into the hiking planer within March 2008. The planing tool has been widened with a GPS Export module. With this extension the user is able to export his preferred hike in all common GPS file formats. An interface to famous GPS softwares as “Google Earth”, “Google Maps” or “SwissMap 25” is developed, too (Sieber 2008).

We see another possibility where our system could be enlarged. The Swiss hiking association is looking for more data about hikers and hiking trails. The association would like to survey numbers giving an idea of use of certain hiking trails. This data has a great importance. With numbers dealing with frequency of use the importance of the hiking trail can be deduced (Schweizer Wanderwege 2007b). Out of it the association could graduate the amount of maintenance necessary for a trail or which hiking routes need to be looked after more carefully. With the hiking tool, such data is already stored (each calculated path is stored): it is an easy extension to query the database in a form which gives out this data. On the side of the project, the outlook into the

future consists more or less of such possibilities.

On the side of technologies we have to wait, how SVG and browser producers will carry on. Information saying Adobe will abolish support for it's Adobe SVG Viewer by beginning of 2009. Their argumentation is that there are still enough browsers on the market with native SVG support. Assuming this information to be true (Adobe 2007c), we can only hope Microsoft's newest release of Internet Explorer will support SVG. Trusting sources close to Microsoft there are many indications alleging that still the next version, being IE 8, will natively support the open SVG standard. The release is planned for the autumn 2008 (IE Blog 2007). This news propitiate us for an ongoing and development of the SVG standard. We believe the SVG standard to play an important role for future geographers and, therefore, a wider propagation and a reduction of obstacles would indirectly support the geographer's guild in a sustainable manner.

References

[•] **Text sources:**

[1] Bartelme, N. (1995): *Geoinformatik - Modelle, Strukturen, Funktionen.* Berlin: Springer Verlag.

[2] Bernreuther M. (2000): *Kürzester-Weg-Suche mit dem Dijkstra-Algorithmus.* Stuttgart: Universitätsverlag.

[3] Bondy J.A. and Murty U. (1976): *Graph Theory with applications.* New York: Elsevier Science Publishing Inc.

[4] Collet C. (2007): *Methods for Environmental Analysis: Distance Modelling.* Course material from Master Course UniFR. Fribourg: Departement of Geosciences - Geography, Univerity of Fribourg.

[5] Diestel R. (2005): *Graph Theory.* Heidelberg: Springer Verlag.

[6] Dijkstra E.W. (1959): *A note on two problems in connexion with graphs.* in: Numerische Mathematik. 1 (1959), p. 269-271.

[7] Dransch D. (1997): *Computer-Animation in der Kartographie: Theorie und Praxis.* Berlin: Springer Verlag.

[8] Eisenberg J. D. (2002): *SVG Essentials.* Sebastopol: O'Reilly & Associates Inc.

[9] Goad C. (2002): *Flash/SWF for GIS.* Winnetka: Directions Media.

[10] Gross J. and Yellen J. (1999): *Graph Theory and its Applications.* Boca Raton: CRC Press LLC.

[11] Hartmann A. and Weigt M. (2006): *Phase Transitions in Combinatorial Optimization Problems.* Weinheim: Wiley-VCH Verlag GmbH & Co.

[12] Kantonal st.gallische Wanderwege (2002): *Wanderkarte Rheintal - Appenzellerland: Spezialzusammensetzung des Bundesamtes für Landestopographie 1:25'000.* Wabern: Bundesamt für Landestopographie.

[13] Kruskal J.B. (1956): *On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem.* in: Proceedings of the American Mathematical Society. 1 (1956), p. 48-50.

[14] Leica (2003): *Leica Geosystems: Die "Klick"-Zukunft im GIS.* in: Geospatial Solutions. January 2003, p. 22-27.

[15] MacEachren A. and Taylor F. (1994): *Visualization in modern cartography.* Oxford: Elsevier Science Ltd.

[16] Meinike T. (2006): *XML Praxis: SVG mit Firefox.* in: XML Magazin und Web Services. February 2006, p. 132 - 135.

[17] Monmonier M. (1982): *Computer assisted cartograph: principles and prospects.* Englewood Cliffs: Prentice-Hall Inc.

[18] Muller J.C. (1991): *Advances in Cartography.* New York: Elsevier Science Publishers Ltd.

[19] Olbrich G., Quick M., Schweikart J. (1996): *Computerkartographie: Eine Einführung in das Desktop Mapping am PC.* Berlin: Springer Verlag.

[20] Reichenbacher T. (2005): *SVG: Potential und Anwendungsbeispiele in der Geoinformationswissenschaft.* München: Kursunterlagen Technische Universität München.

[21] Sedgewick R. (2002): *Algorithms in C (third edition): Part 5 Graph Algorithms.* Boston: Addison-Wesley.

[22] Sieber U. (2008): *Konvertierungen in gängige GPS-Datenformate: Erweiterung eines online Wanderplaners für GPS Geräte und GPS Software.* University of Fribourg: Faculty of Sciences, Unit of Geography.

[23] Swisstopo (1996): *Digitales Höhenmodell der Schweiz DHM25 und DHM75.* Bern: Bundesamt für Landestopographie.

[24] Swisstopo (2003): *Landeskarten der Schweiz: Zeichenerklärung und weitere Informationen zu den Landeskarten.* Bern: Bundesamt für Landestopographie.

[25] Tagblatt (2007): St.Galler Tagblatt 08-08-2007: *Die Schweiz auf silbernen Scheiben.*

[26] Tages-Anzeiger (2007): Tages-Anzeiger 11-15-2007*Google tritt mit Karten-Dienst gegen Post an.*

[27] Tanenbaum A.S. (2003): *Computer Networks. International edition.* Upper Saddle River: Pearson Education Inc.

[28] Tomlin D. (1990): *Geographic Information Systems and Cartographic Modelling.* Englewoods Cliffs: Prentice-Hall.

[29] Ueberschär N., Winter A. (2006): *Visualisiern von Geodaten mit SVG im Internet: Band 1 Scalable Vector Graphics - Einführung, clientseitige Interaktionen und Dynamik.* Heidelberg: Herbert Wichmann Verlag.

[•] Internet sources:

[30] Adobe (2007a): *Adobe IllustratorCS3.* Webpage (consulted 10-12-2007):
`http://www.adobe.com/de/products/illustrator/`

[31] Adobe (2007b): *Adobe: Scalable Vector Graphics.* Webpage (consulted 10-13-2007):
`http://www.adobe.com/svg/main.html`

[32] Adobe (2007c): *Adobe to discontinue Adobe SVG Viewer.* Webpage (consulted 12-03-2007):
`http://www.adobe.com/svg/eol.html`

[33] Appenzeller Wanderwege (2007): *Appenzeller Wanderwege: Wandern im Appenzellerland.* Webpage (consulted 10-02-2007):
`http://www.appenzeller-wanderwege.ch/img/wandervorschl_karte.pdf`

[34] Carto:Net (2007): *Carto:net: SVG, scalable vector graphics: tutorials, examples, widgets and libraries.* Webpage (consulted 10-23-2007):
`http://www.carto.net/papers/svg/samples/get_parse.shtml`

[35] Code Dread (2007): *Code Dread*. Webpage (consulted 10-13-2007):
http://codedread.com/svg-support.php

[36] Corel (2007): *Corel Draw 11: Grafiksuite*. Webpage (consulted 10-12-2007):
http://www.pcp.dtpnet.de/html/bericht_corel_draw_11.htm

[37] Entwickler Blog (2007): *Header location und Sessions-Variablen: Ein Problem in PHP* . Webpage (consulted 11-28-2007):
http://www.entwickler-blog.de/archiv/
23-Header-Location-und-Sessions-Variablen-ein-PHP-Problem.html

[38] ESRI (2007): *ESRI: Press release*. Webpage (consulted 10-12-2007):
http://www.esri.com/news/releases/06_1qtr/arcgis_svg.html

[39] ESRI Support (2007): *ESRI Support Center: Knowledge Base*. Webpage (consulted 08-01-2007):
http://support.esri.com/index.cfm?fa=knowledgeBase.gateway

[40] Feike R. (2007): *HTML MIME Mail - MIME Mail mit PHP versenden (PHP4)*. Webpage (consulted 11-09-2007):
http://www.feike.biz/php_mime_mail_de.html

[41] GeoMedia Technology (2007): *GeoMedia WebMap Basismodul 2*. Webpage (consulted 07-17-2007):
http://195.141.205.118/bm2_admin/

[42] Gitta (2007): *Geographic Information Technology Training Alliance: Modul Erreichbarkeit*. Webpage (consulted 09-11-2007):
http://gitta.info/Accessibilit/de/html/UncProxAnaly_learningObject4.html

[43] Golem (2007): *Golem: IT-News für Profis*. Webpage (consulted 11-30-2007):
http://www.golem.de/0504/37798.html

[44] Google Maps (2007): *Maps.Google: Willkommen bei Google Maps*. Webpage (consulted 10-05-2007):
http://maps.google.de/maps

[45] IE Blog (2007): *The future of Internet Explorer and Web Standards*. Webpage (consulted 12-05-2007):
http://blog.codedread.com/archives/2006/05/04/
the-future-of-internet-explorer-and-web-standards

[46] Janco (2007): *Janco Associates Inc*. Webpage (consulted 10-13-2007):
http://www.e-janco.com/browser.htm

[47] Living Internet (2007): *Tim Berners-Lee, Robert Cailliau, and the World Wide Web*. Webpage (consulted 07-12-2007):
http://www.livinginternet.com

[48] Map 24 (2007): *Map24: Kostenloser Routenplaner*. Webpage (consulted 10-05-2007):
http://www.ch.map24.com

[49] Mozilla (2007): *Mozilla Development Center: Ajax - Getting started*. Webpage (consulted 11-01-2007):
http://developer.mozilla.org/de/docs/AJAX:Getting_Started

[50] MySQL (2007): *MySQL: Volltextsuche.* Webpage (consulted 11-23-2007):
http://dev.mysql.com/doc/refman/5.1/de/fulltext-search.html

[51] MySQL Databases (2007): *MySQL: The most popular way to drive databese in the world.* Webpage (consulted 09-28-2007):
http://www.mysql.de

[52] Neumann A. (2003): *Kartographie im Internet auf Vektorbasis, mit Hilfe von SVG nun möglich.* Paper (consulted 10-12-2007):
http://www.carto.net/papers/svg/index_d.shtml

[53] OpenSource (2007): *OpenSource: History of the OSI.* Webpage (consulted 07-13-2007):
http://www.opensource.org/history/

[54] OpenSwf (2007): *Openswf: What you need, when you need it.* Webpage (consulted 07-13-2007):
http://www.openswf.org/

[55] PHP.net (2007): *PHP: Hypertext Preprocessor.* Webpage (consulted 10-11-2007):
http://www.php.net/

[56] Regiun Surselva (2007a): *Regiun Surselva: GPS Touren.* Webpage (consulted 10-02-2007):
http://www.regiun-surselva.ch/GPS-Touren.1126.0.html?&L=1

[57] Regiun Surselva (2007b): *Regiun Surselva: Kulturführer.* Webpage (consulted 10-02-2007):
http://www.regiun-surselva.ch/Kulturfuehrer.1233.0.html?&L=1

[58] Schweizer Wanderwege (2007a): *Schweizer Wanderwege: Wegweisend für ihr Wandererlebnis.* Webpage (consulted 09-27-2007):
http://www.swisshiking.ch/fileadmin/wandern/
Dokumente/Wandern__Wanderwege/mzb_2007.zip

[59] Schweizer Wanderwege (2007b): *Schweizer Wanderwege: Aktuelle Projekte Wander-Monitoring.* Webpage (consulted 12-06-2007):
http://swisshiking.ch/index.php?id=13

[60] Selfhtml (2007): *Selfhtml / Navigationshilfen: Eingebettete Frames.* Webpage (consulted 11-23-2007):
http://de.selfhtml.org/html/frames/eingebettete.htm

[61] St.Galler Wanderwege (2007): *Kantonal St.Galler Wanderwege: Die schönsten 100 Wanderungen im Kanton St.Gallen.* Webpage (consulted 10-02-2007):
http://www.sg-wanderwege.ch/100wander.htm

[62] Strassenprogrammierer (2007): *MySQL: Optimieren Sie Ihre Abfragen.* Webpage (consulted 10-09-2007):
http://www.strassenprogrammierer.de/
mysql-abfragen-optimieren-laufzeit_tipp_428.html

[63] SVG Developers (2007): *zoom/pan svg image: only inside a frame.* Webpage (consulted 11-02-2007):
http://tech.groups.yahoo.com/group/svg-developers/message/50789

[64] SVG Wiki (2007): *SVG Wiki: the place for help on Scalable Vector Graphics.* Webpage (consulted 10-13-2007):
`http://wiki.svg.org/Internet_Explorer`

[65] Swisstopo (2007): *Freedata: Swiss municipalities gravity centre.* Webpage (consulted 11-13-2007):
`http://www.swisstopo.ch/de/download/freedata/gravitycentre`

[66] Toggenburg (2007): *Toggenburg: Gut aufgehoben - GPS Touren.* Webpage (consulted 10-02-2007):
`http://www.toggenburg.org/de/GPS_tracks_hike.cfm`

[67] Typo3 (2007): *Typo3: Get content right.* Webpage (consulted 09-20-2007):
`http://www.typo3.org`

[68] Webland (2007): *Webland.ch: WebServer Applikationen.* Webpage (consulted 09-28-2007):
`http://www.webland.ch/support/webserver_applikationen.aspx`

[69] W3C (2007): *W3C Recommendation: Namepsaces in XML 1.0 (Second Edition).* Webpage (consulted 11-09-2007):
`http://www.w3.org/TR/xml-names/`

[70] W3C DOM (2007): *W3C Architecture Domain: Document Object Model (DOM).* Webpage (consulted 11-09-2007):
`http://www.w3.org/DOM/`

[71] Zonum (2007): *Zonum Solutions: Shape2Text file convertion.* Webpage (consulted 06-12-2007):
`http://www.zonums.com/shp2text.html`

[•] Interview and data sources:

[72] IGUF (1996): *University of Fribourg: Institute of Geography.*

[73] Styger V. (2007a): *Styger Viktor: responsible person St.Galler Wanderwege.* Interview on Friday, 20-07-2007.

[74] Styger V. (2007b): *Styger Viktor: responsible person St.Galler Wanderwege.* Interview on Monday, 01-10-2007.

[75] Styger V. (2008): *Styger Viktor: responsible person St.Galler Wanderwege.* Comments on Wednesday, 05-03-2008.

[76] Vermessungsamt St.Gallen (2007): *Vermessungsamt des Kantons St.Gallen: Digitale Daten des Kantons St.Gallen.*

A Calculation Tables

A.1 Tests of heuristic and algorithm runtime

Long path with heuristic on real topology		
Loops	Time [s]	Average [s]
20	65.57	3.28
20	62.32	3.12
20	62.32	3.11
Average over all tests: 3.17 seconds		

Table 7: *Calculation table for testing a long path with heuristic on real topology.*

Long path without heuristic on real topology		
Loops	Time [s]	Average [s]
20	86.80	4.34
20	87.17	4.36
20	86.56	4.32
Average over all tests: 4.34 seconds		

Table 8: *Calculation table for testing a long path without heuristic on real topology.*

Short path with heuristic on real topology		
Loops	Time [s]	Average [s]
20	6.49	0.32
20	6.40	0.32
20	6.48	0.32
Average over all tests: 0.32 seconds		

Table 9: *Calculation table for testing a short path with heuristic on real topology.*

Short path without heuristic on real topology		
Loops	Time [s]	Average [s]
20	5.78	0.29
20	5.88	0.29
20	5.90	0.30
Average over all tests: 0.29 seconds		

Table 10: *Calculation table for testing a short path without heuristic on real topology.*

A.2 Server load tests

Duration of response		
DURATION UNDER STRESS [S]	DURATION AS SINGLE QUERY [S]	TIME DIFFERENCE [%]
3.75	5.05	35
5.28	6.72	27
4.60	6.07	32
2.98	3.81	28
7.05	9.91	41
3.19	4.25	33
2.55	3.34	31
4.21	5.40	28
2.23	3.73	67
Average difference: 0.36%		

Table 11: *Time comparison between queries with heavy loaded server and with disburden server infrastructure. An average time difference of 35% results.*

B Digital Data on DVD

On the DVD attached, all available digital files are copied. The reader may look into and explore some scripts on a Computer directly. The following data is duplicated on the DVD:

- Digital Masterthesis document in `.pdf` format with links to parts, sections and references, raw figures
- Raw data: orthophotographs, `.shp` hiking network files and auxiliaries, `.shp` files containing roads, rivers, borders
- All scripts and auxiliary files on the server
- Topology: an `.sql` export of the whole database with topology and user tables
- Presentation of topology build up

x

C Paste of the Source Code

A set of code files are listed in this section. We are aware of copying all the files in the appendix. There are just the scripts the most important echoed. To watch the whole set of source files, please rely on the digital versions in section B.

C.1 Compiler codes

In this section, all developed compilers to initialize our basegeometry out of the received data is pasted. The files are listed in ascending manner as they are used to follow a step-by-step transformation of the raw data into the wanted topological dataset and its corresponding content tables. The compilers purpose is written on every file header as comment.

C.1.1 0CompilerNodesFromSignpostep.php

```
<?php
//Opens File with Coordinates and Nodes (with accesory data)
//the file has to be named "basenodes.txt". No header is accepted, the
//nodes have to be directly listed in the file.
//The File has to be stored in the same folder as this compiler file.
$datei = fopen("basenodes.txt", "r") ;

//DB Connection
$db=mysql_connect("mysql5.webland.ch", "virto_hike", "euueeuue");
mysql_select_db("virto_hike");
$cStrTABLENode = "basenodes";

while (feof($datei)==FALSE) {
 //read lines out of file
 $zeile = fgets($datei, 1024) ;
 $cnt = explode(" ", $zeile);
 $x = $cnt[0];
 $y = $cnt[1];

 //Check line if contains a coordinate as Beginning for a new node
 if($x>0 AND $y>0){
 $anfrage = "INSERT INTO $cStrTABLENode VALUES (
'".$cnt[0]."' , '".$cnt[1]."', '".$cnt[2]."', '".$cnt[3]."',
'".$cnt[4]."', '".$cnt[5]."', '".$cnt[6]."', '".$cnt[7]."',
'".$cnt[8]."', '".$cnt[9]."', '".$cnt[10]."')";
$ergebnis = mysql_query($anfrage);
 }//endIF
} //endWhile

mysql_close($db);
fclose($datei);
?>
```

C.1.2 1CompilerEdges.php

```
<?php
//Open File with Coordinates and Edges, the file has to be named "baseedges.txt"
//and has to be stored in the same folder as this compiler file.
$datei = fopen("baseedge1.txt", "r") ;

//DB Connection
$db=mysql_connect("mysql5.webland.ch", "virto_hike", "euueeuue");
mysql_select_db("virto_hike");
$cStrTABLEFrame = "baseedges";

while (feof($datei)==FALSE) {
 //read lines out of file
 $zeile = fgets($datei, 512) ;

 //Check line if contains "Node" as Beginning for a new Edge
 if(strpos($zeile, "Node")!==false){
```

```
      //initializings
      $i = 0;
      $dist = 0;
      $txt = "";
      $svg = "";
      //Number of Segments in the Edge
      $Segments = intval($zeile);

      //Work through all Segments
      for($i; $i<$Segments; $i++){
       $trim = trim(fgets($datei, 512));
       $cnt = explode(" ", $trim);
       $x = $cnt[0];
       $y = $cnt[1];

        //Calculation for a SVG Path
        if($i>0){
          $xcalc = $x - $xbuffer;
          $ycalc = $y - $ybuffer;
          //Increments the distance
          $dist = $dist + sqrt(pow($xcalc,2)+pow($ycalc,2));
          // Renews the path
          $svg .= "L".$xcalc.",".$ycalc."";}
          else{  //i==0, Start SVG Path
           $svg .= "M".$x.",".$y."";}

       $xbuffer = $x;
       $ybuffer = $y;

       //Check if Startnode of Edge (From-Node)
       if($i==0){
         $sx = $x;
         $sy = $y;}

       //Check if Endnode of Edge (To-Node)
       if($i==$Segments-1){
         $txt .= "".$trim."";
         $ex = $x;
         $ey = $y;
         //Insertion into database
         $anfrage = "INSERT INTO $cStrTABLEFrame VALUES (
'' , '".$dist."', '', '".$sx."', '".$sy."', '".$ex."', '".$ey."', '".$txt."', '".$svg."')";
$ergebnis = mysql_query($anfrage);
        }
         else{
          $txt .= "".$trim.",";}
       } //endFor
 } //endIf
} //endWhile

mysql_close($db);
fclose($datei);
?>
```

C.1.3 2CompilerSurfaces.php

```
<?php
//Open File with surface values, the file has to be named "surface.txt"
//and has to be stored in the same folder as this compiler file.
$datei = fopen("surface.txt", "r") ;

//DB Connection
$db=mysql_connect("mysql5.webland.ch", "virto_hike", "euueeuue");
mysql_select_db("virto_hike");
$cStrTABLEFrame = "baseedges";

//Initialization
$ID = 0;
$counter = 0;
$sum = 0;
```

```
    $ratio = 0;
16  $buffer = 1; //has to match the first entry ID

    while (feof($datei)==FALSE) {
     //read lines out of file
20   $zeile = fgets($datei, 512) ;
      $cnt = explode(",", $zeile);
        $ID = $cnt[0];
        $type = $cnt[1];
24
     if($ID == $buffer){
      $counter += 1;
    $sum += $type;
28  $ratio = $sum/$counter;}
    else{
          //Update on database
          $anfrage = "UPDATE ".$cStrTABLEFrame." SET Surf = \"".$ratio."\" WHERE ID = ".$buffer."";
32    $ergebnis = mysql_query($anfrage);
      //Re-Initializing
     $counter = 1;
     $sum = $type;
36   $ratio = $sum/$counter;}

     $buffer = $ID;
    } //endWhile
40
    mysql_close($db);

    fclose($datei);
44  ?>
```

C.1.4 3CompilerInitNodes.php

```
    <?php
    //Reads and inserts nodes from baseedges into n_nodes table
    //Creates the basegeometry's nodes out of the start- and
 4  //endpoints of each edge.

    //DB Connection
    $db=mysql_connect("mysql5.webland.ch", "virto_hike", "euueeuue");
 8  mysql_select_db("virto_hike");
    $cStrTABLEEdge = "baseedges";
    $cStrTABLENode = "n_nodes";

12        //Select Startnodes
          $eanfrage = "SELECT sx, sy FROM ".$cStrTABLEEdge."";
      $edges = mysql_query($eanfrage);
          //Select Endnodes
16        $eanfrage = "SELECT ex, ey FROM ".$cStrTABLEEdge."";
      $eedges = mysql_query($eanfrage);

     while($ezeile = mysql_fetch_row($edges)){
20  //Compares if node is already entried
    $anfrageCOMP = "SELECT n_x, n_y FROM ".$cStrTABLENode."
    WHERE n_x = ".$ezeile[0]." AND n_y = ".$ezeile[1]."";
    $ergebnisCOMP = mysql_query($anfrageCOMP);
24  if(mysql_fetch_row($ergebnisCOMP)== TRUE){
    ;}
    else{
    $anfrage = "INSERT INTO ".$cStrTABLENode."
28  VALUES ('".$ezeile[0]."', '".$ezeile[1]."', '3','','','','','','','','','','')";
         $ergebnis = mysql_query($anfrage);}

    } //endWhile
32
     while($ezeile = mysql_fetch_row($eedges)){
      //Compares if node is already entried
    $anfrageCOMP = "SELECT n_x, n_y FROM ".$cStrTABLENode."
36  WHERE n_x = ".$ezeile[0]." AND n_y = ".$ezeile[1]."";
    $ergebnisCOMP = mysql_query($anfrageCOMP);
```

```
    if(mysql_fetch_row($ergebnisCOMP)== TRUE){
    ;}
40  else{
    $anfrage = "INSERT INTO ".$cStrTABLENode."
    VALUES ('".$ezeile[0]."', '".$ezeile[1]."', '3','','','','','','','','','','')";
         $ergebnis = mysql_query($anfrage);}
44
    } //endWhile

    mysql_close($db);
48  ?>
```

C.1.5 4CompilerSignpost2Nodes.php

```
    <?php
    //Connects the basenodes SG Wanderwege to the graph's topology
    //Links the nodedata to the topology out of the edges
 4
    //DB Connection
    $db=mysql_connect("mysql5.webland.ch", "virto_hike", "euueeuue");
    mysql_select_db("virto_hike");
 8  $cStrTABLESignpost = "basenodes";
    $cStrTABLENode = "n_nodes";
    $cStrTABLETopo = "topology";

12  //Limit to connect a node from SIGNPOSTEP to a node out of the
    //basegeometry in [m]

    $limit = 50;
16
         //Select Edges
         $eanfrage = "SELECT * FROM ".$cStrTABLESignpost."";
      $signpost = mysql_query($eanfrage);
20
      while($ezeile = mysql_fetch_row($signpost)){
      //Select every exisiting node out of basegeometry
         $nanfrage = "SELECT n_GID, n_x, n_y FROM ".$cStrTABLENode."";
24    $nodes = mysql_query($nanfrage);
      $init = 10000000;
      $ID = 0;
       while($nzeile = mysql_fetch_row($nodes)){
28     //Calculating every distance from a node out of SIGNPOSTEP to every node in
       //our basegeometry
    $dist = pow((($ezeile[0]-$nzeile[1])*($ezeile[0]-$nzeile[1])+
    ($ezeile[1]-$nzeile[2])*($ezeile[1]-$nzeile[2])),0.5);
32  if($init>$dist){
    $ID = $nzeile[0];
    $init = $dist;}
       } //endWhile
36
     if($init < $limit){
      $anfrageSQL =  "UPDATE ".$cStrTABLENode." SET type = \"2\", GID = \"".$ezeile[2]."\",
      n_NAME = \"".$ezeile[3]."\", n_ZZ = \"".$ezeile[4]."\",  n_POSNR = \"".$ezeile[5]."\",
40    n_MAPNR = \"".$ezeile[6]."\",n_HEIGHT = \"".$ezeile[7]."\",n_Location =
      \"".$ezeile[8]."\",n_Municipality = \"".$ezeile[9]."\",n_District =
      \"".$ezeile[10]."\" WHERE n_GID = ".$ID."";
      $ergebnisSQL = mysql_query($anfrageSQL);}
44
      } //endWhile

    mysql_close($db);
48
    ?>
```

C.1.6 5CompilerHeights.php

```
    <?php
    //Applies the hight above sealevel to all nodes in the basegeometry of our graph
    //If there are values from SG Wanderwege SIGNPOSTEP than copy them into the graph nodes
 4  //Else calculate a medium distance with the weighted triangualtion method
```

```
    //DB Connection
    $db=mysql_connect("mysql5.webland.ch", "virto_hike", "euueeuue");
 8  mysql_select_db("virto_hike");
    $cStrTABLESignpost = "basenodes";
    $cStrTABLENode = "n_nodes";
    $cStrTABLETopo = "topology";
12
         //Select Edges that do not have a height out of SIGNPOSTEP (type=3)
         $eanfrage = "SELECT n_GID, n_x, n_y FROM ".$cStrTABLENode." WHERE type = 3";
      $zeroh = mysql_query($eanfrage);
16
      while($ezeile = mysql_fetch_row($zeroh)){
      //Select Height and Distances from SIGNPOSTEP to our corresponding nodes in n_nodes
         $nanfrage = "SELECT GID, HEIGHT,
20       POW((POW((x-".$ezeile[1]."),2)+POW((y-".$ezeile[2]."),2)),0.5)
         AS 'Distance' FROM ".$cStrTABLESignpost." ORDER BY Distance ASC";
      $distances = mysql_query($nanfrage);

24    //Store the 3 nearest nodes
      $dist1 = mysql_fetch_row($distances);
      $dist2 = mysql_fetch_row($distances);
      $dist3 = mysql_fetch_row($distances);
28
      //Triangualtion Method with weighted influences of 3 nearest nodes
      $sumdist = (1/$dist1[2])+(1/$dist2[2])+(1/$dist3[2]);
      //Weight for each node (inverse proportional to their distance)
32    $w1 = (1/$dist1[2])/$sumdist;
      $w2 = (1/$dist2[2])/$sumdist;
      $w3 = (1/$dist3[2])/$sumdist;
      //Calculated height out of the three nearest nodes with triangulation (inverse proportional)
36    $sumheight = ($w1*$dist1[1])+($w2*$dist2[1])+($w3*$dist3[1]);

      $anfrageSQL =  "UPDATE ".$cStrTABLENode." SET n_HEIGHT = \"".$sumheight."\"
      WHERE n_GID = ".$ezeile[0]."";
40    $ergebnisSQL = mysql_query($anfrageSQL);

      } //endWhile

44  mysql_close($db);

    ?>
```

C.1.7 6CompilerInitTopo.php

```
    <?php
    //Initialization of topology via Edges Table and n_nodes table
    //Calculation of difference in height for every entry.
 4
    //DB Connection
    $db=mysql_connect("mysql5.webland.ch", "virto_hike", "euueeuue");
    mysql_select_db("virto_hike");
 8  $cStrTABLEEdge = "baseedges";
    $cStrTABLENode = "n_nodes";
    $cStrTABLETopo = "topology";

12       //Select Edges
         $eanfrage = "SELECT ID, Dist, sx, sy, ex, ey FROM ".$cStrTABLEEdge."";
      $edges = mysql_query($eanfrage);

16    while($ezeile = mysql_fetch_row($edges)){

         $nanfrage = "SELECT n_GID, n_HEIGHT FROM ".$cStrTABLENode."
         WHERE n_x=".$ezeile[2]." AND n_y=".$ezeile[3]."";
20    $start = mysql_query($nanfrage);
         $nanfrage2 = "SELECT n_GID, n_HEIGHT FROM ".$cStrTABLENode."
         WHERE n_x=".$ezeile[4]." AND n_y=".$ezeile[5]."";
      $end = mysql_query($nanfrage2);
24
      $snode = mysql_fetch_row($start);
```

```
    $endnode = mysql_fetch_row($end);

28  $hdiff =    $endnode[1]-$snode[1];
    $antihdiff = $hdiff*(-1);
    $hvdist = pow((pow($ezeile[1], 2)+pow($hdiff,2)),0.5);

32  $anfrageSQL = "INSERT INTO ".$cStrTABLETopo."  VALUES ('".$ezeile[0]."',
    '".$snode[0]."', '".$endnode[0]."', '".$hdiff."', '', '', '".$ezeile[1]."', '".$hvdist."')";
    $ergebnisSQL = mysql_query($anfrageSQL);
    //Other way Edge INSERTION
36  $anfrageSQL2 = "INSERT INTO ".$cStrTABLETopo."  VALUES ('".$ezeile[0]."',
    '".$endnode[0]."', '".$snode[0]."', '".$antihdiff."', '', '', '".$ezeile[1]."', '".$hvdist."')";
    $ergebnisSQL2 = mysql_query($anfrageSQL2);

40  } //endWhile

  mysql_close($db);

44 ?>
```

C.1.8 7CompilerTopoTime.php

```
  <?php
  //calcualtes walking time out of distance horizontal and differece in height.

 4 //DB Connection
  $db=mysql_connect("mysql5.webland.ch", "virto_hike", "euueeuue");
  mysql_select_db("virto_hike");
  $cStrTABLETopo = "topology";
 8
  //Polynome's constants
  $c0 = 14.271 ;
  $c1 =  0.36992;
12 $c2 =  0.025922;
  $c3 =  -0.0014384;
  $c4 =  0.000032105;
  $c5 =  0.0000081542;
16 $c6 =  -0.000000090261;
  $c7 =  -0.000000020757;
  $c8 =  0.00000000010192;
  $c9 =  0.000000000028588;
20 $c10 =  -0.000000000000057466;
  $c11 =  -0.000000000000021842;
  $c12 =  1.5176*pow(10,-17);
  $c13 =  8.6894*pow(10,-18);
24 $c14 =  -1.3584*pow(10,-21);
  $c15 =  -1.4026*pow(10,-21);

      //Select Edges
28    $eanfrage = "SELECT Edge, From_node, To_node, Hdiff, hdist FROM ".$cStrTABLETopo."";
    $edges = mysql_query($eanfrage);

    while($ezeile = mysql_fetch_row($edges)){
32
      //Function calculating walking time out of hdist L and gradient S
    //Polynome of degree 15
    //This polynome limits the calculations for gradients -40%<$S<+40%
36  //Out of this limits it calculates wrong and irreal values!

    //Limits $S to -40%<$S<+40%, very important to avoid irreal values
    $S = 100*$ezeile[3]/$ezeile[4];
40  if($S>40){$S=40;}
    if($S<-40){$S=-40;}

    //horizontal distance. If $L is just some meters and height that is calculated
44  //approximatively varies in some meters, the gradient becomes so steep that
    //irreal values for time are its consequence. So we limit the length, whereas
    //steepness does not count anymore just because nodes are side-by-side each other
    $L = $ezeile[4];
48  $limit = 8; //Limit in meters
```

```
    if($L < $limit){
     $S = 0;}

52  //Polynome with degree 15
        $sec_calc = ($L * ($c0 + ($c1 * pow($S,1)) + ($c2 * pow($S,2)) + ($c3 * pow($S,3)) +
        ($c4 * pow($S,4)) + ($c5 * pow($S,5)) + ($c6 * pow($S,6)) + ($c7 * pow($S,7)) +
        ($c8 * pow($S,8)) + ($c9 * pow($S,9)) + ($c10 * pow($S,10)) + ($c11 * pow($S,11)) +
56      ($c12 * pow($S,12)) + ($c13 * pow($S,13)) + ($c14 * pow($S,14)) +
        ($c15 * pow($S,15)))) / 1000;

    $anfrageSQL = "UPDATE ".$cStrTABLETopo." SET Time = \"".$sec_calc."\",
60  Gradient = \"".$S."\" WHERE Edge = ".$ezeile[0]." AND From_node = ".$ezeile[1]."
    AND To_node = ".$ezeile[2]."";
    $ergebnisSQL = mysql_query($anfrageSQL);

64  } //endWhile

  mysql_close($db);

68 ?>
```

C.2 rasterHeightsComp.php

The source code to extract the corresponding heights out of the DHM25. Input files contain the raster data of heights and the coordinates of the approximated nodes.

```
  <?php
  //The files have to be named as follows:
  $datei = fopen("rasterHeights.txt", "r");
 4 $heights = fopen("nodesHeight.txt", "r");

  $xVersatz = 721987.5; //min x-Edge
  $yVersatz = 210012.5; //min y_Edge
 8 $matrix = array();

  while (feof($datei)==FALSE) {
   //read lines out of file
12  $i = 0;
   $zeile = fgets($datei, 16384);
   $cnt = explode(" ", $zeile);
   $matrix[] = $cnt;
16 $i = $i+1;
  } //endWhile

  fclose($datei);
20 //Length of matrix to get inverse index
  $l = count($matrix);

  //echo ('TEST '.$matrix[200][400].'<br>');
24 //echo ('TEST2 '.$matrix[1][1].'<br>');

  while (feof($heights) == FALSE){
  $coor = fgets($heights, 1024);
28 $values = explode(" ", $coor);
  $x = floor(($values[0]-$xVersatz)/25); //DHM25
  $y = floor(($values[1]-$yVersatz)/25); //DHM25
  echo (".$matrix[$l-$y][$x]."<br>");
32 } //endWhile
  fclose($heights);
  ?>
```

C.3 Codes for Dijkstra's algorithm and the heuristic

This chapter gives a copy of all relevant codes for the calculation of the paths. Whereas `dijkstraClass.php` contains the main class and the implementation, `costConstants.inc` lists the defined constants for the heuristic. Finally `DijkstraMitHeuristik.php` takes all files together and builds the framework for the calculations.

C.3.1 dijkstraClass.php

```
<?php
//Object-oriented PHP, Definition of the main class "TrailDijkstra"
class trailDijkstra {

    var $visited = array();
    var $distance = array();
    var $previousNode = array();
    var $startnode = null;
    var $tonode = null;
    var $graph = array();
    var $infiniteDistance = 0;
    var $numberOfNodes = 0;
    var $bestPath = 0;
    var $matrixWidth = 0;

//trailDijkstra: Main class-method
    function trailDijkstra(&$ourGraph, $infiniteDistance) {
        $this -> infiniteDistance = $infiniteDistance;
        $this -> graph = &$ourGraph;
        $this -> numberOfNodes = count($ourGraph);
        $this -> bestPath = 0;
    }

//trailShortestPath: Method initializes all the nodes to be ready
//for computing
//Starts the loop of the searching algorithm in the while-case
    function trailShortestPath($start,$to) {
        $this -> startnode = $start;
        $this -> tonode = $to;
        for ($i=0;$i<$this -> numberOfNodes;$i++) {
            if ($i == $this -> startnode) {
                $this -> visited[$i] = true;
                $this -> distance[$i] = 0;
            } else {
                $this -> visited[$i] = false;
                $this -> distance[$i] = isset($this -> graph[$this -> startnode][$i])
                    ? $this -> graph[$this -> startnode][$i]
                    : $this -> infiniteDistance;
            }
            $this -> previousNode[$i] = $this -> startnode;
        }
//Set this may be manually to reduce buffer overflows
//$maxTries = $this -> numberOfNodes;
        $maxTries = 1000;
        $tries = 0;
        while (in_array(false,$this -> visited,true) && $tries <= $maxTries) {
            $this -> bestPath = $this->trailBestPath($this->
             distance,array_keys($this -> visited,false));
            if($to !== null && $this -> bestPath === $to) {
                break;
            }
            $this -> updateDistanceAndPrevious($this -> bestPath);
            $this -> visited[$this -> bestPath] = true;
            $tries++;
        }
    }

//trailBestPath: Is called by trailShortestPath in the while-loop to
//look for the min weighted trail
    function trailBestPath($ourDistance, $ourNodesLeft) {
        $bestPath = $this -> infiniteDistance;
        $bestNode = 0;
        for ($i = 0,$m=count($ourNodesLeft); $i < $m; $i++) {
            if($ourDistance[$ourNodesLeft[$i]] < $bestPath) {
                $bestPath = $ourDistance[$ourNodesLeft[$i]];
                $bestNode = $ourNodesLeft[$i];
            }
        }
```

```
            return $bestNode;
        }

    //updateDistanceAndPrevious: Updates the actual Distance of the calculated
    //track
        function updateDistanceAndPrevious($obp) {
            for ($i=0;$i<$this -> numberOfNodes;$i++) {
                if(     (isset($this->graph[$obp][$i]))
                    &&    (!($this->graph[$obp][$i] == $this->infiniteDistance) ||
                     ($this->graph[$obp][$i] == 0 ))
                    &&    (($this->distance[$obp] + $this->graph[$obp][$i]) < $this -> distance[$i])
                )
                {
                        $this -> distance[$i] = $this -> distance[$obp] + $this -> graph[$obp][$i];
                        $this -> previousNode[$i] = $obp;
                }
            }
        }

    //printGraph: Visualizing method for our graph to check the weight-table graphically
        function printGraph(&$graph) {
            $placeholder = ' %' . strlen($this -> infiniteDistance) .'d';
            $foo = '';
            for($i=0,$im=count($graph);$i<$im;$i++) {
                for ($k=0,$m=$im;$k<$m;$k++) {
                    $foo.= sprintf($placeholder, isset($graph[$i][$k]) ? $graph[$i][$k] :
                    $this -> infiniteDistance);
                }
                $foo.= "\n";
            }
            return $foo;
        }

    //getResults: Output method of trailDijkstra class returns a string containing
    //the shortest path and the distance/weight of this path.
        function getResults($to) {
            $ourShortestPath = array();
            for ($i = 0; $i < $this -> numberOfNodes; $i++) {
                if($to !== null && $to !== $i) {
                    continue;
                }
                $ourShortestPath[$i] = array();
                $endNode = null;
                $currNode = $i;
                $ourShortestPath[$i][] = $i;
                while ($endNode === null || $endNode != $this -> startnode) {
                    $ourShortestPath[$i][] = $this -> previousNode[$currNode];
                    $endNode = $this -> previousNode[$currNode];
                    $currNode = $this -> previousNode[$currNode];
                }
                $ourShortestPath[$i] = array_reverse($ourShortestPath[$i]);
                if (($to === null || $to === $i) AND $i == $this -> tonode) {
                if($this -> distance[$i] >= $this -> infiniteDistance) {
                    $track = array('no','no');
                    //$track = sprintf("no route from %d to %d. \n",$this -> startnode,$i);
                } else {
                    $cost  = sprintf('%d',$this -> distance[$i]);
                    $path  = sprintf('%s',implode(',',$ourShortestPath[$i]));
                    $track = array($cost,$path);
                }
            //    $track .= str_repeat('-',20) . "\n";
                    if ($to === $i) {
                        break;
                    }
                }
            }
    //return $ourShortestPath;
            return $track;
            //Return the Graph visualized
            //return $graphVisualized;
```

```
    }
} // end class
?>
```

C.3.2 costConstants.inc

```
<?php
// -------------------------------------------------------
// Constants defining widening of focus area for heuristic

function fctcostFactor($a){
$costfact = 0;
if($a ==  "time"){$costfact = 5000;}
elseif (stristr($a, "gradient") == TRUE){$costfact = 15000;}
elseif ($a ==  "hvdist"){$costfact = 3000;}
elseif ($a ==  "hdiff"){$costfact = 15000;}
elseif (stristr($a, "surface") == TRUE){$costfact = 20000;}
else {$costfact = 20000;}
return $costfact;}


function fctInfinite($a){
$inf = 0;
if(stristr($a, "time") == TRUE){$inf = 24*60;}
elseif ($a ==  "gradient"){$inf = 100;}
elseif ($a ==  "hdiff"){$inf = 5000;}
elseif (stristr($a, "dist") == TRUE){$inf = 100000;}
else {$inf = 10000;}
return $inf;}


function fctQuery($a,$Topo,$Node,$Edge,$x,$y){
if($a ==  "time"){
$q = "SELECT from_node, to_node, time,
POW((POW((n_x-".$x."),2)+POW((n_y-".$y."),2)),0.5) AS 'Distance'
FROM ".$Topo.", ".$Node." WHERE n_id = from_node";}
elseif ($a ==  "gradient"){
$q = "SELECT from_node, to_node, gradient,
POW((POW((n_x-".$x."),2)+POW((n_y-".$y."),2)),0.5) AS 'Distance'
FROM ".$Topo.", ".$Node." WHERE n_id = from_node";}
elseif ($a ==  "hvdist"){
$q = "SELECT from_node, to_node, hvdist,
POW((POW((n_x-".$x."),2)+POW((n_y-".$y."),2)),0.5) AS 'Distance'
FROM ".$Topo.", ".$Node." WHERE n_id = from_node";}
elseif ($a ==  "hdiff"){
$q = "SELECT from_node, to_node, hdiff,
POW((POW((n_x-".$x."),2)+POW((n_y-".$y."),2)),0.5) AS 'Distance'
FROM ".$Topo.", ".$Node." WHERE n_id = from_node";}
elseif ($a ==  "time_surface_nat" OR $a ==  "time_surface_art"){
$q = "SELECT from_node, to_node, time,
POW((POW((n_x-".$x."),2)+POW((n_y-".$y."),2)),0.5) AS 'Distance', surf
FROM ".$Topo.", ".$Node.", ".$Edge." WHERE n_id = from_node AND id = Edge";}
elseif ($a ==  "dist_surface_nat" OR $a ==  "dist_surface_art"){
$q = "SELECT from_node, to_node, hvdist,
POW((POW((n_x-".$x."),2)+POW((n_y-".$y."),2)),0.5) AS 'Distance', surf
FROM ".$Topo.", ".$Node.", ".$Edge." WHERE n_id = from_node AND id = Edge";}
elseif ($a ==  "time_gradient"){
$q = "SELECT from_node, to_node, time,
POW((POW((n_x-".$x."),2)+POW((n_y-".$y."),2)),0.5) AS 'Distance', gradient
FROM ".$Topo.", ".$Node." WHERE n_id = from_node";}
return $q;}

function fctWeight($a,$b1,$b2){
if($a ==  "time"){
$q = $b1;}
elseif ($a ==  "gradient"){
$q = abs($b1);} //avoid negative values
elseif ($a ==  "hvdist"){
$q = $b1;}
elseif ($a ==  "hdiff"){
```

```
    $q = abs($b1);} //avoid negative values
    elseif ($a ==  "time_surface_nat" OR $a ==  "dist_surface_nat"){
    if($b2 == 0){$q = $b1;}
68  else{$q = $b1 * $b2;}
    }
    elseif ($a ==  "time_surface_art" OR $a ==  "dist_surface_art"){
    if($b2 == 0){$q = $b1;}
72  else{$q = abs($b2-3) * $b1;}
    }
    elseif ($a ==  "time_gradient"){
    $q = $b1 * abs($b2);} //avoid negative values
76  return $q;}
    ?>
```

C.3.3 DijkstraMitHeuristik.php

```
    <?php

    //Constants initialization
 4  include("constants.php");
    $cDB = $LOCAL_CONS["DB"];

    //Session load
 8  session_start();

    //URL
       $host   = $_SERVER['HTTP_HOST'];
12     $uri   = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');

    // DATABASE:-----------------------------------------------
    define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
16  define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
    define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
20  define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
    define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    define ('cStrTABLEUserQ',$cDB['cStrTABLEUserQ']);
    // -------------------------------------------------------
24
    include("dijkstra_class.php");

    //Including constants for widening focus area
28  include("costConstants.inc");

    function fctDijkstra($Start, $End, $costFactor){

32  $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);


36  $ourGraph = array();

    //Setting start- and endnode
    //$Start = 1739; // Berneck:
40  //$Start = 50;
    //$End = 30;
    /* Possible cost factors:
    - time "time"
44  - distance "hvdist"
    - gradient min "gradient"
    - m in altitude min "hdiff"
    - natural surface preferred "time_surface_nat"
48  - artificial surface preferred "time_surface_art"
    - time gradient min "time_gradient"
    - distance natural surface min "dist_surface_nat"
    - distance hard surface min "dist_surface_art"
52  */

    // -------------------------------------------------------
```

```
    //Implementing the heuristic to reduce graph's size and accelerate computation
 56


    //Defining diameter and center out of distance and cost factor
 60 $loc1 = "SELECT n_x, n_y from ".cStrTABLENodes." WHERE
    n_id = ".$Start." OR n_id = ".$End."";
    $resLoc1 = mysql_query($loc1);
    $n1 = mysql_fetch_row($resLoc1);
 64 $n2 = mysql_fetch_row($resLoc1);
    $dist = (abs($n1[0]-$n2[0])^2 + abs($n1[1]-$n2[1])^2)^(0.5);
    $diameter = ($dist / 2) + fctcostFactor($costFactor);
    $center_x = ($n1[0]+$n2[0])/2;
 68 $center_y = ($n1[1]+$n2[1])/2;
    $infdist = fctInfinite($costFactor);
    // ------------------------------------------------------

 72   //Applying the heuristic
        $anfrageAll = fctQuery($costFactor,cStrTABLETopo,cStrTABLENodes,
         cStrTABLEEdges,$center_x,$center_y);
      $ergebnisAll = mysql_query($anfrageAll);
 76   $num_rows = mysql_num_rows($ergebnisAll);

    //Filling the graph
     for($count = 0; $count < $num_rows; $count++) {
 80 $row = mysql_fetch_row($ergebnisAll);
    // Read in the points and push them into the graph
          //Initializing if Graph should be symmetric or asymmetric.
    //Trail Case is asymmetric, because A to B not equals B to A.
 84 //Different costs are possible depending on direction.
    if($row[3] < $diameter){
    $ourGraph[$row[0]][$row[1]] = fctWeight($costFactor,$row[2],$row[4]);
    //in a symmetric case:
 88 //$ourGraph[$row[1]][$row[0]] = fctWeight($costFactor,$row[2],$row[4]);
    }
    else { $ourGraph[$row[0]][$row[1]] = $infdist;}
         } //endFor
 92 mysql_close($db);

    // ensure that the distance from a node to itself is always zero
    // Purists may want to edit this bit out.
 96 for ($i=0; $i < $num_rows; $i++) {
        for ($k=0; $k < $$num_rows; $k++) {
            if ($i == $k) $ourGraph[$i][$k] = 0;
        }
100 }

    // ------------------------------------------------------
    // I is the initializing infinite distance that limits the focus area
104 define('I',$infdist);

    // initialize the algorithm class
    $dijkstra = new trailDijkstra($ourGraph, I);
108
    //Find shortest path from start- to endnode
    $dijkstra->trailShortestPath($Start,$End);

112 //To find all shortest pathes from node 0
    //$dijkstra->trailShortestPath(0);
    return($dijkstra->getResults());
    }
116
    // Display the results, dealing with via's (max. 2 via's can be set)
    // via's have to be marked with a "xxx" between for further treatment
    // output is an array "sto" containing in Element 0 the total cost and in
120 // Element 1 the String containing the routenodes.
    function resDijkstra($start, $end, $via1, $via2, $costFactor){
    if ($via2 > 0){
    $res1 = fctDijkstra($start,$via1,$costFactor);
124 $res2 = fctDijkstra($via1,$via2,$costFactor);
```

```
    $res3 = fctDijkstra($via2,$end,$costFactor);
    $track = $costFactor.',$res1[1].',xxx'.$res2[1].',xxx'.$res3[1];
    $cost = $res1[0]+$res2[0]+$res3[0];
    $sto = array("cost" => $cost, "res_path" => $track);
    return ($sto);
    //echo $track.' Kosten: '.$cost;
    }
    else if ($via1 > 0){
    $res1 = fctDijkstra($start,$via1,$costFactor);
    $res2 = fctDijkstra($via1,$end,$costFactor);
    $track = $costFactor.',$res1[1].',xxx'.$res2[1];
    $cost = $res1[0]+$res2[0];
    $sto = array("cost" => $cost, "res_path" => $track);
    return ($sto);
    //echo $track.' Kosten: '.$cost;
    }
    else if ($start > 0){
    $res1 = fctDijkstra($start,$end,$costFactor);
    $track = $costFactor.',$res1[1];
    $cost = $res1[0];
    $sto = array("cost" => $cost, "res_path" => $track);
    return ($sto);
    //echo $track.' Kosten: '.$cost;
    }
    else {return ("Bad calculation process. Try again!<br>");
    }
    }

    //TESTS
    //$start = 30;
    //$end = 268;
    //$costFactor = 'hvdist';
    //$via1 = 50;  //Default = -1
    //$via2 = -1;  //Default = -1
    //echo ("".$_SESSION['From']."<br>");
    //echo ("".$_SESSION['To']."<br>");
    //echo ("".$_SESSION['Via1']."<br>");
    //echo ("".$_SESSION['Via2']."<br>");
    //echo ("".$_SESSION['Weight']."<br>");
    $userpath = resDijkstra($_SESSION['FromID'], $_SESSION['ToID'],
    $_SESSION['Via1ID'], $_SESSION['Via2ID'], $_SESSION['Weight']);
    //echo ($userpath["res_path"]);


    //Introduction into database for further treatment
    //only if Path was found
    if($userpath["res_path"] != 'no'){
    $tstamp = time();
    $IP = $_SERVER['REMOTE_ADDR'];

    //Set Sessions
    $_SESSION['totalCost'] = $userpath["cost"];
    $_SESSION['totalPath'] = $userpath["res_path"];
    $_SESSION['time'] = $tstamp;
    $_SESSION['IP'] = $IP;

    $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);

    $insertQ = "INSERT INTO ".cStrTABLEUserQ." VALUES (
    '' , '".$IP."', '".$tstamp."', '".$userpath["res_path"]."', '".$userpath["cost"]."')";
    $ins1 = mysql_query($insertQ);

    mysql_close($db);

    header("location: resMap.php?".session_name()."=".session_id()."&L=".$_REQUEST['L']."");
    }
    else{
    echo ("Leider keine Route zu ihren Daten gefunden.<br>");
```

```
    }
196
    ?>
```

C.3.4 dijkstraExMySQL.php

The test situation can be reproduced by looking at the setting listed in this file.

```
    <?php
    function fctDijkstra($Start, $End){

 4  include("dijkstra_class.php");

    //Including constants for widening focus area
    include("costConstants.inc");
 8
    // Get the topology out of the DB:
    // DATABASE:-----------------------------------------------
    $cStrHOSTDB = "mysql5.webland.ch";
12  $cStrUSERDB = "virto_hike";
    $cStrPASSWORDDB = "euueeuue";
    $cStrDB = "virto_hike";
    $cStrTABLETopo = "topology";
16  $cStrTABLENodes = "n_nodes";
    $cStrTABLEEdges = "baseedges";

    $db=mysql_connect($cStrHOSTDB, $cStrUSERDB, $cStrPASSWORDDB);
20  mysql_select_db($cStrDB);
    // -------------------------------------------------------

    $ourGraph = array();
24
    //Setting start- and endnode
    //$Start = 1739; // Berneck:
    //$Start = 50;
28  //$End = 30;
    /* Possible cost factors:
    - time "time"
    - distance "hvdist"
32  - gradient min "gradient"
    - m in altitude min "hdiff"
    - natural surface preferred "time_surface_nat"
    - artificial surface preferred "time_surface_art"
36  - time gradient min "time_gradient"
    - distance natural surface min "dist_surface_nat"
    - distance hard surface min "dist_surface_art"
    */
40  $costFactor = "hvdist";

    // -------------------------------------------------------
    //Implementing the heuristic to reduce graph's size and accelerate computation
44



    //Defining diameter and center out of distance and cost factor
48  $loc1 = "SELECT n_x, n_y from $cStrTABLENodes WHERE
    n_id = ".$Start." OR n_id = ".$End."";
    $resLoc1 = mysql_query($loc1);
    $n1 = mysql_fetch_row($resLoc1);
52  $n2 = mysql_fetch_row($resLoc1);
    $dist = (abs($n1[0]-$n2[0])^2 + abs($n1[1]-$n2[1])^2)^(0.5);
    $diameter = ($dist / 2) + fctcostFactor($costFactor);
    $center_x = ($n1[0]+$n2[0])/2;
56  $center_y = ($n1[1]+$n2[1])/2;
    $infdist = fctInfinite($costFactor);
    // -------------------------------------------------------

60  $db=mysql_connect($cStrHOSTDB, $cStrUSERDB, $cStrPASSWORDDB);
    mysql_select_db($cStrDB);
      //Applying the heuristic
          $anfrageAll = fctQuery($costFactor,$cStrTABLETopo,$cStrTABLENodes,
```

```
64          $cStrTABLEEdges,$center_x,$center_y);
      $ergebnisAll = mysql_query($anfrageAll);
      $num_rows = mysql_num_rows($ergebnisAll);

68  //Filling the graph
     for($count = 0; $count < $num_rows; $count++) {
    $row = mysql_fetch_row($ergebnisAll);
    // Read in the points and push them into the graph
72        //Initializing if Graph should be symmetric or asymmetric.
    //Trail Case is asymmetric, because A to B not equals B to A.
    //Different costs are possible depending on direction.
    if($row[3] < $diameter){
76  $ourGraph[$row[0]][$row[1]] = fctWeight($costFactor,$row[2],$row[4]);
    //in a symmetric case:
    //$ourGraph[$row[1]][$row[0]] = fctWeight($costFactor,$row[2],$row[4]);
    }
80  else { $ourGraph[$row[0]][$row[1]] = $infdist;}
         } //endFor
    mysql_close($db);

84  // ensure that the distance from a node to itself is always zero
    // Purists may want to edit this bit out.
    for ($i=0; $i < $num_rows; $i++) {
        for ($k=0; $k < $$num_rows; $k++) {
88          if ($i == $k) $ourGraph[$i][$k] = 0;
        }
    }

92  // ------------------------------------------------------
    // I is the initializing infinite distance that limits the focus area
    define('I',$infdist);

96  // initialize the algorithm class
    $dijkstra = new trailDijkstra($ourGraph, I);

    //Find shortest path from start- to endnode
100 $dijkstra->trailShortestPath($Start,$End);

    //To find all shortest pathes from node 0
    //$dijkstra->trailShortestPath(0);
104 return($dijkstra->getResults());
    }

    // Display the results
108 echo '<pre>';
    echo "the graph table looks like:\n\n";
    echo "\n\nthe shortest path from $Start to $End:\n";
    echo fctDijkstra(50,30);
112 echo '</pre>';
    ?>
```

C.4 Map navigation and additional load of details

The AJAX implementation and the JavaScript functions for full navigation possibilities are listed in this chapter. These are all core functions for our project and are loaded into the SVG map files at the very beginning with the PHP command `include()`.

C.4.1 naviJS.php

All used navigation functions to imply user interactivity are programmed in this file. The biggest part is used to implement all different cases for the diverging browsers and SVG viewers.

```
    <?php

    echo("
 4      <![CDATA[
    //*****************************************************************************
    //Checks viewport to not overlap the main map coordiantes
```

```
        function myCheckVB(myVB){
 8      myDefVB = document.getElementById('myOverviewMap').getAttributeNS(null,'viewBox').split(' ');
        myDefVB[0] = parseFloat(myDefVB[0]);
        myDefVB[1] = parseFloat(myDefVB[1]);
        myDefVB[2] = parseFloat(myDefVB[2]);
12      myDefVB[3] = parseFloat(myDefVB[3]);

    if(myVB[2] < myDefVB[2] && myVB[3] < myDefVB[3]){
        //LEFT BORDER
16      if(myVB[0] < myDefVB[0]){
        myVB[0] = myDefVB[0];
        //RIGHT BORDER
        } else if(myVB[0] > myDefVB[0] + myDefVB[2] - myVB[2]){
20      myVB[0] = myDefVB[0] + myDefVB[2] - myVB[2];
        }
     //TOP BORDER
        if(myVB[1] < myDefVB[1]){
24      myVB[1] = myDefVB[1];
  //BOTTOM BORDER
  } else if(myVB[1] > myDefVB[1] + myDefVB[3] - myVB[3]){
        myVB[1] = myDefVB[1] + myDefVB[3] - myVB[3];
28      }
    //IF TOO BIG
    } else {
        myVB[0] = myDefVB[0];
32      myVB[1] = myDefVB[1];
        myVB[2] = myDefVB[2];
        myVB[3] = myDefVB[3];
    }
36      return myVB;
        }

  //*****************************************************************************
40  //PANNING VIA CLICK
        function myPan(myXDir,myYDir){
         //STEP IN METERS
         var myStep = 750;
44      myMapVBArray = document.getElementById('myMap').getAttributeNS(null,'viewBox').split(' ');
        myMapVBArray[0] = parseFloat(myMapVBArray[0]) + myXDir * myStep;
        myMapVBArray[1] = parseFloat(myMapVBArray[1]) + myYDir * myStep;
        //TEST EXTENSION
48      myMapVBArray = myCheckVB(myMapVBArray);
        document.getElementById('myMap').setAttributeNS(null,'viewBox',myMapVBArray.join(' '));
        mySetOverviewRect(myMapVBArray);
          }
52
  //*****************************************************************************
    //ZOOMING VIA CLICK
        function myZoom(myVal){
56       //Change into an array
        myMapVBArray = document.getElementById('myMap').getAttributeNS(null,'viewBox').split(' ');
        //FACTOR
        myVal *= 1.5;
60
        //ZOOM IN
        if(myVal > 0){
        //Reduce height and width
64  myMapVBArray[2] = parseFloat(myMapVBArray[2]) / myVal;
    myMapVBArray[3] = parseFloat(myMapVBArray[3]) / myVal;
    //heeding the central point
    myMapVBArray[0] = parseFloat(myMapVBArray[0]) + myMapVBArray[2]* ((myVal - 1) / 2);
68  myMapVBArray[1] = parseFloat(myMapVBArray[1]) + myMapVBArray[3]* ((myVal - 1) / 2) ;
       }

        //ZOOM OUT
72      else {
    myMapVBArray[0] = parseFloat(myMapVBArray[0]) - myMapVBArray[2] *
    ((Math.abs(myVal) - 1) / 2) ;
    myMapVBArray[1] = parseFloat(myMapVBArray[1]) - myMapVBArray[3] *
76  ((Math.abs(myVal) - 1) / 2) ;
```

```
    //Widening of height and width
    myMapVBArray[2] = parseFloat(myMapVBArray[2]) * Math.abs(myVal);
    myMapVBArray[3] = parseFloat(myMapVBArray[3]) * Math.abs(myVal);
80      }

        //Test if still in main extension
        myMapVBArray = myCheckVB(myMapVBArray);
84      document.getElementById('myMap').setAttributeNS(null,'viewBox',myMapVBArray.join(' '));
        mySetOverviewRect(myMapVBArray);
          }

88  //*****************************************************************************
      //Resets the viewport to full map extension
          function myReset(){
        document.getElementById('myMap').setAttributeNS(null,'viewBox','725000 -268000 45000 55466');
92      mySetOverviewRect([725000, -268000, 45000, 55466]);
          }

    //*****************************************************************************
96    // Sets the sensitive rectangle on the overview map
      function mySetOverviewRect(myMapVBArray){ //
       var myRect = document.getElementById('myOverviewRect');
    myRect.setAttributeNS(null,'x',myMapVBArray[0]);
100 myRect.setAttributeNS(null,'y',myMapVBArray[1]);
    myRect.setAttributeNS(null,'width',myMapVBArray[2]);
    myRect.setAttributeNS(null,'height',myMapVBArray[3]);
    }
104
    //*****************************************************************************
      // Transforms screen into map coordinates
      function myCoordTransform(myCoordArray,myObjID){
108
       var myMatrix = null;

    try {
112 myMatrix = document.getElementById(myObjID).getScreenCTM();
    }
    //Does not work with AVS3! Does not support getScreenCTM()
    catch(e){
116
    var myInner2RootMatrix = document.documentElement.createSVGMatrix();
    var myViewBoxArray = document.getElementById(myObjID).getAttributeNS(null,'viewBox').split(' ');
    // Get the scaling factor
120 var myScal = parseFloat(myViewBoxArray[2]) /
     document.getElementById(myObjID).getAttributeNS(null,'width');
    // Set the transformation matrix manually
    myInner2RootMatrix.a = myScal;
124 myInner2RootMatrix.b = 0; // no rotation
    myInner2RootMatrix.c = 0;
    myInner2RootMatrix.d = myScal;
    myInner2RootMatrix.e = parseFloat(myViewBoxArray[0]) -
128 (myScal * document.getElementById(myObjID).getAttributeNS(null,'x'));
    myInner2RootMatrix.f = parseFloat(myViewBoxArray[1]) -
    (myScal * document.getElementById(myObjID).getAttributeNS(null,'y'));
    myRoot2ScreenMatrix = myGetScreenCTM(document.documentElement);
132 myMatrix = myRoot2ScreenMatrix.multiply(myInner2RootMatrix.inverse());
    }
    // Transformation applied
    if(myMatrix){
136 // Initialize fictive coordinate pair
    var mySVGPoint = document.documentElement.createSVGPoint();
    mySVGPoint.x = myCoordArray[0];
    mySVGPoint.y = myCoordArray[1];
140 mySVGPoint = mySVGPoint.matrixTransform(myMatrix.inverse());

    return [mySVGPoint.x,mySVGPoint.y];
    }
144 }

    //*****************************************************************************
```

```
  // Selecting screen coordinates and transform them directly
  // Disposing the coordinates in map coordiante system for real time display
  function myMouseCoords(evt){
var myTransformedCoords = myCoordTransform([evt.clientX,evt.clientY],'myMap');
document.getElementById('myCursorMapX').firstChild.data =
'".$lang["xcoor"]." ' + Math.round(myTransformedCoords[0]);
document.getElementById('myCursorMapY').firstChild.data =
'".$lang["ycoor"]." ' + Math.round(myTransformedCoords[1] * (-1));
  }

//*****************************************************************************
  // Moving of the sensitive area over the overview map
  var myDragLock = null;
  function myDrag(evt){

   if(evt.type == 'mousedown'){
//Safe starting position to get offset
myDragLock = myCoordTransform([evt.clientX,evt.clientY],'myOverviewMap');
var myRectNow = [];
myRectNow[0] = document.getElementById('myOverviewRect').getAttributeNS(null,'x');
myRectNow[1] = document.getElementById('myOverviewRect').getAttributeNS(null,'y');
myDragLock[0] -= myRectNow[0];
myDragLock[1] -= myRectNow[1];

} else if(evt.type == 'mousemove'){
  if(myDragLock){
  // Store coordinates of actual position
  var myTarget = myCoordTransform([evt.clientX,evt.clientY],'myOverviewMap');

  // Adding moved distance in correct coordinates
  document.getElementById('myOverviewRect').setAttributeNS
   (null,'x', myTarget[0] - myDragLock[0]);
  document.getElementById('myOverviewRect').setAttributeNS
   (null,'y', myTarget[1] - myDragLock[1]);
  } //endIf
  } else if(evt.type == 'mouseup' || evt.type == 'mouseout'){
     var myVBNowArray = document.getElementById('myMap').getAttributeNS(null,'viewBox').split(' ');
 myVBNowArray[0] = document.getElementById('myOverviewRect').getAttributeNS(null,'x');
 myVBNowArray[1] = document.getElementById('myOverviewRect').getAttributeNS(null,'y');
     //Check new values
     myVBNowArray = myCheckVB(myVBNowArray);
     document.getElementById('myOverviewRect').setAttributeNS(null,'x', myVBNowArray[0]);
     document.getElementById('myOverviewRect').setAttributeNS(null,'y', myVBNowArray[1]);
     document.getElementById('myMap').setAttributeNS(null,'viewBox',myVBNowArray.join(' '));
     myDragLock = null;

}
  }

//*****************************************************************************
// Implementing getScreenCTM() for ASV3 etc.
// THIS FUNCTION IS BASED ON THE WORK BY HOLGER WILL
// http://groups.yahoo.com/group/svg-developers/message/50789.
function myGetScreenCTM(doc){
var de = document.documentElement;
var sCTM = de.createSVGMatrix();
var tr = de.createSVGMatrix();
var par = null;
if(doc.hasAttributeNS(null,'preserveAspectRatio')){
par = doc.getAttributeNS(null,'preserveAspectRatio');
}
if (par == null || par == ''){
par='xMidYMid meet'; // setting to default value
}
var parX = par.substring(0,4); // xMin;xMid;xMax
var parY = par.substring(4,8); // YMin;YMid;YMax
var ma = par.split(' ');
var mos = ma[1]; // meet;slice
// set dimensions of the viewport
sCTM.a = 1;
```

```
    sCTM.d = 1;
    sCTM.e = 0;
    sCTM.f = 0;
220 // ## Batik doesn't know about innerWidth/innerHeight!
    var w = null;
    if(doc.hasAttributeNS(null,'width')){
    w = doc.getAttributeNS(null,'width');
224 }
    if (w == null || w == '' || w.match(/%$/)){
    w = innerWidth;
    }
228 var h = null
    if(doc.hasAttributeNS(null,'height')){
    h = doc.getAttributeNS(null,'height');
    }
232 if (h == null || h == '' || h.match(/%$/)){
    h = innerHeight;
    }
    var vba = null;
236 if(doc.hasAttributeNS(null,'viewBox')){
    vba = doc.getAttributeNS(null,'viewBox');
    }
    if(vba == null || vba == ''){
240 vba = '0 0 ' + w + ' ' + h;
    }
    var vb = vba.split(' '); // get the viewBox into an array
    // create a matrix with current user transformation
244 tr.a = de.currentScale;
    tr.d = de.currentScale;
    tr.e = de.currentTranslate.x;
    tr.f = de.currentTranslate.y;
248 //scale factors
    var sx = w / vb[2];
    var sy = h / vb[3];
    // preserveAspectRatio='none'
252 if (par == 'none'){
    sCTM.a = sx; // scaleX
    sCTM.d = sy; // scaleY
    sCTM.e = -vb[0] * sx; // translateX
256 sCTM.f = -vb[0] * sy; // translateY
    sCTM = tr.multiply(sCTM) ; // taking user transformations into acount
    return sCTM;
    }
260 //meetOrSlice
    var s = null;
    if(mos == 'slice'){
    s = (sx > sy ? sx : sy)
264 } else {
    s = (sx < sy ? sx : sy)
    }
    sCTM.a = s; // scaleX
268 sCTM.d = s; // scaleY
    switch(parX){
    case 'xMid':
    sCTM.e = ((w - vb[2]* s) / 2) - vb[0]*s; // translateX
272 break;
    case 'xMin':
    sCTM.e = -vb[0] * s; //translateX
    break;
276 case 'xMax':
    sCTM.e = (w - vb[2] * s) - vb[0] * s; // translateX
    break;
    }
280 switch(parY){
    case 'YMid':
    sCTM.f = ( h - vb[3] * s) / 2 - vb[1] * s; // translateY
    break;
284 case 'YMin':
    sCTM.f = -vb[1] * s; //translateY
    break;
```

```
    case 'YMax':
288 sCTM.f = ( h - vb[3] * s) - vb[1] * s; // translateY
    break;
    }
    sCTM = tr.multiply(sCTM); //taking user transformations into acount
292 return sCTM
    } // END OF getScreenCTM() FOR ASV3


296      ]]>

    ");

300 ?>
```

C.4.2 loadOfDetails.php

The AJAX implementation with the different cases for the different XML transmission methods supported by different browsers can be studied in this file. The code is in some parts leaned on the ideas given in Ueberschär and Winter (Ueberschär 2006).

```
    <?php

    echo("
 4       <![CDATA[
    // Delete content within a group container
    function myDelEle(myElem2Clear){
    myElem2Clear = document.getElementById(myElem2Clear);
 8  while(myElem2Clear.childNodes.length > 0){
    myElem2Clear.removeChild(myElem2Clear.firstChild);
    }
    }
12
    function myXMLAdd(myURL,myDestination,myEncoding){

        // Implementation for Adobe SVG Viewer
16   if (window.getURL){
    // alert(\"GetURL ENTRY\");
    getURL(myURL, myFileLoaded,myEncoding);
    }
20
        // Implementation for SVG Support in Firefox etc.
     else if (window.XMLHttpRequest) {
      // alert(\"HttpRequest ENTRY\");
24  var myReq = new XMLHttpRequest();
    if(myReq) {
          myReq.open('GET', myURL, true);
    myReq.onreadystatechange = function(){
28  //Check status of asynchronous load, 4 = fully loaded
    if (myReq.readyState == 4){
       document.getElementById(myDestination).
       appendChild(myReq.responseXML.documentElement);
32      }
    }
    myReq.send(null);
    }
36  }
        else {
    alert(\"".$cColors['nativeWithPlugin']."\");
    }
40
        function myFileLoaded(myData){
      var myString = '';
      if(myData.success) {
44      myString = myData.content;
    }
      else {
    alert(\"".$cColors['nativeWithPlugin']."\");
48  return;
    }
```

```
      document.getElementById(myDestination).
      appendChild(parseXML(myString, document));
52    }
    }

    //Main function dealing with deletes and with loads.
56  //To mark the status of a layer, the filling of the
    //layers rectangle is changed with the status
    function myLayer(myURL, myDest, myEncoding, myLayerID){
    if(!document.getElementById(myDest).firstChild){
60  myXMLAdd(myURL,myDest,myEncoding);
    document.getElementById(myLayerID).
    setAttributeNS(null, 'fill-opacity', '0');
    }
64  else {
    myDelEle(myDest);
    document.getElementById(myLayerID).
    setAttributeNS(null, 'fill-opacity', '1');
68  }
    }
        ]]>
    ");
72
    ?>
```

C.5 Scripts generating the layer's XML content

All files in this section do the same: they generate the XML code when executed. The purpose of each file is described in the first comment directly in the scripts.

C.5.1 LayerEnvpoly.php for the use with vector maps

```
    <?php
    //Layer printing the landmarks: Rivers, Lakes, cantonal frontiers

 4  header("Content-Type: image/svg+xml");

    //Constants initialization
    include("constants.php");
 8  $cDB = $LOCAL_CONS["DB"];
    $cColors = $LOCAL_CONS["LayerEdges"];

    // DATABASE:-----------------------------------------------
12  define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
    define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
16  define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
    define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
    define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    define ('cStrTABLEEnvvec',$cDB['cStrTABLEEnvvec']);
20  define ('cStrTABLEEnvpoly',$cDB['cStrTABLEEnvpoly']);
    // -------------------------------------------------------

    //DB Connection
24  $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);

    echo ('<g xmlns="http://www.w3.org/2000/svg"
28    xmlns:xlink="http://www.w3.org/1999/xlink">');

    // SELECT Rivers
        $eanfrage = "SELECT svg FROM ".cStrTABLEEnvvec." WHERE kind = 'river'";
32    $nodes = mysql_query($eanfrage);

      echo ('<g id="myRivers" stroke="#0A4C84"
    stroke-width="45" fill="none"
36  stroke-linecap="butt" stroke-linejoin="round">
```

```
    <g id="innerMyRivers">
40     <path d="');
          while($ezeile = mysql_fetch_row($nodes)){
          echo ('.$ezeile[0].'');
          } //endWhile
44        echo('"/>'); //endPath
    echo ('</g>
       <use id="myRiversMiddle" xlink:href="#innerMyRivers" stroke="#AFD9FD"
        stroke-width="32"/>
48      </g>');

    // SELECT Cantons
          $eanfrage = "SELECT svg FROM ".cStrTABLEEnvpoly." WHERE kind = 'canton'";
52    $nodes = mysql_query($eanfrage);

      echo ('<g id="myCanton" stroke="#B8FF83"
    stroke-width="80" fill="none"
56  stroke-linecap="butt" stroke-linejoin="round">
    <g id="innerMyCantons">
       <path d="');
          while($ezeile = mysql_fetch_row($nodes)){
60        echo ('.$ezeile[0].'');
          } //endWhile
          echo('"/>'); //endPath
    echo ('</g>
64     <use id="myCantonMiddle" xlink:href="#innerMyCantons" stroke="#388500"
        stroke-width="15"/>
        </g>');

68  // SELECT Lakes
          $eanfrage = "SELECT svg FROM ".cStrTABLEEnvpoly." WHERE kind = 'lake'";
      $nodes = mysql_query($eanfrage);

72    echo ('<g id="myLakes" stroke="#0A4C84"
    stroke-width="35" fill="#AFD9FD"
    stroke-linecap="butt" stroke-linejoin="round">
       ');
76        while($ezeile = mysql_fetch_row($nodes)){
          echo ('<polygon points="'.$ezeile[0].'"/>');
          } //endWhile
          echo(''); //endPoly
80  echo ('</g>');
    echo('<g font-style="italic" font-family="Times,sans-serif" letter-spacing="200">
    <text  x="753000" y="-264850" id="LakeName1" fill="#0A4C84"  font-size="1200pt">
    B O D E N S E E</text>
84  <text x="731000" y="-220500" id="LakeName2" fill="#0A4C84" font-size="800pt">
    W A L E N S E E</text>
      </g>');

88  echo ('</g>');
    ?>
```

C.5.2 LayerEnvpolySat.php for the use with aerial images

```
    <?php
    //Layer printing the landmarks: Rivers, Lakes, cantonal frontiers
    //For the use with aerial images!!
 4
    header("Content-Type: image/svg+xml");

    //Constants initialization
 8  include("../constants.php");
    $cDB = $LOCAL_CONS["DB"];
    $cColors = $LOCAL_CONS["LayerEdges"];

12  // DATABASE:-----------------------------------------------
    define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
    define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
```

```
define ('cStrDB',$cDB['cStrDB']);
define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
define ('cStrTABLEEnvvec',$cDB['cStrTABLEEnvvec']);
define ('cStrTABLEEnvpoly',$cDB['cStrTABLEEnvpoly']);
// ------------------------------------------------------

//DB Connection
$db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
mysql_select_db(cStrDB);

echo ('<g xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">');

// SELECT Cantons
      $eanfrage = "SELECT svg FROM ".cStrTABLEEnvpoly." WHERE kind = 'canton'";
  $nodes = mysql_query($eanfrage);

  echo ('<g id="myCanton" stroke="#B8FF83"
stroke-width="80" fill="none"
stroke-linecap="butt" stroke-linejoin="round">
<g id="innerMyCantons">
   <path d="');
      while($ezeile = mysql_fetch_row($nodes)){
      echo ('.$ezeile[0].'');
      } //endWhile
      echo('"/>'); //endPath
echo ('</g>
   <use id="myCantonMiddle" xlink:href="#innerMyCantons" stroke="#388500"
    stroke-width="15"/>
    </g>');

// SELECT Lakes
      $eanfrage = "SELECT svg FROM ".cStrTABLEEnvpoly." WHERE kind = 'lake'";
  $nodes = mysql_query($eanfrage);

  echo ('<g id="myLakes" stroke="#0A4C84"
stroke-width="35" fill="#AFD9FD"
stroke-linecap="butt" stroke-linejoin="round">
   ');
      while($ezeile = mysql_fetch_row($nodes)){
      echo ('<polygon points="'.$ezeile[0].'"/>');
      } //endWhile
      echo(''); //endPoly
echo ('</g>');
echo('<g font-style="italic" font-family="Times,sans-serif" letter-spacing="200">
<text  x="753000" y="-264850" id="LakeName1" fill="#0A4C84"  font-size="1200pt">
B O D E N S E E</text>
<text x="731000" y="-220500" id="LakeName2" fill="#0A4C84" font-size="800pt">
W A L E N S E E</text>
  </g>');

echo ('</g>');
?>
```

C.5.3 LayerEnvcomm.php

```
<?php
//Layer printing the communal border and the names on
//the center of gravity

header("Content-Type: image/svg+xml");

//Constants initialization
include("constants.php");
$cDB = $LOCAL_CONS["DB"];
$cColors = $LOCAL_CONS["LayerEdges"];

// DATABASE:-----------------------------------------
```

```
define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
define ('cStrUSERDB',$cDB['cStrUSERDB']);
define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
define ('cStrDB',$cDB['cStrDB']);
define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
define ('cStrTABLEEnvvec',$cDB['cStrTABLEEnvvec']);
define ('cStrTABLEEnvpoly',$cDB['cStrTABLEEnvpoly']);
// ------------------------------------------------------

//DB Connection
$db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
mysql_select_db(cStrDB);

echo ('<g xmlns="http://www.w3.org/2000/svg"
        xmlns:xlink="http://www.w3.org/1999/xlink">');

// Rectangle covering the underlaying borders that do not exactly
// match these borders.
echo ("<rect id=\"myOverlap\" stroke=\"white\" stroke-width=\"0\"
fill=\"white\" x=\"725000\" y=\"-268000\" width=\"45000\" height=\"55466\"/>");

// SELECT Communities
      $eanfrage = "SELECT svg FROM ".cStrTABLEEnvpoly." WHERE kind = 'community'";
  $nodes = mysql_query($eanfrage);

  echo ('<g id="myComm" stroke="#444444" stroke-dasharray="80,80"
stroke-width="7" fill="none"
stroke-linecap="butt" dasharray="5,15" stroke-linejoin="round">
   <path d="');
      while($ezeile = mysql_fetch_row($nodes)){
      echo (''.$ezeile[0].'');
      } //endWhile
      echo('"/>'); //endPath
echo ('</g>');

//Display Comm names
echo ('<g id="CommNames"
        font-size="150" letter-spacing="5pt" font-weight="bold"
        opacity="0.5" font-family="Times,sans-serif" text-anchor="middle">

<text x="741431" y="-264587">Muolen</text>
<text x="743339" y="-262146">H&#228;ggenschwil</text>
<text x="750438" y="-262308">Steinach</text>
<text x="759645" y="-260775">Thal</text>
<text x="730170" y="-260056">Niederhelfenschwil</text>
<text x="748672" y="-261176">Berg</text>
<text x="746321" y="-259129">Wittenbach</text>
<text x="751882" y="-261071">T&#252;bach</text>
<text x="752959" y="-260074">Goldach</text>
<text x="761676" y="-259906">Rheineck</text>
<text x="740556" y="-258713">Waldkirch</text>
<text x="749832" y="-259320">M&#246;rschwil</text>
<text x="755726" y="-259360">Rorschacherberg</text>
<text x="754989" y="-260350">Rorschach</text>
<text x="733936" y="-258295">Niederb&#252;ren</text>
<text x="725404" y="-259161">Zuzwil</text>
<text x="764937" y="-258134">St. Margrethen</text>
<text x="754122" y="-257113">Eggersriet</text>
<text x="752029" y="-257847">Untereggen</text>
<text x="731432" y="-256048">Oberb&#252;ren</text>
<text x="726596" y="-256465">Uzwil</text>
<text x="742818" y="-255577">Gaiserwald</text>
<text x="738965" y="-256055">Andwil</text>
<text x="746142" y="-254174">St. Gallen</text>
<text x="736948" y="-253828">Gossau</text>
<text x="765843" y="-255153">Au</text>
<text x="764012" y="-254922">Berneck</text>
<text x="728435" y="-253697">Oberuzwil</text>
```

```
    <text x="732769" y="-251988">Flawil</text>
 84 <text x="766082" y="-252761">Widnau</text>
    <text x="763741" y="-252270">Balgach</text>
    <text x="761786" y="-252021">Rebstein</text>
    <text x="725760" y="-250828">L&#252;tisburg</text>
 88 <text x="766732" y="-250476">Diepoldsau</text>
    <text x="731970" y="-249725">Degersheim</text>
    <text x="761613" y="-250592">Marbach</text>
    <text x="759009" y="-249000">Altst&#228;tten</text>
 92 <text x="760583" y="-244843">Oberriet</text>
    <text x="729941" y="-246477">Mogelsberg</text>
    <text x="725129" y="-248340">Ganterschwil</text>
    <text x="726411" y="-245169">Oberhelfenschwil</text>
 96 <text x="757282" y="-246186">Eichberg</text>
    <text x="733133" y="-243529">St. Peterzell</text>
    <text x="728610" y="-242354">Brunnadern</text>
    <text x="725321" y="-242871">Lichtensteig</text>
100 <text x="732178" y="-239871">Hemberg</text>
    <text x="758446" y="-240346">R&#252;thi</text>
    <text x="754296" y="-234558">Sennwald</text>
    <text x="744986" y="-231318">Wildhaus</text>
104 <text x="750861" y="-230871">Gams</text>
    <text x="735704" y="-229415">Stein</text>
    <text x="753534" y="-225426">Buchs</text>
    <text x="748273" y="-225997">Grabs</text>
108 <text x="730997" y="-224690">Amden</text>
    <text x="725502" y="-222623">Weesen</text>
    <text x="753231" y="-221827">Sevelen</text>
    <text x="743485" y="-221166">Walenstadt</text>
112 <text x="735333" y="-216350">Quarten</text>
    <text x="753095" y="-217734">Wartau</text>
    <text x="741171" y="-213752">Flums</text>
    <text x="752713" y="-213576">Sargans</text>
116         </g>');
    echo ('</g>');
    ?>
```

C.5.4 LayerEnvvec.php

```
    <?php
    //Layer printing the regional traffic lines

  4 header("Content-Type: image/svg+xml");

    //Constants initialization
    include("constants.php");
  8 $cDB = $LOCAL_CONS["DB"];
    $cColors = $LOCAL_CONS["LayerEdges"];

    // DATABASE:-----------------------------------------------
 12 define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
    define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
 16 define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
    define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
    define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    define ('cStrTABLEEnvvec',$cDB['cStrTABLEEnvvec']);
 20 // ----------------------------------------------------

    //DB Connection
    $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
 24 mysql_select_db(cStrDB);

    echo ('<g xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink">');
 28
    // SELECT Highways
          $eanfrage = "SELECT svg FROM ".cStrTABLEEnvvec." WHERE kind = 'highway'";
      $nodes = mysql_query($eanfrage);
```

```
32
      echo ('<g id="myHiway" stroke="#000000"
    stroke-width="15" fill="none"
    stroke-linecap="butt" stroke-linejoin="round">
36  <g id="innerMyHiway">
       <path d="');
          while($ezeile = mysql_fetch_row($nodes)){
          echo (''.$ezeile[0].'');
40        } //endWhile
          echo('"/>'); //endPath
    echo ('</g>
       <use id="myHiwayMiddle" xlink:href="#innerMyHiway" stroke="#FFFFFF"
44      stroke-width="10"/>
        </g>');


48  // SELECT Rail
        $eanfrage = "SELECT svg FROM ".cStrTABLEEnvvec." WHERE kind = 'rail'";
      $nodes = mysql_query($eanfrage);

52    echo ('<g id="mySBB" stroke="#5B2D00"
    stroke-width="25" fill="none"
    stroke-linecap="butt" stroke-linejoin="round">
    <g id="innerMyRail">
56     <path d="');
          while($ezeile = mysql_fetch_row($nodes)){
          echo (''.$ezeile[0].'');
          } //endWhile
60        echo('"/>'); //endPath
    echo ('</g>
       <use id="myRailMiddle" xlink:href="#innerMyRail" stroke="#FFFFFF"
        stroke-width="15" stroke-dasharray="200,200"/>
64      ');
    echo ('</g>');

    // SELECT International Roads
68        $eanfrage = "SELECT svg FROM ".cStrTABLEEnvvec." WHERE kind = 'intRoad'";
      $nodes = mysql_query($eanfrage);

      echo ('<g id="myInt" stroke="#000000"
72  stroke-width="40" fill="none"
    stroke-linecap="butt" stroke-linejoin="round">
    <g id="innerMyInt">
       <path d="');
76        while($ezeile = mysql_fetch_row($nodes)){
          echo (''.$ezeile[0].'');
          } //endWhile
          echo('"/>'); //endPath
80  echo ('</g>
       <use id="myIntMiddle1" xlink:href="#innerMyInt" stroke="#FFFFFF"
        stroke-width="28"/>
       <use id="myIntMiddle2" xlink:href="#innerMyInt" stroke="#000000"
84      stroke-width="3"/>
        </g>');

    echo ('</g>');
88  ?>
```

C.5.5 LayerEdges.php for the use with vector maps

```
    <?php
    //Layer loading the hiking network as linear pathes

 4  header("Content-Type: image/svg+xml");

    //Constants initialization
    include("constants.php");
 8  $cDB = $LOCAL_CONS["DB"];
    $cColors = $LOCAL_CONS["LayerEdges"];
```

```
// DATABASE:-----------------------------------------------
define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
define ('cStrUSERDB',$cDB['cStrUSERDB']);
define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
define ('cStrDB',$cDB['cStrDB']);
define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
// -----------------------------------------------------

//DB Connection
$db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
mysql_select_db(cStrDB);

// SELECT Edges
      $eanfrage = "SELECT sx, sy, ex, ey FROM ".cStrTABLEEdges."";
  $nodes = mysql_query($eanfrage);
  echo ('<g xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink" id="myEdges"
   fill="none" stroke="'.$cColors["edgeColor"].'"
   stroke-width="'.$cColors["edgeWidth"].'" stroke-linecap="round"><path d="');
    while($ezeile = mysql_fetch_row($nodes)){
    echo ('M '.round($ezeile[0]).' -'.round($ezeile[1]).'
     L '.round($ezeile[2]).' -'.round($ezeile[3]).'');
    }
     echo('"/></g>');
?>
```

C.5.6 LayerEdgesSat.php for the use with aerial images

```
<?php
//Layer loading the hiking network as linear pathes
//For the use with aerial images in the background
//better contrasting network!!

header("Content-Type: image/svg+xml");

//Constants initialization
include("../constants.php");
$cDB = $LOCAL_CONS["DB"];
$cColors = $LOCAL_CONS["LayerEdges"];

// DATABASE:-----------------------------------------------
define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
define ('cStrUSERDB',$cDB['cStrUSERDB']);
define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
define ('cStrDB',$cDB['cStrDB']);
define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
// -----------------------------------------------------

//DB Connection
$db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
mysql_select_db(cStrDB);

// SELECT Edges
      $eanfrage = "SELECT sx, sy, ex, ey FROM ".cStrTABLEEdges."";
  $nodes = mysql_query($eanfrage);
  echo ('<g xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink" id="myEdges" fill="none"
  stroke="black" stroke-width="17" stroke-linecap="round">
  <g id="myInnerEdges">
  <path d="');
     while($ezeile = mysql_fetch_row($nodes)){
     echo ('M '.round($ezeile[0]).' -'.round($ezeile[1]).'
      L '.round($ezeile[2]).' -'.round($ezeile[3]).'');
     }
      echo('"/></g>
      <use id="myEdgesOuter" xlink:href="#myInnerEdges" stroke="red"
```

```
        stroke-width="13"/></g>
          ');

?>
```

C.5.7 LayerNodesSecondary.php

```
<?php
//Layer placing the subnodes connecting edges
//without signpost location as nodes

header("Content-Type: image/svg+xml");

//Constants initialization
include("constants.php");
$cDB = $LOCAL_CONS["DB"];
$cColors = $LOCAL_CONS["LayerNodesSecondary"];

// DATABASE:-----------------------------------------------
define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
define ('cStrUSERDB',$cDB['cStrUSERDB']);
define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
define ('cStrDB',$cDB['cStrDB']);
define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
// -------------------------------------------------------

//DB Connection
$db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
mysql_select_db(cStrDB);

//Select Nodes
     $eanfrage = "SELECT n_x, n_y FROM ".cStrTABLENodes." WHERE type = 3";
  $nodes = mysql_query($eanfrage);

  echo ('<g xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink"
   fill="'.$cColors["nodeFill"].'" stroke="'.$cColors["nodeStrokeColor"].'"
   stroke-width="'.$cColors["nodeStrokeWidth"].'">');
     while($ezeile = mysql_fetch_row($nodes)){
     echo ('<circle id="Ort_3" cx="'.round($ezeile[0]).'"
      cy="-'.round($ezeile[1]).'" r="'.$cColors["nodeSize"].'"/>');
      } //endWhile

 echo('</g>');
?>
```

C.5.8 LayerNodesSignpost.php

```
<?php
//Layer printing the signpost locations and the
//important nodes from the topology

header("Content-Type: image/svg+xml");

//Constants initialization
include("constants.php");
$cDB = $LOCAL_CONS["DB"];
$cColors = $LOCAL_CONS["LayerNodesSignpost"];

// DATABASE:-----------------------------------------------
define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
define ('cStrUSERDB',$cDB['cStrUSERDB']);
define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
define ('cStrDB',$cDB['cStrDB']);
define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
// -------------------------------------------------------
```

```
    //DB Connection
    $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
24  mysql_select_db(cStrDB);


    //Select Nodes type = 2
28        $eanfrage = "SELECT n_x, n_y, n_municipality
             FROM ".cStrTABLENodes." WHERE type = 2";
      $nodes = mysql_query($eanfrage);
      echo ('<g xmlns="http://www.w3.org/2000/svg"
32     xmlns:xlink="http://www.w3.org/1999/xlink"
       fill="'.$cColors["nodeFill"].'" stroke="'.$cColors["nodeStrokeColor"].'"
       stroke-width="'.$cColors["nodeStrokeWidth"].'">');

36       while($ezeile = mysql_fetch_row($nodes)){
         echo ('<circle cx="'.round($ezeile[0]).'"
          cy="-'.round($ezeile[1]).'" r="'.$cColors["nodeSize"].'"/>');
         } //endWhile
40

      echo('</g>');
    ?>
```

C.5.9 LayerNodeNames.php

```
    <?php
    //Layer placing the location's names

 4  header("Content-Type: image/svg+xml");

    //Constants initialization
    include("constants.php");
 8  $cDB = $LOCAL_CONS["DB"];
    $cColors = $LOCAL_CONS["LayerNodeNames"];

    // DATABASE:-----------------------------------------------
12  define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
    define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
16  define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
    define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
    define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    // -------------------------------------------------------
20
    //DB Connection
    $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);
24
    //SELECT Names
          $eanfrage = "SELECT n_x, n_y, n_location FROM ".cStrTABLENodes." WHERE type = 2";
      $nodes = mysql_query($eanfrage);
28    echo ('<g xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"
          text-anchor="'.$cColors["textAnchor"].'" fill="'.$cColors["textFill"].'"
          transform="translate('.$cColors["translatex"].' '.$cColors["translatey"].')"
          font-size="'.$cColors["fontSize"].'">');
32       while($ezeile = mysql_fetch_row($nodes)){
         $y1 = (-1)*round($ezeile[1]);
         $x1 = round($ezeile[0]);
     echo ('<text x="'.$x1.'" y="'.$y1.'">'.htmlspecialchars($ezeile[2]).'</text>');
36     }
      echo('</g>');
    ?>
```

C.5.10 getPathMap.php

```
    <?php
    //Layer producing all necessary data for the calculated path
    //Data transmission from Dijkstra's algorithm via SESSIONS.
 4
```

```
    header("Content-Type: image/svg+xml");

    //Session load
 8  session_start();

    //Constants initialization
    include("constants.php");
12  $cDB = $LOCAL_CONS["DB"];
    $cColors = $LOCAL_CONS["ResultLayer"];

    // DATABASE:-----------------------------------------------
16  define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
    define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
20  define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
    define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
    define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    define ('cStrTABLEUserQ',$cDB['cStrTABLEUserQ']);
24  // -------------------------------------------------------

    function getDataPath($input){

28  $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);
    //Clean the result string of unwanted doubles through via's
    //by eliminating Elements containing "xxx"
32  $stripInput = preg_replace('/,xxx(\d+)/', '', $input);
    //echo $stripInput;
    $path = explode(",",$stripInput);
    $length = count($path);
36  $dataPath = array();
    //Begin for buckle with i=1 because in Element 0 the costFactor is stored.
    for ($i = 1; $i <= $length; $i++){
    //Look up data for node
40  $dNode = "SELECT n_x, n_y, n_height, n_location
         FROM ".cStrTABLENodes." WHERE n_id = ".$path[$i]."";
    $res = mysql_query($dNode);
    $n1 = mysql_fetch_row($res);
44  //Look up data for Edge
    $pNode = "SELECT
     baseedges.svg FROM ".cStrTABLEEdges.", ".cStrTABLETopo." WHERE
         ".cStrTABLEEdges.".id = ".cStrTABLETopo.".edge
48       AND ".cStrTABLETopo.".from_node = ".$path[$i]."
         AND ".cStrTABLETopo.".to_node = ".$path[$i+1]."";
    $res2 = mysql_query($pNode);
    $p1 = mysql_fetch_row($res2);
52  //Store data in an multidimesional, assoziative array
    $dataPath[$i-1] = array("n_x" => $n1[0], "n_y" => $n1[1], "n_height" => $n1[2],
      "n_location" => $n1[3], "svg" => $p1[0]);
    //Tests
56  //$inhalt = $dataPath[$i-1]["time"];
         //$inhalt .= $dataPath[$i-1]["n_location"]."<br>";
         //echo $inhalt;
         }
60       mysql_close($db);
         return ($dataPath);
    }

64  $mapdata = getDataPath($_SESSION['totalPath']);
    //echo ($_SESSION['totalPath']);

    //Create the Layer with the map on
68  $text = '';
    $mappath = '';
    $nodes = '';
    for($x=0; $x < count($mapdata)-1; $x++)
72       {
         $i = $mapdata[$x];
            $text .= '<text x="'.$i["n_x"].'" y="-'.$i["n_y"].'">'.
```

```
            htmlspecialchars($i["n_location"]).'</text>';
76          $nodes .= '<circle cx="'.$i["n_x"].'" cy="-'.$i["n_y"].'"
             r="'.$cColors["resNodeSize"].'"/>';
            $mappath .= $i["svg"];
        }
80
    //Generate the XML
    echo ('<g xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink">');
84  echo ('<g id="myResPath" stroke="'.$cColors["resPathOuter"].'"
    stroke-width="'.$cColors["resPathOuterStroke"].'" fill="none"
    stroke-linecap="butt" stroke-linejoin="round">
    <g id="innerMyResPath">
88  <path d="'.$mappath.'"/>
    </g>
       <use id="myResPathMiddle" xlink:href="#innerMyResPath"
       stroke="'.$cColors["resPathInner"].'"
92      stroke-width="'.$cColors["resPathInnerStroke"].'"/>
          </g>');
    echo ('<g id="myResText" transform="translate('.$cColors["translatex"].'
    '.$cColors["translatey"].')" font-size="'.$cColors["resTextSize"].'"
96  fill="'.$cColors["resTextFill"].'">
    '.$text.'
          </g>');
    echo ('<g id="myResNodes" stroke="'.$cColors["resNodeStrokeColor"].'"
100   stroke-width="'.$cColors["resNodeStrokeWidth"].'" fill="'.$cColors["resNodeFill"].'">
    '.$nodes.'
          </g>');
    echo('</g>');
104
    ?>
```

C.5.11 aerialMap.php

The layer showing the aerial images in the background for the calculated path.

```
    <?php
    header("Content-Type: image/svg+xml");

 4  //Session load
    session_start();

    //Constants initialization
 8  include("../constants.php");
    $cDB = $LOCAL_CONS["DB"];
    $cColors = $LOCAL_CONS["Colors"];
    $cMAP = $LOCAL_CONS["MAP"];
12
    //Language selection
    include("../locallang.php");
    $langID = $_REQUEST['L'];
16  if ($langID == 1){
    $langcode = "en";}
    else if ($langID == 0){
    $langcode = "default";}
20  else {$langcode = "default";}
    $lang = $LOCAL_LANG[$langcode];

    // DATABASE:-----------------------------------------------
24  define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
    define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
28  define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
    define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
    define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    define ('cStrTABLEUserQ',$cDB['cStrTABLEUserQ']);
32  // -------------------------------------------------------

    $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);
```

```
36
    //include getSat methods for satellite images matching the resulted path
    function getDataSat($input){

40  //Clean the result string of unwanted doubles through via's
    //by eliminating Elements containing "xxx"
    $stripInput = preg_replace('/,xxx(\d+)/', '', $input);
    //echo $stripInput;
44  $path = explode(",",$stripInput);
    $length = count($path);
    $dataPath = array();
    //Begin for buckle with i=1 because in Element 0 the costFactor is stored.
48  for ($i = 1; $i < $length; $i++){
    //Look up data for node
    $dNode = "SELECT n_x, n_y
    FROM ".cStrTABLENodes."
52  WHERE n_id = ".$path[$i]."";
    $res = mysql_query($dNode);
    $n1 = mysql_fetch_row($res);
    //transform for file names
56  $tempx = (floor((floor($n1[0]/1000)-6)/8)*8)+6;
    $tempy = (floor((floor($n1[1]/1000)-6)/8)*8)+6+8;
    //Store data in an multidimesional, assoziative array
    $dataPath[$i-1] = "".$tempx.",".$tempy."";
60       }
         $res = array_unique($dataPath);
         $inner = 0;
         foreach ($res as $i) {
64      if ($i != "") {
           $ret[$inner] = $i;
           $inner++;
         }
68  }
    //echo (count($res));
         //print_r($res);
         return ($ret);
72  }

    function getSat($sat){
    $img = "";
76  foreach($sat as $i){
    $a = explode(',',$i);
    $ax = $a[0]*1000;
    $ay = $a[1]*(-1000);
80  $img .= "<image xlink:href=\"".$a[0]."_".$a[1]."_2m.jpg\"
    x=\"".$ax."\" y=\"".$ay."\" width=\"8000\" height=\"8000\" />";
    }
    return $img;
84  }

    function together($r){
    $o = getSat(getDataSat($r));
88  return $o;}
    $img = together($_SESSION['totalPath']);


92  //---------------------------------------------------------------
    // Get the viewport focussed on the calculated Path, if there is one.
    if($_SESSION['totalPath'] != ''){
    //Get coordinates from start and end
96  $PathArray = explode(',',$_SESSION['totalPath']);
    $qLoc = "SELECT n_x, n_y
    FROM ".cStrTABLENodes."
    WHERE ".$_SESSION['FromID']." = n_id
100 OR ".$_SESSION['ToID']." = n_id";
    $res = mysql_query($qLoc);
    $n1 = mysql_fetch_row($res);
    $n2 = mysql_fetch_row($res);
104 $offset = $cMAP['offsetFocus'];  //Widening in meters
    $xDia = abs($n1[0]-$n2[0]);
```

```
$yDia = abs($n1[1]-$n2[1]);
if($yDia*$cMAP['xDia']/$cMAP['yDia'] > $xDia){
$yDia = $yDia+$cMAP['offsetFocus'];
$xDia = $yDia*($cMAP['xDia']/$cMAP['yDia']);
if($n1[1]<$n2[1]){$cornery = $n2[1]*(-1)-($cMAP['offsetFocus']/2);}
else{$cornery = $n1[1]*(-1)-($cMAP['offsetFocus']/2);}
$cornerx = (($n1[0]+$n2[0])/2)-($xDia/2);
}
else{
$xDia = $xDia + $cMAP['offsetFocus'];
$yDia = $xDia/($cMAP['xDia']/$cMAP['yDia']);
if($n1[0]<$n2[0]){$cornerx = $n1[0]-($cMAP['offsetFocus']/2);}
else{$cornerx = $n2[0]-($cMAP['offsetFocus']/2);}
$cornery = ((($n1[1]+$n2[1])/2)+($yDia/2))*(-1);
}
}
else{
$xDia = $cMAP['xDia'];
$yDia = $cMAP['yDia'];
$cornerx = $cMAP['xC'];
$cornery = $cMAP['yC'];
} //endElse

//No transit over the Map borders
if($cornerx > $cMAP['xC']+$cMAP['xDia']-$xDia){
$cornerx = $cMAP['xC']+$cMAP['xDia']-$xDia;
}
else if($cornerx < $cMAP['xC']){
$cornerx = $cMAP['xC'];
}
else if($cornery < $cMAP['yC']){
$cornery = $cMAP['yC'];
}
else if($cornery > $cMAP['yC']+$cMAP['yDia']-$yDia){
$cornery = $cMAP['yC']+$cMAP['yDia']-$yDia;
}


echo("
<svg onmousemove=\"myMouseCoords(evt)\" zoomAndPan=\"disable\"
viewBox=\"0 0 560 480\" width=\"100%\" height=\"100%\"
xmlns=\"http://www.w3.org/2000/svg\" version=\"1.0\"
xmlns:xlink=\"http://www.w3.org/1999/xlink\">
<title>".$lang["title"]."</title>
<defs>

<symbol id=\"mySymbArrow\" overflow=\"visible\" fill=\"none\"
stroke=\"".$cColors["symbAndBorder"]."\" stroke-width=\"1\">
  <polyline points=\"-3,-5 3,-5 3,1 5,1 0,5 -5,1 -3,1 -3,-5\"/>
</symbol>

<script>");
include("../naviJS.php");
include("../loadOfDetails.php");

//The Reset fct to get the focus back on the calculated track
//has to be definied locally
echo("
       function myResetCalc(){
      document.getElementById('myMap').setAttributeNS(null,'viewBox','".$cornerx."
      ".$cornery." ".$xDia." ".$yDia."');
      mySetOverviewRect([".$cornerx.", ".$cornery.", ".$xDia.", ".$yDia."]);
        }
        ");

   echo ("</script>
  </defs>

<rect id=\"backGroundDelimiter\" x=\"0\" y=\"0\" width=\"562\"
height=\"485\" fill=\"".$cColors["bgProject"]."\"/>
```

```
176    <rect id=\"mapAreaDelimiter\" x=\"171\" y=\"4\" width=\"384\"
       height=\"473\" fill=\"".$cColors["symbAndBorder"]."\"/>

     <svg id=\"myMap\" viewBox=\"".$cornerx." ".$cornery." ".$xDia." ".$yDia."\"
180  x=\"172\" y=\"5\" width=\"382\" height=\"471\">
     <rect id=\"mapBG\" x=\"".$cMAP['xC']."\" y=\"".$cMAP['yC']."\"
     width=\"".$cMAP['xDia']."\" height=\"".$cMAP['yDia']."\" fill=\"".$cColors["bgMap"]."\"/>

184  <image id=\"myTopo\" xlink:href=\"../Topo5.jpg\" x=\"725000\"
     y=\"-267500\" width=\"".$cMAP['xDia']."\" height=\"".$cMAP['yDia']."\"/>

     <g id=\"myPassiveElements\">");
188  //Include the satellite images matching the result
     echo (''.$img.'');

     //Layers to copy by user input into this viewport
192  echo("
     <g id='LayerGroup1'>
     <g id='LayerEnvpoly'/>
     <g id='useEnvpoly'> </g>
196  <g id='LayerEnvcomm'/>
     <g id='LayerEnvvec'/>
     <g id='LayerEdges'/>
     <g id='LayerNodesSignpost'/>
200  <g id='LayerNodesSecondary'/>
     <g id='LayerNodeNames'/>
     <g id='LayerRes'/>
     <g id='LayerNodeCommunities'/>
204  </g>");


     echo ("
208
     <g id=\"myMapElementsOverTheGrid\">
     <g id=\"myScaleBar\">
     <path id=\"strecke\" stroke=\"black\" stroke-width=\"100\"
212  fill=\"none\" d=\"M727000 -214800 v 400 h 5000 v -400\"/>
     <text id=\"m_text\" fill=\"black\" font-size=\"1000\"
     font-weight=\"bold\" x=\"729500\" y=\"-213300\" text-anchor=\"middle\">5000 m</text>
     </g>
216  </g>
     </g>
     </svg>

220  <rect id=\"infoPanelDelimiter\" x=\"20\" y=\"5\" width=\"132\"
     height=\"471\" fill=\"".$cColors["bgInfoPanel"]."\"/>
     <g id=\"myNavig\" transform=\"translate(30,9) scale(1.6)\"
     stroke=\"".$cColors["symbAndBorder"]."\" stroke-width=\"1\"
224  onmouseover=\"evt.target.setAttributeNS(null,'fill','".$cColors["bgInfoPanel"]."')\"
     cursor=\"pointer\" onmouseout=\"evt.target.setAttributeNS(null,'fill','".$cColors["navMouseout"]."')\">
     <g shape-rendering=\"crispEdges\" fill=\"".$cColors["navMouseout"]."\">
     <rect id=\"myPanNW\" x=\"0\" y=\"0\" width=\"20\" height=\"20\" onclick=\"myPan(-1,-1)\"/>
228  <rect id=\"myPanNN\" x=\"25\" y=\"0\" width=\"20\" height=\"20\" onclick=\"myPan(0,-1)\"/>
     <rect id=\"myPanNE\" x=\"50\" y=\"0\" width=\"20\" height=\"20\" onclick=\"myPan(1,-1)\"/>
     <rect id=\"myPanWW\" x=\"0\" y=\"25\" width=\"20\" height=\"20\" onclick=\"myPan(-1,0)\"/>
     <rect id=\"myResetAll\" x=\"25\" y=\"25\" width=\"20\" height=\"20\" onclick=\"myResetCalc()\"/>
232  <rect id=\"myPanEE\" x=\"50\" y=\"25\" width=\"20\" height=\"20\" onclick=\"myPan(1,0)\"/>
     <rect id=\"myPanSW\" x=\"0\" y=\"50\" width=\"20\" height=\"20\" onclick=\"myPan(-1,1)\"/>
     <rect id=\"myPanSS\" x=\"25\" y=\"50\" width=\"20\" height=\"20\" onclick=\"myPan(0,1)\"/>
     <rect id=\"myPanSE\" x=\"50\" y=\"50\" width=\"20\" height=\"20\" onclick=\"myPan(1,1)\"/>
236  </g>
     <circle cx=\"10\" cy=\"84\" r=\"10\" fill=\"white\" onclick=\"myZoom(1)\"/>
     <circle cx=\"35\" cy=\"84\" r=\"10\" fill=\"white\" onclick=\"myZoom(-1)\"/>
     <circle cx=\"60\" cy=\"84\" r=\"10\" fill=\"white\" onclick=\"myReset()\"/>
240  <g id=\"myNavigSymbols\" pointer-events=\"none\">
     <use xlink:href=\"#mySymbArrow\" transform=\"translate(10,10),rotate(135)\" />
     <use xlink:href=\"#mySymbArrow\" transform=\"translate(35,10),rotate(180)\" />
     <use xlink:href=\"#mySymbArrow\" transform=\"translate(60,10),rotate(-135)\" />
244  <use xlink:href=\"#mySymbArrow\" transform=\"translate(10,35),rotate(90)\" />
     <circle cx=\"35\" cy=\"35\" r=\"7.5\" fill=\"none\"
```

```
stroke=\"".$cColors["symbAndBorder"]."\" stroke-width=\"1\"/>
<circle cx=\"35\" cy=\"35\" r=\"5\" fill=\"".$cColors["symbAndBorder"]."\"
stroke=\"".$cColors["symbAndBorder"]."\"/>
<use xlink:href=\"#mySymbArrow\" transform=\"translate(60,35),rotate(-90)\" />
<use xlink:href=\"#mySymbArrow\" transform=\"translate(10,60),rotate(45)\" />
<use xlink:href=\"#mySymbArrow\" transform=\"translate(35,60)\" />
<use xlink:href=\"#mySymbArrow\" transform=\"translate(60,60),rotate(-45)\" />
<text x=\"10\" y=\"94\" fill=\"".$cColors["symbAndBorder"]."\"
text-anchor=\"middle\" font-size=\"30\">+</text>
<text x=\"35\" y=\"94\" fill=\"".$cColors["symbAndBorder"]."\"
text-anchor=\"middle\" font-size=\"40\">-</text>
<text x=\"60\" y=\"90\" fill=\"".$cColors["symbAndBorder"]."\"
text-anchor=\"middle\" font-size=\"16\">R</text>
</g>
</g>

<g id=\"myCoordPanel\" font-size=\"9\" transform=\"translate(420,-4)\">
<rect fill-opacity=\"0.6\" fill=\"white\" x=\"28\" y=\"450\" width=\"100\" height=\"25\" />
     <text x=\"33\" y=\"460\" id=\"myCursorMapX\">".$lang["xcoor"]."</text>
     <text x=\"33\" y=\"472\" id=\"myCursorMapY\">".$lang["ycoor"]."</text>
</g>");

//-----------------------------------------------------------
//Layer Panels and Names, References to functions
echo("
<g id=\"myLayers\" transform=\"translate(30,359)\" cursor=\"pointer\">
<g onmouseover=\"evt.target.setAttributeNS(null,'fill','".$cColors['fgInfoPanel']."')\"
onmouseout=\"evt.target.setAttributeNS(null,'fill','".$cColors['navMouseout']."')\"
shape-rendering=\"crispEdges\" fill=\"".$cColors['navMouseout']."\" stroke-width=\"1\"
stroke=\"".$cColors['symbAndBorder']."\">
   <g fill=\"".$cColors['symbAndBorder']."\" font-size=\"11\">
    <text id=\"Layer1\" x=\"2\" y=\"15\" >x</text>
    <text id=\"Layer2\" x=\"2\" y=\"30\">x</text>
    <text id=\"Layer3\" x=\"2\" y=\"45\">x</text>
    <text id=\"Layer4\" x=\"2\" y=\"60\">x</text>
    <text id=\"Layer5\" x=\"2\" y=\"0\">x</text>
    <text id=\"Layer6\" x=\"2\" y=\"75\">x</text>
    <text id=\"Layer7\" x=\"2\" y=\"90\">x</text>
    <text id=\"Layer8\" x=\"2\" y=\"105\">x</text>
   </g>
   <g fill-opacity=\"1\">
    <rect id=\"myLayer1\" x=\"0\" y=\"7\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerEdges.php', 'LayerEdges', 'UTF-8', 'myLayer1')\"/>
    <rect id=\"myLayer2\" x=\"0\" y=\"22\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerNodesSignpost.php', 'LayerNodesSignpost', 'UTF-8', 'myLayer2')\"/>
    <rect id=\"myLayer3\" x=\"0\" y=\"37\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerNodeNames.php', 'LayerNodeNames', 'UTF-8', 'myLayer3')\"/>
<a xlink:href='../resMap.php?".session_name()."=".session_id()."&#38;L=".$langID."' target='hike'>
    <rect id=\"myLayer4\" x=\"0\" y=\"52\" width=\"10\" height=\"10\"/></a>
    <rect id=\"myLayer5\" x=\"0\" y=\"-8\" width=\"10\" height=\"10\"
    onclick=\"myLayer('getPathMap.php?".session_name()."=".session_id()."',
    'LayerRes', 'UTF-8', 'myLayer5')\"/>
<a xlink:href='../resSatAll.php?".session_name()."=".session_id()."&#38;L=".$langID."' target='hike'>
    <rect id=\"myLayer6\" x=\"0\" y=\"67\" width=\"10\" height=\"10\"/></a>
    <rect id=\"myLayer7\" x=\"0\" y=\"82\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerEnvpoly.php', 'LayerEnvpoly', 'UTF-8', 'myLayer7')\"/>
    <rect id=\"myLayer8\" x=\"0\" y=\"97\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerEnvcomm.php', 'LayerEnvcomm', 'UTF-8', 'myLayer8')\"/>
   </g>
   </g>
   <g fill=\"".$cColors['fontPanel']."\" font-size=\"9\">
<text x=\"14\" y=\"17\" onclick=\"myLayer('LayerEdges.php', 'LayerEdges',
'UTF-8', 'myLayer1')\">".$lang["layerEdges"]."</text>
<text x=\"14\" y=\"31\" onclick=\"myLayer('LayerNodesSignpost.php',
'LayerNodesSignpost', 'UTF-8', 'myLayer2')\">".$lang["layerSignpost"]."</text>
<text x=\"14\" y=\"46\" onclick=\"myLayer('LayerNodeNames.php', 'LayerNodeNames',
'UTF-8', 'myLayer3')\">".$lang["layerNodeNames"]."</text>
<a xlink:href='../resMap.php?".session_name()."=".session_id()."&#38;L=".$langID."' target='hike'>
<text x=\"14\" y=\"60\">".$lang["vecLabel"]."</text></a>
<text x=\"14\" y=\"1\"
```

```
onclick=\"myLayer('getPathMap.php?".session_name()."=".session_id()."',
'LayerRes', 'UTF-8', 'myLayer5')\"
onload=\"myLayer('getPathMap.php?".session_name()."=".session_id()."',
'LayerRes', 'UTF-8', 'myLayer5')\">".$lang["layerRes"]."</text>
<a xlink:href='../resSatAll.php?".session_name()."=".session_id()."&#38;L=".$langID."' target='hike'>
<text x=\"14\" y=\"75\">".$lang["satAllLabel"]."</text></a>
<text x=\"14\" y=\"90\"
onclick=\"myLayer('LayerEnvpoly.php','LayerEnvpoly', 'UTF-8', 'myLayer7')\"
onload=\"myLayer('LayerEnvpoly.php','LayerEnvpoly', 'UTF-8', 'myLayer7')\"
>".$lang["layerEnv"]."</text>
<text x=\"14\" y=\"105\" onclick=\"myLayer('LayerEnvcomm.php',
'LayerEnvcomm', 'UTF-8', 'myLayer8')\">".$lang["layerComm"]."</text>
<text x=\"-4\" y=\"-174\">".$lang["dragMap"]."</text>
<text x=\"-4\" y=\"-164\" font-size='8'>".$lang["dragMapSubtitle"]."</text>
</g>
   </g>

<rect  x=\"25\" y=\"198\" width=\"120\" height=\"148\"/>

<svg id=\"myOverviewMap\" onmouseup=\"myDrag(evt)\" onmousemove=\"myDrag(evt)\"
viewBox=\"".$cMAP['xC']." ".$cMAP['yC']." ".$cMAP['xDia']." ".$cMAP['yDia']."\"
x=\"25\" y=\"198\" width=\"120\" height=\"148\">
<rect id=\"myBackgroundRect\" x=\"".$cMAP['xC']."\"
y=\"".$cMAP['yC']."\" width=\"".$cMAP['xDia']."\" height=\"".$cMAP['yDia']."\"
fill=\"".$cColors["overviewRectBg"]."\" stroke-width=\"0\"/>

<use xlink:href=\"#myTopo\"/>
<use xlink:href=\"#myGrid10km\"/>
<use xlink:href=\"#myGrid5km\"/>
<use xlink:href=\"#LayerEnvpoly\"/>
<use xlink:href=\"#myBorder\"/>

<rect id=\"myOverviewRect\" onmousedown=\"myDrag(evt)\" x=\"".$cornerx."\"
y=\"".$cornery."\" width=\"".$xDia."\" height=\"".$yDia."\"
fill=\"".$cColors["bgProject"]."\" fill-opacity=\".4\"
stroke=\"".$cColors["symbAndBorder"]."\" stroke-opacity=\"1\" stroke-width=\"200\"/>

</svg>

</svg>
");


?>
```

C.6 Diagramm.php

This script generates the profile in SVG code out of the resulted path. All computation is dynamically executed.

```
<?php
header("Content-Type: image/svg+xml");

//Session load
session_start();

//Constants initialization
include("constants.php");
$cDB = $LOCAL_CONS["DB"];
$cColors = $LOCAL_CONS["Colors"];
$cDia = $LOCAL_CONS["Diagramm"];

//Language selection
include("locallang.php");
$langID = $_REQUEST['L'];
if ($langID == 1){
$langcode = "en";}
else if ($langID == 0){
$langcode = "default";}
else {$langcode = "default";}
```

```
    $lang = $LOCAL_LANG[$langcode];

    // DATABASE:-----------------------------------------------
    define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
    define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
    define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
    define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
    define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    define ('cStrTABLEUserQ',$cDB['cStrTABLEUserQ']);
    // -----------------------------------------------------

    $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);

    function getDataOutOfPath($input){

    $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);
    //Clean the result string of unwanted doubles through via's
    //by eliminating Elements containing "xxx"
    $stripInput = preg_replace('/,xxx(\d+)/', '', $input);
    //echo $stripInput;
    $path = explode(",",$stripInput);
    $length = count($path);
    $dataPath = array();
    //Begin for buckle with i=1 because in Element 0 the costFactor is stored.
    for ($i = 1; $i <= $length; $i++){
    //Look up data for node
    $dNode = "SELECT n_x, n_y, n_height, n_location, n_municipality, n_district, n_id
         FROM ".cStrTABLENodes." WHERE n_id = ".$path[$i]."";
    $res = mysql_query($dNode);
    $n1 = mysql_fetch_row($res);
    //Look up data for Edge
    $pNode = "SELECT ".cStrTABLETopo.".hdiff, ".cStrTABLETopo.".gradient,
     ".cStrTABLETopo.".time, ".cStrTABLETopo.".hdist, ".cStrTABLETopo.".hvdist,
     baseedges.svg from ".cStrTABLEEdges.", ".cStrTABLETopo." WHERE
         ".cStrTABLEEdges.".id = ".cStrTABLETopo.".edge
         AND ".cStrTABLETopo.".from_node = ".$path[$i]."
         AND ".cStrTABLETopo.".to_node = ".$path[$i+1]."";
    $res2 = mysql_query($pNode);
    $p1 = mysql_fetch_row($res2);
    //Store data in an multidimesional, assoziative array
    $dataPath[$i-1] = array("n_x" => $n1[0], "n_y" => $n1[1], "n_height" => $n1[2],
      "n_location" => $n1[3], "n_municipality" => $n1[4],
      "n_district" => $n1[5], "n_id" => $n1[6],
      "hdiff" => $p1[0], "gradient" => $p1[1], "time" => $p1[2],
      "hdist" => $p1[3], "hvdist" => $p1[4], "svg" => $p1[5]);
    //Tests
    //$inhalt = $dataPath[$i-1]["time"];
         //$inhalt .= $dataPath[$i-1]["n_location"]."<br>";
         echo $inhalt;
         }
         mysql_close($db);
        return ($dataPath);
    }

    $r = array();
    $r = getDataOutOfPath($_SESSION['totalPath']);
    //$path = $_Request['path'];
    //$r = getDataOutOfPath($path);

    //get the total length and time and heightdiff
    $length = 0;
    $time = 0;
    $altMin = 100000;
    $altMax = 0;
    $xr = count($r) - 1;
    for($i=0; $i < $xr; $i++){
```

```
    $length += $r[$i]['hdist'];
 92 $time += $r[$i]['time'];
    if($altMin > $r[$i]['n_height']){$altMin = $r[$i]['n_height'];}
    if($altMax <= $r[$i]['n_height']){$altMax = $r[$i]['n_height'];}
    }
 96 $hdiff = $altMax-$altMin;
    $lengthKM = $length/1000;
    $timeH = floor($time/60);
    $timeMin = round($time%60);
100 $distFact = 9900/$length;
    $altFact = -5000/$hdiff;
    $opGr = 0.2;

104
    echo("
    <svg zoomAndPan=\"disable\" viewBox=\"0 0 600 400\" width=\"100%\"
    height=\"100%\" xmlns=\"http://www.w3.org/2000/svg\" version=\"1.0\"
108 xmlns:xlink=\"http://www.w3.org/1999/xlink\">
    <title>".$lang["title"]."</title>
    <rect id=\"backGroundDelimiter\" x=\"0\" y=\"0\" width=\"600\"
    height=\"400\" fill=\"".$cColors["bgProject"]."\"/>
112
      <rect id=\"mapAreaDelimiter\" x=\"5\" y=\"4\" width=\"590\" height=\"392\"
      fill=\"".$cColors["symbAndBorder"]."\"/>

116 <svg id=\"myMap\" viewBox=\"-1000 -7000 12000 8000\" x=\"5\" y=\"4\"
    width=\"590\" height=\"392\">
    <rect id=\"mapBG\" x=\"-1000\" y=\"-7000\" width=\"12000\" height=\"8000\"
    fill=\"".$cColors["bgMap"]."\"/>
120

    <g id=\"myActiveElements\"><!-- start dynamic elements -->
    <g id=\"myPoly\" fill='".$cDia['polyFill']."'
124 stroke='".$cDia['strokeColor']."' stroke-width='".$cDia['strokeWidth']."'>");

    //Generating the polygons for the profile segments
    $poly = "";
128 $l = 0;
    $t = 0;
    for($q = 0; $q < count($r)-2; $q++){
    $temp = round(($l + $r[$q]['hdist']) * $distFact);
132 $temp2 = round(($r[$q]['n_height'] - $altMin) * $altFact);
    $temp3 = round(($r[$q + 1]['n_height'] - $altMin) * $altFact);
    //Adapt opacity in dependence of gradient
    $fOp = $opGr + (3 * abs($r[$q]['gradient']) / 100);
136 $temp4 = round($l * $distFact);
    $poly .= "<polygon fill-opacity='".$fOp."'
    points=\"".$temp4." 0, ";
    $poly .= "".$temp4." ".$temp2.", ";
140 $poly .= "".$temp." ".$temp3.",
    ".$temp." 0\" />";
    $l += $r[$q]['hdist'];
    //echo ("$poly <br/>");
144 }

    echo ("$poly");

148 echo ("
    </g>
    <g id=\"myLocNames\" fill=\"".$cColors["Grid"]."\"
    font-size=\"180\" text-anchor=\"start\">");
152
    //Generating Locnames
    $l = 0;
    //Minimal distance between names to avoid overlapping
156 $lAbs = 200;
    $ttime = 0;
    $loc = "";
    $locData = "";
160 for($q = 0; $q < count($r)-1; $q++){
```

```
$tempx = round($l * $distFact) + 40;
$tempy = round(($r[$q]['n_height'] - $altMin) * $altFact) - 150;
$m = $ttime % 60;
$h = floor($ttime / 60);
$km = round($l / 1000,1);
if($lAbs >= 200){
$loc .= "<text x='".$tempx."' y='".$tempy."'
transform='rotate(-90,".$tempx.",".$tempy.")'>
".$r[$q]['n_location']."</text>";
$locData .= "<text x='".$tempx."' y='130'
transform='rotate(-45,".$tempx.",120)'>
".$h."h ".$m."min | ".$km."km</text>";
$lAbs = $r[$q]['hdist']*$distFact;
}
else {
$lAbs += $r[$q]['hdist'] * $distFact;
}
$l += $r[$q]['hdist'];
$ttime += $r[$q]['time'];
} //endFor

   echo ("".$loc."");
echo ("</g>
<g id=\"myTimeDist\" fill=\"".$cColors["Grid"]."\"
font-size=\"140\" text-anchor=\"end\">");
echo ("".$locData."");
echo ("</g>");


//Calculating the height ticks and labels
$tM = floor($altMax / 100);
$tMi = ceil($altMin / 100);
$tH = "";
$pH = "";
//Min / Max
$altMinH = $altMin * $altFact;
$altMaxH = $hdiff * $altFact;
$pH .= "<path d=\"M -550 ".$altMaxH." L 10000 ".$altMaxH."\" />";
$tH .= "<text x='-600' y='".$altMaxH."'>".round($altMax)."</text>";
$tH .= "<text x='-600' y='0'>".round($altMin)."</text>";
if($tMi > $tM){;}
else{
$gr = $tM - $tMi;
for($ww=0 ; $ww <= $gr; $ww++){
$temp3 = round((($tMi + $ww) * 100 - $altMin) * $altFact);
$temp2 = ($tMi + $ww) * 100;
// Avoid overlapping grid lines and labels when
// Max / Min are near 100 limits
if( $temp3 < -500 AND $temp3 > -4500 ){
$tH .= "<text x='-600' y='".$temp3."'>".$temp2."</text>";
$pH .= "<path d=\"M -550 ".$temp3." L 10000 ".$temp3."\" />";
}

}  //endFor
} // endElse


echo ("<g id=\"myHeightTick\" transform='translate(0,50)'
 fill=\"".$cColors["Grid"]."\" font-size=\"180\" text-anchor=\"end\">");
echo(".$tH.");
echo ("</g>");

echo ("<g id=\"myHeightGrid\" fill='black'
 stroke=\"#000000\" stroke-width=\"3\">");
echo(".$pH.");
echo ("</g>");

echo ("</g><!-- end of dynamic elements -->

<g id=\"myPassiveElements\">
```

```
    <g id=\"myMapFrame\">
232
    <g id=\"myAxes\">
    <g id=\"myGridText\" fill=\"".$cColors["Grid"]."\"
    font-size=\"180\" text-anchor=\"start\">
236 <text x='10130' y='-70'>".$lang['distDia']."</text>
    <text x='10130' y='130'>".$lang['timeDia']."</text>
    <text x='-500' y='-5350' transform='rotate(-90,-500,-5350)'>
    ".$lang['heightDia']."</text>
240 </g>
    <g id=\"myGridLines\" fill=\"".$cColors["Grid"]."\"
    stroke-width=\"25\" stroke=\"black\">
    <path d=\"M -500 -5150 L -500 100 M -550 0 L 10000 0\" />
244 </g>
    </g>
    </g>
    </g>
248
    </svg>

    </svg>
252 ");


    ?>
```

C.7 resDataPath.php

The generation of a schedule containing the full detailed information about the calculated path is approached through the following script.

```
    <?php

    //Session load
 4  session_start();

    //Print from file getDataPath.php (include not possible)
    //Constants initialization
 8  include("constants.php");
    $cDB = $LOCAL_CONS["DB"];

    //Language selection
12  include("locallang.php");
    $langID = $_REQUEST['L'];
    if ($langID == 1){
    $langcode = "en";}
16  else if ($langID == 0){
    $langcode = "default";}
    else {$langcode = "default";}
    $lang = $LOCAL_LANG[$langcode];
20
    echo('
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
    <html>
24  <head>
    <title>'.$lang['title'].'</title>
    <meta name="generator" content="http://www.virTours.ch | Urs Sieber">
    <meta name="keywords"
28  content="Wandern, Wanderunge, Hiking, Hike, routeplanning, routenplanug, wegplaner,
    wanderroute, wanderwege, wanderweg, wegweiser, signpost, SVG, dynamic SVG, topology" />
    <link rel="STYLESHEET" href="helpers/style.css" type="text/css">
    </head>
32  <body>
    <table><tr><td class="heading">
    <img border="0" src="helpers/Pfeil_orange_grau.gif" width="11" height="11">  
    '.$lang['dataH1'].'</td></tr></table>
36  <p>'.$lang['dataIntro'].'</p>
    <table><tr><td>
    <ul>
    <li><a href ="resMap.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
```

```
 40  '.$lang['headMap'].'
     </a></li>
     <li>
     '.$lang['headData'].'
 44  </li>
     <li><a href ="resMail.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
     '.$lang['headMail'].'
     </a></li>
 48  <li><a href ="resDia.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
     '.$lang['headDia'].'
     </a></li><br><br>
     <li><a href ="userForm.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
 52  '.$lang['headWeight'].'
     </a></li>
     <li><a href ="destroySession.php?L='.$langID.'" target="_self">
     '.$lang['headNew'].'
 56  </a></li>
     </ul>
     </td></tr></table>');

 60
     // Get the topology out of the DB:
     // DATABASE:-----------------------------------------------
     define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
 64  define ('cStrUSERDB',$cDB['cStrUSERDB']);
     define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
     define ('cStrDB',$cDB['cStrDB']);
     define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
 68  define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
     define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
     define ('cStrTABLEUserQ',$cDB['cStrTABLEUserQ']);
     // -----------------------------------------------------
 72
     //Session load
     session_start();

 76  function getDataOutOfPath($input){

     $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
     mysql_select_db(cStrDB);
 80  //Clean the result string of unwanted doubles through via's
     //by eliminating Elements containing "xxx"
     $stripInput = preg_replace('/,xxx(\d+)/', '', $input);
     //echo $stripInput;
 84  $path = explode(",",$stripInput);
     $length = count($path);
     $dataPath = array();
     //Begin for buckle with i=1 because in Element 0 the costFactor is stored.
 88  for ($i = 1; $i <= $length; $i++){
     //Look up data for node
     $dNode = "SELECT n_x, n_y, n_height, n_location, n_municipality, n_district, n_id
          FROM ".cStrTABLENodes." WHERE n_id = ".$path[$i]."";
 92  $res = mysql_query($dNode);
     $n1 = mysql_fetch_row($res);
     //Look up data for Edge
     $pNode = "SELECT ".cStrTABLETopo.".hdiff, ".cStrTABLETopo.".gradient,
 96   ".cStrTABLETopo.".time, ".cStrTABLETopo.".hdist, ".cStrTABLETopo.".hvdist,
      baseedges.svg from ".cStrTABLEEdges.", ".cStrTABLETopo." WHERE
          ".cStrTABLEEdges.".id = ".cStrTABLETopo.".edge
          AND ".cStrTABLETopo.".from_node = ".$path[$i]."
100       AND ".cStrTABLETopo.".to_node = ".$path[$i+1]."";
     $res2 = mysql_query($pNode);
     $p1 = mysql_fetch_row($res2);
     //Store data in an multidimesional, assoziative array
104  $dataPath[$i-1] = array("n_x" => $n1[0], "n_y" => $n1[1], "n_height" => $n1[2],
     "n_location" => $n1[3], "n_municipality" => $n1[4], "n_district" => $n1[5], "n_id" => $n1[6],
     "hdiff" => $p1[0], "gradient" => $p1[1], "time" => $p1[2],
     "hdist" => $p1[3], "hvdist" => $p1[4], "svg" => $p1[5]);
108       }
          mysql_close($db);
```

```
        return ($dataPath);
    }
//getDataPath("hvdist,30,35,xxx123947192378982372873128 73,40,41,1493,50,xxx41,xxx1493");

    $r = getDataOutOfPath($_SESSION['totalPath']);
    //Table with description of hikeroute
if($_SESSION['Via1'] != ""){
    $via1 = "via ".$_SESSION['Via1']." ";}
    else {
    $via1 = "";}
if($_SESSION['Via2'] != ""){
    $via2 = "".$lang['headAnd']." via ".$_SESSION['Via2']." ";}
    else{
    $via2 = "";}

    echo ('<p>'.$lang['dataRoute1'].' "<b>'.$_SESSION['From'].'</b>"
    '.$via1.''.$via2.' '.$lang['diaRoute2'].' "<b>'.$_SESSION['To'].'</b>".<br />
    '.$lang['factor'].': <b>'.$_SESSION['Weight'].'</b></p>
<p align = "right" class="footer">
    <a href ="resMail.php?'.session_name().'='.session_id().'
    &L='.$langID.'" target="_self"> <img border="0" src="helpers/email.gif"></a>
    <a href ="resDia.php?'.session_name().'='.session_id().'
&L='.$langID.'" target="_self"> <img border="0" src="helpers/diagram.gif"></a>
    <a href ="resDataPathPrint.php?'.session_name().'='.session_id().'&
    L='.$langID.'" target="_blank">   
    <img border="0" src="helpers/printer.gif"></a></p>
');


    //Generation of detail table
echo ("<table width=100%>
    <tr><th>".$lang['dataDist']."</th>
    <th>".$lang['dataTime']."</th>
    <th>".$lang['dataHeight']."</th>
<th>".$lang['dataGradient']."</th>
    </tr>");

    //Initializing counters
$length = 0;
    $lenghtKM = 0;
    $time = 0;
    $timeH = 0;
$timeMin = 0;
    $alt = 0;

    for($w = 0; $w < count($r)-1; $w++){
       echo ("<tr><td ");
       if($r[$w]['n_id'] == $_SESSION['FromID']
        OR $r[$w]['n_id'] == $_SESSION['ToID']
        OR $r[$w]['n_id'] == $_SESSION['Via1ID']
        OR $r[$w]['n_id'] == $_SESSION['Via2ID']){
         echo ("class='dataLoc' ");
         }

       echo ("colspan = 4>".$r[$w]['n_location']."
       ".$r[$w]['n_municipality']."
       ".$r[$w]['n_height']."
       ".$lang['dataSea']." <br>
       ".$lang['dataTime'].": ".round($time)." min (".$timeH." h ".$timeMin." Min),
       ".$lang['dataDist'].": ".$lengthKM." km,
       ".$lang['dataHeight']."; ".round($alt)." m</td></tr>");
       if($w == count($r)-2){
        echo ("<tr>
        <td class='dataEnd' colspan = 4>
        ".$lang['dataFin1']." ".$r[$w]['n_location']."<br>
        in ".$timeH." ".$lang['dataHour']." ".$timeMin." ".$lang['dataMin']."<br>
        ".$lang['dataFin2']." ".$lengthKM."
        ".$lang['dataFin3']." ".$alt." ".$lang['dataFin4']."</td></tr>");
        }
        else{
```

```
180      echo("
         <tr><td class='dataSec'>".round($r[$w]['hvdist'])." m</td>
         <td class='dataSec'>".round($r[$w]['time'])." min</td>
         <td class='dataSec'>".round($r[$w]['hdiff'])." m</td>
184      <td class='dataSec'>".round($r[$w]['gradient'])." %</td>
         </tr>
         ");
         }
188
        //Counters
        $length = $length + $r[$w]['hvdist'];
        $lengthKM = round($length/1000,2);
192     $time = $time + $r[$w]['time'];
        $timeH = floor($time/60);
        $timeMin = round($time%60);
        $alt = $alt + abs($r[$w]['hdiff']);
196
     }  //endFor

     echo ("</table>");
200
     echo ('<br><br>
     <table><tr>
     <td><a href="http://www.virtours.ch" target="_blank">
204  <img src="MailClass/Logo.jpg" width="100px" border="0"></a></td>
     <td><p class="footer" align="right">'.$lang['mailAdv1'].'<br/>
     </p></td></tr></table>

208  </body>
     </html>

     ');
212
     ?>
```

C.8 Invitation for a hike via email

Implementing the invitation feature letting somebody send an invitation for a hike with the individual hike route proposition happens on behalf of these two files.

C.8.1 resMail.php

The interface showing the html form for the entries to personalize the invitation email. A JavaScript check-submit function tests for wrong entries on client side. The check-submit and the extra added Spam check avoid unwanted mail traffic on the server. The sender gets a copy of his invitation. It is possible to reach multiple recipients when separating the email addresses with commas.

```
     <?php

     $cStrTermin  = $_POST['Termin'];
 4   $cStrName  = $_POST['Name'];
     $cStrEmpfMail = $_POST['EmpfMail'];
     $cStrEmpfName = $_POST['EmpfName'];
     $cStrMail  = $_POST['Mail'];
 8   $cStrText  = $_POST['Text'];
     $cStrSpam  = $_POST['Spam'];
     $cStrSend  = $_POST['Send'];

12      $host   = $_SERVER['HTTP_HOST'];
        $uri   = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');

     //Session load
16   session_start();

     //Constants initialization
     include("constants.php");
```

```
$cDB = $LOCAL_CONS["DB"];

//Language selection
include("locallang.php");
$langID = $_REQUEST['L'];
if ($langID == 1){
$langcode = "en";}
else if ($langID == 0){
$langcode = "default";}
else {$langcode = "default";}
$lang = $LOCAL_LANG[$langcode];


// DATABASE:---------------------------------------------
define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
define ('cStrUSERDB',$cDB['cStrUSERDB']);
define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
define ('cStrDB',$cDB['cStrDB']);
define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
define ('cStrTABLEUserQ',$cDB['cStrTABLEUserQ']);
// -----------------------------------------------------

// Tests wheter the entries exist
$db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
mysql_select_db(cStrDB);


//Include MailClass for MIME Mails
require_once("MailClass/MIMEMailxPHP4_V2.inc");
//Instantiate MailClass
$mail = new MIMEMailxPHP4_V2();
$mail->addBinaryFileAsAttachment("MailClass/Logo.jpg", "image/jpg", "Logo.jpg", "logobild");

if($cStrSend != TRUE){
$tstamp = time();
$SPAM = substr($tstamp,-4);
$_SESSION['SPAM'] = $SPAM;
}

//function to get ID of userquery
function getID($path){
$getID = "SELECT id
     FROM ".cStrTABLEUserQ."
     WHERE userpath = '".$path."'";
$res = mysql_query($getID);
$n1 = mysql_fetch_row($res);
return $n1[0];}

$urlID = getID($_SESSION['totalPath']);
$url = "http://www.virtours.ch/t3/index.php?id=96&userID=".$urlID."&L=".$langID."&mail=1";

$mailtext = "
<p>".$lang['mailDear']." ".$cStrEmpfName."<br/>
".$cStrName." ".$lang['mailtext1']." ".$cStrName." ".$lang['mailtext2']."<p/>
<p>".nl2br(htmlspecialchars($Text))."</p>
<table>
<tr>
<td colspan = '2'>".$lang['mailLink']."<br><a href='".$url."'
target='_blank'>".$lang['mailLink2']." ".$cStrName."</a></td>
</tr>
<tr>
<td colspan = '2'>".$lang['mailPossTermin']."<br>".$cStrTermin."</td>
</tr>
<tr>
<td colspan = '2'>".$lang['mailContact']."<br>
<a href='mailto:".$cStrMail."'>E-Mail ".$cStrName."</a></td>
</tr>
<tr>
```

```
    <td>".$lang['mailAdv1']."</td><td align='right'>
    <br><br><br><br><br>
 92 <p align = 'right'>
    <a href='http://www.virtours.ch/hiking' target='_blank'>
    <img style='border:0; width: 180px; height: 44px;'
    alt='Logo virTours.ch' src='cid:logobild'></a></p></td>
 96 </tr>
    </table>
    ";

100
    echo('
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
    <html>
104 <head>
    <title>'.$lang['title'].'</title>
    <meta name="generator" content="http://www.virTours.ch | Urs Sieber">
    <meta name="keywords"
108 content="Wandern, Wanderunge, Hiking, Hike, routeplanning, routenplanug, wegplaner,
    wanderroute, wanderwege, wanderweg, wegweiser, signpost, SVG, dynamic SVG, topology" />
    <link rel="STYLESHEET" href="helpers/style.css" type="text/css">

112 ');


    echo('
116 </head>
    <body>
    <table><tr><td class="heading">
    <img border="0" src="helpers/Pfeil_orange_grau.gif" width="11" height="11"> 
120 '.$lang['mailH1'].'
    </td></tr></table>
    <p>'.$lang['mailIntro'].'</p>
    <table><tr><td>
124 <ul>
    <li><a href ="resMap.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
    '.$lang['headMap'].'
    </a></li>
128 <li><a href ="resDataPath.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
    '.$lang['headData'].'
    </a></li>
    <li>
132 '.$lang['headMail'].'
    </li>
    <li><a href ="resDia.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
    '.$lang['headDia'].'
136 </a></li><br><br>
    <li><a href ="userForm.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
    '.$lang['headWeight'].'
    </a></li>
140 <li><a href ="destroySession.php?L='.$langID.'" target="_self">
    '.$lang['headNew'].'
    </a></li>
    </ul>
144 </td></tr></table>');


    //Send Mail, if Form was filled in
148 if($cStrSend == TRUE){
    $cStrSubject = "$cStrName ".$lang['mailSubject']."";
    $mail->setHTMLContent($mailtext);
    $mail->setFromHeader("Wanderplaner @ virTours.ch <".$cStrMail.">");
152 $mail->setRecievers(Array($cStrEmpfMail));
    $mail->setCCRecievers($cStrMail);
    $mail->setBCCRecievers("urs.sieber@virtours.ch");
    $mail->setSubject($cStrSubject);
156 if($cStrSpam == $_SESSION['SPAM']){
    //SendMail
    $mail->setEOL("\r\n");
    $rc = $mail->sendMail();
```

```
160  echo ('<p>'.$lang['mailSucc1'].' '.$cStrEmpfName.' ('.$cStrEmpfMail.')
     '.$lang['mailSucc2'].' ('.$cStrMail.') '.$lang['mailSucc3'].'</p>');
     }
     else {echo (''.$lang['mailMessSpam'].'');
164  }
     }

     //Check the form for false entries and Spam
168  echo ('<script type="text/javascript">
     function checksubmit()
     {
     if (document.getElementById("f1").Name.value == "")
172  {
     alert("'.$lang['mailFrom'].'")
     document.getElementById("f1").Name.focus()
     return false
176  }
     if (document.getElementById("f1").Mail.value == "")
     {
     alert("'.$lang['mailFromMail'].'")
180  document.getElementById("f1").Mail.focus()
     return false
     }
     if (document.getElementById("f1").EmpfName.value == "")
184  {
     alert("'.$lang['mailTo'].'")
     document.getElementById("f1").EmpfName.focus()
     return false
188  }
     if (document.getElementById("f1").EmpfMail.value == "")
     {
     alert("'.$lang['mailToMail'].'")
192  document.getElementById("f1").EmpfMail.focus()
     return false
     }
     if (document.getElementById("f1").Text.value == "")
196  {
     alert("'.$lang['mailMessage'].'")
     document.getElementById("f1").Text.focus()
     return false
200  }
     if (document.getElementById("f1").Spam.value == "")
     {
     alert("'.$lang['mailSpam'].'")
204  document.getElementById("f1").Spam.focus()
     return false
     }

208  }
     </script>');


212  if($_SESSION['Via1'] != "") {$via1 = "via ".$_SESSION['Via1']." ";} else {$via1 = "";}
     if($_SESSION['Via2'] != "") {$via2 = "und via ".$_SESSION['Via2']." ";} else {$via2 = "";}

     echo ('
216  <p>Ihre Route f&#252;hrt vom Startpunkt "<b>'.$_SESSION['From'].'</b>"
     '.$via1.''.$via2.' zum Zielstandort "<b>'.$_SESSION['To'].'</b>".</p>
     <p>'.$lang['mailForm'].'</p>

220  <p align = "right" class="footer">
     <a href ="resMail.php?'.session_name().'='.session_id().'
     &L='.$langID.'" target="_self"> <img border="0" src="helpers/email.gif"></a>
     <a href ="resDia.php?'.session_name().'='.session_id().'
224  &L='.$langID.'" target="_self"> <img border="0" src="helpers/diagram.gif"></a>
     </p>
     ');

228
     echo ('
```

```
    <form name="userInput" id="f1" method="POST"
    action="'.$_SERVER['PHP_SELF'].'?L='.$langID.'" onsubmit="return checksubmit()">
232 <fieldset style="padding: 10">
    <legend>'.$lang['mailFieldset'].'</legend>
    <table>
    <tr>
236 <td>* '.$lang['mailFrom'].'<br>
    <input type="text" name="Name" id="Name" size="40" value="'.$cStrName.'"></td>
    </tr>
    <tr>
240 <td>* '.$lang['mailFromMail'].'<br>
    <input type="text" name="Mail" id="Mail" size="40" value="'.$cStrMail.'"></td>
    </tr>
    <tr>
244 <td>* '.$lang['mailTo'].'<br>
    <input type="text" name="EmpfName" id="EmpfName" size="40" value="'.$cStrEmpfName.'"></td>
    </tr>
    <tr>
248 <td>* '.$lang['mailToMail'].'<br>
    <input type="text" name="EmpfMail" id="EmpfMail" size="40" value="'.$cStrEmpfMail.'"></td>
    </tr>
    <tr>
252 <td>'.$lang['mailTermin'].'<br>
    <input type="text" name="Termin" id="Termin" size="40" value="'.$cStrTermin.'"></td>
    </tr>
    <tr>
256 <td>* '.$lang['mailMessage'].'<br>
    <textarea rows="3" name="Text" id="Text" cols="40">'.$cStrMessage.'</textarea></td>
    </tr></table></fieldset>
    <table><tr>
260 <td><br><br>* '.$lang['mailSpam'].'<br>
    <b>'.$_SESSION['SPAM'].'</b>   
    <input type="text" name="Spam" id="Spam" size="20" value=""></td>
    </tr>
264 <tr>
    <td><br><br>
    <img border="0" src="helpers/Pfeil_orange_grau.gif" width="11" height="11">
        
268 <input type="submit" value="'.$lang['mailSend'].'" name="Send"></td>
    </tr>
    </table>
    </form>
272
    ');

    echo ('<br><br>
276 <table><tr>
    <td><a href="http://www.virtours.ch" target="_blank">
    <img src="MailClass/Logo.jpg" width="100px" border="0"></a></td>
    <td><p class="footer" align="right">'.$lang['mailAdv1'].'<br/>
280 </p></td></tr></table>
    </body>
    </html>

284 ');

    ?>
```

C.8.2 eMail.php

This file works in the background and is executed only if a user clicks the URL to inspect the route in a received invitation email. The script looks for the correspondign path in the database and sets up the new session with the data for the path sent by the addressee. Finally it automatically redirects to the map showing the sent path in focus.

```
    <?php

    //Session load
 4  session_start();
```

```
    //Load new session
    session_unset();

 8     $host   = $_SERVER['HTTP_HOST'];
       $uri   = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');

    //Constants initialization
12  include("constants.php");
    $cDB = $LOCAL_CONS["DB"];

    //Language selection
16  include("locallang.php");
    $lang = $LOCAL_LANG["default"];

    // DATABASE:-------------------------------------------
20  define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
    define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
24  define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
    define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
    define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    define ('cStrTABLEUserQ',$cDB['cStrTABLEUserQ']);
28  // -----------------------------------------------------

    // Tests wheter the entries exist
    $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
32  mysql_select_db(cStrDB);

    // Gets location out of ID
    function getNameOfID($ID){
36  $getID = "SELECT n_id, n_location, n_municipality
         FROM ".cStrTABLENodes."
         WHERE n_id = '".$ID."'";
    $res = mysql_query($getID);
40  $n1 = mysql_fetch_row($res);
    return $n1;}

    function getPath($ID){
44  $getID = "SELECT userpath
         FROM ".cStrTABLEUserQ."
         WHERE id = '".$ID."'";
    $res = mysql_query($getID);
48  $n1 = mysql_fetch_row($res);
    return $n1[0];}

    function searchXXX($p){
52  $temp = array();
    for($i=0;$i<count($p);$i++){
    if(strpos($p[$i], 'xxx') === 0){
    array_push($temp, $p[$i]);}
56  }
    return $temp;
    }

60

    //Getting the path out of URL ID
    $path = getPath($_REQUEST['userID']);

64  $pathArr = explode(',', $path);
    $_SESSION['Weight'] = $pathArr[0]; //echo (".$_SESSION['Weight']."    <br>");
    $_SESSION['FromID'] = $pathArr[1]; //echo (".$_SESSION['FromID']."   From ID <br>");
    $_SESSION['ToID'] = array_pop($pathArr); //echo (".$_SESSION['ToID']." To ID   <br>");
68

    //Filter for vias
    $q = searchXXX($pathArr);

72  if(count($q) == 2){
    $_SESSION['Via1ID'] = substr($q[0],3); //echo (".$_SESSION['Via1ID']."   V1 ID <br>");
    $_SESSION['Via2ID'] = substr($q[1],3); //echo (".$_SESSION['Via2ID']."   V2 ID <br>");
```

```
    $V1 = getNameOfID($_SESSION['Via1ID']);
76  $V2 = getNameOfID($_SESSION['Via2ID']);
    $_SESSION['Via1'] = "".$V1[2]." | ".$V1[1]."";
    $_SESSION['Via2'] = "".$V2[2]." | ".$V2[1]."";
    }
80  else if (count($q) == 1){
    $_SESSION['Via1ID'] = substr($q[0],3); //echo ("".$_SESSION['Via1ID']."   V1 ID <br>");
    $V1 = getNameOfID($_SESSION['Via1ID']);
    $_SESSION['Via1'] = "".$V1[2]." | ".$V1[1]."";
84  }
    else {;}

    $From = getNameOfID($_SESSION['FromID']);
88  $To = getNameOfID($_SESSION['ToID']);
    $_SESSION['From'] = "".$From[2]." | ".$From[1]."";
    $_SESSION['To'] = "".$To[2]." | ".$To[1]."";

92  $_SESSION['totalPath'] = $path;

    //Redirect after Session settings
    header("location: http://".$host.$uri."/resMap.php?
96  ".session_name()."=".session_id()."&L=".$_REQUEST['L']."");


    ?>
```

C.9 User input and web interface

All user input is guided by one single script "userForm.php". Together with it's additional subscripts, it manages user input and transfers the input data to further treatment.
Data output is implemented with html inline frames and simple link navigation menus in the headers.

C.9.1 userForm.php

The core script for user input management is quoted below.

```
    <?php

    //Constants initialization
 4  include("constants.php");
    $cDB = $LOCAL_CONS["DB"];

    //Language selection
 8  include("locallang.php");
    $langID = $_REQUEST['L'];
    if ($langID == 1){
    $langcode = "en";}
12  else if ($langID == 0){
    $langcode = "default";}
    else {$langcode = "default";}
    $lang = $LOCAL_LANG[$langcode];
16
    //Different queries before start
       $Loc = explode('qqq', $_REQUEST['loc']);

20     //print_r($_POST);

       $cStrDiagramm  = $_POST['dia'];
       $cStrNew = $_POST["new"];
24     $cStrCalcDijkstra= $_POST["calcDijkstra"];
       $cStrWeight = $_POST["Weight"];
       $cStrFromSel = $_POST["FromSel"];
       $cStrToSel = $_POST["ToSel"];
28     $cStrVia1Sel = $_POST["Via1Sel"];
       $cStrVia2Sel = $_POST["Via2Sel"];
       $cStrFrom = $_POST["From"];
       $cStrTo = $_POST["To"];
```

```
32      $cStrVia1 = $_POST["Via1"];
        $cStrVia2 = $_POST["Via2"];
        $cStrFromHide = $_POST["FromHide"];
        $cStrToHide = $_POST["ToHide"];
36      $cStrVia1Hide = $_POST["Via1Hide"];
        $cStrVia2Hide = $_POST["Via2Hide"];
        $boolCalc = $_POST["Calc"];
        $boolFrom = $_POST["okFrom"];
40      $boolTo = $_POST["okTo"];
        $boolVia1 = $_POST["okVia1"];
        $boolVia2 = $_POST["okVia2"];
        $host   = $_SERVER['HTTP_HOST'];
44      $uri   = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');

        //Check the correctness of entries, initialize
        $boolF = 1;
48      $boolT = 1;
        $boolV1 = 1;
        $boolV2 = 1;

52  //Session load
    session_start();
    if ($cStrNew){
       header("location: http://".$host.$uri."/destroySession.php?L=".$langID."");
56     }

    // DATABASE:-----------------------------------------------
    define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
60  define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
    define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
64  define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
    define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    define ('cStrTABLEUserQ',$cDB['cStrTABLEUserQ']);
    // -----------------------------------------------------
68
    // Tests wheter the entries exist
    $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);
72

    // Substitute öäü
    function umlaute($c){
76  $c = eregi_replace('ä','ae',$c);
    $c = eregi_replace('ü','ue',$c);
    $c = eregi_replace('ö','oe',$c);
    return  $c;
80  }

    // Gets location out of ID
    function getNameOfID($ID){
84  $getID = "SELECT n_id, n_location, n_municipality
         FROM ".cStrTABLENodes."
         WHERE n_id = '".$ID."'";
    $res = mysql_query($getID);
88  $n1 = mysql_fetch_row($res);
    return $n1;}

    //Renew Session vars when calculating hikepath
92  if ($boolCalc OR $boolTo OR $boolFrom OR $boolVia1 OR $boolVia2 OR $cStrCalcDijkstra){
    $_SESSION['Weight']= $cStrWeight;
    $_SESSION['FromID']= $cStrFromHide;
    $_SESSION['ToID']= $cStrToHide;
96  $_SESSION['Via1ID']= $cStrVia1Hide;
    $_SESSION['Via2ID']= $cStrVia2Hide;
    $_SESSION['From']= $cStrFrom;
    $_SESSION['To']= $cStrTo;
100 $_SESSION['Via1']= $cStrVia1;
    $_SESSION['Via2']= $cStrVia2;
```

```
    //Check if coming from DropDown
104 if($cStrFromSel != ""){
    $t1 = getNameOfID($cStrFromHide);
    $_SESSION['From'] = "".$t1[2]." | ".$t1[1]."";
    }
108 if($cStrToSel != ""){
    $t1 = getNameOfID($cStrToHide);
    $_SESSION['To'] = "".$t1[2]." | ".$t1[1]."";
    }
112 if($cStrVia1Sel != ""){
    $t1 = getNameOfID($cStrVia1Hide);
    $_SESSION['Via1'] = "".$t1[2]." | ".$t1[1]."";
    }
116 if($cStrVia2Sel != ""){
    $t1 = getNameOfID($cStrVia2Hide);
    $_SESSION['Via2'] = "".$t1[2]." | ".$t1[1]."";
    }
120 }
    //

    // Check if coming from SVG linkage
124 if ($Loc[1] != ''){
       $cStrLoc = $Loc[0];
       $cStrKind = $Loc[1];
       $temp = getNameOfID($cStrLoc);
128    // echo $temp[1]; echo $temp[2];
       if($cStrKind == 'From'){
        $_SESSION['FromID'] = $cStrLoc;
        $_SESSION['From'] = "".$temp[2]." | ".$temp[1]."";
132     }
       else if ($cStrKind == 'To'){
        $_SESSION['ToID'] = $cStrLoc;
        $_SESSION['To'] = "".$temp[2]." | ".$temp[1]."";
136     }
       else if ($cStrKind == 'Via1'){
        $_SESSION['Via1ID'] = $cStrLoc;
        $_SESSION['Via1'] = "".$temp[2]." | ".$temp[1]."";
140     }
       else if ($cStrKind == 'Via2'){
        $_SESSION['Via2ID'] = $cStrLoc;
        $_SESSION['Via2'] = "".$temp[2]." | ".$temp[1]."";
144     }
     }

    // Test for all Fields, if matching or not.
148 // If not, give similar locations in a drop down.

    if($_SESSION['From'] != "" OR $cStrFromSel != ""){
    if(strpos($_SESSION['From']," | ") OR $cStrFromSel != ""){
152 $Where = "n_id = ".$_SESSION['FromID']."";
    }
    else{
    $Where = "WHERE n_location = '".trim($_SESSION['From'])."'
156          OR n_location = '".trim(umlaute($_SESSION['From']))."'";}
    $qFrom = "SELECT n_id, n_location, n_municipality
         FROM ".cStrTABLENodes."
         WHERE ".$Where."";
160 $res = mysql_query($qFrom);
    $n1 = mysql_fetch_row($res);
    if($n1[0] > 0){
    $_SESSION['FromID'] = $n1[0];
164 $_SESSION['From'] = "".$n1[2]." | ".$n1[1]."";
    $boolF = 100;
    $messageF = $lang['messageFa'];
    }
168 else{
    $qFrom = "SELECT DISTINCT n_id, n_location, n_municipality
         FROM ".cStrTABLENodes."
         WHERE
```

```
172       MATCH (n_location, n_municipality) AGAINST ('".$_SESSION['From']."')
          OR n_location LIKE '%".$_SESSION['From']."%'
          OR n_municipality LIKE '%".$_SESSION['From']."%'
          OR n_location LIKE '%".umlaute($_SESSION['From'])."%'
176       OR n_municipality LIKE '%".umlaute($_SESSION['From'])."%'
          ORDER BY n_municipality";
     $res = mysql_query($qFrom);
     $num_rows = mysql_num_rows($res);
180  $n1 = mysql_fetch_row($res);
     if($n1[0]>0){
     $messageF = $lang['messageFb'];
     $selectFrom = "<select name='FromSel' size='1' id='FromSel'
184  onChange=\"transfer('FromSel','FromHide')\">
     <option value=''>".$lang['alternative']."</option>";
     while($n1){
     $selectFrom .= '<option value="'.$n1[0].'">';
188  $selectFrom .= ''.$n1[2].' | '.$n1[1].'';
     $selectFrom .= '</option>';
     $n1 = mysql_fetch_row($res);
     }
192  $selectFrom .= '</select>';
     $_SESSION['FromID'] = 0;
     $boolF = 2;
     } //endIf
196  else{
     $boolF = 3;
     $messageF = $lang['messageFc'];
     } //endElse
200  } //endElse
     } //endIf
     else{  //if field ""
     $boolF = 4;
204  $messageF = $lang['messageFd'];
     }  //endElse


208  if($_SESSION['To'] != "" OR $cStrToSel != ""){
     if(strpos($_SESSION['To']," | ") OR $cStrToSel != ""){
     $Where = "n_id = ".$_SESSION['ToID']."";}
     else{
212  $Where = "WHERE n_location = '".trim($_SESSION['To'])."'
              OR n_location = '".trim(umlaute($_SESSION['To']))."'";}
     $qTo = "SELECT n_id, n_location, n_municipality
          FROM ".cStrTABLENodes."
216       WHERE ".$Where."";
     $res = mysql_query($qTo);
     $n1 = mysql_fetch_row($res);
     if($n1[0] > 0){
220  $_SESSION['ToID'] = $n1[0];
     $_SESSION['To'] = "".$n1[2]." | ".$n1[1]."";
     $boolT = 100;
     $messageT = $lang['messageTa'];
224  }
     else{
     $qTo = "SELECT DISTINCT n_id, n_location, n_municipality
          FROM ".cStrTABLENodes."
228       WHERE
          MATCH (n_location, n_municipality) AGAINST ('".$_SESSION['To']."')
          OR n_location LIKE '%".$_SESSION['To']."%'
          OR n_municipality LIKE '%".$_SESSION['To']."%'
232       OR n_location LIKE '%".umlaute($_SESSION['To'])."%'
          OR n_municipality LIKE '%".umlaute($_SESSION['To'])."%'
          ORDER BY n_municipality";
     $res = mysql_query($qTo);
236  $num_rows = mysql_num_rows($res);
     $n1 = mysql_fetch_row($res);
     if($n1[0]>0){
     $messageT = $lang['messageTb'];
240  $selectTo = "<select name='ToSel' size='1' id='ToSel'
     onChange=\"transfer('ToSel','ToHide')\">
```

```
    <option value=''>".$lang['alternative']."</option>";
    while($n1){
244 $selectTo .= '<option value="'.$n1[0].'">';
    $selectTo .= ''.$n1[2].' | '.$n1[1].'';
    $selectTo .= '</option>';
    $n1 = mysql_fetch_row($res);
248 }
    $selectTo .= '</select>';
    //Re-Initialization beacuse no match!
    $_SESSION['ToID'] = 0;
252 $boolT = FALSE;
    $boolT = 2;
    } //endIf
    else{
256 $boolT = 3;
    $messageT = $lang['messageTc'];
    } //endElse
    } //endElse
260 } //endIf
    else{  //if field ""
    $boolT = 4;
    $messageT = $lang['messageTd'];
264 }  //endElse


    if($_SESSION['Via1'] != "" OR $cStrVia1Sel != ""){
268 if(strpos($_SESSION['Via1']," | ") OR $cStrVia1Sel != ""){
    $Where = "n_id = ".$_SESSION['Via1ID']."";}
    else{
    $Where = "WHERE n_location = '".trim($_SESSION['Via1'])."'
272          OR n_location = '".trim(umlaute($_SESSION['Via1']))."'";}
    $qVia1 = "SELECT n_id, n_location, n_municipality
         FROM ".cStrTABLENodes."
         WHERE ".$Where."";
276 $res = mysql_query($qVia1);
    $n1 = mysql_fetch_row($res);
    if($n1[0] > 0){
    $_SESSION['Via1ID'] = $n1[0];
280 $_SESSION['Via1'] = "".$n1[2]." | ".$n1[1]."";
    $boolV1 = 100;
    $messageV1 = $lang['messageV1a'];
    }
284 else{
    $qVia1 = "SELECT DISTINCT n_id, n_location, n_municipality
         FROM ".cStrTABLENodes."
         WHERE
288      MATCH (n_location, n_municipality) AGAINST ('".$_SESSION['Via1']."')
         OR n_location LIKE '%".$_SESSION['Via1']."%'
         OR n_municipality LIKE '%".$_SESSION['Via1']."%'
         OR n_location LIKE '%".umlaute($_SESSION['Via1'])."%'
292      OR n_municipality LIKE '%".umlaute($_SESSION['Via1'])."%'
         ORDER BY n_municipality";
    $res = mysql_query($qVia1);
    $num_rows = mysql_num_rows($res);
296 $n1 = mysql_fetch_row($res);
    if($n1[0]>0){
    $messageV1 = $lang['messageV1b'];
    $selectVia1 = "<select name='Via1Sel' size='1' id='Via1Sel'
300 onChange=\"transfer('Via1Sel','Via1Hide')\">
    <option value=''>".$lang['alternative']."</option>";
    while($n1){
    $selectVia1 .= '<option value="'.$n1[0].'">';
304 $selectVia1 .= ''.$n1[2].' | '.$n1[1].'';
    $selectVia1 .= '</option>';
    $n1 = mysql_fetch_row($res);
    }
308 $selectVia1 .= '</select>';
    //Re-Initialization beacuse no match!
    $_SESSION['Via1ID'] = 0;
    $boolV1 = 2;
```

```
312  } //endIf
     else{
     $boolV1 = 3;
     $messageV1 = $lang['messageV1c'];
316  } //endElse
     } //endElse
     } //endIf
     else{  //if field ""
320  $boolV1 = 4;
     $messageV1 = $lang['messageV1d'];
     }  //endElse

324


     if($_SESSION['Via2'] != "" OR $cStrVia2Sel != ""){
     if(strpos($_SESSION['Via2']," | ") OR $cStrVia2Sel != ""){
328  $Where = "n_id = ".$_SESSION['Via2ID']."";}
     else{
     $Where = "WHERE n_location = '".trim($_SESSION['Via2'])."'
              OR n_location = '".trim(umlaute($_SESSION['Via2']))."'";}
332  $qVia2 = "SELECT n_id, n_location, n_municipality
          FROM ".cStrTABLENodes."
          WHERE ".$Where."";
     $res = mysql_query($qVia2);
336  $n1 = mysql_fetch_row($res);
     if($n1[0] > 0){
     $_SESSION['Via2ID'] = $n1[0];
     $_SESSION['Via2'] = "".$n1[2]." | ".$n1[1]."";
340  $boolV2 = 100;
     $messageV2 = $lang['messageV2a'];
     }
     else{
344  $qVia2 = "SELECT DISTINCT n_id, n_location, n_municipality
          FROM ".cStrTABLENodes."
          WHERE
          MATCH (n_location, n_municipality) AGAINST ('".$_SESSION['Via2']."')
348       OR n_location LIKE '%".$_SESSION['Via2']."%'
          OR n_municipality LIKE '%".$_SESSION['Via2']."%'
          OR n_location LIKE '%".umlaute($_SESSION['Via2'])."%'
          OR n_municipality LIKE '%".umlaute($_SESSION['Via2'])."%'
352       ORDER BY n_municipality";
     $res = mysql_query($qVia2);
     $num_rows = mysql_num_rows($res);
     $n1 = mysql_fetch_row($res);
356  if($n1[0]>0){
     $messageV2 = $lang['messageV2b'];
     $selectVia2 = "<select name='Via2Sel' size='1' id='Via2Sel'
     onChange=\"transfer('Via2Sel','Via2Hide')\">
360  <option value=''>".$lang['alternative']."</option>";
     while($n1){
     $selectVia2 .= '<option value="'.$n1[0].'">';
     $selectVia2 .= ''.$n1[2].' | '.$n1[1].'';
364  $selectVia2 .= '</option>';
     $n1 = mysql_fetch_row($res);
     }
     $selectVia2 .= '</select>';
368  //Re-Initialization beacuse no match!
     $_SESSION['Via2ID'] = 0;
     $boolV2 = 2;
     } //endIf
372  else{
     $boolV2 = 3;
     $messageV2 = $lang['messageV2c'];
     } //endElse
376  } //endElse
     } //endIf
     else{  //if field ""
     $boolV2 = 4;
380  $messageV2 = $lang['messageV2d'];
     }  //endElse
```

```
    //Checking if a calculation is possible
384 if($boolF == 100 AND $boolT == 100 AND
    (($_SESSION['Via1'] != "" AND $boolV1 == 100) OR $_SESSION['Via1'] == "")
    AND
    (($_SESSION['Via2'] != "" AND $boolV2 == 100) OR $_SESSION['Via2'] == "")
388 ){
    $calcButton = '<input class="sub" type="submit"
     value="'.$lang['calcButton'].'" name="calcDijkstra"
     onmouseout="style.backgroundColor=\'#E5FF9E\'"
392  onmouseover="style.backgroundColor=\'#85B30B\'";>  ';
    // Update SESSIONS
    }
    else{
396 $calcButton = '';}

    // If Map is choosen, redirect to the map
    if ($boolFrom){
400 //sleep(5);
    header("location: http://".$host.$uri."/resUserLoadMap.php?loc=From&id=".$_SESSION['FromID']."
    &name=".htmlspecialchars($_SESSION['From'])."&L=".$langID."");}
    if ($boolTo){
404 //sleep(0.25);
    header("location: http://".$host.$uri."/resUserLoadMap.php?loc=To&id=".$_SESSION['ToID']."
    &name=".$_SESSION['To']."&L=".$langID."");}
    if ($boolVia1){
408 //sleep(0.25);
    header("location: http://".$host.$uri."/resUserLoadMap.php?loc=Via1&id=".$_SESSION['Via1ID']."
    &name=".$_SESSION['Via1']."&L=".$langID."");}
    if ($boolVia2){
412 //sleep(0.25);
    header("location: http://".$host.$uri."/resUserLoadMap.php?loc=Via2&id=".$_SESSION['Via2ID']."
    &name=".$_SESSION['Via2']."&L=".$langID."");}
    if ($cStrCalcDijkstra){
416 //sleep(0.25);
    header("location: http://".$host.$uri."/DijkstraMitHeuristik.php?L=".$langID."");}

    //Generating the Form
420 echo ("
    <html><head>

    <title>".$lang['title']."</title>
424 <meta name='generator' content='http://www.virTours.ch | Urs Sieber'>
    <meta name='keywords' content='Wandern, Wanderunge,
      Hiking, Hike, routeplanning, routenplanug, wegplaner,
    wanderroute, wanderwege, wanderweg, wegweiser, signpost, SVG, dynamic SVG, topology' />
428 <link rel='STYLESHEET' href='helpers/style.css' type='text/css'>
    <script type=\"Text/JavaScript\">

      function transfer(fr,to)
432    {
       document.getElementById(to).value = document.getElementById(fr).value;
       }

436 </script>

    </head><body>
    <table><tr><td class='heading'>
440 <img border='0' src='helpers/Pfeil_orange_grau.gif' width='11' height='11'> 
    ".$lang['formH1']."
    </td></tr></table>
    <p>".$lang['formIntro']."</p>
444 <form name=\"userInput\" method=\"POST\" action=\"".$_SERVER['PHP_SELF']."?L=".$langID."\">
    <p align='middle'>".$calcButton."</p>
    ");

448 echo ('<table><tr><td><fieldset style="padding: 5"><legend>'.$lang['labelFrom'].'</legend>
       <br>'.$messageF.' <br>');
    echo ('<input type="hidden" name="FromHide" id="FromHide" size="40" value="'.$_SESSION['FromID'].'">');
    if ($boolF == 2){
```

```
452  echo (''.$selectFrom.'
               
      <input class="sub" type="submit" value="'.$lang['formSubmitDD'].'" name="Calc"
      onmouseout="style.backgroundColor=\'#E5FF9E\'"
456   onmouseover="style.backgroundColor=\'#85B30B\'";>
      ');
        }
     else {
460  echo ('<input type="text" name="From" id="From" size="40" value="'.$_SESSION['From'].'">
               
      <input class="sec" type="submit" value="'.$lang['formSubmitDB'].'" name="Calc"
      onmouseout="style.backgroundColor=\'#E5FF9E\'"
464   onmouseover="style.backgroundColor=\'#85B30B\'";>
      ');
     }
     echo ('  <input class="sec" type="submit" value="'.$lang['formSubmitMap'].'" name="okFrom"
468   onmouseout="style.backgroundColor=\'#E5FF9E\'"
      onmouseover="style.backgroundColor=\'#85B30B\'";>
     <br>
     </fieldset></td></tr></table>
472  ');

     echo ('<table><tr><td><fieldset style="padding: 5"><legend>'.$lang['labelTo'].'</legend>
        <br>'.$messageT.'<br>');
476  echo ('<input type="hidden" name="ToHide" id="ToHide" size="40" value="'.$_SESSION['ToID'].'">');
     if ($boolT == 2){
     echo (''.$selectTo.'
                
480   <input class="sub" type="submit" value="'.$lang['formSubmitDD'].'" name="Calc"
      onmouseout="style.backgroundColor=\'#E5FF9E\'"
      onmouseover="style.backgroundColor=\'#85B30B\'";>
      ');
484     }

     else {
     echo ('<input type="text" name="To" id="To" size="40" value="'.$_SESSION['To'].'">
488            
      <input class="sec" type="submit" value="'.$lang['formSubmitDB'].'" name="Calc"
      onmouseout="style.backgroundColor=\'#E5FF9E\'"
      onmouseover="style.backgroundColor=\'#85B30B\'";>
492   ');
     }
     echo ('  <input class="sec" type="submit" value="'.$lang['formSubmitMap'].'" name="okTo"
      onmouseout="style.backgroundColor=\'#E5FF9E\'"
496   onmouseover="style.backgroundColor=\'#85B30B\'";>
     <br>
     </fieldset></td></tr></table>
     ');
500
     echo ('<table><tr><td><fieldset style="padding: 5"><legend>'.$lang['labelV1'].'</legend>
        <br>'.$messageV1.'<br>');
     echo ('<input type="hidden" name="Via1Hide" id="Via1Hide" size="40" value="'.$_SESSION['Via1ID'].'">');
504  if ($boolV1==2){
     echo (''.$selectVia1.'
                
      <input class="sub" type="submit" value="'.$lang['formSubmitDD'].'" name="Calc"
508   onmouseout="style.backgroundColor=\'#E5FF9E\'"
      onmouseover="style.backgroundColor=\'#85B30B\'";>
      ');
        }
512  else {
     echo ('<input type="text" name="Via1" id="Via1" size="40" value="'.$_SESSION['Via1'].'">
               
      <input class="sec" type="submit" value="'.$lang['formSubmitDB'].'" name="Calc"
516   onmouseout="style.backgroundColor=\'#E5FF9E\'"
      onmouseover="style.backgroundColor=\'#85B30B\'";>
      ');
     }
520  echo ('  <input class="sec" type="submit" value="'.$lang['formSubmitMap'].'" name="okVia1"
      onmouseout="style.backgroundColor=\'#E5FF9E\'"
```

```
 onmouseover="style.backgroundColor=\'#85B30B\'";>
<br>
</fieldset></td></tr></table>
');

echo ('<table><tr><td><fieldset style="padding: 5"><legend>'.$lang['labelV2'].'</legend>
   <br>'.$messageV2.' <br>');
echo ('<input type="hidden" name="Via2Hide" id="Via2Hide" size="40" value="'.$_SESSION['Via2ID'].'">');
if ($boolV2==2){
echo (''.$selectVia2.'
           
 <input class="sub"  type="submit" value="'.$lang['formSubmitDD'].'" name="Calc"
 onmouseout="style.backgroundColor=\'#E5FF9E\'"
 onmouseover="style.backgroundColor=\'#85B30B\'";>
 ');
   }
else {
echo ('<input type="text" name="Via2" id="Via2" size="40" value="'.$_SESSION['Via2'].'">
          
 <input class="sec" type="submit" value="'.$lang['formSubmitDB'].'" name="Calc"
 onmouseout="style.backgroundColor=\'#E5FF9E\'"
 onmouseover="style.backgroundColor=\'#85B30B\'";>
 ');
}
echo ('  <input class="sec" type="submit" value="'.$lang['formSubmitMap'].'" name="okVia2"
 onmouseout="style.backgroundColor=\'#E5FF9E\'"
 onmouseover="style.backgroundColor=\'#85B30B\'";>
</fieldset></td></tr></table>
');

/* Possible cost factors:
- time "time"
- distance "hvdist"
- gradient min "gradient"
- m in altitude min "hdiff"
- natural surface preferred "time_surface_nat"
- artificial surface preferred "time_surface_art"
- time gradient min "time_gradient"
- distance natural surface min "dist_surface_nat"
- distance hard surface min "dist_surface_art"
*/

if($_SESSION['Weight'] == "time"){
$time = "checked";
}
else if ($_SESSION['Weight'] == "hvdist"){
$hvdist = "checked";
}
else if ($_SESSION['Weight'] == "gradient"){
$gradient = "checked";
}
else if ($_SESSION['Weight'] == "hdiff"){
$hdiff = "checked";
}
else if ($_SESSION['Weight'] == "time_gradient"){
$time_gradient = "checked";
}
else if ($_SESSION['Weight'] == "time_surface_art"){
$time_surface_art = "checked";
}
else if ($_SESSION['Weight'] == "time_surface_nat"){
$time_surface_nat = "checked";
}
else if ($_SESSION['Weight'] == "dist_surface_art"){
$dist_surface_art = "checked";
}
else if ($_SESSION['Weight'] == "dist_surface_nat"){
$dist_surface_nat = "checked";
}
else {
```

```
592  $time = "checked";
     }

     echo ('<fieldset style="padding: 5"><legend>'.$lang['labelWeight'].'</legend><table><tr><td><br>
596  <input type="radio" value="time" '.$time.' name="Weight"> '.$lang['time'].' <br>
     <input type="radio" value="hvdist" '.$hvdist.' name="Weight"> '.$lang['hvdist'].' <br>
     <input type="radio" value="gradient" '.$gradient.' name="Weight"> '.$lang['gradient'].'<br>
     <input type="radio" value="hdiff" '.$hdiff.' name="Weight"> '.$lang['hdiff'].' <br>
600  <input type="radio" value="time_gradient" '.$time_gradient.'
     name="Weight"> '.$lang['time_gradient'].' <br></td><td>
     <input type="radio" value="time_surface_art" '.$time_surface_art.'
     name="Weight"> '.$lang['time_surface_art'].' <br>
604  <input type="radio" value="time_surface_nat" '.$time_surface_nat.'
     name="Weight"> '.$lang['time_surface_nat'].' <br>
     <input type="radio" value="dist_surface_art"  '.$dist_surface_art.'
     name="Weight"> '.$lang['dist_surface_art'].' <br>
608  <input type="radio" value="dist_surface_nat"  '.$dist_surface_nat.'
     name="Weight"> '.$lang['dist_surface_nat'].' <br>
     </td></tr></table></fieldset>
     ');
612
     echo ('<p>'.$calcButton.'
     <input class="sub"  type="submit" value="'.$lang['formSubmitDD'].'" name="Calc"
      onmouseout="style.backgroundColor=\'#E5FF9E\'"
616   onmouseover="style.backgroundColor=\'#85B30B\'";>  
     <input  class="sub"  type="submit" value="'.$lang['formNewSearch'].'" name="new"
      onmouseout="style.backgroundColor=\'#E5FF9E\'"
      onmouseover="style.backgroundColor=\'#85B30B\'";>
620  </p>

     </form>

624  <table><tr>
     <td><a href="http://www.virtours.ch" target="_blank">
     <img src="MailClass/Logo.jpg" width="100px" border="0"></a></td>
     <td><p class="footer" align="right">'.$lang['mailAdv1'].'<br/>
628  </p></td></tr></table>
     </body></html>
     ');
```

C.9.2 LayerChoose.php

This script computes the layer where the user can choose from it's preferred location. The search is limited due to minimize data traffic and loading time to the perimeter lying in the near surrounding of the search.

```
     <?php
     header("Content-Type: image/svg+xml");

 4   //Session load
     session_start();

     //Constants initialization
 8   include("constants.php");
     $cDB = $LOCAL_CONS["DB"];
     $cColors = $LOCAL_CONS["LayerNodesSignpost"];
     $cNames = $LOCAL_CONS["LayerNodeNames"];
12

     // DATABASE:-----------------------------------------------
     define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
16   define ('cStrUSERDB',$cDB['cStrUSERDB']);
     define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
     define ('cStrDB',$cDB['cStrDB']);
     define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
20   define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
     define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
     // -------------------------------------------------------

24   //DB Connection
```

```
    $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);

28     $loc  = explode('qqq',$_REQUEST['loc']);
       $cStrKind = $loc[0];
       $cStrLoc = html_entity_decode($loc[2]);
       $cStrID = $loc[1];
32     $langID = $loc[3];

       $Radius = 5000; //extra Layer with data in 10km distance
       $mRadius = (-1)*$Radius;
36     $host   = $_SERVER['HTTP_HOST'];
       $uri   = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');

    // Substitute öäü
40  function umlaute($c){
    $c = eregi_replace('ä','ae',$c);
    $c = eregi_replace('ü','ue',$c);
    $c = eregi_replace('ö','oe',$c);
44  return  $c;
    }

    // Select averages
48     if($cStrLoc != ""){
    if($cStrID > 0){
    $t1 = "WHERE n_id = ".$cStrID."";}
    else{
52  $t1 = "WHERE
         n_municipality LIKE '%".$cStrLoc."%'
         OR n_municipality LIKE '%".umlaute($cStrLoc)."%'
         OR n_location LIKE '%".$cStrLoc."%'
56       OR n_location LIKE '%".umlaute($cStrLoc)."%'";}

    $qLoc = "SELECT AVG(n_x), AVG(n_y)
         FROM ".cStrTABLENodes."
60       ".$t1."";

    $res = mysql_query($qLoc);
    $n1 = mysql_fetch_row($res);
64  }

    //Select Nodes and Names for extra Layer
    // Check if average found, else show every node with linked name
68  if($n1[0]>0){
    $Where = "type = 2 AND
             n_x - ".$n1[0]." < ".$Radius." AND
             n_x - ".$n1[0]." > ".$mRadius." AND
72           n_y - ".$n1[1]." < ".$Radius." AND
             n_y - ".$n1[1]." > ".$mRadius."";
             }
        else{
76       $Where = "type = 2";
         }

          $eanfrage = "SELECT n_x, n_y, n_location, n_id FROM ".cStrTABLENodes."
80           WHERE ".$Where."";
      $nodes = mysql_query($eanfrage);

      $circ = '';
84    $tex = '';
       while($ezeile = mysql_fetch_row($nodes)){
          $url = "loc=".$ezeile[3]."qqq".$cStrKind."";
         $circ .= "<a xlink:href='userForm.php?".$url."&#38;L=".$langID."' target='hike'>
88        <circle cx=\"".round($ezeile[0])."\" cy=\"-".round($ezeile[1])."\"
          r=\"".$cColors['nodeSize']."\"/></a>";
         $tex .= "<a xlink:href='userForm.php?".$url."&#38;L=".$langID."' target='hike'>
         <text x='".$ezeile[0]."' y='-".$ezeile[1]."'>".htmlspecialchars($ezeile[2])."</text></a>";
92       }

      echo ('<g xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
```

```
      <g fill="'.$cColors["nodeFill"].'" stroke="'.$cColors["nodeStrokeColor"].'"
96     stroke-width="'.$cColors["nodeStrokeWidth"].'">');
       echo ('.$circ.'</g>');
       echo ('<g text-anchor="'.$cNames["textAnchor"].'" fill="'.$cNames["textFill"].'"
        transform="translate('.$cNames["translatex"].' '.$cNames["translatey"].')"
100     font-size="'.$cNames["fontSize"].'">');
      echo ('.$tex.');
       echo('</g>');
     echo('</g>');
104 ?>
```

C.9.3 LayerChooseAll.php

If the search word is not found, the searching perimeter for a preferred location is widened to the full extension and every existing location is printed.

```
    <?php
    header("Content-Type: image/svg+xml");

 4  //Session load
    session_start();

    //Constants initialization
 8  include("constants.php");
    $cDB = $LOCAL_CONS["DB"];
    $cColors = $LOCAL_CONS["LayerNodesSignpost"];
    $cNames = $LOCAL_CONS["LayerNodeNames"];
12
    // DATABASE:-----------------------------------------------
    define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
    define ('cStrUSERDB',$cDB['cStrUSERDB']);
16  define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
    define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
    define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
20  define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    // -------------------------------------------------------

    //DB Connection
24  $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);

       $loc  = explode('qqq',$_REQUEST['loc']);
28     $cStrKind = $loc[0];


       $host   = $_SERVER['HTTP_HOST'];
32     $uri   = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');


    // Select averages
36
          $eanfrage = "SELECT n_x, n_y, n_location, n_id FROM ".cStrTABLENodes."
             WHERE type = 2";
      $nodes = mysql_query($eanfrage);
40
      $circ = '';
      $tex = '';
       while($ezeile = mysql_fetch_row($nodes)){
44        $url = "loc=".$ezeile[3]."qqq".$cStrKind."";
         $circ .= "<a xlink:href='userForm.php?".$url."&#38;L=".$_REQUEST['L']."' target='hike'>
          <circle cx=\"".round($ezeile[0])."\" cy=\"-".round($ezeile[1])."\"
          r=\"".$cColors['nodeSize']."\"/></a>";
48        $tex .= "<a xlink:href='userForm.php?".$url."&#38;L=".$_REQUEST['L']."' target='hike'>
          <text x='".$ezeile[0]."' y='-".$ezeile[1]."'>".htmlspecialchars($ezeile[2])."</text></a>";
          }

52    echo ('<g xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
       <g fill="'.$cColors["nodeFill"].'" stroke="'.$cColors["nodeStrokeColor"].'"
```

```
       stroke-width="'.$cColors["nodeStrokeWidth"].'">');
         echo ('.$circ.'</g>');
56       echo ('<g text-anchor="'.$cNames["textAnchor"].'" fill="'.$cNames["textFill"].'"
          transform="translate('.$cNames["translatex"].' '.$cNames["translatey"].')"
          font-size="'.$cNames["fontSize"].'">');
       echo ('.$tex.');
60       echo('</g>');
      echo('</g>');
    ?>
```

C.9.4 userLoadMap.php

The subscript which echoes the SVG map on which users can choose their preferred location is listed below.

```
    <?php
    header("Content-Type: image/svg+xml");
    //sleep(2);
 4  //Session load
    session_start();

    //Constants initialization
 8  include("constants.php");
    $cDB = $LOCAL_CONS["DB"];
    $cColors = $LOCAL_CONS["Colors"];
    $cMAP = $LOCAL_CONS["MAP"];
12
    //Language selection
    include("locallang.php");
    $langID = $_REQUEST['L'];
16  if ($langID == 1){
    $langcode = "en";}
    else if ($langID == 0){
    $langcode = "default";}
20  else {$langcode = "default";}
    $lang = $LOCAL_LANG[$langcode];

    // DATABASE:-----------------------------------------------
24  define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
    define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
28  define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
    define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
    define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    define ('cStrTABLEUserQ',$cDB['cStrTABLEUserQ']);
32  // -------------------------------------------------------


    // Substitute öää
36  function umlaute($c){
    $c = eregi_replace('ä','ae',$c);
    $c = eregi_replace('ü','ue',$c);
    $c = eregi_replace('ö','oe',$c);
40  return  $c;
    }

       $cStrKind = $_REQUEST['loc'];
44     $cStrLoc = html_entity_decode($_REQUEST['name']);
       $cStrID = $_REQUEST['id'];

    // Tests wheter the entries exist
48  $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);

    // Select AVG from selection to get a center of municipality
52  if($cStrLoc != ""){

    if($cStrID > 0){
```

```
    $t1 = "WHERE n_id = ".$cStrID."";}
 56 else{
    $t1 = "WHERE
         n_municipality LIKE '%".$cStrLoc."%'
         OR n_municipality LIKE '%".umlaute($cStrLoc)."%'
 60      OR n_location LIKE '%".$cStrLoc."%'
         OR n_location LIKE '%".umlaute($cStrLoc)."%'";}

    $qLoc = "SELECT AVG(n_x), AVG(n_y)
 64      FROM ".cStrTABLENodes." ".$t1."";
    $res = mysql_query($qLoc);
    $n1 = mysql_fetch_row($res);
    }
 68
    // Get the viewport focussed on the userInput
    // if a match is found
    if($n1[0]>0){
 72 $Win = $cMAP['areaChoose'];
    $xDia = $Win*($cMAP['xDia']/$cMAP['yDia']);
    $xMap = $n1[0]-($xDia/2);
    $yMap = ((-1)*$n1[1])-($Win/2);
 76 }
    else{
    $Win = $cMAP['yDia'];
    $xDia = $cMAP['xDia'];
 80 $xMap = $cMAP['xC'];
    $yMap = $cMAP['yC'];
    }

 84 //No transit over the Map borders
    if($xMap > $cMAP['xC']+$cMAP['xDia']-$xDia){
    $xMap = $cMAP['xC']+$cMAP['xDia']-$xDia;
    }
 88 else if($xMap < $cMAP['xC']){
    $xMap = $cMAP['xC'];
    }
    else if($yMap < $cMAP['yC']){
 92 $yMap = $cMAP['yC'];
    }
    else if($yMap > $cMAP['yC']+$cMAP['yDia']-$Win){
    $yMap = $cMAP['yC']+$cMAP['yDia']-$Win;
 96 }

    echo("
    <svg zoomAndPan=\"disable\" viewBox=\"0 0 560 480\" width=\"100%\" height=\"100%\"
100  xmlns=\"http://www.w3.org/2000/svg\" version=\"1.0\" xmlns:xlink=\"http://www.w3.org/1999/xlink\">
    <title>".$lang["title"]."</title>
    <defs>

104 <symbol id=\"mySymbArrow\" overflow=\"visible\" fill=\"none\" stroke=\"#79AA3D\" stroke-width=\"1\">
      <polyline points=\"-3,-5 3,-5 3,1 5,1 0,5 -5,1 -3,1 -3,-5\"/>
    </symbol>

108 <script>");
    include("naviJS.php");
    include("loadOfDetails.php");

112 //The Reset fct to get the focus back on the calculated track
    //has to be definied locally
    echo("
          function myResetCalc(){
116       document.getElementById('myMap').
          setAttributeNS(null,'viewBox','".$xMap." ".$yMap." ".$xDia." ".$Win."');
          mySetOverviewRect([".$xMap.", ".$yMap.", ".$xDia.", ".$Win."]);
            }
120         ");

       echo ("</script>
      </defs>
124 <rect id=\"backGroundDelimiter\" x=\"0\" y=\"0\" width=\"562\"
```

```
height=\"485\" fill=\"".$cColors["bgProject"]."\"/>

  <rect id=\"mapAreaDelimiter\" x=\"171\" y=\"4\" width=\"384\"
  height=\"473\" fill=\"".$cColors["symbAndBorder"]."\"/>

  <!-- VERSCHACHTELTES SVG-ELEMENT MIT DER HAUPTKARTE -->
<svg id=\"myMap\" viewBox=\"".$xMap." ".$yMap." ".$xDia." ".$Win."\"
x=\"172\" y=\"5\" width=\"382\" height=\"471\">
<rect id=\"mapBG\" x=\"".$cMAP['xC']."\" y=\"".$cMAP['yC']."\"
width=\"".$cMAP['xDia']."\" height=\"".$cMAP['yDia']."\" fill=\"".$cColors["bgMap"]."\"/>

<g id=\"myPassiveElements\">");

//Layers to copy by user input into this viewport
echo("
<g id='LayerGroup1'>
<g id='LayerEnvpoly'/>
<g id='useEnvpoly'> </g>
<g id='LayerEnvcomm'/>
<g id='LayerEnvvec'/>
<g id='LayerEdges'/>
<g id='LayerNodesSignpost'/>
<g id='LayerNodesSecondary'/>
<g id='LayerNodeNames'/>
<g id='LayerRes'/>
<g id='LayerNodeCommunities'/>
<g id='LayerChoose'/>
</g>
");

echo("<g id=\"myMapFrame\">
<path id=\"myGrid10km\" stroke=\"".$cColors["Grid"]."\"
stroke-width=\"5\" fill=\"none\" d=\"M730000 ".$cMAP['yC']."
v ".$cMAP['yDia']." M740000 ".$cMAP['yC']." v ".$cMAP['yDia']."
M750000 ".$cMAP['yC']." v ".$cMAP['yDia']." M760000 ".$cMAP['yC']."
v ".$cMAP['yDia']." M".$cMAP['xC']." -220000 h ".$cMAP['xDia']."
M".$cMAP['xC']." -230000 h ".$cMAP['xDia']." M".$cMAP['xC']."
-240000 h ".$cMAP['xDia']." M".$cMAP['xC']." -250000 h
".$cMAP['xDia']." M".$cMAP['xC']." -260000 h ".$cMAP['xDia']."\"/>
<path id=\"myGrid5km\" stroke=\"".$cColors["Grid"]."\"
stroke-width=\"3\" fill=\"none\" d=\"M735000 ".$cMAP['yC']."
v ".$cMAP['yDia']." M745000 ".$cMAP['yC']." v ".$cMAP['yDia']."
M755000 ".$cMAP['yC']." v ".$cMAP['yDia']." M765000 ".$cMAP['yC']."
v ".$cMAP['yDia']." M".$cMAP['xC']." -215000 h ".$cMAP['xDia']."
M".$cMAP['xC']." -225000 h ".$cMAP['xDia']." M".$cMAP['xC']." -235000
h ".$cMAP['xDia']." M".$cMAP['xC']." -245000 h ".$cMAP['xDia']."
M".$cMAP['xC']." -255000 h ".$cMAP['xDia']." M".$cMAP['xC']."
-265000 h ".$cMAP['xDia']."\"/>

<g id=\"myBorder\">
<rect id=\"myBorderOuter\" stroke=\"".$cColors["mapBorderOuter"]."\"
stroke-width=\"900\" fill=\"none\" x=\"725000\" y=\"-268000\"
width=\"".$cMAP['xDia']."\" height=\"55466\"/>
<rect id=\"myBorderInner\" stroke=\"".$cColors["mapBorderInner"]."\"
stroke-width=\"800\" fill=\"none\" x=\"725000\" y=\"-268000\"
width=\"".$cMAP['xDia']."\" height=\"55466\"/>
<g id=\"myGridCoords\" fill=\"".$cColors["Grid"]."\"
font-size=\"400\" text-anchor=\"middle\">
<text id=\"myGridCoords1\" x=\"740000\" y=\"-267670\">740'000</text>
<text id=\"myGridCoords2\" x=\"730000\" y=\"-267670\">730'000</text>
<text id=\"myGridCoords3\" x=\"750000\" y=\"-267670\">750'000</text>
<text id=\"myGridCoords4\" x=\"760000\" y=\"-267670\">760'000</text>
<text id=\"myGridCoords5\" x=\"725330\" y=\"-220000\"
transform=\"rotate(-90 725330 -220000)\">220'000</text>
<text id=\"myGridCoords6\" x=\"725330\" y=\"-230000\"
transform=\"rotate(-90 725330 -230000)\">230'000</text>
<text id=\"myGridCoords7\" x=\"725330\" y=\"-240000\"
transform=\"rotate(-90 725330 -240000)\">240'000</text>
<text id=\"myGridCoords8\" x=\"725330\" y=\"-250000\"
transform=\"rotate(-90 725330 -250000)\">250'000</text>
```

```
    <text id=\"myGridCoords9\" x=\"725330\" y=\"-260000\"
196 transform=\"rotate(-90 725330 -260000)\">260'000</text>
    </g>
    </g>
    </g>
200 ");

    echo ("

204 <g id=\"myMapElementsOverTheGrid\">
    <g id=\"myScaleBar\">
    <path id=\"strecke\" stroke=\"black\" stroke-width=\"100\"
    fill=\"none\" d=\"M727000 -214800 v 400 h 5000 v -400\"/>
208 <text id=\"m_text\" fill=\"black\" font-size=\"1000\"
    font-weight=\"bold\" x=\"729500\" y=\"-213300\" text-anchor=\"middle\">5000 m</text>
    </g>
    </g>
212 </g><!-- ENDE PASSIVER ELEMENTE -->
    </svg><!-- ENDE DES VERSCHACHTELTEN SVG-ELEMENTS MIT DER HAUPTKARTE -->

    <rect id=\"infoPanelDelimiter\" x=\"20\" y=\"5\" width=\"132\"
216 height=\"471\" fill=\"".$cColors["bgInfoPanel"]."\"/>
    <g id=\"myNavig\" transform=\"translate(30,50) scale(1.6)\"
    stroke=\"".$cColors["symbAndBorder"]."\" stroke-width=\"1\"
     onmouseover=\"evt.target.setAttributeNS(null,'fill','".$cColors["bgInfoPanel"]."')\"
220  cursor=\"pointer\"
     onmouseout=\"evt.target.setAttributeNS(null,'fill','".$cColors["navMouseout"]."')\">
    <g shape-rendering=\"crispEdges\" fill=\"".$cColors["navMouseout"]."\">
    <rect id=\"myPanNW\" x=\"0\" y=\"0\" width=\"20\" height=\"20\" onclick=\"myPan(-1,-1)\"/>
224 <rect id=\"myPanNN\" x=\"25\" y=\"0\" width=\"20\" height=\"20\" onclick=\"myPan(0,-1)\"/>
    <rect id=\"myPanNE\" x=\"50\" y=\"0\" width=\"20\" height=\"20\" onclick=\"myPan(1,-1)\"/>
    <rect id=\"myPanWW\" x=\"0\" y=\"25\" width=\"20\" height=\"20\" onclick=\"myPan(-1,0)\"/>
    <rect id=\"myResetAll\" x=\"25\" y=\"25\" width=\"20\" height=\"20\" onclick=\"myResetCalc()\"/>
228 <rect id=\"myPanEE\" x=\"50\" y=\"25\" width=\"20\" height=\"20\" onclick=\"myPan(1,0)\"/>
    <rect id=\"myPanSW\" x=\"0\" y=\"50\" width=\"20\" height=\"20\" onclick=\"myPan(-1,1)\"/>
    <rect id=\"myPanSS\" x=\"25\" y=\"50\" width=\"20\" height=\"20\" onclick=\"myPan(0,1)\"/>
    <rect id=\"myPanSE\" x=\"50\" y=\"50\" width=\"20\" height=\"20\" onclick=\"myPan(1,1)\"/>
232 </g>
    <circle cx=\"10\" cy=\"87\" r=\"10\" fill=\"white\" onclick=\"myZoom(1)\"/>
    <circle cx=\"35\" cy=\"87\" r=\"10\" fill=\"white\" onclick=\"myZoom(-1)\"/>
    <circle cx=\"60\" cy=\"87\" r=\"10\" fill=\"white\" onclick=\"myReset()\"/>
236 <g id=\"myNavigSymbols\" pointer-events=\"none\">
    <use xlink:href=\"#mySymbArrow\" transform=\"translate(10,10),rotate(135)\" />
    <use xlink:href=\"#mySymbArrow\" transform=\"translate(35,10),rotate(180)\" />
    <use xlink:href=\"#mySymbArrow\" transform=\"translate(60,10),rotate(-135)\" />
240 <use xlink:href=\"#mySymbArrow\" transform=\"translate(10,35),rotate(90)\" />
    <circle cx=\"35\" cy=\"35\" r=\"7.5\" fill=\"none\" stroke=\"#79AA3D\" stroke-width=\"1\"/>
    <circle cx=\"35\" cy=\"35\" r=\"5\" fill=\"#79AA3D\" stroke=\"#79AA3D\"/>
    <use xlink:href=\"#mySymbArrow\" transform=\"translate(60,35),rotate(-90)\" />
244 <use xlink:href=\"#mySymbArrow\" transform=\"translate(10,60),rotate(45)\" />
    <use xlink:href=\"#mySymbArrow\" transform=\"translate(35,60)\" />
    <use xlink:href=\"#mySymbArrow\" transform=\"translate(60,60),rotate(-45)\" />
    <text x=\"10\" y=\"97\" fill=\"".$cColors["symbAndBorder"]."\"
248 text-anchor=\"middle\" font-size=\"30\">+</text>
    <text x=\"35\" y=\"97\" fill=\"".$cColors["symbAndBorder"]."\"
    text-anchor=\"middle\" font-size=\"40\">-</text>
    <text x=\"60\" y=\"93\" fill=\"".$cColors["symbAndBorder"]."\"
252 text-anchor=\"middle\" font-size=\"16\">R</text>
    </g>
    </g>

256 <g id=\"myCoordPanel\" transform=\"translate(0,-430)\" fill=\"".$cColors["fontPanel"]."\">
         <text x=\"30\" y=\"453\" id=\"myCursorMapX\" font-size=\"12\" font-weight=\"bold\">
         ".$lang["mapChoose".$cStrKind.""]."</text>
         <text x=\"30\" y=\"465\" id=\"myCursorMapY\" font-size=\"7\">(
260      ".$lang["mapChooseHowTo"].")</text>
    </g>");

    //-----------------------------------------------------------
264 //Layer Panels and Names, References to functions
```

```
echo("
<g id=\"myLayers\" transform=\"translate(30,390)\" cursor=\"pointer\">
<g onmouseover=\"evt.target.setAttributeNS(null,'fill','".$cColors['fgInfoPanel']."')\"
onmouseout=\"evt.target.setAttributeNS(null,'fill','".$cColors['navMouseout']."')\"
shape-rendering=\"crispEdges\" fill=\"".$cColors['navMouseout']."\" stroke-width=\"1\"
stroke=\"".$cColors['symbAndBorder']."\">
   <g fill=\"".$cColors['symbAndBorder']."\" font-size=\"11\">
    <text id=\"Layer1\" x=\"2\" y=\"30\" >x</text>
    <text id=\"Layer2\" x=\"2\" y=\"15\">x</text>
    <text id=\"Layer3\" x=\"2\" y=\"45\">x</text>
    <text id=\"Layer4\" x=\"2\" y=\"60\">x</text>
    <text id=\"Layer7\" x=\"2\" y=\"75\">x</text>
   </g>
   <g fill-opacity=\"1\">
    <rect id=\"myLayer1\" x=\"0\" y=\"22\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerEdges.php', 'LayerEdges', 'UTF-8', 'myLayer1')\"/>
    <rect id=\"myLayer2\" x=\"0\" y=\"7\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerChooseAll.php?loc=".$cStrKind."qqq".$cStrID."
    qqq".umlaute($cStrLoc)."qqq".$_REQUEST['L']."',
     'LayerNodesSignpost', 'UTF-8', 'myLayer2')\"/>
    <rect id=\"myLayer3\" x=\"0\" y=\"37\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerEnvcomm.php', 'LayerEnvcomm', 'UTF-8', 'myLayer3')\"/>
    <rect id=\"myLayer4\" x=\"0\" y=\"52\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerChoose.php?loc=".$cStrKind."qqq".$cStrID."
    qqq".umlaute($cStrLoc)."qqq".$_REQUEST['L']."',
     'LayerChoose', 'UTF-8', 'myLayer4')\"
    onload=\"myLayer('LayerChoose.php?loc=".$cStrKind."qqq".$cStrID."
    qqq".umlaute($cStrLoc)."qqq".$_REQUEST['L']."',
     'LayerChoose', 'UTF-8', 'myLayer4')\"/>
    <rect id=\"myLayer7\" x=\"0\" y=\"67\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerEnvpoly.php', 'LayerEnvpoly', 'UTF-8', 'myLayer7')\"/>
   </g>
   </g>
   <g fill=\"".$cColors['fontPanel']."\" font-size=\"9\">
<text x=\"14\" y=\"31\" onclick=\"myLayer('LayerEdges.php', 'LayerEdges',
'UTF-8', 'myLayer1')\">".$lang["layerEdges"]."</text>
<text x=\"14\" y=\"17\" onclick=\"myLayer('LayerChooseAll.php?loc=".$cStrKind."qqq".$cStrID."
qqq".umlaute($cStrLoc)."qqq".$_REQUEST['L']."',
'LayerNodesSignpost', 'UTF-8', 'myLayer2')\">".$lang["layerChooseAll"]."</text>
<text x=\"14\" y=\"46\" onclick=\"myLayer('LayerEnvcomm.php', 'LayerEnvcomm',
'UTF-8', 'myLayer3')\">".$lang["layerComm"]."</text>
<text x=\"14\" y=\"60\"
onclick=\"myLayer('LayerChoose.php?loc=".$cStrKind."qqq".$cStrID."
qqq".umlaute($cStrLoc)."qqq".$_REQUEST['L']."', 'LayerChoose',
'UTF-8', 'myLayer4')\">".$lang["layerChoose"]."</text>
<text x=\"14\" y=\"75\"
onclick=\"myLayer('LayerEnvpoly.php','LayerEnvpoly', 'UTF-8', 'myLayer7')\"
onload=\"myLayer('LayerEnvpoly.php','LayerEnvpoly', 'UTF-8', 'myLayer7')\">
".$lang["layerEnv"]."</text>
<text x=\"-4\" y=\"-158\">".$lang["dragMap"]."</text>
</g>
   </g>


<rect  x=\"25\" y=\"238\" width=\"120\" height=\"148\"/>

<svg id=\"myOverviewMap\" onmouseup=\"myDrag(evt)\" onmousemove=\"myDrag(evt)\"
viewBox=\"725000 -268000 45000 55466\" x=\"25\" y=\"238\" width=\"120\" height=\"148\">
<rect id=\"myBackgroundRect\" x=\"".$cMAP['xC']."\" y=\"".$cMAP['yC']."\"
width=\"".$cMAP['xDia']."\" height=\"".$cMAP['yDia']."\"
fill=\"".$cColors["overviewRectBg"]."\" stroke-width=\"0\"/>

<use xlink:href=\"#myGrid10km\"/>
<use xlink:href=\"#myGrid5km\"/>
<use xlink:href=\"#LayerEnvpoly\"/>
<use xlink:href=\"#myBorder\"/>

<rect id=\"myOverviewRect\" onmousedown=\"myDrag(evt)\"
x=\"".$xMap."\" y=\"".$yMap."\" width=\"".$xDia."\"
height=\"".$Win."\" fill=\"#C2DD71\" fill-opacity=\".4\"
```

```
    stroke=\"".$cColors["symbAndBorder"]."\" stroke-opacity=\"1\" stroke-width=\"100\"/>
336
    </svg>
    <!-- ENDE INFOBLOCK -->
    </svg>
340 ");

    ?>
```

C.9.5 resMap.php

An example script which generates a html frameset containing a main frame with header and footer as an inline frame loading the SVG graphic into the frameset.

```
    <?php

    //Session load
 4  session_start();


    //Language selection
 8  include("locallang.php");
    $langID = $_REQUEST['L'];
    if ($langID == 1){
    $langcode = "en";}
12  else if ($langID == 0){
    $langcode = "default";}
    else {$langcode = "default";}
    $lang = $LOCAL_LANG[$langcode];
16
    echo('
    <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
    <html>
20  <head>
    <title>'.$lang['title'].'</title>
    <meta name="generator" content="http://www.virTours.ch | Urs Sieber">
    <meta name="keywords" content="Wandern, Wanderunge,
24  Hiking, Hike, routeplanning, routenplanug, wegplaner,
    wanderroute, wanderwege, wanderweg, wegweiser, signpost, SVG, dynamic SVG, topology" />
    <link rel="STYLESHEET" href="helpers/style.css" type="text/css">
    </head>
28  <body>
    <table><tr><td class="heading">
    <img border="0" src="helpers/Pfeil_orange_grau.gif" width="11" height="11">
      '.$lang['mapH1'].'</td></tr></table>
32  <p>'.$lang['mapIntro'].'</p>
    <table><tr><td>
    <ul>
    <li>
36  '.$lang['headMap'].'
    </li>
    <li><a href ="resDataPath.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
    '.$lang['headData'].'
40  </a></li>
    <li><a href ="resMail.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
    '.$lang['headMail'].'
    </a></li>
44  <li><a href ="resDia.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
    '.$lang['headDia'].'
    </a></li>
    <li><a href ="BlankMap.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_blank">
48  '.$lang['mapWin'].'
    </a></li><br><br>
    <li><a href ="userForm.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_self">
    '.$lang['headWeight'].'
52  </a></li>
    <li><a href ="destroySession.php?L='.$langID.'" target="_self">
    '.$lang['headNew'].'
    </a></li>
```

```
</ul>
</td></tr></table>');

if($_SESSION['Via1'] != "")
{$via1 = "via ".$_SESSION['Via1']." ";} else {$via1 = "";}
if($_SESSION['Via2'] != "")
{$via2 = "".$lang['headAnd']." via ".$_SESSION['Via2']." ";}
else {$via2 = "";}

echo ('
<p>'.$lang['mapRoute1'].' "<b>'.$_SESSION['From'].'</b>
" '.$via1.''.$via2.' '.$lang['diaRoute2'].' "<b>'.$_SESSION['To'].'</b>".<br />
'.$lang['factor'].': <b>'.$_SESSION['Weight'].'</b></p>

<p align = "right" class="footer">
<a href ="resMail.php?'.session_name().'='.session_id().'&L='.$langID.'"
target="_self"> <img border="0" src="helpers/email.gif"></a>
<a href ="BlankMap.php?'.session_name().'='.session_id().'&L='.$langID.'"
target="_blank"> <img border="0" src="helpers/zoom.gif"></a>
<a href ="resDia.php?'.session_name().'='.session_id().'&L='.$langID.'"
target="_self"> <img border="0" src="helpers/diagram.gif"></a>
<a href ="resMapPrint.php?'.session_name().'='.session_id().'&L='.$langID.'"
target="_blank">   <img border="0" src="helpers/printer.gif"></a></p>

<iframe frameborder="0" src="vorlageProjektPHP.php?'.session_name().'='.session_id().'&L='.$langID.'"
WIDTH="100%" HEIGHT="600">
  <p>'.$lang['noIframe'].' <br/>
  <a href ="VorlageProjektPHP.php?'.session_name().'='.session_id().'&L='.$langID.'" target="_blank">
'.$lang['mapWin'].'</a></p>
</iframe>

<table><tr>
<td><a href="http://www.virtours.ch" target="_blank">
<img src="MailClass/Logo.jpg" width="100px" border="0"></a></td>
<td><p class="footer" align="right">'.$lang['foot1'].'<br/>
<a href ="diagramm.php?'.session_name().'='.session_id().'" target="_blank">
'.$lang['foot2'].'</a><br>
'.$lang['foot3'].'
<a href="http://www.adobe.com/svg/viewer/install/auto/" target="_blank">
'.$lang['foot4'].'</a><br/>
'.$lang['foot5'].'
<a href ="resDataPath.php?'.session_name().'='.session_id().'" target="_self">
'.$lang['foot6'].'</a>
</p></td></tr></table>


</body>
</html>

');

?>
```

C.9.6 mainMap.php

Here we echo the full script generating the map output with navigation panel, different viewports and layer management console. This file is given as example script for all map generating files (map for Safari browser, print map, aerial image map etc.).

```
<?php
header("Content-Type: image/svg+xml");

//Session load
session_start();

//Constants initialization
include("constants.php");
$cDB = $LOCAL_CONS["DB"];
$cColors = $LOCAL_CONS["Colors"];
```

```
    $cMAP = $LOCAL_CONS["MAP"];
12
    //Language selection
    include("locallang.php");
    $langID = $_REQUEST['L'];
16  if ($langID == 1){
    $langcode = "en";}
    else if ($langID == 0){
    $langcode = "default";}
20  else {$langcode = "default";}
    $lang = $LOCAL_LANG[$langcode];

    // DATABASE:---------------------------------------------
24  define ('cStrHOSTDB',$cDB['cStrHOSTDB']);
    define ('cStrUSERDB',$cDB['cStrUSERDB']);
    define ('cStrPASSWORDDB',$cDB['cStrPASSWORDDB']);
    define ('cStrDB',$cDB['cStrDB']);
28  define ('cStrTABLETopo',$cDB['cStrTABLETopo']);
    define ('cStrTABLENodes',$cDB['cStrTABLENodes']);
    define ('cStrTABLEEdges',$cDB['cStrTABLEEdges']);
    define ('cStrTABLEUserQ',$cDB['cStrTABLEUserQ']);
32  // ------------------------------------------------------

    $db=mysql_connect(cStrHOSTDB, cStrUSERDB, cStrPASSWORDDB);
    mysql_select_db(cStrDB);
36
    // Get the viewport focussed on the calculated Path, if there is one.
    if($_SESSION['totalPath'] != ''){
    //Get coordinates from start and end
40  $PathArray = explode(',',$_SESSION['totalPath']);
    $qLoc = "SELECT n_x, n_y
    FROM ".cStrTABLENodes."
    WHERE ".$_SESSION['FromID']." = n_id
44  OR ".$_SESSION['ToID']." = n_id";
    $res = mysql_query($qLoc);
    $n1 = mysql_fetch_row($res);
    $n2 = mysql_fetch_row($res);
48  $offset = $cMAP['offsetFocus'];  //Widening in meters
    $xDia = abs($n1[0]-$n2[0]);
    $yDia = abs($n1[1]-$n2[1]);
    if($yDia*$cMAP['xDia']/$cMAP['yDia'] > $xDia){
52  $yDia = $yDia+$cMAP['offsetFocus'];
    $xDia = $yDia*($cMAP['xDia']/$cMAP['yDia']);
    if($n1[1]<$n2[1]){$cornery = $n2[1]*(-1)-($cMAP['offsetFocus']/2);}
    else{$cornery = $n1[1]*(-1)-($cMAP['offsetFocus']/2);}
56  $cornerx = (($n1[0]+$n2[0])/2)-($xDia/2);
    }
    else{
    $xDia = $xDia + $cMAP['offsetFocus'];
60  $yDia = $xDia/($cMAP['xDia']/$cMAP['yDia']);
    if($n1[0]<$n2[0]){$cornerx = $n1[0]-($cMAP['offsetFocus']/2);}
    else{$cornerx = $n2[0]-($cMAP['offsetFocus']/2);}
    $cornery = ((($n1[1]+$n2[1])/2)+($yDia/2))*(-1);
64  }
    }
    else{
    $xDia = $cMAP['xDia'];
68  $yDia = $cMAP['yDia'];
    $cornerx = $cMAP['xC'];
    $cornery = $cMAP['yC'];
    } //endElse
72
    //No transit over the Map borders
    if($cornerx > $cMAP['xC']+$cMAP['xDia']-$xDia){
    $cornerx = $cMAP['xC']+$cMAP['xDia']-$xDia;
76  }
    else if($cornerx < $cMAP['xC']){
    $cornerx = $cMAP['xC'];
    }
80  else if($cornery < $cMAP['yC']){
```

```
$cornery = $cMAP['yC'];
}
else if($cornery > $cMAP['yC']+$cMAP['yDia']-$yDia){
$cornery = $cMAP['yC']+$cMAP['yDia']-$yDia;
}

//MAIN MAP
echo("
<svg onmousemove=\"myMouseCoords(evt)\" zoomAndPan=\"disable\"
viewBox=\"0 0 560 480\" width=\"100%\" height=\"100%\"
xmlns=\"http://www.w3.org/2000/svg\" version=\"1.0\" xmlns:xlink=\"http://www.w3.org/1999/xlink\">
<title>".$lang["title"]."</title>
<defs>

<symbol id=\"mySymbArrow\" overflow=\"visible\" fill=\"none\"
stroke=\"".$cColors["symbAndBorder"]."\" stroke-width=\"1\">
  <polyline points=\"-3,-5 3,-5 3,1 5,1 0,5 -5,1 -3,1 -3,-5\"/>
</symbol>

<script>");
include("naviJS.php");
include("loadOfDetails.php");

//The Reset fct to get the focus back on the calculated track
//has to be definied locally
echo("
      function myResetCalc(){
     document.getElementById('myMap').setAttributeNS(null,'viewBox',
     '".$cornerx." ".$cornery." ".$xDia." ".$yDia."');
     mySetOverviewRect([".$cornerx.", ".$cornery.", ".$xDia.", ".$yDia."]);
       }
       ");

   echo ("</script>
  </defs>


<rect id=\"backGroundDelimiter\" x=\"0\" y=\"0\" width=\"562\"
height=\"485\" fill=\"".$cColors["bgProject"]."\"/>

  <rect id=\"mapAreaDelimiter\" x=\"171\" y=\"4\" width=\"384\"
  height=\"473\" fill=\"".$cColors["symbAndBorder"]."\"/>

<svg id=\"myMap\" viewBox=\"".$cornerx." ".$cornery." ".$xDia." ".$yDia."\"
x=\"172\" y=\"5\" width=\"382\" height=\"471\">
<rect id=\"mapBG\" x=\"".$cMAP['xC']."\" y=\"".$cMAP['yC']."\"
width=\"".$cMAP['xDia']."\" height=\"".$cMAP['yDia']."\" fill=\"".$cColors["bgMap"]."\"/>

<image id=\"myTopo\" xlink:href=\"Topo5.jpg\" x=\"725000\" y=\"-267500\"
width=\"".$cMAP['xDia']."\" height=\"".$cMAP['yDia']."\"/>
<g id=\"myPassiveElements\">");

//Layers to copy by user input into this viewport
echo("
<g id='LayerGroup1'>
<g id='LayerEnvpoly'/>
<g id='useEnvpoly'> </g>
<g id='LayerEnvcomm'/>
<g id='LayerEnvvec'/>
<g id='LayerEdges'/>
<g id='LayerNodesSignpost'/>
<g id='LayerNodesSecondary'/>
<g id='LayerNodeNames'/>
<g id='LayerRes'/>
<g id='LayerNodeCommunities'/>
</g>");

//MAP GRID
echo("<g id=\"myMapFrame\">
<path id=\"myGrid10km\" stroke=\"".$cColors["Grid"]."\" stroke-width=\"5\"
```

```
    fill=\"none\" d=\"M730000 ".$cMAP['yC']." v ".$cMAP['yDia']." M740000 ".$cMAP['yC']."
152 v ".$cMAP['yDia']." M750000 ".$cMAP['yC']." v ".$cMAP['yDia']." M760000 ".$cMAP['yC']."
    v ".$cMAP['yDia']." M".$cMAP['xC']." -220000 h ".$cMAP['xDia']." M".$cMAP['xC']." -230000
    h ".$cMAP['xDia']." M".$cMAP['xC']." -240000 h ".$cMAP['xDia']." M".$cMAP['xC']." -250000
    h ".$cMAP['xDia']." M".$cMAP['xC']." -260000 h ".$cMAP['xDia']."\"/>
156 <path id=\"myGrid5km\" stroke=\"".$cColors["Grid"]."\" stroke-width=\"3\" fill=\"none\"
    d=\"M735000 ".$cMAP['yC']." v ".$cMAP['yDia']." M745000 ".$cMAP['yC']."
    v ".$cMAP['yDia']." M755000 ".$cMAP['yC']." v ".$cMAP['yDia']." M765000 ".$cMAP['yC']."
    v ".$cMAP['yDia']." M".$cMAP['xC']." -215000 h ".$cMAP['xDia']." M".$cMAP['xC']." -225000
160 h ".$cMAP['xDia']." M".$cMAP['xC']." -235000 h ".$cMAP['xDia']." M".$cMAP['xC']." -245000
    h ".$cMAP['xDia']." M".$cMAP['xC']." -255000 h ".$cMAP['xDia']." M".$cMAP['xC']." -265000
    h ".$cMAP['xDia']."\"/>

164 <g id=\"myBorder\">
    <rect id=\"myBorderOuter\" stroke=\"".$cColors["mapBorderOuter"]."\" stroke-width=\"900\"
    fill=\"none\" x=\"725000\" y=\"-268000\" width=\"".$cMAP['xDia']."\" height=\"55466\"/>
    <rect id=\"myBorderInner\" stroke=\"".$cColors["mapBorderInner"]."\" stroke-width=\"800\"
168 fill=\"none\" x=\"725000\" y=\"-268000\" width=\"".$cMAP['xDia']."\" height=\"55466\"/>
    <g id=\"myGridCoords\" fill=\"".$cColors["Grid"]."\" font-size=\"400\" text-anchor=\"middle\">
    <text id=\"myGridCoords1\" x=\"740000\" y=\"-267670\">740’000</text>
    <text id=\"myGridCoords2\" x=\"730000\" y=\"-267670\">730’000</text>
172 <text id=\"myGridCoords3\" x=\"750000\" y=\"-267670\">750’000</text>
    <text id=\"myGridCoords4\" x=\"760000\" y=\"-267670\">760’000</text>
    <text id=\"myGridCoords5\" x=\"725330\" y=\"-220000\"
    transform=\"rotate(-90 725330 -220000)\">220’000</text>
176 <text id=\"myGridCoords6\" x=\"725330\" y=\"-230000\"
    transform=\"rotate(-90 725330 -230000)\">230’000</text>
    <text id=\"myGridCoords7\" x=\"725330\" y=\"-240000\"
    transform=\"rotate(-90 725330 -240000)\">240’000</text>
180 <text id=\"myGridCoords8\" x=\"725330\" y=\"-250000\"
    transform=\"rotate(-90 725330 -250000)\">250’000</text>
    <text id=\"myGridCoords9\" x=\"725330\" y=\"-260000\"
    transform=\"rotate(-90 725330 -260000)\">260’000</text>
184 </g>
    </g>
    </g>
    ");
188

    echo ("

192 <g id=\"myMapElementsOverTheGrid\">
    <g id=\"myScaleBar\">
    <path id=\"strecke\" stroke=\"black\" stroke-width=\"100\" fill=\"none\"
    d=\"M727000 -214800 v 400 h 5000 v -400\"/>
196 <text id=\"m_text\" fill=\"black\" font-size=\"1000\" font-weight=\"bold\"
    x=\"729500\" y=\"-213300\" text-anchor=\"middle\">5000 m</text>
    </g>
    </g>
200 </g>
    </svg>

    <!-- NAVIGATION PANEL -->
204 <rect id=\"infoPanelDelimiter\" x=\"20\" y=\"5\" width=\"132\" height=\"471\"
    fill=\"".$cColors["bgInfoPanel"]."\"/>
    <g id=\"myNavig\" transform=\"translate(30,9) scale(1.6)\" stroke=\"".$cColors["symbAndBorder"]."\"
    stroke-width=\"1\" onmouseover=\"evt.target.setAttributeNS(null,’fill’,’".$cColors["bgInfoPanel"]."’)\"
208 cursor=\"pointer\" onmouseout=\"evt.target.setAttributeNS(null,’fill’,’".$cColors["navMouseout"]."’)\">
    <g shape-rendering=\"crispEdges\" fill=\"".$cColors["navMouseout"]."\">
    <rect id=\"myPanNW\" x=\"0\" y=\"0\" width=\"20\" height=\"20\" onclick=\"myPan(-1,-1)\"/>
    <rect id=\"myPanNN\" x=\"25\" y=\"0\" width=\"20\" height=\"20\" onclick=\"myPan(0,-1)\"/>
212 <rect id=\"myPanNE\" x=\"50\" y=\"0\" width=\"20\" height=\"20\" onclick=\"myPan(1,-1)\"/>
    <rect id=\"myPanWW\" x=\"0\" y=\"25\" width=\"20\" height=\"20\" onclick=\"myPan(-1,0)\"/>
    <rect id=\"myResetAll\" x=\"25\" y=\"25\" width=\"20\" height=\"20\" onclick=\"myResetCalc()\"/>
    <rect id=\"myPanEE\" x=\"50\" y=\"25\" width=\"20\" height=\"20\" onclick=\"myPan(1,0)\"/>
216 <rect id=\"myPanSW\" x=\"0\" y=\"50\" width=\"20\" height=\"20\" onclick=\"myPan(-1,1)\"/>
    <rect id=\"myPanSS\" x=\"25\" y=\"50\" width=\"20\" height=\"20\" onclick=\"myPan(0,1)\"/>
    <rect id=\"myPanSE\" x=\"50\" y=\"50\" width=\"20\" height=\"20\" onclick=\"myPan(1,1)\"/>
    </g>
220 <circle cx=\"10\" cy=\"84\" r=\"10\" fill=\"white\" onclick=\"myZoom(1)\"/>
```

```
<circle cx=\"35\" cy=\"84\" r=\"10\" fill=\"white\" onclick=\"myZoom(-1)\"/>
<circle cx=\"60\" cy=\"84\" r=\"10\" fill=\"white\" onclick=\"myReset()\"/>
<g id=\"myNavigSymbols\" pointer-events=\"none\">
<use xlink:href=\"#mySymbArrow\" transform=\"translate(10,10),rotate(135)\" />
<use xlink:href=\"#mySymbArrow\" transform=\"translate(35,10),rotate(180)\" />
<use xlink:href=\"#mySymbArrow\" transform=\"translate(60,10),rotate(-135)\" />
<use xlink:href=\"#mySymbArrow\" transform=\"translate(10,35),rotate(90)\" />
<circle cx=\"35\" cy=\"35\" r=\"7.5\" fill=\"none\" stroke=\"".$cColors["symbAndBorder"]."\"
stroke-width=\"1\"/>
<circle cx=\"35\" cy=\"35\" r=\"5\" fill=\"".$cColors["symbAndBorder"]."\"
stroke=\"".$cColors["symbAndBorder"]."\"/>
<use xlink:href=\"#mySymbArrow\" transform=\"translate(60,35),rotate(-90)\" />
<use xlink:href=\"#mySymbArrow\" transform=\"translate(10,60),rotate(45)\" />
<use xlink:href=\"#mySymbArrow\" transform=\"translate(35,60)\" />
<use xlink:href=\"#mySymbArrow\" transform=\"translate(60,60),rotate(-45)\" />
<text x=\"10\" y=\"94\" fill=\"".$cColors["symbAndBorder"]."\"
text-anchor=\"middle\" font-size=\"30\">+</text>
<text x=\"35\" y=\"94\" fill=\"".$cColors["symbAndBorder"]."\"
text-anchor=\"middle\" font-size=\"40\">-</text>
<text x=\"60\" y=\"90\" fill=\"".$cColors["symbAndBorder"]."\"
text-anchor=\"middle\" font-size=\"16\">R</text>
</g>
</g>

<g id=\"myCoordPanel\" font-size=\"9\" transform=\"translate(420,-4)\">
     <text x=\"33\" y=\"460\" id=\"myCursorMapX\">".$lang["xcoor"]."</text>
     <text x=\"33\" y=\"472\" id=\"myCursorMapY\">".$lang["ycoor"]."</text>
</g>");

//-----------------------------------------------------------
//Layer Panels and Names, References to functions
echo("
<g id=\"myLayers\" transform=\"translate(30,359)\" cursor=\"pointer\">
<g onmouseover=\"evt.target.setAttributeNS(null,'fill','".$cColors['fgInfoPanel']."')\"
onmouseout=\"evt.target.setAttributeNS(null,'fill','".$cColors['navMouseout']."')\"
shape-rendering=\"crispEdges\" fill=\"".$cColors['navMouseout']."\" stroke-width=\"1\"
stroke=\"".$cColors['symbAndBorder']."\">
   <g fill=\"".$cColors['symbAndBorder']."\" font-size=\"11\">
    <text id=\"Layer1\" x=\"2\" y=\"15\" >x</text>
    <text id=\"Layer2\" x=\"2\" y=\"30\">x</text>
    <text id=\"Layer3\" x=\"2\" y=\"45\">x</text>
    <text id=\"Layer4\" x=\"2\" y=\"60\">x</text>
    <text id=\"Layer5\" x=\"2\" y=\"0\">x</text>
    <text id=\"Layer6\" x=\"2\" y=\"90\">x</text>
    <text id=\"Layer7\" x=\"2\" y=\"75\">x</text>
    <text id=\"Layer8\" x=\"2\" y=\"105\">x</text>
   </g>
   <g fill-opacity=\"1\">
    <rect id=\"myLayer1\" x=\"0\" y=\"7\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerEdges.php', 'LayerEdges', 'UTF-8', 'myLayer1')\"/>
    <rect id=\"myLayer2\" x=\"0\" y=\"22\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerNodesSignpost.php', 'LayerNodesSignpost', 'UTF-8', 'myLayer2')\"/>
    <rect id=\"myLayer3\" x=\"0\" y=\"37\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerNodeNames.php', 'LayerNodeNames', 'UTF-8', 'myLayer3')\"/>
<a xlink:href='resSat.php?".session_name()."=".session_id()."&#38;L=".$langID."' target='hike'>
    <rect id=\"myLayer4\" x=\"0\" y=\"52\" width=\"10\" height=\"10\"/></a>
    <rect id=\"myLayer5\" x=\"0\" y=\"-8\" width=\"10\" height=\"10\"
    onclick=\"myLayer('getPathMap.php?".session_name()."=".session_id()."',
    'LayerRes', 'UTF-8', 'myLayer5')\"/>
    <rect id=\"myLayer6\" x=\"0\" y=\"82\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerEnvvec.php', 'LayerEnvvec', 'UTF-8', 'myLayer6')\"/>
    <rect id=\"myLayer7\" x=\"0\" y=\"67\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerEnvpoly.php', 'LayerEnvpoly', 'UTF-8', 'myLayer7')\"/>
    <rect id=\"myLayer8\" x=\"0\" y=\"97\" width=\"10\" height=\"10\"
    onclick=\"myLayer('LayerEnvcomm.php', 'LayerEnvcomm', 'UTF-8', 'myLayer8')\"/>
   </g>
   </g>
   <g fill=\"".$cColors['fontPanel']."\" font-size=\"10\">
<text x=\"14\" y=\"17\" onclick=\"myLayer('LayerEdges.php', 'LayerEdges',
'UTF-8', 'myLayer1')\">".$lang["layerEdges"]."</text>
```

```
    <text x=\"14\" y=\"31\" onclick=\"myLayer('LayerNodesSignpost.php',
292 'LayerNodesSignpost', 'UTF-8', 'myLayer2')\">".$lang["layerSignpost"]."</text>
    <text x=\"14\" y=\"46\" onclick=\"myLayer('LayerNodeNames.php', 'LayerNodeNames',
    'UTF-8', 'myLayer3')\">".$lang["layerNodeNames"]."</text>
    <a xlink:href='resSat.php?".session_name()."=".session_id()."&#38;L=".$langID."' target='hike'>
296 <text x=\"14\" y=\"60\">".$lang["satLabel"]."</text></a>
    <text x=\"14\" y=\"1\"
    onclick=\"myLayer('getPathMap.php?".session_name()."=".session_id()."',
    'LayerRes', 'UTF-8', 'myLayer5')\"
300 onload=\"myLayer('getPathMap.php?".session_name()."=".session_id()."',
    'LayerRes', 'UTF-8', 'myLayer5')\">".$lang["layerRes"]."</text>
    <text x=\"14\" y=\"90\" onclick=\"myLayer('LayerEnvvec.php',
    'LayerEnvvec', 'UTF-8', 'myLayer6')\">".$lang["layerTraffic"]."</text>
304 <text x=\"14\" y=\"75\"
    onclick=\"myLayer('LayerEnvpoly.php','LayerEnvpoly', 'UTF-8', 'myLayer7')\"
    onload=\"myLayer('LayerEnvpoly.php','LayerEnvpoly', 'UTF-8', 'myLayer7')\"
    >".$lang["layerEnv"]."</text>
308 <text x=\"14\" y=\"105\" onclick=\"myLayer('LayerEnvcomm.php',
    'LayerEnvcomm', 'UTF-8', 'myLayer8')\">".$lang["layerComm"]."</text>
    <text x=\"-4\" y=\"-174\">".$lang["dragMap"]."</text>
    <text x=\"-4\" y=\"-164\" font-size='8'>".$lang["dragMapSubtitle"]."</text>
312 </g>
       </g>


316   <!-- START OVERVIEW MAP -->
    <rect  x=\"25\" y=\"198\" width=\"120\" height=\"148\"/>

    <!-- THE SAME  VIEWPORT DIMENSIONS AS THE MAIN MAP BUT SMALLER SIZE -->
320 <svg id=\"myOverviewMap\" onmouseup=\"myDrag(evt)\" onmousemove=\"myDrag(evt)\"
    viewBox=\"".$cMAP['xC']." ".$cMAP['yC']." ".$cMAP['xDia']." ".$cMAP['yDia']."\"
    x=\"25\" y=\"198\" width=\"120\" height=\"148\">
    <rect id=\"myBackgroundRect\" x=\"".$cMAP['xC']."\" y=\"".$cMAP['yC']."\"
324 width=\"".$cMAP['xDia']."\" height=\"".$cMAP['yDia']."\"
    fill=\"".$cColors["overviewRectBg"]."\" stroke-width=\"0\"/>

    <!--STILL EXISTING LAYERS AND CONTENT ARE COPIED INTO THE OVERVIEW -->
328
    <use xlink:href=\"#myTopo\"/>
    <use xlink:href=\"#myGrid10km\"/>
    <use xlink:href=\"#myGrid5km\"/>
332 <use xlink:href=\"#LayerEnvpoly\"/>
    <use xlink:href=\"#myBorder\"/>

    <!-- RECTANGLE SHOWING THE POSITION OF ACTUAL MAP CUT, SENSITIVE AREA TO DRAG AND DROP THE MAP -->
336 <rect id=\"myOverviewRect\" onmousedown=\"myDrag(evt)\" x=\"".$cornerx."\" y=\"".$cornery."\"
    width=\"".$xDia."\" height=\"".$yDia."\" fill=\"".$cColors["bgProject"]."\" fill-opacity=\".4\"
    stroke=\"".$cColors["symbAndBorder"]."\" stroke-opacity=\"1\" stroke-width=\"200\"/>

340 </svg>
    </svg>
    ");

344
    ?>
```

C.9.7 constants.php

All possible constants concerning database identifiers, colors, layout data etc. are outsourced into this managing file. It is built with the same structure as the file containing all language labels with a multidimensional array guiding the allocation.

```
    <?php
    /**
     * All constants
 4   */

    $LOCAL_CONS = Array (
    'DB' => Array (
 8  'cStrHOSTDB' => 'mysql5.webland.ch',
```

```
    'cStrUSERDB' => 'xxxx',
    'cStrPASSWORDDB' => 'xxxx',
    'cStrDB' => 'xxxx',
    'cStrTABLETopo' => 'topology',
    'cStrTABLENodes' => 'n_nodes',
    'cStrTABLEEdges' => 'baseedges',
    'cStrTABLEUserQ' => 'userqueries',
    'cStrTABLEEnvvec' => 'envvector',
    'cStrTABLEEnvpoly' => 'envpolygon',
    ),
    'Mail' => Array (
    'mailURL' => 'http://www.virtours.ch/t3/index.php?id=38',
    ),
    'MAP' => Array (
    'xC' => '725000',
    'yC' => '-268000',
    'xDia' => '45000',
    'yDia' => '55466',
    'offsetFocus' => '1000', //offset of focus area calc path in m
    'areaChoose' => '6000', //area focussed when looking for entry in m

    ),
    'Diagramm' => Array(
    'strokeColor' => '#FFFFFF',
    'polyFill' => 'green',
    'strokeWidth' => '5'
    ),
    'Colors' => Array (
    'bgInfoPanel' => '#C2DD71',
    'fgInfoPanel' => '#C2DD71',  //Lighter Foreground and mouseover
    'navMouseout' => '#FFFFFF',
    'symbAndBorder' => '#537413', //Symbols and Border of nav Elements
    'overviewRectBg' => '#FFFFFF',
    'bgProject' => '#79AA3D',  //Bg of mainFrame
    'bgMap' => '#FFFFFF',
    'Grid' => '#000000',
    'mapBorderOuter' => '#000000',
    'mapBorderInner' => '#FFFFFF',
    'fontPanel' => '#000000'
    ),
    'ResultLayer' => Array(
    'resPathInner' => 'orange',
    'resPathInnerStroke' => '15',
    'resPathOuter' => '#000000',
    'resPathOuterStroke' => '25',
    'resTextFill' => '#3F4B56',
    'resTextSize' => '50',
    'resTextSizePrint' => '190',
    'translatex' => '20',
    'translatey' => '-40',
    'translatexPrint' => '50',
    'translateyPrint' => '-70',
    'resNodeFill' => '#4CA2EB',
    'resNodeSize' => '25',
    'resNodeStrokeWidth' => '4',
    'resNodeSizePrint' => '80',
    'resNodeStrokeWidthPrint' => '8',
    'resNodeStrokeColor' => '#000000',
    'satFill' => 'black',
    'satSize' => '70',
    'printPathOuter' => '75',
    'printPathInner' => '60',
    ),
    'LayerNodeNames' => Array(
    'fontSize' => '50',
    'translatex' => '20',
    'translatey' => '-40',
    'textAnchor' => 'left',
    'textFill' => '#3F4B56',
    'satFill' => 'black',
```

```
    'satSize' => '70',
80  ),
    'LayerNodesSignpost' => Array(
    'nodeFill' => '#FFFC00',
    'nodeSize' => '20',
84  'nodeStrokeWidth' => '4',
    'nodeStrokeColor' => '#000000',
    ),
    'LayerEdges' => Array(
88  'edgeColor' => 'red',
    'edgeWidth' => '8',
    ),
    'LayerNodesSecondary' => Array(
92  'nodeFill' => 'darkorange',
    'nodeSize' => '16',
    'nodeStrokeWidth' => '3',
    'nodeStrokeColor' => '#000000',
96  ),
    );
    ?>
```