ETH Zürich

Vertiefungsblock Wintersemester 2004/05

Erstellung eines Touristeninformationssystems für die Gemeinde Flüelen UR

Marius Schäuble und Peter Süess

2. Februar 2005

Institut für Kartographie
Prof. Dr. L. Hurni

Vorwort

Diese Arbeit entstand im Rahmen eines Vertiefungsblockes an der ETH Zürich im Wintersemester 2004/05. Angeboten und betreut wurde dieser VTB vom Institut für Kartographie (IKA).

Die gestellte Aufgabe gab uns die Möglichkeit, die bisher erworbenen Kenntnisse im Bereich der Herstellung von Internetseiten mittels XML, SVG und zusätzlichen Skriptsprachen aufzufrischen und zu vertiefen. Bei Fragen oder Anregungen zur vorliegenden Arbeit stehen die Verfasser gerne zur Verfügung.

Allen, die uns in irgendeiner Form bei unserer Arbeit in den vergangenen 14 Wochen unterstützt haben, möchten wir recht herzlich danken.

- Professor Dr. Lorenz Hurni für die Leitung des Vertiefungsblockes
- Tobias Dahinden für die Betreuung und die fachliche Unterstützung
- Claudia Matthys für die Unterstützung im Bereich EDV

Zürich, 2. Februar 2005

Marius Schäuble Peter Süess

Zusammenfassung

In der vorliegenden Arbeit geht es darum, ein Touristeninformationssystem für die Gemeinde Flüelen UR zu erstellen. Dieses System soll es Touristen ermöglichen, sich im Vorfeld eines Besuches dieser Gemeinde über die Möglichkeiten, die in Bereichen Unterkunft, Kulinarisches, Sehenswürdigkeiten, usw. geboten werden, zu informieren. Andererseits sollen Touristen vor Ort mit einem mobilen Gerät (z.B. PDA (Personal Digital Assistant)) Informationen abfragen können, die von Bedeutung oder Interesse sind. Im Mittelpunkt des Informationssystems soll eine geografische Karte stehen, welche die Beziehung zwischen Information und Ort herstellen soll.

Diese Arbeit beschränkt sich nur auf eine Gemeinde, weil die zur Verfügung stehende Zeit die Bearbeitung eines grösseren Gebietes nicht zulässt. Unser selbst gestecktes Ziel ist es dennoch, ein Informationssystem zu entwickeln, das ohne grössere Schwierigkeiten auf weitere Gemeinden erweitert werden kann. Aus diesem Grund ist eine gut durchdachte Datenhaltung von grosser Bedeutung.

Die Umsetzung erfolgt durch Benutzung von sogenannten Markup-Sprachen (Auszeichnungssprachen), wie dies bei der Herstellung von Web-Seiten, die von Browsern interpretiert werden können, üblich ist. Dabei handelt es sich um SVG (Scalable Vector Graphics) und HTML (Hypertext Markup Language), die durch CSS (Cascading Stylesheets) und ECMAScript ergänzt werden. Ausserdem verwenden wir für die Speicherung aller Daten eine MySQL-Datenbank. Diese ist relativ leicht zu bedienen und ermöglicht eine gute Strukturierung der Daten. Der Zugriff auf diese Datenbank erfolgt über die Programmiersprache Perl. Innerhalb eines solchen Perlskripts kann ausserdem SVG- und HTML-Code geschrieben werden, welcher beim Aufruf des Skripts als solcher erkannt wird. Perl kann somit als Bindeglied zwischen Web-Seite und Datenbank verstanden werden.

Das Ergebnis dieses Vertiefungsblocks ist ein Touristeninformationssystem. Eingeteilt in sechs Themen, können Touristen bei der Urlaubsplanung oder auch vor Ort Informationen ihres Interesses abfragen. Unser Ziel einer möglichst leichten Erweiterbarkeit des Systems auf weitere Gemeinden haben wir, so weit es vom Standpunkt der Entwicklung aus zu sagen ist, erreicht. Dies lässt uns mit Zufriedenheit auf die entstandene Arbeit zurückblicken.

Inhaltsverzeichnis

1 Einleitung

1.1 Aufgabenstellung

Wer kennt ihn nicht, den Wunsch sich an irgendeinem Ort der Schweiz oder der Welt zu befinden und gleichzeitig nützliche Informationen zur aktuellen Lokalität zu erhalten? Die immer stärker aufkommenden mobilen Kleincomputer (PDAs, Handys, etc.) machen dies möglich, indem sie z.B. über einen Server positionsbasierte Dienste anbieten können. Aus diesem Grund wollen wir in diesem Vertiefungsblock einen Prototyp für ein Touristeninformationssystem erstellen, welches sowohl vor Ort als auch als Planungswerkzeug im Vorfeld eines Besuchs verwendet werden kann.

Um den Rahmen dieser Arbeit nicht zu sprengen, dient als Testregion die Gemeinde Flüelen (UR). Diese ist besonders gut geeignet, da dort eine Vielzahl unterschiedlicher Lokalitäten, wie Bahnhof, Schiff, Hotels, Kunsträume, Wanderwege, etc. auf engem Raum vorkommen. Flüelen liegt am Eingang zum Urner Reusstal. Die Gemeinde ist von Süden her der Verzweigungspunkt der Gotthardroute in Richtung Zürich oder Luzern. Die herrliche Lage am Urnersee lädt zum Wandern oder zu einer Pause auf der Durchreise ein. Ausserdem ist Flüelen ein optimaler Ausgangspunkt, um die Urner Bergwelt zu erkunden.

Zur Realisierung dieses Informationssystems sollen Daten aus Prospekten, vor Ort erhobene Daten, Daten aus Webseiten und kartografische Daten zusammengetragen werden. Diese sind schlussendlich in ansprechender, leicht interpretierbarer und gut benutzbarer Form darzustellen.

1.2 Ziel - Vision

Ziel dieser Semesterarbeit ist es, einen Grundstein zu legen für eine eventuelle Weiterführung dieser Arbeit im Sinne einer Erweiterung auf Tourismusgemeinden der gesamten Schweiz. Das soll heissen, dass von unserer Seite aus Wert darauf gelegt wird, bei der Lösung der einzelnen Probleme stets dieser Vision nachzugehen. Als Ergebnis streben wir aus diesem Grund ein Informationssystem an, welches nicht haargenau auf die Gemeinde Flüelen zugeschnitten ist. Eine Erweiterung des Systems soll mit deutlich reduziertem Aufwand möglich sein.

1.3 Positionierung der Arbeit

Die Entwicklung von Touristeninformationssystemen steckt noch in den Kinderschuhen. Es gibt bereits einige theoretische Abhandlungen zu diesem Thema, praktische Anwendungen sind jedoch kaum vorhanden. Wenn ja, sind diese noch nicht sehr ausgereift, z.B. [14]. Dies ist zu erkennen, wenn man mit der Suchmaschine 'google' nach 'Touristeninformationssystem' sucht.

Ein ähnliches Bild zeigt sich in [1]. Darin sind viele theoretische Abhandlungen zum Thema 'Location Based Services & Telecartography' zu finden. Tatsächliche Umsetzungen sind nur spärlich vorhanden.

Eine gute Übersicht geben Markus Uhlirz und Julius Kindler in ihrer Arbeit 'A View on Location-Based Services - 1000 Days After the Hype'. Darin bestätigen sie die noch geringe Verbreitung von LBS aufgrund fehlender Akzeptanz auf dem Markt und zu hoher Kosten. Anzumerken ist jedoch, dass Technologie, Software und die notwendigen Schnittstellen verfügbar sind und zum Gebrauch bereit stehen. Auch die Möglichkeiten zur Positionierung sind mit genügend Genauigkeits- und Auflösungsvermögen vorhanden. Ihrer Ansicht nach werden sich LBS innerhalb der nächsten Jahre etablieren [1].

Die Verwendung von SVG zusammen mit einer Datenbank ist nichts Neues. Am Institut für Kartographie gibt es bereits einige Arbeiten mit diesem Konzept:

- 'Interaktive, SVG-basierte Flugplatzhinderniskarte', Diplomarbeit Sandra Demarmels 2004 [12]
- 'Interaktive Webkarte vom Katasterplan Morcote', Vertiefungsblock Christian Höhener 2003/04 [12]

Unter diesem Aspekt ist unsere Arbeit als Erweiterung dieser Sammlung zu sehen.

2 Grundkonzept

2.1 Funktionalität

Dem Benutzer wird eine selbsterklärende Oberfläche präsentiert, welche das Zoomen und Pannen per Mausklick erlaubt und die gewünschten Informationen einfach abrufen lässt. In einer Menüleiste sind die Informationen, welche für einen Touristen von Interesse sind, in verschiedene Hauptthemen unterteilt:

- Unterkunft
- Kulinarisches
- Einkaufen
- Verkehrsmittel
- Aktivitäten
- Sehenswürdigkeiten

Durch die Auswahl eines Themas wechselt die Darstellung auf der Grundkarte und der Anwender kann sich weiter vertiefen, bis er die gewünschten Informationen erhält. Dies geschieht durch direktes Anklicken eines Symbols in der Karte oder einer Liste, auf welcher alle Angebote zu einem jeweiligen Thema aufgeführt sind. Jedes Angebot (z.B. Informationen zu 'Hotel Weisses Kreuz') wird in einem separaten Fenster vorgestellt. Die Angaben dazu werden direkt aus einer Datenbank abgerufen.

2.2 Darstellung

Die Navigation wird von einer Menüleiste oben im Bildschirm gesteuert. Via Hauptthemen können verschiedene Untermenüs angeklickt werden. Darauf wird die Seite neu dargestellt. Einerseits ändert die Informationsliste links neben der Karte, wo die entsprechenden Angebote zum Untermenü aufgeführt werden. Andererseits werden diese Angebote auch auf der Karte mit einem passenden Symbol dargestellt. Der Text in der Liste und das zugehörige Symbol werden ausserdem animiert, d.h. beim Überfahren mit der Maus werden beide hervorgehoben, was eine visuelle Zuordnung ermöglicht. Ohne dies könnte kaum ein Zusammenhang zwischen den Symbolen und den Namen hergestellt werden. Durch Anklicken können schliesslich die Detailinformationen abgerufen werden.

Die Informationsliste auf der linken Seite enthält ausserdem Buttons, welche das Zoomen und Pannen der Karte ermöglichen. Bei der Anzeige des Themas 'Verkehrsmittel' soll ausserdem eine Übersichtskarte mit den wichtigsten Verkehrsverbindungen angezeigt werden.

2.3 Umsetzung

Da die mobilen Computer normalerweise einen Browser haben, kann die Darstellung mit einer browserverständlichen Sprache geschrieben werden. Wir könnten auch selber ein Programm schreiben, das die PDAs ausführen können, jedoch wäre der Aufwand grösser. Wir entscheiden uns für SVG, da sich diese Sprache gut für grafische Darstellungen eignet. Ebenfalls können Animationen direkt programmiert werden.

Um die Informationen zu den einzelnen Themen abzurufen, wird der Seite eine Datenbank hinterlegt (abfragbar mit SQL). Teile der Verbindung zwischen der Darstellung und der Datenbank werden mit einem Perl-Programm realisiert. Mit der Auswahl eines Themas werden die entsprechenden Objekte in der Datenbank selektiert und mit Hilfe von Koordinateninformationen in der Karte dargestellt. So sollte die Aktualisierung unabhängig vom SVG-Dokument möglich sein.

Die verwendete Karte muss die Gemeinde Flüelen in genügend hoher Auflösung darstellen. Die wichtigsten Elemente, wie Strassen, Gewässer, Gebäude, müssen vorhanden sein. Für die Karte beabsichtigen wir nicht allzu grossen Aufwand zu betreiben.

3 Grundlagedaten

3.1 Grundlagenkarte

Als Grundlagenkarte verwenden wir einen Ausschnitt der LK 1:25'000. Da die digital vorhandene Karte aufgrund einer zu geringen Scan-Auflösung nicht den Anforderungen entspricht, wird das analog vorhandene Kartenblatt der LK 1:25'000 mit einer Auflösung von 2400 dpi eingescannt. In Photoshop wird die Karte auf den notwendigen Bereich zugeschnitten und ausserdem noch komprimiert, damit die Ladezeiten später nicht zu lange sind. Sie liegt im JPEG-Format vor und hat eine Grösse von 3.3MB. Die Verwendung der vorhandenen digitalen Vektorkarte (Vector 25 der swisstopo) als Hintergrundkarte haben wir sehr schnell ausgeschlossen, da der Aufwand für die notwendige Symbolisierung zu viel Zeit in Anspruch genommen hätte.

3.2 Symbolisierung

Unser Touristeninformationssystem soll Besuchern möglichst viele, touristisch interessante Informationen zur Verfügung stellen. Diese werden unter anderem durch Symbole auf der Karte dargestellt. Aus diesem Grund müssen wir für die im Grobkonzept angesprochenen Themen einzelne Symbole schaffen. Dies machen wir auf zwei Arten. Einerseits suchen und finden wir im Internet Symbole (Bilder im .gif-Format), die bereits bestehen. Andererseits kreieren wir weitere Symbole. Schlussendlich erstellen wir einen Ordner, in dem alle zu den Themen passenden Symbole enthalten sind.

3.3 Datenbank - Inhalt

Ein sehr wichtiger Teil der Grundlagedaten dieser Arbeit ist die Sammlung der eigentlichen Informationen zur Gemeinde Flüelen. Um diese Daten sinnvoll und übersichtlich strukturiert speichern und verwalten zu können, entscheiden wir uns für eine Datenbank. Die Wahl fällt auf MySQL, wobei auch andere Datenbanktypen möglich gewesen wären. Darin sind folgende Themen mit Unterthemen enthalten:

- Unterkunft: Camping, Hotel, Privatunterkunft
- Kulinarisches: Bistro, Pizzeria, Pub, Restaurant
- Verkehrsmittel: Bergbahn, Bus, Postauto, SBB, Schiff
- Einkaufen: Elektronik, Kiosk, Lebensmittel, Post, Sonstiges, Souvenirs
- Sehenswürdigkeiten: Denkmal, Gebäude, Kunst
- Aktivitäten: Baden, Surfen, Wandern, Wasserski

Diese Einteilung ist ein Vorschlag von uns und kann jederzeit in der Datenbank geändert werden.

Zur weiteren inhaltlichen Strukturierung verwenden wir Attribute, die genauere Informationen zu den einzelnen Themen enthalten. Zum besseren Verständnis ein kleines Beispiel:

- Thema: Unterkunft
  - Attribute: Id (Bsp.: Hotel), Name (Bsp.: Hirschen), Strasse, PLZ, Ort, Besitzer, Tel., Fax, www, email, Öffnungszeiten, Ruhetag, Bettenzahl, x, y (für Platzierung des Symbols)

Datenquellen Die Hauptquelle unserer Daten ist die Homepage der Gemeinde Flüelen mit der Adresse 'www.fluelen.ch'. Hier finden wir im Prinzip alles, was für unser Touristeninformationssystem von Bedeutung ist. Weitere Webseiten, wie z.B. die der SBB, und die Ortskenntnisse von Peter Süess (Wohnort Altdorf UR) dienen uns als zusätzliche Quellen.

Damit die Ortskenntnisse der beiden Vertiefungsblockteilnehmer etwa auf den gleichen Stand kommen und um letzte örtliche Unklarheiten, wie z.B. Standpunkte von Gebäuden und Objekten, beseitigen zu können, wird gegen Ende des Semesters eine Exkursion in die Gemeinde Flüelen durchgeführt.

4 Kurzeinführung in die verwendeten Werkzeuge

4.1 HTML

HTML bedeutet Hypertext Markup Language und ist eine Auszeichnungssprache für Internetseiten, die vom World Wide Web Consortium (W3C) standardisiert wurde. Damit wird nur das Layout der Webseite bestimmt, *nicht* der Inhalt. HTML ist eine Sprache zur Strukturierung von Texten, wobei aber auch die Möglichkeit besteht, Grafiken und multimediale Inhalte in Form einer Referenz einzubinden und in den Text zu integrieren.

HTML-Webseiten können in jedem Browser betrachtet werden. Die HTML-Dateien sind Textdateien, deren Inhalt mittels sogenannter Tags strukturiert und formatiert wird. Bis auf wenige Ausnahmen werden alle HTML-Elemente durch ein einleitendes Starttag und ein abschliessendes Endtag markiert. Der Inhalt steht zwischen den Tags.

Vor dem eigentlichen Dokument steht meist eine Dokumenttyp-Definition (DTD). Im Kopf der HTML-Seite werden die Metainformationen angegeben, die in der Regel aus der Angabe der verwendeten Sprache (Bsp.: ISO-8859-1) und dem Titel bestehen.

HTML-Dokumente sind hierarchisch gegliedert, das heisst, sie haben ein Dokument-Tag (<html>), welches wiederum in einen Kopfteil (<head>, für Metainformationen) und einen Rumpfteil (<body>, für den Inhalt) gegliedert ist. Im Rumpfteil können Elemente in einer beliebigen Reihenfolge stehen. Der Browser liest diese Elemente von oben nach unten aus und stellt sie dar. Die ersten Elemente werden also auch zuerst gezeichnet. Zudem können die Elemente ineinander verschachtelt sein.

Ab der HTML-Version 4.x spricht man auch von DHTML (Dynamic Hypertext Markup Language), da hier HTML mit den Sprachen CSS (Cascading Stylesheets, für Layout) und ECMAScript (für Interaktionen) erweitert wurde. [2]

4.2 SVG

SVG bedeutet Scalable Vector Graphics und ist eine Sprache zur Beschreibung und Integration von Vektorgrafiken (aber auch Rastergrafiken und Text) für das Internet. SVG ist ein Standard des World Wide Web Consortiums (W3C). Es basiert auf XML, der eXtensible Markup Language (erweiterte Auszeichnungssprache), ist also ein Dialekt von XML. Das Format ist rein textbasiert, kann somit in jedem Texteditor gelesen und geschrieben werden. Ein weiterer Vorteil ist die geringe Dateigrösse von SVG-Dateien, auch wenn mit vielen Vektorgrafiken gearbeitet wird. Dies ist eine gute Voraussetzung für das Internet.

Ähnlich HTML benutzt SVG Tags. Diese werden aber nicht zur Textauszeichnung, sondern hauptsächlich zur Darstellung von Vektorobjekten benutzt. Mit SVG können beliebige zweidimensionale Grafiken gezeichnet werden. Weiterhin ist die Einbindung von Rastergrafiken, CSS, Sound, Interaktionen (z.B. mittels ECMAScript), Animationen und Filtern möglich. Wie der Name schon sagt, können Elemente in SVG-Anwendungen verschoben, vergrössert und verkleinert werden, ohne dass ein Informationsverlust eintritt.

SVG-Grafiken können in Standardbrowsern (IE, Netscape, Opera) nicht ohne weiteres betrachtet werden. Es ist (wie bei jedem Vektorgrafikformat) ein Plugin erforderlich, um die SVG-Datei betrachten zu können.

SVG-Dokumente sind hierarchisch gegliedert, das heisst, sie haben ein Dokument-Starttag (<svg>) und ein Dokument-Endtag (</svg>). Zwischen diesen steht der Inhalt der SVG-Datei. Der Browser liest die Inhaltselemente von oben nach unten aus und stellt sie dar. Die

ersten Elemente werden zuerst gezeichnet, die weiteren darüber.

Da SVG eine Erweiterung der Sprache XML ist, muss das vor dem <svg>-Starttag entsprechend gekennzeichnet werden. Mit 'encoding' wird die Codierung angegeben, in der das SVG-Dokument verfasst ist. ISO-8859-1 kennzeichnet die normalen westeuropäischen Sprachen und unterstützt auch deutsche Umlaute wie ä, ö und ü. Weitere Zeichensätze, die ebenfalls verwendet werden können, sind z.B. UTF-8 und UTF-16. Diese enthalten mehr Zeichen als ISO-8859-1. Es ist anzuraten, vor dem Starttag ausserdem eine Dokumenttyp-Definition (DTD) zu stellen. Diese überprüft, ob es ungültige Elemente in der SVG-Datei gibt. [3] [7]

4.3 CSS

Cascading Stylesheets ist eine Erweiterung von HTML, die zur Definition der Formateigenschaften von HTML-Elementen dient. CSS ist ein Standard des W3C. Damit kann man z.B. die Schriftgrösse und Schriftart der Überschrift, die Ausrichtung von Absätzen und das Layout von Listen und Tabellen festlegen. Inhalte können pixelgenau auf der Webseite platziert werden. Die neuesten Browser unterstützen CSS der Version 2.

CSS kann in eine HTML- bzw. SVG-Datei eingebettet werden (internes CSS). Damit gilt es jedoch nur für diese eine Datei. Soll ein Layout zentral für mehrere Webseiten definiert werden, wird ein externes Stylesheet eingebunden. Der Inhalt von externem und internem CSS unterscheidet sich nicht, nur die Art der Einbindung ändert sich. [2]

4.4 ECMAScript

ECMAScript ist die von der European Computer Manufacturers Association standardisierte Version von JavaScript. Es dient der Interaktion der Nutzer mit der Internetseite. Es besteht letztendlich aus einer Reihe von Befehlen, die der ECMAScript-Interpreter des Webbrowsers in Maschinencode umsetzt, der auf dem Rechner des Anwenders ausführbar ist.

ECMAScript kann, wie Cascading Stylesheets, in die jeweilige HTML- bzw. SVG-Datei eingebettet werden (internes ECMAScript). Das bedeutet, dass der Code nur für die jeweilige Datei gilt. Es kann aber auch als externes ECMAScript eingebunden werden. Dies sollte man dann tun, wenn man ECMAScript-Code in mehreren HTML- bzw. SVG-Dateien nutzen will, z.B. für die Navigation mit Buttons. [4] [11]

4.5 Perl

Perl steht für Practical Extraction and Report Language. Ziel des Autors Larry Wall bei der Erstellung von Perl war es, zwei wichtige Programmfunktionalitäten zu verbinden. Einerseits enthält Perl die wichtigsten Programmierbefehle wie Schleifen, Verzweigungen, etc. und bietet andererseits auch die Möglichkeit, Such- und Ersetzungsoperationen durchzuführen. Perl wird oft als eine Synthese der Programmiersprache C und den UNIX-Funktionen 'sed' und 'awk' angesehen. Perl-Skripte werden zur Laufzeit über einen Interpreter ausgeführt. Perl-Interpreter gibt es mittlerweile für alle gängigen Betriebssysteme, sodass Perl nahezu plattformunabhängig ist. [6] [7]

4.5.1 DBI-Modul

Das DBI-Modul setzt die Schnittstelle zwischen Perl-Skript und Datenbank um. DBI definiert einen Satz Funktionen, Variablen und Vereinbarungen, die eine beständige Datenbankschnittstelle darstellen, unabhängig von der tatsächlich verwendeten Datenbank.

Es ist wichtig, sich daran zu erinnern, dass DBI nur eine Schnittstelle ist. Eine dünne Schicht 'Klebstoff' zwischen der Applikation und einem oder mehreren Datenbank-Treibern. Die Treiber erledigen die eigentliche Arbeit. DBI stellt die Standardschnittstelle zur Verfügung und steuert die entsprechenden Treiber an.

4.6 CGI

Die CGI-Schnittstelle definiert die Art und Weise, wie an einen Server abfragen gestellt werden und wie darauf geantwortet werden soll. Im Unterschied zu ECMAScript wird Perl vor dem Übertragen auf der Serverseite ausgeführt, während ECMAScripts normalerweise mit der Seite mitgeliefert werden und lokal auf dem Rechner des Clients funktionieren. Deshalb sind Perl-Skripts in Verbindung mit dem Web nur ausführbar, wenn eine HTTP-basierte Kommunikation zwischen Web-Browser und Web-Server stattfindet.

Ein CGI-Skript kann auch Datenbanken abfragen. Dabei muss die Datenbank nicht zwingend auf demselben Rechner installiert sein. Bei entsprechendem Aufruf kann ein CGI-Programm gespeicherte Daten auslesen und daraus einfachen Text generieren. Üblicherweise ist dies ein Quell-Code, welcher beim Client durch den Browser dargestellt wird. Dieser 'dynamisch' erzeugte Quell-Code enthält zum Beispiel den aktuellen Zugriffszählerstand einer Web-Seite, die bisherigen Einträge in einem Gästebuch oder Ergebnisse einer Datenbanksuche. CGI-Programmierung und Perl werden mittlerweile oft in einem Atemzug genannt und als zusammengehörig dargestellt. Es ist nur wichtig zu wissen, dass die Verbindung von CGI und Perl nichts naturgegebenes oder notwendiges ist. CGI ist lediglich eine Norm für eine Programmier-Schnittstelle, die Web-Server-Software unterstützen sollte, und Perl ist eine universell einsetzbare Skriptsprache, die sich allerdings auf Grund ihrer Charakteristik hervorragend für die CGI-Programmierung eignet. Anstatt Perl könnten auch andere Sprachen eingesetzt werden, z.B. PHP. [7]

4.7 MySQL-Datenbank

4.7.1 Über MySQL

Der Datenbankserver MySQL ist die populärste Open-Source-Datenbank der Welt. MySQL wird in über fünf Millionen Installationen für Web-Sites und geschäftskritische Unternehmensanwendungen genutzt. Hierzu zählen führende Unternehmen wie Associated Press, Google, DaimlerChrysler, die HypoVereinsbank und T-Systems.

Auf den meisten Betriebssystemen ist MySQL lauffähig: u.a. Windows, Unix, Linux, Solaris. Das bedeutet, dass es für unser Vorhaben geeignet. Der eigentliche Grund für die Verwendung von MySQL liegt darin, dass der Datenbankserver gratis zur Vrfügung gestellt wird. Ausserem ist MySQL einfach zu bedienen bezüglich Erstellung und Manipulation der Datenbank. [13]

4.7.2 Aufbau einer MySQL-Datenbank

Grundlayout Eine MySQL-Datenbank besteht normalerweise aus einer gewissen Anzahl von Datenbanken. Diese werden von einem Administrator auf einem Server verwaltet. Jeder Benutzer hat dann seine eigene Datenbank, in der er arbeiten kann. Ihm wird von der Administration eine Datenbank zugeteilt, die mit einem Passwort zugänglich ist. Der Aufbau einer gesamthaften Datenbank sieht dann folgendermassen aus.

MySQL-Datenbank	
Datenbank 1	Datenbank 2
Datenbank 3	Datenbank 4

Innerhalb der zugeteilten Datenbank kann der Benutzer dann Aktionen durchführen, welche von den gegebenen Rechten her gestattet sind, z.B Datensätze einfügen, löschen oder verändern.

Erstellen von Tabellen Zuerst müssen die Tabellen, die für die Strukturierung der Daten benötigt werden, erstellt werden. Nützlich ist, dass bestehende Tabellen jederzeit gelöscht oder neue Tabellen erstellt werden können. Erst in den Tabellen können dann die Informationen gespeichert werden. Jede Tabelle hat ihren eigenen Namen, z.B. News. Ausserdem besitzt jede Tabelle Spalten. Man kann sich das wie in einer Excel-Tabellenkalkulation vorstellen, mit dem Unterschied, dass die Spalten in MySQL nicht A, B, C, etc. heissen, sondern bestimmte Namen haben. Eine Tabelle hat nicht unendlich viele Spalten, sondern nur eine vorher festgelegte Anzahl. Die Spalten in einer Tabelle News könnten Inhalt, Titel, Datum und Autor heissen. Dies würde grafisch wie folgt aussehen:

Tabelle News			
Titel	Inhalt	Datum	Autor

In dieser Tabelle können dann die gewünschten Daten gespeichert werden. Ein Eintrag eines Datensatzes könnte so aussehen:

Tabelle News			
Titel	Inhalt	Datum	Autor
Nachricht 1	Das ist eine Nachricht	12.12.03	Meier

Die Anzahl der Datensätze in einer Tabelle ist unbegrenzt. Die einzige Grenze ist der Festplattenspeicher oder gegebenenfalls eine Einschränkung durch den Administrator der MySQL-Datenbank.

Verbindung und Abfragen Um auf die Daten der MySQL-Datenbank zugreifen zu können, muss zuerst eine Verbindung mit dieser Datenbank hergestellt werden. In unserem Fall verwenden wir für alle Datenbankverbindungen Perl. Innerhalb dieses Skripts kann mit einem 'DBI -> connect'-Befehl auf eine bestimmte Datenbank zugegriffen werden. Dazu wird natürlich auch der Name der Datenbank und das Passwort benötigt.

Wenn die Verbindung hergestellt ist, können mit Perl die einzelnen Tabellen anhand ihrer Namen angesprochen werden, d.h. die Namen werden bei der DB-Abfrage in Variablen gespeichert. Somit hat man nun Zugriff auf den Inhalt der einzelnen Tabellen der MySQL-Datenbank. Mit dem 'select'-Befehl können die gewünschten Informationen abgefragt und anschliessend weiter verwendet werden.

Datenbankinhalt hinzufügen/löschen In einer Datenbank können Informationen jederzeit hinzugefügt oder auch gelöscht werden. Dies geschieht mit SQL-Befehlen, die hier nicht näher erläutert werden sollen. Für den Benutzer bedeutet das eine grosse Flexibilität bezüglich auftretender Veränderungen. Wenn sich also etwas, das den Inhalt der Datenbank betrifft, ändern

sollte, muss nicht die Gesamtheit der Datenbank geändert oder gar neu angelegt werden, sondern es können genau die relevanten Informationen angepasst werden. Dies hat auch auf die Abfragen keinen Einfluss, weil diese ja im Prinzip unabhängig vom Inhalt sind.

Dies ist grundsätzlich auch mit einem einfachen Text-File möglich. Diesem gegenüber hat die Datenbank aber entscheidende Vorteile:

- Die Datenbankabfrage ist schneller.
- Die Datenbankabfrage ist einfacher zu programmieren.
- Die Verwaltung von grossen Datensätzen ist komfortabler.

Gerade in Bezug auf die in der Einleitung von uns erwähnte Vision einer beliebigen Erweiterbarkeit des Informationssystems, ist dies ein wichtiger Punkt. Denn ein Grossteil der Informationen zur Gemeinde liegt in einer Datenbank. Es müssen also nur weitere Informationen von anderen Gemeinden gesammelt und eingefügt werden, um das ganze Informationssystem bezüglich Textdaten auszubauen.

5 Verwendung der Werkzeuge in unserer Arbeit

5.1 HTML

HTML verwenden wir für die Abfrage der SBB-Verbindungen, die Webcam-/Wetterseite und das Impressum.

5.2 SVG

SVG verwenden wir für die Hauptseite des Informationssystems. Der Grund dafür liegt darin, dass es bei SVG, wie der Name schon sagt, um skalierbare Vektorgrafiken geht. Der grosse Vorteil dabei ist, dass egal wie gross etwas dargestellt wird, die Informationsmenge bleibt immer dieselbe. Für die Beschreibung einer Web-Seite ist dies ein grosser Vorteil. Dies erlaubt vektorgrafische Darstellungen mit geringen Dateigrössen, die sich beliebig zoomen lassen.

5.3 CSS

CSS verwenden wir, um das gesamte Layout (Schriftarten, Schriftgrössen, Farben, Linienstärken, Opazitäten, Positionierung von Elementen, etc.) der Seite an einem zentralen Ort festzulegen.

5.4 ECMAScript

ECMAScript verwenden wir für die Definition aller Funktionen, wie Zooming oder Panning, und für Animationen, wie das Herunterfahren des Menüs. Zudem verwenden wir es, um beispielsweise per Mausklick auf Texte oder Symbole weitere Aktionen in Gang zu setzen. Gewisse Objekte, wie z.B. die Rechtecke der Menüleiste, kreieren wir mit ECMAScript.

ECMAScript ist die standardisierte Version von JavaScript.

5.5 Perl

Die verschiedenen Perl-Skripte, auf welche unsere Seite zugreift, stellen das Zwischenstück von der Web-Seite zur Datenbank dar. Die Aufgabe von Perl in diesem Fall besteht darin, SVG-Code zu schreiben. Allein zu diesem Zweck könnte man auch direkt auf ein SVG-Dokument zugreifen. Der Inhalt des SVG-Codes ist aber nicht statisch, sondern wird ergänzt mit Daten aus der Datenbank. An dieser Stelle kommt das DBI-Modul zum Zug. Mit diesem Modul wird während der Ausführung des Perl-Skripts die Datenbank mit den entsprechenden Parametern abgefragt. Das Resultat wird an den jeweiligen Stellen des Codes eingefügt. Mit Perl und DBI wird also der SVG-Code geschrieben, mit Inhalten aus der Datenbank an den entsprechenden Stellen. Die CGI-Schnittstelle schliesst die Lücke zwischen dem generierten Code und der Web-Seite. Bei einem Aufruf ohne CGI würde der Perl-Code an den Absender zurückgeschickt. CGI ist dafür verantwortlich, dass bei einem Aufruf das Perl-Skript ausgeführt wird und der generierte SVG-Code an den Absender zurückkommt.

5.6 CGI

In unserem Fall ist CGI dafür verantwortlich, dass auf eine Anfrage an ein Perl-Skript, dieses zuerst ausgeführt und als Antwort der generierte Source-Code folgt. Konkret existiert auf dem 'karto'-Server in unserem Verzeichnis ein Ordner mit dem Namen 'CGI_Bin', in welchen wir unsere Perl-Skripte abgespeichert haben. Sobald über das Web auf diesen Ordner zugegriffen

wird, 'weiss' der Server, dass er zuerst die Perl-Skripte ausführen muss, bevor er die Antwort auf die Anfrage zurückschickt.

5.7 MySQL-Datenbank

Unsere Datenbank befindet sich auf dem 'karto'-Server und hat den Namen 'turist'. Sie wurde uns vom IKA für die Zwecke dieser Arbeit eingerichtet. In dieser Datenbank können wir, soweit es uns von den (Benutzer-)Rechten her gestattet ist, wirken. In unserem Fall ist der Aufbau des Inhalts der Datenbank nicht allzu schwierig. Wir verwenden lediglich 6 Tabellen, die unabhängig von einander sind, d.h. es gibt keine Verknüpfungen. Die Struktur der einzelnen Tabellen ist je nach Inhalt leicht verschieden. Dies soll mit den folgenden Darstellungen verdeutlicht werden.

Unterkunft	
Attribut	Typ
id	char(25)
name	char(30)
strasse	char(25)
plz	int(4)
ort	char(20)
besitzer	char(50)
tel	char(15)
fax	char(15)
www	char(40)
email	char(40)
oeffnungszeiten	char(20)
ruhetag	char(30)
bettenzahl	int(2)
xkoord	int(4)
ykoord	int(4)

Kulinarisches	
Attribut	Typ
id	char(25)
name	char(30)
strasse	char(25)
plz	int(4)
ort	char(20)
besitzer	char(50)
tel	char(15)
fax	char(15)
www	char(40)
email	char(40)
oeffnungszeiten	char(20)
ruhetag	char(30)
xkoord	int(4)
ykoord	int(4)

Verkehrsmittel	
Attribut	Typ
id	char(25)
name	char(30)
besitzer	char(50)
strasse	char(40)
plz	int(4)
ort	char(20)
tel	char(15)
fax	char(15)
email	char(40)
intFahrplan	char(150)
SBBFahrplan	char(4)
xkoord	int(4)
ykoord	int(4)

Einkaufen	
Attribut	Typ
id	char(25)
name	char(30)
strasse	char(25)
plz	int(4)
ort	char(20)
besitzer	char(50)
tel	char(15)
fax	char(15)
www	char(40)
email	char(40)
xkoord	int(4)
ykoord	int(4)

Sehenswürdigkeiten	
Attribut	Typ
id	char(25)
name	char(30)
strasse	char(25)
plz	int(4)
ort	char(20)
xkoord	int(4)
ykoord	int(4)

Aktivitäten	
Attribut	Typ
id	char(25)
name	char(30)
besitzer	char(50)
strasse	char(50)
plz	int(4)
ort	char(20)
tel	char(15)
www	char(100)
email	char(100)
xkoord	int(4)
ykoord	int(4)

Erklärungen zu den Tabellen

Allgemein Die Attribute zu den einzelnen Hauptthemen sind so gewählt, dass sie diese möglichst gut repräsentieren. Bei der Definition der Typen verwenden wir 'char' für Zeichenketten und 'int' für ganze Zahlen. Zudem haben wir darauf geachtet, dass alle Attribute in voller Länge in die Datenbank eingefügt werden können. Beispielsweise gibt es www- oder email-Adressen mit sehr vielen Zeichen.

Attribut 'name' Der Typ des Attributs 'name' ist 'char (30)'. Er ist in allen Tabellen gleich definiert. Der Grund dafür ist die Breite der Informationsliste links, welche nur eine Länge von 30 Zeichen zulässt.

Attribut 'tel' Das Attribut 'tel' haben wir nicht als 'int', sondern als 'char' definiert, weil nur bei 'char' die '0' am Anfang der Zeichenkette geschrieben wird.

5.8 Schema

Die Zusammenhänge des Informationssystems, d.h. wie die einzelnen Teile miteinander verknüpft sind, soll anhand von Abb 1 gezeigt werden.

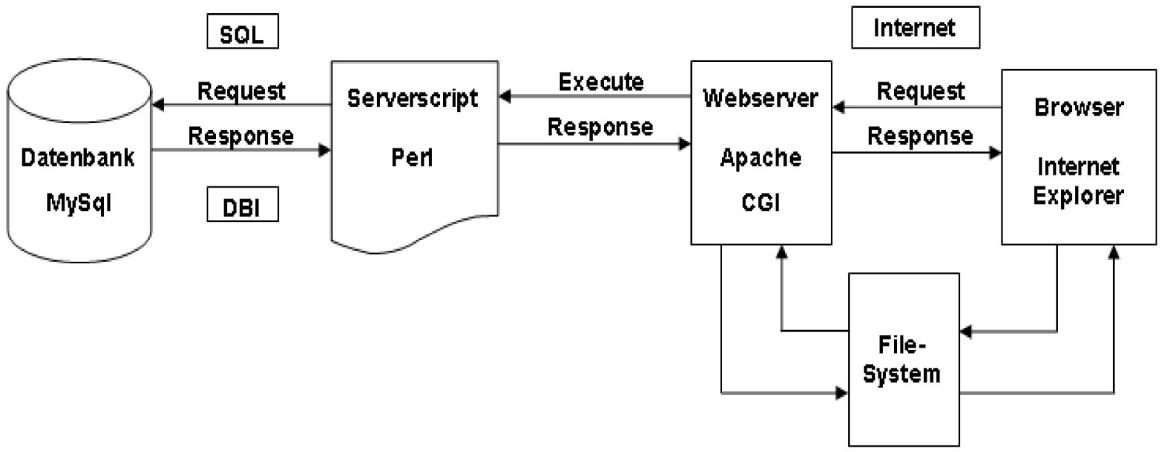

Abbildung 1: Schema der einzelnen Komponenten

6 Das Touristeninformationssystem Flüelen

6.1 Strukturierung

Bei einer Web-Seite mit kartographischem Inhalt sollte es so sein, dass die Karte eine zentrale Stellung einnimmt, sofern sie eine zentrale Bedeutung hat. Das heisst, dass sie einen relativ grossen Teil der Seite ausmachen sollte, wie dies in unserer Arbeit der Fall ist. Sie stellt den geografischen Bezug zu den thematischen Informationen her und ist dem Benutzer eine wichtige Hilfe bei der Nutzung dieses Informationssystems. Sie wird durch einen Massstab ergänzt.

Die Karte wird von zahlreichen weiteren Anwendungsmöglichkeiten umgeben. Zum einen sind dies die Elemente, die Veränderungen an der Karte ermöglichen, wie das Verschieben der Karte in 4 Richtungen, hinein- und herauszoomen sowie das Laden des Startbilds der Karte.

Desweiteren nimmt die Menüleiste oberhalb der Karte eine bedeutende Stellung ein. Sie ist der Zugang zu vielen Informationen, die für einen Touristen interessant sein können. Diese Leiste dient, wie schon in Kapitel 2.1 erwähnt, der Unterteilung der Informationen in verschiedene Themen. Werden die einzelnen Felder mit der Maus überfahren, erscheint ein weiteres Menü mit Unterthemen zu jedem Thema. Wenn man nun mit der Maus auf eines der Unterthemen klickt, wird eine Liste mit Einträgen zum jeweiligen Unterthema am linken Rand neben der Karte geladen. Ausserdem erscheinen Symbole zu den Einträgen auf der Karte. Durch Klicken auf das Feld eines Eintrags oder das entsprechende Symbol erscheinen nun in einem sich separat öffnenden Fenster genauere Informationen, beispielsweise zu Hotels oder Restaurants.

Damit nicht immer erst alle Menüs durchsucht werden müssen, haben wir für die Unterthemen 'Hotels' und 'Restaurants' Suchfunktionen eingerichtet. Sie befinden sich links neben den Zoomfunktionen, gekennzeichnet durch ein Feld mit der Aufschrift 'Suchen...'.

Ein weiteres Element der Web-Seite ist der Button am linken unteren Rand mit der Aufschrift 'Webcam/Wetter'. Hinter ihm versteckt sich ein aktueller Blick über den Urnersee mit einem Bergmassiv im Hintergrund und ausserdem eine Statistik zur aktuellen Wetterlage.

Ergänzt wird die Web-Seite durch das Wappen der Gemeinde Flüelen, eine Kontaktmöglichkeit zu den Erstellern der Web-Seite per e-mail, einen Link zur Hauptquelle dieser Arbeit, nämlich 'www.fluelen.ch', und das Impressum, welches Informationen zu diesem Vertiefungsblock enthält.

6.2 Einzelne Bestandteile

6.2.1 Karte

Die Karte wird so in die Web-Seite integriert, dass Interaktionen, wie Zoomen und Pannen, möglich sind. Dafür wird in der <defs>-Sektion mit dem sogenannten <clipPath>-Element ein Fenster definiert. Dieses Fenster gibt die Grösse des Kartenausschnitts an, der auf der Seite dargestellt wird. Was sich ausserhalb dieses Fensters befindet, ist nicht mehr sichtbar. Die Karte wird, je nach Interaktion, hinter diesem Fenster verschoben, vergrössert oder verkleinert. Im SVG-Hauptteil wird für die Integration der Karte sowie der Symbole eine eigene Gruppe definiert, die Bezug auf das definierte <clipPath>-Element nimmt.

Verschiedene andere Objekte und Operationen stehen mit der Karte in Verbindung. Da sind zum einen die Navigationsmöglichkeiten, wie 'Verschieben der Karte' und die 'Zoom'-Funktionen (s. Abb. 2). Beides wird mit Hilfe von ECMAScript-Funktionen (fluelen.js) bewerkstelligt. Dafür gibt es eine Hauptfunktion, in unserem Fall 'zoom-scale' genannt, die die dafür notwendigen Translationen und Skalierungen berechnet. Hierbei ist noch zu erwähnen, dass beim Laden der Web-Seite immer ein vordefinierter Startzustand erscheint, der einen

von uns gewählten Kartenausschnitt anzeigt. Die dafür zuständige Funktion heisst 'zoom-init' und wird in 'fluelen.svg' im <svg>-Starttag mit dem 'onload'-Befehl aufgerufen. Darin werden Startwerte für die Translation und die Skalierung definiert.

Das Verschieben der Karte wird durch die vier schwarzen Pfeile am Rand der Karte ermöglicht. Jede Verschiebungsrichtung hat eine eigene kleine Funktion, in der ein vordefinierter Verschiebungsbetrag zur derzeitigen Position in x- oder y-Richtung addiert oder von ihr subtrahiert wird. Die eigentliche Berechnung des neuen Kartenausschnitts mit den neuen Werten findet wieder in der oben erwähnten Hauptfunktion statt.

Das Zoomen geschieht auf ähnliche Weise. Der Unterschied liegt nur darin, dass nicht Translationswerte geändert werden, sondern die Skalierung. Ihr aktueller Wert wird in der Funktion 'zoomin' mit einem definierten Faktor multipliziert und in der Funktion 'zoomout' durch den gleichen Wert dividiert. Die Hauptfunktion berechnet dann wiederum die Karte neu. Die Zoom-Buttons befinden sich am oberen Rand des Feldes neben der Karte. Wie man sehen kann, handelt es sich dabei nicht nur um zwei, sondern drei Lupen. Die dritte Lupe mit dem 'S' darin ermöglicht das Laden der Anfangseinstellungen der Karte.

Ergänzt wird die Karte durch einen Massstabsbalken. Da die Karte jedoch nicht statisch ist, sondern wie wir eben gesehen haben unter anderem gezoomt werden kann, muss sich auch der Massstabsbalken an die jeweilige Zoomeinstellung anpassen. Damit immer ein gutes Erscheinungsbild erreicht wird, verwenden wir zwei in der Beschriftung unterschiedliche Balken. Einer gibt die Länge '1 km' an und der andere '500 m'. Die korrekte Länge des Balkens wurde aus der verwendeten Karte mit Hilfe der Kilometrierungslinien abgeleitet. Es ist vom momentanen Zoomfaktor abhängig, welcher Massstabsbalken angezeigt wird. Die Länge des Balkens innerhalb einer der beiden Zoom-Bereiche wird erneut mit der Hauptfunktion berechnet. Dabei handelt es sich um eine Skalierung in x-Richtung.

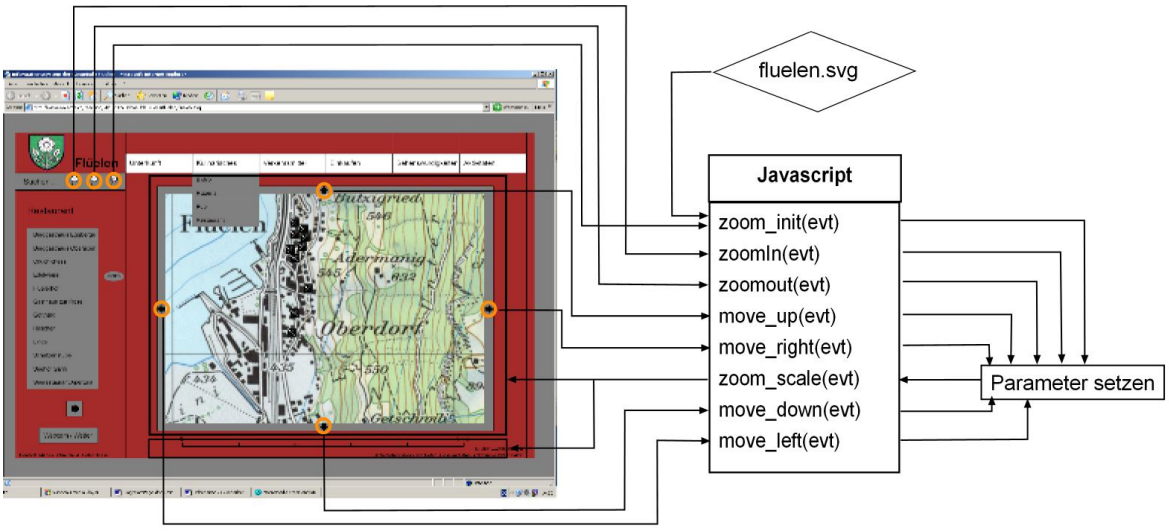

Abbildung 2: Funktionen der Karte

6.2.2 Menüleiste

Über die verwendeten Begriffe dieses Kapitels gibt Abb. 3 Aufschluss. Die Unterthemen der einzelnen Hauptthemen sind nicht von vornherein bekannt, sondern abhängig von den Einträgen in der Datenbank. Daher werden die Menüs durch ein Perl-Skript generiert. Dieses wird gleich zu Beginn beim initialisieren der Web-Seite aufgerufen (Abb. 4). Die Aufgabe des Perl-Skripts besteht darin, die Datenbank nach den verschiedenen Unterthemen abzufragen und diese unter den entsprechenden Hauptthemen aufzulisten. Zusätzlich ist ein Umweg über ein ECMAScript nötig, welches den vom Perl-Skript geschriebenen SVG-Code an der richtigen Stelle einfügt.

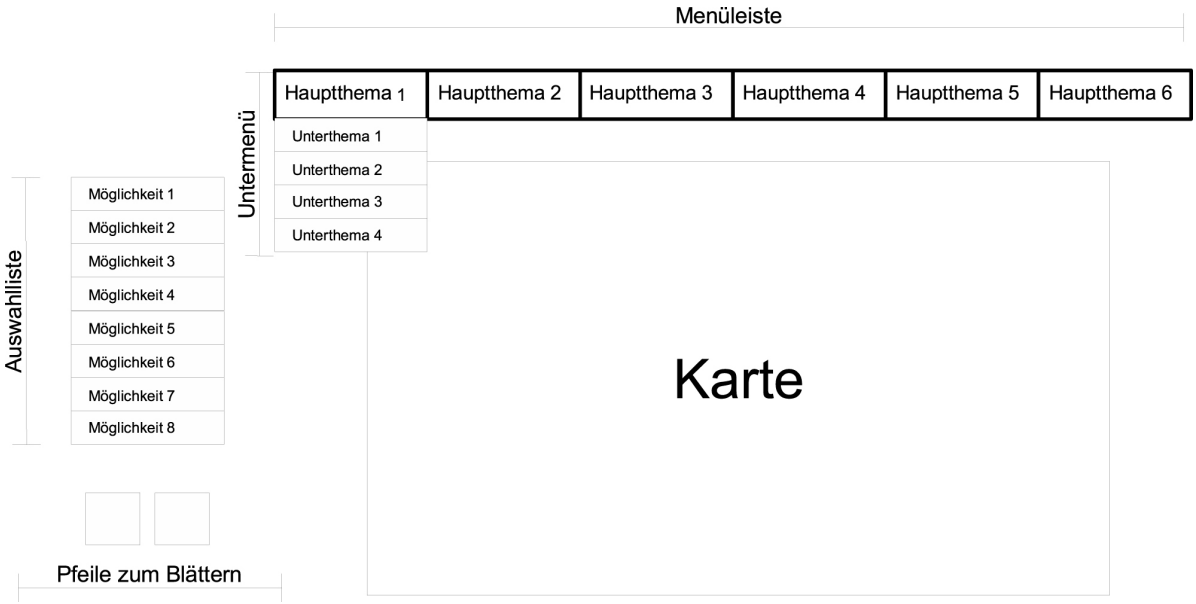

Abbildung 3: Schema mit den verwendeten Begriffen im Text

Aufruf von 'ladenMenu.pl' durch ECMAScript Beim Initialisieren der Web-Seite wird eine Funktion namens 'ladenMenu()' aufgerufen. Diese enthält nur den Aufruf des Perl-Skripts durch

```
getURL('../../../fluelen-cgi/ladenMenu.pl', Menu).
```

Dabei handelt es sich um eine ECMAScript-Funktion, welche über eine URL einen Quell-Code abfragt und diesen in der Funktion 'Menu(data)' weiterverarbeitet. Die Funktion 'Menu(data)' übernimmt nun den SVG-Code aus dem Perl-Skript und hängt diesen in der Web-Seite der Gruppe mit der ID 'Menu' als Kindknoten an. Es ist nicht zwingend, dass die Funktion den gleichen Namen hat wie die Gruppe im SVG-Code. Falls keine Daten aus dem Perl-Skript zurückkommen, gibt die Funktion eine Fehlermeldung aus.

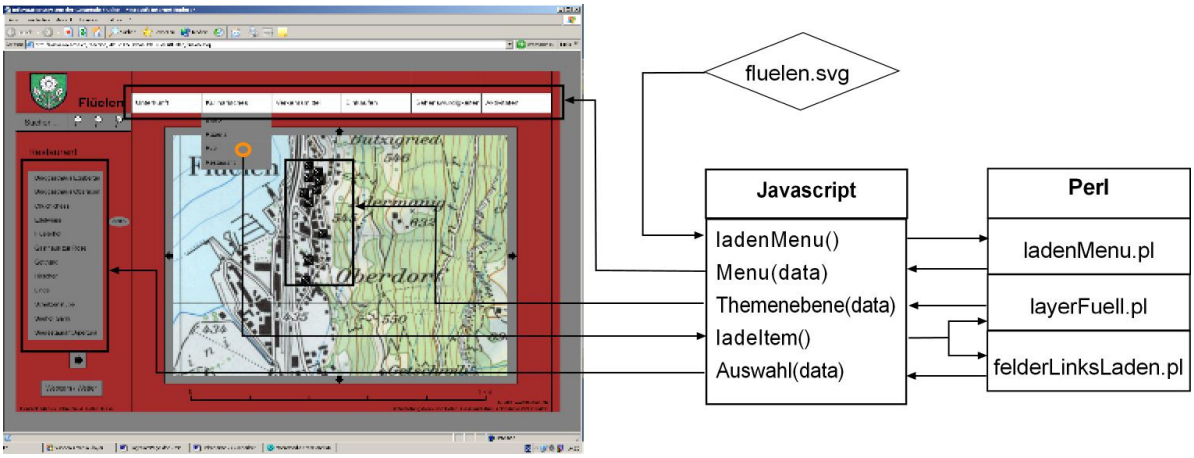

Abbildung 4: Darstellung der Auswahlliste und der Symbole

Perl-Skript 'ladenMenu.pl'

Datenbankabfrage Da die Aktualisierung der Web-Seite lediglich über die Datenbank laufen soll, müssen die Menüs bei jedem Aufruf neu aktualisiert werden, d.h. die Datenbank wird bei jedem Neuladen nach verschiedenen Unterthemen abgefragt. Das unter Kapitel 4.5.1 auf der Seite 13 vorgestellte Interface DBI stellt eine Reihe Funktionen zur Verfügung, welche in diesem Zusammenhang sehr nützlich sind. Die Datenbankverbindung erfolgt über den Befehl

```
my $dbh = DBI->connect( $data_source, $username, $password)
        or die "Can't connect to $data_source\n";
```

Nun wird mittels einer Schleife für jedes Hauptthema eine Abfrage durchgeführt. Erst wird das SQL-Statement in die Variable $sth gespeichert:

```
$sth[$i]=$dbh->prepare("SELECT DISTINCT id FROM $menuTitel[$i]");
```

anschliessend die Abfrage ausgeführt:

```
$resp[$i]=$sth[$i]->execute
        or die "Unable to execute your query ...\n";
```

und schliesslich das Ergebnis in einem Array gespeichert.

```
$table[$i]=$sth[$i]->fetchall_arrayref
        or die "$sth[$i]-errstr\n";
```

Generierter SVG-Code

Runter- und Zurückgleiten der Menüs Im Gegensatz zu HTML existieren in der verwendeten Version von SVG keine Funktionalitäten, welche das erstellen von Auswahlmenüs erleichtern. Auf den ersten Blick sieht die Aufgabe einfach aus: Die Unterthemen sollten beim Überfahren des Kastens des Hauptthemas heruntergleiten und sobald die Maus sich wieder aus dem Menübereich bewegt, sollten die Unterthemen wieder verschwinden.

Zuerst orientieren wir uns an einer Vorlage von Andreas Neumann vom Institut für Kartografie an der ETHZ. Diese Vorlage ist aber eher komplex und schwierig zu begreifen. Daher wird ein eigener Vorschlag entwickelt.

Das Herzstück bilden zwei Animationen, welche die Gruppe, in welcher die Untermenüs untergebracht sind, nach unten resp. nach oben bewegen. Die Animation, welche das Menü nach unten bewegt wird ausgelöst, wenn der Maus-Cursor das Hauptthemenfeld überfährt. Die Animation darf hingegen nicht ausgelöst werden, wenn der Maus-Cursor vom bereits geöffneten Menüfeld ins Hauptthemenfeld bewegt wird. Die Empfindlichkeit des Hauptthemenfeldes wird mit der ECMAScript-Funktion 'menuUp()' gesteuert. Das Attribut 'begin' der Nach-Unten-Animation wird ausgeschaltet und erst beim Verlassen des ganzen Feldes wieder eingeschaltet. Gleichzeitig muss auch das Schliessen des Menüs organisiert werden. Zwar ist es möglich eine ganze Gruppe mit einer ECMAScript-Funktion anzusprechen, die Funktion wird so aber für jedes Gruppenelement einzeln ausgeführt. So würde z.B. die Schliessen-Funktion beim Verlassen jedes einzelnen Textes ausgelöst, was nicht zum gewünschten Ergebnis führt. Stattdessen wird beim Öffnen des Menüs ein leeres Rechteck mit einem äusserst dicken, durchsichtigen Rahmen (400px!) um die ganze Gruppe, d.h. Hauptthemakasten und Untermenüs, gelegt (Abb. 5).

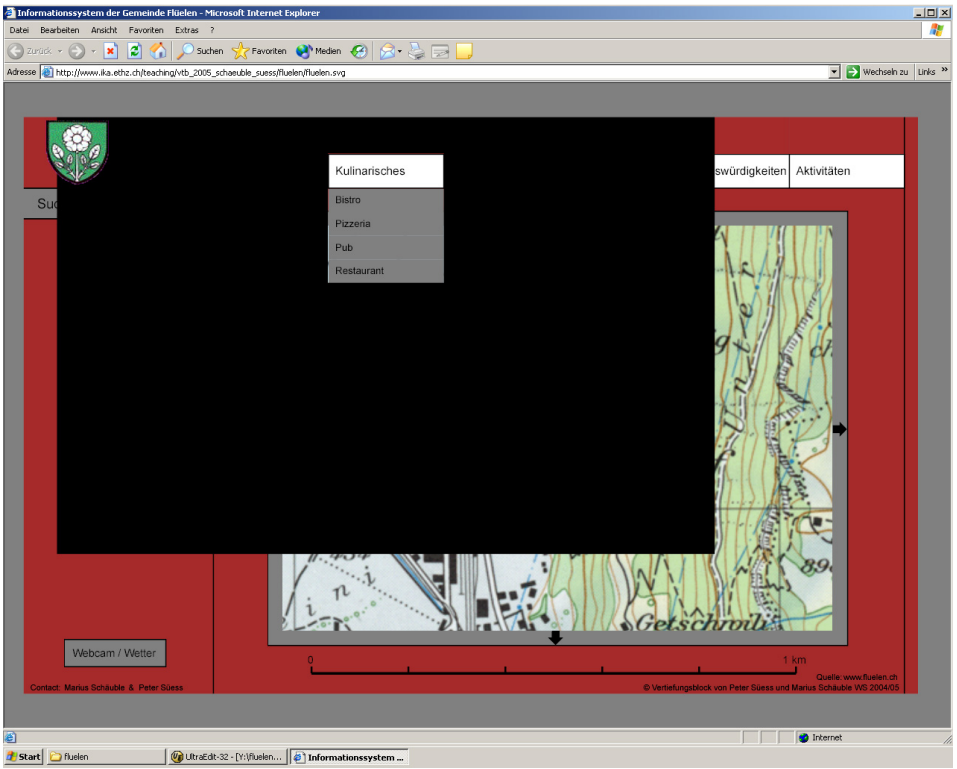

Abbildung 5: Rahmen, welcher die Funktion menuClose() auslöst

Sobald der Maus-Cursor den Rahmen des leeren Rechtecks berührt, wird die Funktion 'menuClose()' ausgelöst. Diese macht nichts anderes, als den Ursprungszustand wieder herzustellen. Aus kosmetischen Gründen wird die Gruppe mit den Untermenüs mit einem Rechteck abgedeckt, damit sie unter dem Hauptthemakasten nicht sichtbar ist.

Aufleuchten der Untermenüs Jedes einzelne Rechteck der Untermenüs ist überlagert mit einem Rechteck, welches nur sichtbar ist, wenn sich die Maus darüber befindet. Wiederum

über eine ECMAScript-Funktion (farbeHigh()) wird die Stilklasse (CSS-File) des Rechtecks geändert, d.h. das Rechteck verändert das Attribut 'opacity' von Null auf 0.3, was das Feld heller erscheinen lässt. Ausserdem bekommt das Rechteck einen weissen Rand, damit es sich etwas abhebt. Der Nachteil bei dieser Methode ist die Verblassung des Textes. Bei der gewählten Farbkombination fällt dies aber nicht ins Gewicht.

6.2.3 Symbole

Überblick Ein Satz Symbole, z.B. alle Hotels, wird durch Anklicken eines Unterthemas angezeigt. Um die einzelnen Möglichkeiten (Abb. 3) identifizieren zu können, erscheint das Symbol beim Überfahren grösser. Gleichzeitig wird der Name in der Auswahlliste heller dargestellt. Diese Identifikation der Elemente wird in Kapitel 6.2.4 auf Seite 27 ausgeführt.

Aufruf von 'layerFuell.pl' durch ECMAScript Der Ablauf geschieht ähnlich wie in Kapitel Menüleiste (6.2.2) unter 'Aufruf von 'ladenMenu.pl' durch ECMAScript' auf Seite 22 beschrieben. Das ECMAScript ruft ein Perl-Skript auf, welches den entsprechenden SVG-Code schreibt und fügt diesen an der richtigen Stelle ein, nämlich in der Gruppe 'Themenlayer'. Der Unterschied besteht darin, dass in diesem Fall angegeben werden muss, um welches Unterthema es sich handelt. Es soll nicht irgendein Unterthema dargestellt werden, sondern dasjenige, welches der Benutzer angeklickt hat. Diese Information steckt in der ID des Objekts, auf welches geklickt wird. Sie wird mit dem ECMAScript-Befehl

```
var feld = evt.getTarget();
actAttr = feld.getAttribute("id");
```

abgefragt und im Aufruf des Perl-Skripts als sogenanntes 'path-info' mitgegeben:

```
getURL('../../../fluelen-cgi/layerFuell.pl'+"/"+actAttr, Themenebene)
```

Der vom Perl-Skript generierte SVG-Code wird in der Funktion 'Themenebene()' weiterverarbeitet. Dort wird die Rückgabe zuerst nach Inhalt überprüft. War die Abfrage nicht erfolgreich wird eine Fehlermeldung ausgegeben. Danach müssen allfällige schon vorhandene Inhalte aus der Gruppe 'Themenlayer' in der Web-Seite entfernt werden. Dies ist der Fall wenn vorher schon eine Auswahl getroffen wurde. Danach wird der SVG-Code eingefügt.

Datenbankabfrage im Perl-Skript 'layerFuell.pl' Bevor die Abfrage erfolgen kann, muss zuerst die Information aus dem Aufruf extrahiert werden. Dazu kennt das CGI-Paket den einfachen Befehl

```
my $path_info=$q->path_info;,
```

welcher die gewünschte Information in die Variable '$path_info' schreibt. Diese Variable enthält nun zwei Informationen, erstens, um welches Hauptthema (=Tabellenname in der Datenbank) es sich handelt und zweitens, welches Unterthema der Benutzer angeklickt hat. Diese zwei Angaben sind getrennt durch einen Unterstrich (z.B. Unterkunft_Hotel) und müssen nun in je eine Variable geschrieben werden mit dem Befehl

```
($sparte, $art)=split /_/,$path_info;.
```

Ausserdem muss der Variablen '$sparte' noch der Schrägstrich abgetrennt werden mit

```
my $sparte = substr($sparte,1,length($sparte));.
```

Die Datenbankabfrage geschieht wiederum mit Hilfe der Befehle des DBI-Moduls. Das SQL Statement wird in eine Variable '$sth' gepackt:

```
my $sth=$dbh->prepare("SELECT name,xkoord,ykoord
                       FROM $sparte WHERE id = '$art' ");
```

Durch dessen Ausführen erhält man die Attribute 'name', 'xkoord' und 'ykoord', wobei der Name als Identifikator eingesetzt wird und daher nicht zweimal im gleichen Unterthema vorkommen darf. Zum Beispiel würden zwei 'Hotel Sternen' zu Problemen führen. Hingegen ein 'Hotel Sternen' und ein 'Restaurant Sternen' können problemlos existieren. Die Attribute 'xkoord' und 'ykoord' sind die Koordinaten der Symbole, mit welchen sie in die Karte eingefügt werden.

SVG-Code aus dem Perl-Skript 'layerFuell.pl' Grundsätzlich werden die Symbole als Bilder (.gif) in die Gruppe 'Karte' in der Web-Seite eingefügt. Somit sind auch die Funktionalitäten der Gruppe 'Karte', wie Zoomen und Pannen, auch für die Symbole gewährleistet. Ausserdem müssen sie im Ordner Symbole/'Hauptthema' als 'Unterthema'.gif gespeichert sein. Das Symbol für die Hotels befindet sich z.B. unter 'Symbole/Unterkunft/Hotel.gif'.

Die Intuitivität der Symbole ist nicht von grösster Bedeutung, da immer nur eine Art von Symbolen auf der Karte erscheint. Natürlich ist es sinnvoll die Symbole einigermassen geschickt zu wählen, damit keine Verwirrung entsteht. Die Symbole werden mit einer ID versehen. Diese ID lautet gleich wie der Name der Lokalität, welches das Symbol darstellt. Dies ist nötig, um das Symbol anzusprechen für die Identifikation zwischen der Auswahlliste und den Symbolen in der Karte. Dieses Thema wird in Kapitel 6.2.4 auf Seite 27 unter dem Titel 'Identifikation von Symbol und zugehörigem Namen' ausführlich behandelt.

Für das Unterthema 'Wandern' werden nicht Punktelemente dargestellt, wie bei allen anderen Unterthemen, sondern Linienelemente entlang des Verlaufs des Wanderweges. Die Wanderwege werden mit dem Programm 'inkscape' digitalisiert. Das Resultat ist ein Pfad in SVG-Code, d.h. ein Pfad bestehend aus Koordinatenpaaren. Dieser Pfad wird in einem .txt-File separat abgespeichert. Neben dem Symbol wird nun auch das Linienelement in die Karte eingefügt. Einerseits wollen wir damit zeigen, dass es auch möglich ist Linienelemente darzustellen und zweitens glauben wir, ist es berechtigt, das Unterthema Wandern speziell zu behandeln, da es in der Schweiz praktisch überall Wanderwege gibt.

6.2.4 Auswahlliste

Überblick Sobald der Benutzer im Menü ein Unterthema anklickt, wird das Perl-Skript 'felderLinksLaden' aufgerufen. Die erhaltenen Daten, resp. der generierte SVG-Code, wird in der Gruppe 'Auswahlliste' eingefügt. Der Aufruf ist in Kapitel 6.2.3 auf Seite 25 beschrieben. Die Datenbankabfrage funktioniert analog zu den Symbolen, mit dem Unterschied, dass die Koordinaten für die Symbolplatzierung nicht gebraucht werden.

SVG-Code aus dem Perl-Skript 'felderLinksLaden.pl' Als Ausgangspunkt für die Platzierung der Auswahlliste gilt die linke obere Ecke des obersten Namensfeldes. Alle anderen Werte wie Textplatzierungen oder Platzierungen der GOTO-Ellipse werden daraus abgeleitet.

Die Liste ist nicht unendlich lang, sie bietet Platz für 12 Einträge. Sobald mehr als zwölf Einträge vorhanden sind, werden mehrere Listen erstellt, welche nacheinander über ein Pfeilsymbol aufgerufen werden können. Die verantwortliche Funktion für diese Funktionalität wird später beschrieben. Die verschiedenen Listen mit je zwölf Einträgen werden in dieser Funktion

mit dem visibility-Attribut sichtbar oder eben unsichtbar gemacht. Das Perl-Skript generiert also eine sichtbare Liste 'liste0' mit den ersten zwölf Einträgen, alle weiteren Listen sind unsichtbar mit entsprechender ID 'liste1', 'liste2', etc..

Nun wird überprüft, ob es sich um das Unterthema 'Wandern' handelt oder nicht. Der Grund liegt darin, dass die Elemente bei einem Mausklick unterschiedliche Links aufrufen. Beim Anklicken eines Namens unter dem Unterthema 'Wandern' wird eine externe URL aufgerufen, während bei allen anderen ein internes Infofenster generiert wird.

Die Namen aus der Datenbank werden in einem Rechteck dargestellt. Diese beiden Elemente werden wiederum überlagert von einem Rechteck, welches

- das Aufleuchten des Feldes auslöst,
- die ID des Feldes trägt,
- den Link für weitere Informationen trägt.

Für das Aufleuchten werden dabei die gleichen Funktionen verwendet wie schon in der Menüleiste bei den Unterthemen (Kapitel 6.2.2, Seite 24).

Identifikation von Symbol und zugehörigem Namen Die reine Darstellung von Symbolen in der Karte und den Namen in der Auswahlliste bringt eigentlich noch nicht viel. Der Betrachter möchte z.B. wissen, wo sich welches Hotel befindet. Für diese Identifikation leuchtet einerseits der Name in der Auswahlliste auf, während andererseits das Symbol in der Karte grösser wird. Dies wird ausgelöst beim Überfahren von einem der beiden Elemente. Verantwortlich dafür ist die ECMAScript-Funktion 'identifikationLinks()' beim Überfahren der Auswahlliste und die Funktion 'identifikationSymbol()' beim Überfahren der Symbole. Beide Funktionen sind praktisch identisch, einzig die Variablen sind vertauscht. Leider ist es in SVG nicht möglich über ECMAScript zwei Elemente gleichzeitig anzusprechen, welche die gleiche ID besitzen. Dies ist aber der Fall bei den Rechtecken in der Auswahlliste und den Symbolen. Dort wird jeweils der Name als ID des Objektes gespeichert. Deshalb wird erst die ID des unter der Maus liegenden Objektes abgefragt und in einer Variablen gespeichert. Nun wird die ID desselben (auf dem sich die Maus befindet) geändert. Es existiert nun also nur noch ein Element mit dieser ID in der Seite. Dieses Element (auf welchem sich die Maus nicht befindet) wird nun mit der ID aufgerufen und die entsprechende Aktion über Attributänderungen ausgeführt. Die Funktion 'identifikationOut()' bringt beim Verlassen des jeweiligen Elementes schliesslich wieder alles in Ordnung. Die betroffenen Objekte werden in globalen Variablen abgespeichert, es ist also kein Problem, die Attribute dieser Objekte wieder zurück in den Normalzustand zu stellen.

Als Alternative könnte man den beiden Objekten verschiedene IDs vergeben, z.B. 'Name-Symbol' und 'NameFeld'. In diesem Fall müsste aber aus der jeweiligen ID der Teil 'Name' extrahiert und mit 'Feld' resp. 'Symbol' ergänzt werden, um das zugehörige Element zu identifizieren. Beide Varianten erscheinen uns etwa gleich aufwändig.

GOTO-Funktion Beim Überfahren eines Rechtecks in der Auswahlliste erscheint rechts davon eine Ellipse mit der Inschrift GOTO. Dabei wird zuerst geprüft, ob schon eine GOTO-Ellipse vorhanden ist. Diese wird allenfalls versteckt (visibility = 'hidden'), sodass immer nur eine Ellipse zu sehen ist. Beim Anklicken kommt die ECMAScript-Funktion 'ZentTrans' zum Einsatz (Abb. 6). Sie fragt die Koordinaten des entsprechenden Symbols ab und zentriert die Karte anschliessend nach diesen Koordinaten. Falls zu diesem Namen kein Symbol existiert, wird

die Fehlermeldung 'Leider keine Informationen auf der Karte vorhanden!' ausgegeben. Dies ist z.B. bei Ausflugszielen, die sich ausserhalb der Karte befinden, der Fall, wie z.B. dem Bergrestaurant Eggberge. Es wäre auch denkbar Ausflugsziele ausserhalb des Ortes aufzuführen, welche bequem zu erreichen sind, wie z.B. der Vergnügungs- und Freizeitpark in Morschach. Schliesslich ist es für einen Touristen sicherlich von Interesse, welche Möglichkeiten an Aktivitäten sich in der Nähe bieten, nicht nur innerhalb des jeweiligen Ortes.

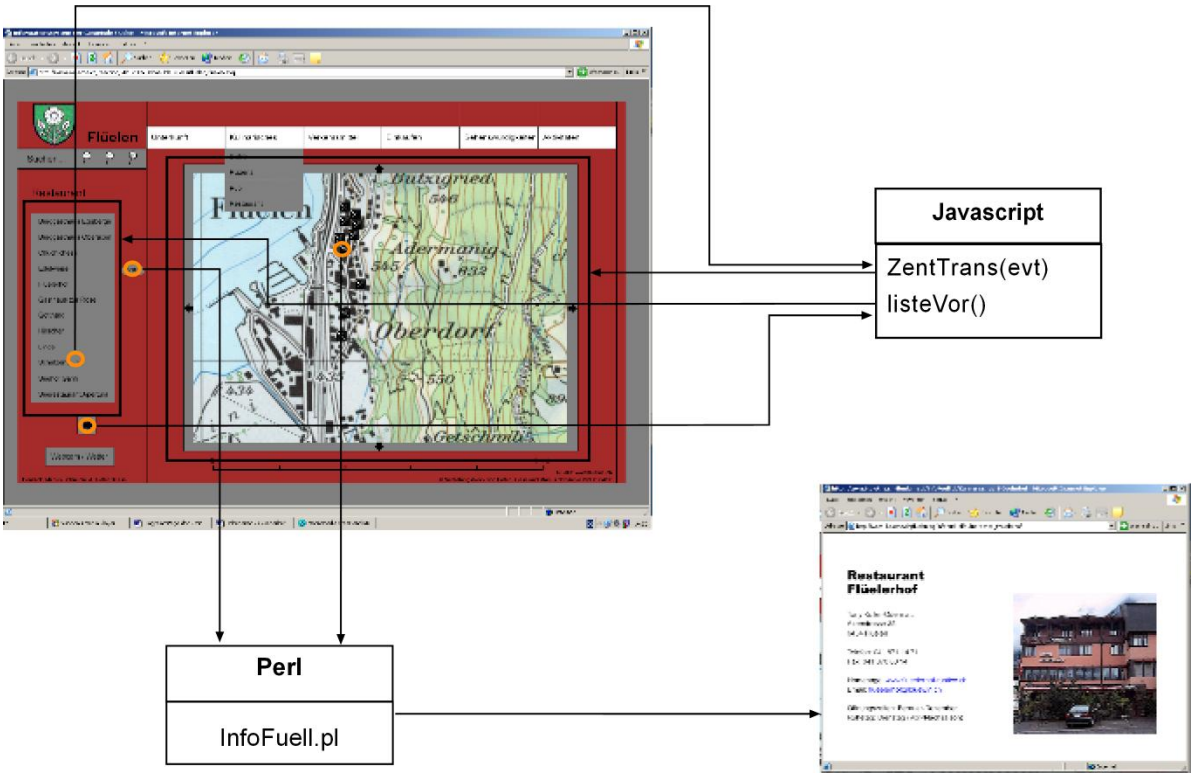

Abbildung 6: Funktionen der Auswahlliste

Blättern der Auswahllisten Im Kapitel zum Perl-Skript der Auswahlliste auf Seite 26 wurde gesagt, dass nur zwölf Einträge in eine Liste passen. Für allfällige weitere Einträge werden weitere Listen erstellt. Die Listen können über das Attribut 'visibility' ein und ausgeschaltet werden. Dazu sind die Pfeile unter der Auswahlliste da, einer nach links und einer nach rechts (Abb. 6). Die Pfeile müssen soweit 'intelligent' sein, dass sie nur erscheinen, wenn auch wirklich eine Liste vor oder zurück geblättert werden kann. Für die Identifizierung existiert die Variable 'listennr', welche die Nummer der Liste angibt und zusammengesetzt mit 'liste' kann nun simpel die Liste aufgerufen werden.

6.2.5 Informationsfenster

Überblick Das Anklicken eines Rechtecks in der Auswahlliste oder eines Symbols in der Karte führt zum Aufruf des Informationsfensters (Abb. 6). In diesem werden die in der Datenbank vorhandenen Informationen dargestellt. Eine Ausnahme bildet das Unterthema 'Wandern', bei welchem direkt eine externe Internetseite aufgerufen wird. Denkbar wäre hier ein Link zu einem Wanderroutenplaner, wie er von [5] gemacht wurde. Für den Aufruf des Informationsfensters

kommt wiederum ein Perl-Skript zum Zug. In diesem Fall kann aber der Umweg über das ECMAScript ausgelassen werden, da der generierte Code nicht in die Haupt-Seite eingefügt, sondern in einem separaten Fenster dargestellt wird.

Datenbankabfrage Dem Perl-Skript werden wiederum die Parameter für die Datenbankabfrage via URL übergeben. Die Abfrage selbst ist praktisch identisch mit den Abfragen aus den schon vorgestellten Perl-Skripten, mit dem Unterschied, dass alle Informationen abgefragt werden und diese in einem Hash statt einem Array abgespeichert werden.

Der Hash ist eine Datenstruktur welche einem Wörterbuch ähnelt.

Schlüsselwort	**Inhalt**		
Strasse	street	road	avenue
Tisch	desk	table	
Haus	house	home	
Bildschirm	screen		

Tabelle 1: Beispiel eines Hash

Ein Hash beinhaltet ein oder mehrere Schlüsselwörter und jedem Schlüsselwort sind ein oder mehrere Inhalte zugeordnet (s. Tabelle 1). Die Schlüsselwörter könnten z.B. eine Anzahl deutscher Wörter sein und deren Inhalte wären ein oder mehrere englische Wörter. Über das Schlüsselwort kann nun dessen Inhalt abgefragt werden. In diese Datenstruktur werden nun die Informationen aus der Datenbank, inklusive deren Attributsnamen, abgespeichert. Ein Hash enthält z.B. die Keywords 'ID', 'name', 'adresse', 'ort', etc. und deren Inhalte lauten 'Hotel', 'Sternen', 'Flüelerstrasse', 'Flüelen', etc.

Dies ist äusserst praktisch, da man alle Informationen der Tabelle abfragt, aber nicht im Vornherein weiss, um welche Tabelle es sich handelt. Infolge dessen können die Attribute in der Abfrage unterschiedlich lauten. Wäre das Resultat in einem Array gespeichert, hätte man keine Attributsinformationen und wüsste nicht wie man die einzelnen Informationen im Array verarbeiten soll. Mit einem Hash ist es möglich, die Existenz eines Schlüsselworts, z.B. der Faxnummer, abzufragen.

Der SVG-Code aus dem Perl-Skript 'infoFuell.pl'

Allgemein Der SVG-Code aus diesem Perl-Skript beginnt mit dem üblichen XML Header, da das Informationsfenster separat dargestellt und nicht in die Seite eingefügt wird. Dann werden die einzelnen Informationen geschrieben. Die verschiedenen Hauptthemen werden verschieden dargestellt (Abb. 7). Zuoberst steht der Titel, gefolgt von Name, Adresse und Ort. Diese Attribute sind in allen Tabellen vorhanden, ausser in der Tabelle 'Sehenswürdigkeiten'.

Es ist zu sagen, dass jedes Attribut auf seinen Inhalt geprüft wird. Enthält das Attribut keinen Inhalt, entsteht nicht eine Lücke in der Darstellung im Informationsfenster, sondern jede folgende Information wird in einem bestimmten Abstand an die Vorhergehende angehängt. Dies ist mit einer Art Zeiger realisiert, welcher die aktuelle Stelle zeigt, an welcher die Informationen geschrieben werden. Nach der Adresse folgt die Telefonnummer und die Faxnummer und ebenso die Adresse einer allfälligen Homepage und eine E-Mail Adresse. Die letzteren zwei sind in blau

geschrieben und führen per Mausklick direkt auf die entsprechende Seite, resp. zum installierten Mail-Programm.

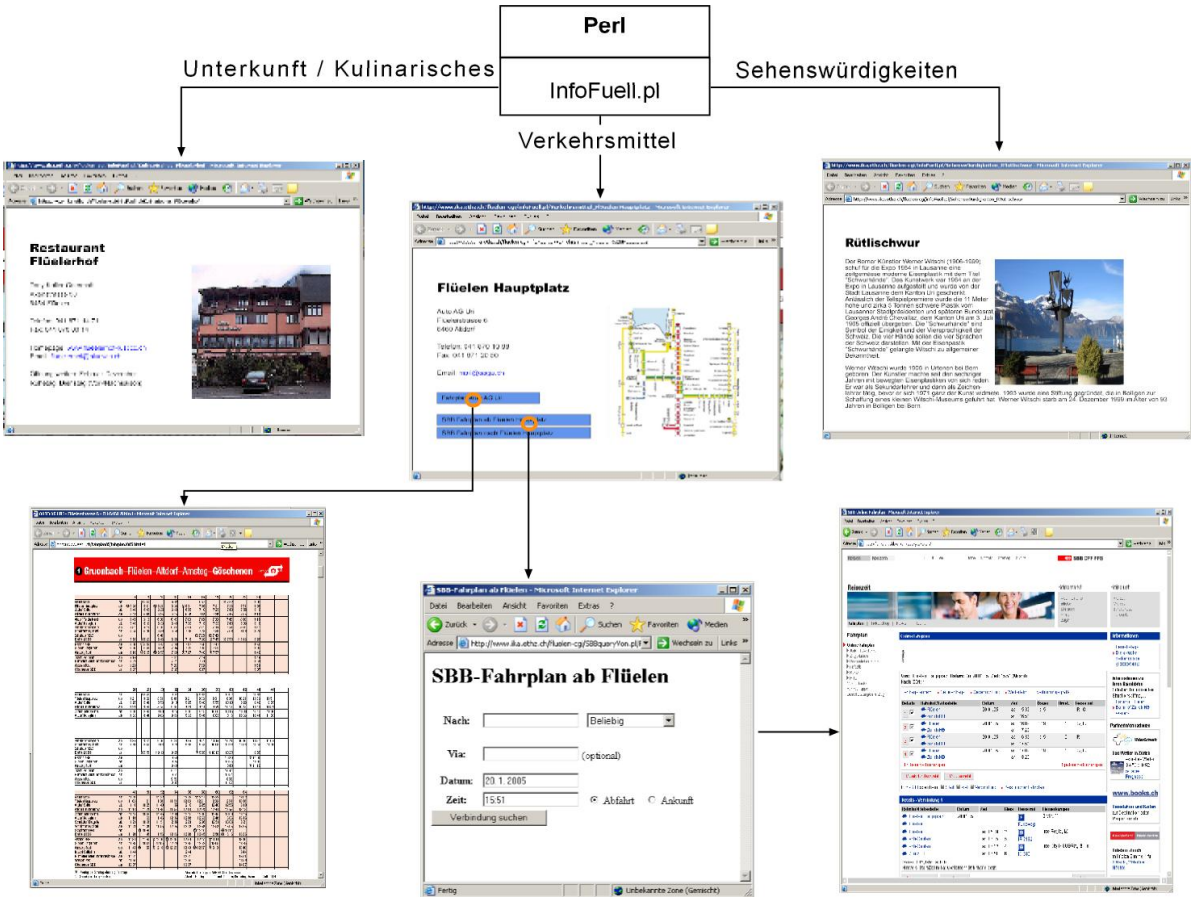

Abbildung 7: Verschiedene Darstellungen des Infofensters

Unterkunft und Kulinarisches Im Anschluss folgen die themenspezifischen Informationen. Für die Hauptthemen 'Unterkunft' und 'Kulinarisches' sind dies die Öffnungszeiten, der Ruhetag und allenfalls die Bettenzahl.

Sehenswürdigkeiten Für den Fall der 'Sehenswürdigkeiten' wird aus einem separaten Textfile der Text eingelesen und in den SVG-Code eingefügt. Leider bietet SVG noch keine Möglichkeit den Text in eine Form, z.B. in ein Rechteck mit automatischem Zeilenumbruch, zu schreiben. Im Textfile muss daher jeder Zeilenumbruch so eingefügt werden, wie er im Informationsfenster erscheinen soll.

Verkehrsmittel Bei den Verkehrsmitteln ist, neben der Lokalisierung und den Anschlussmöglichkeiten, der Fahrplan von Interesse. Viele Verkehrsmittelanbieter stellen auf dem Internet ihren eigenen Fahrplan zur Verfügung. Die Web-Adresse kann in die Datenbank eingegeben werden, diese wird im Informationsfenster als Link dargestellt.

Der SBB-Fahrplan [15] ist in der Schweiz wohl der umfangreichste und bietet zudem komfortable Abfrage-Möglichkeiten. Diese machen wir uns zu Nutze und zwar nicht mit einem

einfachen Link, sondern das Resultat sollte direkt angezeigt werden durch Eingabe der Zeit und des Ziel- oder Abfahrtsortes. Dazu haben wir aus der HTML-Seite der SBB alles Unnötige rausgelöscht, sodass nur noch die Eingabefelder für Ziel- oder Abfahrtsort, die Zeit und das Datum übrig bleiben. Die letzteren zwei werden mit Default-Werten gefüllt, die Zeit mit dem aktuellen Zeitpunkt plus vier Stunden und dem aktuellen Datum.

Das ausgefüllte HTML-Formular wird nun an die Abfrageadresse der SBB geschickt und als Antwort erhält man die gewünschten Resultate. Dies funktioniert natürlich nur, wenn die entsprechenden Verkehrsmittel oder dessen Haltestellen im SBB-Fahrplan vorhanden sind. In der Tabelle 'Verkehrsmittel' existiert dazu eine Spalte, welche mit 'ja' oder 'nein' Auskunft darüber gibt. Für den Aufruf sind im Informationsfenster zwei Links angezeigt, einer für die Hinfahrt (nach Flüelen) und einer für die Rückfahrt (von Flüelen). Dies ist nötig, da der Ausgangs- resp. Zielort mittels Perl direkt in das HTML-Eingabefenster gespeichert wird. Der Ausgangs- resp. Zielort ist bekannt, da der Benutzer das entsprechende Symbol angeklickt hat.

Bild In jedem Informationsfenster ist es möglich ein Bild darzustellen. Dieses muss in einem Ordner am gleichen Ort wie die Web-Seite im Stil 'Bilder/Haupthema/Name.gif' abgespeichert sein. Das Bild des Restaurants Sternen wäre also unter 'Bilder/Kulinarisches/Sternen.gif' zu finden. Wenn kein Bild vorhanden ist, wird der Text 'Leider kein Bild vorhanden' angezeigt.

6.2.6 Webcam / Wetter

Als interessante Information für jeden Touristen sehen wir den Knopf 'Webcam/Wetter' mit der dahinterliegenden HTML-Seite. Hier können alle einen Blick über den 'Urnersee' werfen und sich ein Bild von der aktuellen Wetterlage machen. Ergänzt wird der Blick durch die Webcam mit weiteren statistischen Daten zum Wetter, wie Temperatur, Luftfeuchtigkeit, Luftdruck, etc.. Wir haben die Webcam nicht selbst installiert, sondern wir verwenden den Service, der auf [8] angeboten wird. Wir haben die dafür benutzte HTML-Seite etwas ausgedünnt und von unnötiger Werbung befreit und anschliessend mit unserer Web-Seite über den angesprochenen Button verlinkt.

6.2.7 Suchfunktion

Die implementierte Suchfunktion ermöglicht es, nach Namen von Unterkünften und Gaststätten zu suchen. Dabei muss der Name genau so eingegeben werden, wie er in der Datenbank steht. Die Funktionalität ist nicht sehr ausgereift. Aus Zeitgründen haben wir uns auf das Wichtigste beschränkt. Die Suche ist nur für die Hauptthemen 'Unterkunft' und 'Kulinarisches' möglich, da wir diese als die Wichtigsten erachten. Die gesamte Funktion ist im Wesentlichen eine geschickte Aneinanderreihung schon vorhandener Funktionen. Unter dem Button 'Suchen...' verstecken sich ein Restaurant- und ein Hotel-Symbol. Durch Anklicken erscheint eine Eingabeaufforderung, in welche der Name des gesuchten Objektes eingegeben werden kann. Für die jeweiligen Hauptthemen 'Restaurant' und 'Hotel' existieren separate Funktionen, da wir sonst das Perl-Skript 'infoFuell.pl' nicht hätten verwenden können. Dem Skript 'infoFuell.pl' muss das Hauptthema als Parameter übergeben werden. Die Suche beinhaltet nicht mehr, als dass sie versucht, das Programm 'infoFuell.pl' mit dem eingegebenen Namen als Parameter auszuführen. Gelingt dies, wird ein Informationsfenster geöffnet, analog dem Anklicken eines Symbols oder einem Rechteck in der Auswahlliste. Ist die Suche nicht erfolgreich, wird die Meldung 'Nichts gefunden unter *Suchbegriff*' zurückgegeben. Um eine ausgereifte Suche zu kreieren, erscheint uns SVG nicht sonderlich geeignet, da uns keine Methode bekannt ist,

ein Eingabefeld anzuzeigen. Daher wird für die Eingabeaufforderung ECMAScript verwendet. Auch diese Variante hat einen Haken: Die Eingabeaufforderung 'promt(...)' funktioniert zum Zeitpunkt unserer Arbeit nur mit dem Internet Explorer.

6.2.8 Impressum

Das Impressum unserer Web-Seite besteht aus drei Teilen. Am linken unteren Rand wird den Benutzern dieser Seite die Möglichkeit gegeben, die Autoren per e-mail zu kontaktieren. Durch Klicken auf einen der Namen wird automatisch ein kleines Fenster geöffnet, mit welchem eine Nachricht an den jeweiligen Autor gerichtet werden kann.

Auf der rechten unteren Seite gibt es zum einen die Möglichkeit, die Homepage der Gemeide Flüelen, welches die Hauptquelle unserer Arbeit ist, zu besuchen. Ausserdem ist mit dem darunterliegenden Schriftzug 'Vertiefungsblock von...' eine HTML-Seite verlinkt, die alle notwendigen Informationen, wie Studiengang, Institut, Thema, Bearbeitung, Leitung, Betreuung, Bearbeitungszeitraum, Grundlagenkarte und Quelle, zu dieser Arbeit enthält.

6.3 In dieser Arbeit nicht vorgesehen

6.3.1 Übersichtskarte

Wir haben kurzzeitig daran gedacht, eine zusätzliche Übersichtskarte im Feld links neben der Karte anzubringen. Da jedoch die verwendete Karte der Gemeinde Flüelen nicht übermässig gross ist, haben wir dies nicht umgesetzt. Der Fensterinhalt kann relativ schnell von einem Ende zum anderen geschoben werden und durch genügendes Herauszoomen kann man auch eine Übersicht über die ganze Gemeinde erhalten. Will man das System nun um weitere Gemeinden erweitern, wird eine grössere Grundlagenkarte nötig sein, was ab einer bestimmten Grösse auch eine Übersichtskarte sinnvoller machen würde, um den Überblick zu behalten.

6.3.2 Legende

Eine Legende, vor allem zur Erklärung der einzelnen Symbole, ist unserer Ansicht nach in dieser Arbeit nicht notwendig. Es wird immer nur ein Themenlayer geladen, was heisst, dass stets nur eine Sorte Symbole auf der Karte erscheint. Je nach gewähltem Thema weiss man dann sowieso schon, was ein Symbol darstellen soll. Im Prinzip sind die Symbole nur für die örtliche Beschreibung der einzelnen Objekte zuständig, d.h. es könnten auch Punkte sein.

6.3.3 Museen, Theater, etc.

Unser Thema 'Sehenswürdigkeiten' in der Menüleiste besteht bislang aus den Unterthemen 'Gebäude', 'Denkmal' und 'Kunst'. Weitere Unterthemen, wie 'Museen', 'Theater', 'Ausstellungen' usw., haben wir in das bestehende Informationssystem nicht integriert, weil es dergleichen in Flüelen nicht gibt. Bei einer Erweiterung des Systems auf andere Gemeinden wäre aber eine Ergänzung solcher Unterthemen ohne Probleme möglich. Es müssten lediglich neue Einträge in die Datenbank erfolgen. Das gilt natürlich für alle anderen Themenbereiche auch.

6.3.4 Routenplaner

Ein Wanderroutenplaner ist aus zwei Gründen nicht Bestandteil unserer Arbeit. Der Eine ist, dass die Entwicklung eines solchen Planers den Rahmen der Arbeit gesprengt hätte. Der zweite

Grund ist, dass ein Wanderroutenplaner in einem früheren Vertiefungsblock am IKA bereits erstellt wurde [5], weshalb wir hier gerne auf diese Arbeit verweisen wollen.

6.3.5 Grundgerüst mit HTML

Für das Grundgerüst unserer Arbeit verwenden wir SVG, also nicht HTML. Der Grund dafür liegt zum einen darin, dass SVG, vor allem Vorteile im Bereich der Darstellung und Skalierung von Vektorgrafiken hat. Ein Problem bei der Verwendung von HTML als Grundgerüst wäre dessen Kombination mit SVG. Es gibt Browser, die mit dem Zusammenspiel in dieser Hierarchieform grosse Probleme haben, weshalb wir uns für die andere Variante entschieden haben.

Theoretisch könnte HTML ohne SVG in dieser Arbeit verwendet werden. Jedoch wäre dies sehr viel aufwendiger und Grafiken müssten z.B. alle in einem Grafikformat (JPEG, GIF, etc.) integriert werden. Dies würde die Datenmenge und damit auch die Rechenzeit enorm erhöhen. Eine Skalierbarkeit ohne Verzerrungen wäre nicht mehr möglich.

6.4 Probleme

6.4.1 Umlaute

Schon bei den ersten Versuchen, Inhalte aus der Datenbank in unserer Web-Seite darzustellen, stellen wir fest, dass die Umlaute nicht richtig decodiert werden. Stattdessen werden irgendwelche Sonderzeichen dargestellt. Nicht selten führt dies sogar zum Abbruch der Datenübertragung aus der Datenbank. Die Daten werden in der Konsequenz nur teilweise dargestellt oder es wird sogar der Fehler 404 (Page not found) angezeigt. Bei der Fehlersuche stellen wir fest, dass die Web-Seite in SVG, das Perl-Programm sowie die Datenbank mit Umlauten arbeiten können. Einzig die Kommunikation zwischen diesen Komponenten scheint das Problem zu sein.

Eine Ausnahme bildet ECMAScript, wo sämtliches Auftreten von Umlauten zu einem Absturz der Programmkette führt. Auch in Kommentaren zum Programm ist das Verwenden von Umlauten tabu. Es kommt dabei keine Fehlermeldung zurück, sondern das Ausführen des ECMAScripts ist einfach nicht möglich. Die Umlaute können lediglich in Variablen gespeichert werden, indem z.B. von der SVG-Seite ein Attribut abgefragt wird. Für die Kommunikation zwischen den verschiedenen Programmteilen versuchen wir zuerst die Codes und somit auch die Variablen, welche von einem Programmteil zum anderen überführt werden, nach dem gleichen Standard zu codieren. Zum einen ist dies der Standard ISO-8859-1, welcher für XML im deutschsprachigen Raum üblich ist. Zum anderen ist dies der Standard UTF-8, welcher für die meisten Sprachen und deren Sonderzeichen verwendet werden kann [8].

Der Standard wird einerseits im Header der Web-Seite festgelegt. Andererseits spielt es auch eine Rolle, in welcher Codierung der Code oder das Skript abgespeichert werden. Über die genauen Hintergründe haben wir uns nur ansatzweise informiert, da wir uns nicht sicher waren, ob dieser Weg zum Erfolg geführt hätte. Wir arbeiten nach der Trial-and-Error-Methode und ziehen verschiedene Leute des Instituts für Kartografie zu Rate.

Nach und nach können wir das Problem eingrenzen. Die Kommunikation zwischen Perl-Skript und Datenbank funktioniert unter ISO-8859-1 und für den Weg vom Perl-Skript zur Web-Seite werden die Umlaute entsprechend mit der Unicode-Codierung (z.B. ü entspricht ü) codiert. Einzig auf dem Weg von der Web-Seite zum Perl-Skript scheitern wir. Dies hat auch einen einfachen Grund: Die Parameter für die Datenbankabfrage werden mittels URL weitergegeben, welche ebenfalls Umlaute enthalten können. In der URL sind aber Umlaute und

auch die meisten Sonderzeichen nicht erlaubt [16]. Damit löst die Unicode-Codierung dieses Problem nicht.

Diese Sackgasse zwingt uns, uns nach einem anderen Lösungsweg umzuschauen. Wir entscheiden uns dafür, die Umlaute intern zu codieren und zwar indem dem Normallaut die Zahl neun vorangestellt wird. Aus 'ü' wird so z.B. '9u'. Sämtliche IDs von Objekten in der Web-Seite und auch von den Bildern, Symbolen etc., welche in einer URL erscheinen können, müssen auf Umlaute überprüft und gegebenenfalls ersetzt werden. Dies führt dazu, dass teilweise Namen im Perl-Skript doppelt geführt werden müssen. Beispielsweise ist in einer Variablen 'Flüelerhof' abgespeichert, um den Namen in der Auswahlliste darzustellen und in einer anderen 'Fl9uelerhof', um die IDs von SVG-Objekten zu schreiben oder Bilder aufzurufen. Daher enthalten alle Perl-Skripts Umwandlungs-Befehle, welche die Variablen eingangs umformatieren oder für den Output entsprechend codieren. Die Daten in der Datenbank sind aber ausgenommen von der '9'-Codierung.

6.4.2 Textdarstellung

Die Darstellbarkeit und Formatierung von Text erweist sich in SVG nicht als sehr komfortabel. Dies merken wir vor allem bei der geschichtlichen Beschreibung der einzelnen Gebäude, wo Text um ein Bild herum gelegt wird. Es gibt keine Möglichkeiten ein Textfeld zu definieren, mit welchem man die Form des Textes genau festlegen könnte. Das bedeutet in unserem konkreten Fall, dass der Text im Texteditor genau so zurechtgeschoben werden muss, wie man ihn später haben möchte, was eine relativ mühsame Angelegenheit ist. In der kommenden Version von SVG soll diesbezüglich mehr möglich sein [18].

6.4.3 Fehlende Tools in SVG

Wir haben während unserer Arbeit festgestellt, dass mit SVG vieles oder eigentlich fast alles möglich ist. Man muss jedoch auch praktisch alles selbst kreieren und erarbeiten, was oftmals sehr zeitaufwändig ist. Es wäre manchmal wünschenswert, wenn man für gewisse relativ banale Dinge, wie z.B. das Erzeugen eines Eingabefeldes oder Ähnliches, einfache vordefinierte Tools oder Befehle zur Verfügung hätte, wie dies bei HTML der Fall ist. Man hätte dann einfach mehr Möglichkeiten sich mit den wirklich wichtigen Dingen zu beschäftigen, anstatt unnötig Zeit zu verlieren, bei Sachen, die in anderen Anwendungen längst implementiert sind.

Es sind zwar sogenannte XForms in Entwicklung, welche in diesem Bereich Erleichterungen bringen sollen, jedoch funktionieren diese noch nicht [17].

6.5 Erweiterungsmöglichkeiten

6.5.1 Georeferenzierung der Karte

Der Vorteil einer georeferenzierten Karte wäre der direkte Bezug zwischen den Pixelkoordinaten und einem globalen Koordinatensystem. Positionsbasierte Dienste, die z.B. auf das Schweizerische Landeskoordinatensystem bezogen sind, könnten auf diese Weise leichter verwirklicht werden. Diese würden zwei Vorteile mit sich bringen. Zum einen könnten Koordinaten von Objekten abgefragt werden. Ausserdem könnten GPS-Messungen vor Ort direkt in der Karte angezeigt werden, was dem Benutzer seine Position, seinen zurückgelegten Weg, o.ä. angeben würde.

Die Kombination der beiden Möglichkeiten würde es, zusammen mit einem Wegnetz, zulassen, sich den kürzesten Weg zu einer Örtlichkeit anzeigen zu lassen.

6.5.2 Komplexere Navigation

Panning Das Verschieben der Karte funktioniert momentan mit den vier Pfeilen am Rand der Karte. Eine Verbesserung in diesem Bereich könnte man dadurch erreichen, dass eine Funktion entwickelt wird, bei der die Karte auf den Punkt zentriert wird, auf den man mit der Maus klickt. Mit Hilfe einer Koordinatenabfrage sollte dies kein grosses Problem sein, da die damit erhaltenen Koordinaten nur noch um einen gewissen Betrag verschoben werden müssen.

Zooming Ähnlich wie beim Panning sind auch beim Zooming noch Verbesserungen möglich. Beispielsweise könnte man eine Funktion erarbeiten, mit der man mit der Maus in der Karte ein Fenster beliebiger Grösse aufziehen kann, welches dann die neue Zoomeinstellung sein soll. Wichtig dabei wäre, dass durch diese freie Festlegung des Zoomfensters keine Verzerrungen der Karte entstehen. Die Bestimmung der Eckpunkte des Fensters ist ebenfalls mit einer Koordinatenabfrage möglich.

6.5.3 Erstellung eines übergeordneten Systems

Wir haben uns stets vor Augen gehalten, eine Anwendung zu schaffen, welche sich leicht auf andere Gemeinden übertragen lässt. Es liegt daher nahe ein System zu erstellen, welches mehrere Gemeinden enthält. Dabei wäre unsere Web-Seite ein Vorschlag für die Darstellung einer einzelnen Gemeinde. Es müsste eine Navigation geschaffen werden, welche es zulässt, die verschiedenen Gemeinden auszuwählen und dem Benutzer eine vernünftige Übersicht erlaubt. Im Bereich der Datenbank wären einige Ergänzungen nötig, damit die einzelnen Gemeinden identifiziert werden können. Interessant wäre in diesem Zusammenhang, die Datenbankabfrage gemeindeübergreifend zu gestalten. So könnten zum Beispiel im Hauptthema Aktivitäten nicht nur die Möglichkeiten innerhalb der Gemeinde aufgeführt werden, sondern auch diejenigen der näheren Umgebung.

Zu diskutieren gäbe die Organisation der Karte, welche der Seite zugrunde liegt. Eventuell würde es mehr Sinn machen eine zusammenhängende Karte zu hinterlegen, als für jede Gemeinde einen Ausschnitt aus der LK 1:25'000 einzuscannen. Für eine grössere zusammenhängende Karte wäre das Einscannen sicherlich nicht die optimale Lösung, da die Datenmenge den Rahmen einer vernünftigen Datenübertragung via Internet sprengen würde. Vielmehr müsste man sich überlegen, die Karte im Vektorformat einzufügen. Die Symbolisierung wäre zwar mit einem beträchtlichen Arbeitsaufwand verbunden, was sich aber auszahlen würde je umfangreicher das System gestaltet würde.

Eine Oberfläche, welche es erlaubt die Gemeinden zu verwalten, würde sich als nützlich erweisen. Vor allem, um eine neue Gemeinde einzuführen, aber auch um vorhandene Gemeinden zu verändern. Die Sammlung der Daten für die Datenbank könnte mit einer interaktiven Oberfläche erheblich erleichtert werden. Insbesondere wäre eine interaktive Platzierung der Symbole wünschenswert. Eine Verwaltung würde auch die Möglichkeit der Darstellung des Zustandes einer Gemeinde, wie Vollständigkeit, Aktualität, etc., implizieren.

6.6 Erweiterung auf andere Gemeinden

Da wir stets von einer einfachen Übertragung unserer Web-Seite auf andere Gemeinden sprechen, soll dieses Kapitel Aufschluss darüber geben, wie dies konkret bewerkstelligt werden kann. Folgende Ausführungen sollen als eine Art Anleitung verstanden werden, weshalb sich auch gewisse Teile wiederholen, welche in dieser Arbeit schon einmal angeschitten wurden.

Karte Bei der Übertragung der Seite auf eine andere Gemeinde muss logischerweise auch die entsprechende Karte dazu vorhanden sein. Die Möglichkeiten sind dabei vielfältig. Wir haben uns für ein JPEG-Format von relativ geringer Qualität entschieden, damit sich die Datenmenge in Grenzen hält. Es ist Vorsicht geboten bei grossen Datenmengen. Wir haben die Erfahrung gemacht, dass die Untermenüs beim Heruntergleiten zu stottern beginnen, wenn die Karte in einem zu grossen Format abgespeichert wird. Es ist auch möglich, eine Karte im Vektorformat, z.B. in SVG, einzufügen. Im SVG-File muss die entsprechende Referenz der Karte geändert werden. Hat die Karte eine viel grössere Ausdehnung als die unsere, ist zu überlegen, ob ein neuer Massstab eingeführt werden sollte. Die Seite bedient sich, den Zoomeinstellungen entsprechend, verschiedenen Massstäben. Einer geht von 0 bis 500 m und ein anderer von 0 bis 1 km. Diese werden durch das Attribut 'visibility' ein- und ausgeschaltet.

Hauptthemen Die Hauptthemen sind vorgegeben und sollten eigentlich nicht verändert werden. Trotzdem ist es mit nicht allzu grossem Aufwand möglich, die Namen zu ändern oder gar die Anzahl der Themen zu verändern. Gleich zu Beginn des Perl-Skripts sind die Themen in einem Array definiert. Unter diesen Namen müssen auch die Tabellen in der Datenbank gespeichert sein, da diese Themennamen (s. Kapitel 2.1) für die Datenbankabfrage verwendet werden.

Datenerhebung und Implementierung Die Datenerhebung ist zeitaufwändig, die Implementierung dagegen gestaltet sich einfach. Die Informationen, welche zu jedem Hauptthema gespeichert werden können, sind von uns vorgegeben und in den Tabellen auf Seite 18 ersichtlich. Die Attribute sollten selbsterklärend sein. Theoretisch können auch weitere Attribute hinzugefügt werden, jedoch müsste dann auch das Perl-Skript 'infoFuell.pl' entsprechend nachgeführt werden. Ansonsten sind die Informationen vorhanden, werden aber nicht im Informationsfenster dargestellt. Die Attribute 'id' und 'name' kommen in jeder Tabelle vor und spielen eine wichtige Rolle in der Datenbankabfrage. 'Id' entspricht dem Unterthema und 'name' ist das Attribut, welches das Element identifiziert. Es ist also Vorsicht geboten, falls z.B. zwei Restaurant Sternen in einem Ort existieren.

In der Tabelle 'Verkehrsmittel' sind die Attribute 'intFahrplan' und 'SBBFahrplan' speziell zu erwähnen. 'intFahrplan' enthält die Web-Adresse des internen Fahrplanes des jeweiligen Transportunternehmens. 'SBBFahrplan' informiert mit 'ja' oder 'nein' über das Vorhandensein der Haltestelle im SBB-Fahrplan.

Etwas mühsam dürfte sich die Erhebung der Koordinaten der Symbole gestalten. Wir haben die Symbole probeweise in das 'fluelen.svg'-Dokument eingefügt und uns so an die richtigen Koordinaten herangetastet. Es wäre vielleicht lohnenswert, die Karte zu georeferenzieren und die Koordinaten aus einer LK 1:25'000 herauszulesen.

Zu den Sehenswürdigkeiten wird im Ordner Daten-Geschichte jeweils zu jeder Möglichkeit ein Text abgespeichert. Die Zeilenumbrüche müssen dabei manuell eingefügt werden. Für das richtige Einfügen der Zeilenumbrüche dient am besten ein Beispiel unserer Seite als Vorlage.

Für das Unterthema 'Wandern', welches als einziges speziell behandelt wird, kann neben dem Symbol auch ein Pfad abgespeichert werden. Dieser wird in einem *name*.txt-File in der Form

```
d="M 1295.2116,5576.5985 L 1328.8229,5619.8132 ... L 2269.9414,6162.3968"
```

abgespeichert, wobei die Zahlenpaare Koordinatenpaare bilden, bezogen auf das Bildkoordinatensystem. Diese *name*.txt-Files sind am gleichen Ort abgespeichert wie die Symbole des

Hauptthemas 'Aktivitäten'. *name* entspricht dabei dem Namen, der in der Datenbank in der Spalte 'name' steht. Zur Digitalisierung der Wanderwege empfiehlt sich das Programm 'inkscape'. Dieses Programm erlaubt es, die Koordinaten direkt in den entsprechenden Bildkoordinaten abzuspeichern, im Gegensatz zu anderen Programmen, wie z.B. Adobe Illustrator. Ausserdem wird beim Unterthema 'Wandern' anstatt einem Informationsfenster direkt die Web-Adresse angezeigt, welche in der Datenbank in der Spalte 'www' steht.

Bei Nachführung der Datenbank empfehlen wir, die Daten z.B. in einem Excel-Sheet einzutragen und anschliessend die Daten in ein Text-File zu exportieren, wobei die Spalteneinträge mit Tabulatoren getrennt sind. Dieses Text-File kann mit dem Befehl

```
load infile \textit{path} into table \textit{tablename};
```

in die MySQL-Datenbank eingelesen werden. Was in den einzelnen Feldern steht ist egal, solange es nicht die Vorgaben der Datenbank verletzt, z.B. zu lange Zeichenketten. Die Wahl der Unterthemen ist freigestellt, sie werden automatisch in den Menüs angezeigt.

Symbole Für jedes Unterthema wird ein separates Symbol dargestellt. Diese sind in einem Ordner 'Symbole' abgespeichert. Dieser Ordner enthält aus Übersichtsgründen für jedes Hauptthema einen Unterordner. Zu jedem Unterthema wird im entsprechenden Unterordner ein Bild mit dem Namen *unterthema*.gif gespeichert. Wird ein anderes Format gewünscht, müsste das Perl-Skript 'layerFuell.pl' angepasst werden. Für Unterthemen, welche Umlaute enthalten, müssen diese in unserer '9'-Codierung abgespeichert werden. Aus 'Gebäude' wird z.B. 'Geb9aude'. Dasselbe gilt übrigens für die Unterordner. 'Aktvitäten' und 'Sehenswürdigkeiten' heissen also 'Aktivit9aten' und 'Sehensw9urdigkeiten'. Die Gründe können in Kapitel 6.4 'Probleme' auf Seite 33 nachgelesen werden.

Bilder In jedem Informationsfenster ist es möglich ein Bild anzuzeigen. Die Ordnung der Speicherung ist ungefähr dieselbe wie bei den Symbolen. Im Ordner 'Bilder' gibt es, den Hauptthemen entsprechend, veschiedene Unterordner. In diesen wiederum sind die Bilder mit *name*.jpg gespeichert, wobei *name* dem Datenbankeintrag der Spalte 'name' entsprechen muss. Auch hier wird unsere '9'-Codierung verwendet.

Rahmeninformationen Schliesslich müssen alle Darstellungen geändert werden, welche spezifisch für die Gemeinde Flüelen gelten. Dazu gehört das Wappen, die Überschrift und der Link zur Webcam. Ausserdem muss das Impressum angepasst werden. Diese Änderungen sind alle im .svg-File vorzunehmen.

7 Browserkompatibilität

Die Möglichkeiten SVG darzustellen sind ziemlich umfangreich. Obwohl SVG ein W3C-Standard ist, gibt es wenige Browser, welche SVG standardmässig interpretieren können. Normalerweise muss zusätzlich ein SVG-Plugin installiert werden. Wir haben einige Möglichkeiten getestet und dazu den Adobe SVG Viewer 3.02 verwendet, welcher auf [9] heruntergeladen werden kann. Andere Viewer, z.B den Corel SVG Viewer, haben wir nicht getestet.

7.1 Windows XP

7.1.1 Internet Explorer und Adobe SVG Viewer

Während der Entwicklungsphase dieses Informationssystems haben wir permanent den Internet Explorer verwendet, um die Seite zu testen. Aus diesem Grund ist das System auf diesen Browser ausgerichtet, was heisst, dass damit alle Anwendungen funktionieren.

7.1.2 Netscape Navigator und Adobe SVG Viewer

In diesem Browser kommt unser Informationssystem zu laufen, sobald die nötigen SVG-Plugins vorhanden sind. Bis auf zwei Ausnahmen funktioniert die Seite wie sie sollte. Dabei handelt es sich zum einen um die Suchfunktion, die nicht klappt, weil sie in ECMAScript unter anderem mit dem 'prompt'-Befehl ausgelöst wird und dieser nicht von allen Browsern erkannt werden kann. Zum anderen handelt es sich um das Unterthema 'Gebäude' im Hauptthema 'Sehenswürdigkeiten'. Hier erscheint der Titel 'Gebäude' bei Mausklick zwar links neben der Karte, aber nicht die zugehörige Liste, die alle Gebäude enthält. Der Grund dafür ist wohl der Umlaut in diesem Namen, der Schwierigkeiten macht.

7.1.3 Mozilla Firefox und Adobe SVG Viewer

Auch mit diesem Browser läuft unsere Seite mit den entsprechenden SVG-Plugins. Ansonsten trifft für Mozilla Firefox exakt das Gleiche zu wie für Netscape Navigator. Die Mozilla-Version mit eingebettem SVG habe wir nicht getestet.

7.1.4 Opera und Adobe SVG Viewer

Mit den nötigen SVG-Plugins kann auch Opera als Browser verwendet werden. Es treten wiederum dieselben Probleme auf wie bei Netscape Navigator und Mozilla Firefox, mit dem Unterschied, dass der Titel 'Gebäude' links zwar auch dargestellt wird, jedoch nicht mehr richtig geschrieben ist.

7.2 MacOS

Bei MacOS fällt auf, dass die Schriftgrössen nicht mit denen von Windows übereinstimmen und zwar für alle Browser. Es wird nun auf die Versionen MacOS_9 und MacOS_10 eingegangen.

7.2.1 MacOS_9

Internet Explorer und Adobe SVG Viewer Obwohl das Informationssystem auf Windows für den Internet Explorer ausgerichtet ist, gibt es bei MacOS_9 dennoch Probleme. Die Web-Seite an sich funktioniert, jedoch tauchen die schon bekannten Fehler bei der Suchfunktion und beim Unterthema 'Gebäude' auf.

Netscape Navigator und Adobe SVG Viewer In diesem Browser funktioniert das Informationssystem nicht wie es sollte, weil vermutlich die ECMAScript-Funktionen nicht ausgeführt werden. Dies erkennt man daran, dass die Menüleiste beim Laden nicht erscheint und die Karte nicht das gewünschte Startbild zeigt.

7.2.2 MacOS_10

Safari, Internet Explorer, Netscape Navigator, Mozilla jeweils mit Adobe SVG Viewer In all diesen Browsern funktioniert die Web-Seite wiederum bis auf die Suchfunktion und das Unterthema 'Gebäude'.

7.3 Batik

Batik ist ein Java-basierter Browser, der auf allen Systemen funktionert, welche Java unterstützen. Dieser Browser erscheint uns empfindlicher als die oben Aufgeführten. Während die anderen Browser einen Fehler im CSS-File ignorierten, konnte die Seite in diesem Browser nicht dargestellt werden, bis der Fehler korrigiert war. Die Suchfunktion funktioniert wiederum nicht. Ausserdem wird die Animation, welche die Untermenüs heruntergleiten lässt, nicht ausgeführt, was das weitere Austesten der Web-Seite verunmöglicht.

8 Ergebnis

Das Ergebnis unserer 14-wöchigen Arbeit ist ein funktionstüchtiges, mit reichhaltigen Daten bespicktes, Informationssystem für Touristen für die Gemeinde Flüelen UR.

Da wir uns während unserer Arbeit stark am Grundkonzept orientiert haben, erreichen wir die dort gestellten Anforderungen. Funktionalität und Darstellung werden wie geplant umgesetzt. Aus unserer Sicht sollte sich jeder, ob Laie oder Spezialist, problemlos in der Seite zurechtfinden. Die in Kapitel 1 angesprochene Vision der Erweiterbarkeit ist, soweit wir das beurteilen können, auch erreicht.

Die Bedeutung unserer Arbeit im Kontext der Web-Kartographie ist nicht revolutionär. Viele Anwendungen sind bereits in anderen Arbeiten implementiert. Doch zu billig wollen wir uns nicht verkaufen. Eine Menüliste, die durch eine Animation langsam herunterfährt, sowie ein Massstabsbalken, der sich stets dem aktuellen Massstab anpasst, haben wir noch nirgends gesehen.

Aus persönlicher Sicht sind wir mit unserer Arbeit und dem erreichten Ergebnis sehr zufrieden. Es war durchaus interessant und abwechslungsreich, sich ein Semester lang mit dieser Thematik zu beschäftigen. Das erworbene Wissen wird uns für eventuelle zukünftige Arbeiten ein guter Grundstein sein.

Literatur

[1] Gartner, G.: LOCATION BASED SERVICES & TELEKARTOGRAPHY, Proceedings of the Symposium 2004, Technische Universität Wien, Wien 2004

[2] Schnabel, O., Neumann, A., Dahinden, T.: Web-Kartographie: Einführung in HTML und CSS. Vorlesungsskript. Institut für Kartographie, ETH Zürich 2004

[3] Schnabel, O., Neumann, A., Dahinden, T.: Web-Kartographie: Einführung in SVG. Vorlesungsskript. Institut für Kartographie, ETH Zürich 2004

[4] Schnabel, O., Neumann, A., Dahinden, T.: Web-Kartographie: Einführung ins Scripting. Vorlesungsskript. Institut für Kartographie, ETH Zürich 2004

[5] Spirig, S., Wegmann, M., Zogg, H.-M.: Integration von GIS-Funktionen in einen interaktiven Wanderroutenplaner. Vertiefunsblock. Institut für Kartographie, ETH Zürich 2003

[6] Ziegler, J., Programmieren lernen mit Perl, Springer-Verlag 2002

[7] http://de.selfhtml.org

[8] http://de.selfhtml.org/html/referenz/zeichen.htm

[9] http://www.adobe.com/svg/viewer/install/main.html

[10] http://www.carto.net/papers/svg/samples

[11] http://www.galileocomputing.de

[12] http://www.ika.ethz.ch (Studentenarbeiten)

[13] http://www.mysql.de

[14] http://www.m-vp.de/campingkarte.htm

[15] http://www.sbb.ch/fahrplan

[16] http://www.w3.org/Adressing/rfc1738.txt

[17] http://www.w3.org/MarkUp/Forms

[18] http://www.w3.org/TR/SVG12