MIGRATION ZWISCHEN DER SCHWEIZ UND EUROPA VON 1981 BIS 1999

Eine interaktive und dynamische Karte erstellt mit SVG und JavaScript

Christian Joray
Betreuung: Andreas Neumann
Leitung: Prof. Dr. Lorenz Hurni

2. Februar 2001

Inhaltsverzeichnis

Zusammenfassung

Das Ziel dieser Diplomarbeit war die Erstellung einer interaktiven und dynamischen Internet-Karte mit dem Thema “Migration zwischen der Schweiz und Europa von 1981 bis 1999”. In erster Priorität sollte dabei mit den verwendeten Techniken, vor allem mit SVG (*scalable vector graphics*), experimentiert werden. Die kartographische Umsetzung des Themas war ein weiterer wichtiger Hauptbestandteil der Arbeit.

The goal of this diploma work was the creation of an interactive and dynamic internet-map “migration between switzerland and europe from 1981 to 1999”. The main aspect was to experiment with the used techniques, especially SVG (*scalable vector graphics*), nevertheless the theme’s cartographic implementation was another important part of this diploma work.

Die Karte ist online unter folgender URL zu betrachten:

http://www.karto.ethz.ch/~jorayc/migration/

Dieser Bericht ist im pdf-Format auf der Homepage des Instituts für Kartographie online verfügbar:

http://www.karto.ethz.ch/

Unter der gleichen URL können auch alle zum Projekt gehörenden Dateien in einem zip-File heruntergeladen werden.

Kapitel 1

Einleitung

1.1 zum Thema

Karten sind ein Spiegel der Zeit. Sie vermitteln verschiedenste Themen und Problemstellungen, aber auch Techniken und Medien, die zu ihrer Entstehungszeit zur Verfügung standen. Der Beginn des Computerzeitalters ist daher auch ein Meilenstein in der Geschichte der Kartographie. Ein neues Medium konnte für die Erstellung und Präsentation von Karten in vielen Variationen verwendet werden.
Als Folge der immer grösseren Vernetzung zwischen Computern und der vorhandenen Nachfrage nach Möglichkeiten, über Computer miteinander zu kommunizieren, wurde das Internet aufgebaut. Mit der steigenden Popularität des Internets durch den immer einfacheren Zugang in den letzten Jahren, drängte sich seine Verwendung als Plattform für die Kartographie geradezu auf. Allerdings sind immer noch einige Probleme zu bewältigen. Grosse Grafiken zum Beispiel sind nicht geeignet fürs Internet, da es auf Grund der oft geringen Übertragungsgeschwindigkeit viel zu lange dauert bis der Benutzer diese betrachten kann. Grosse und detaillierte Karten ziehen aber ohne Zweifel grosse Dateien mit sich. Dies betrifft vor allem Rasterformate, welche immer noch die gängigen Formate zur Anzeige von Grafiken im Internet sind. Eine Lösung wäre in diesem Fall natürlich die Verwendung von Vektorformaten zur Anzeige. Dadurch würden nicht nur die Dateigrössen verkleinert, es könnten auch relativ einfach Zoom- und Pan-Funktionen ohne Qualitätsverlust in der Anzeige angewendet werden.

Bis vor kurzem verfügte das Internet über keine ausgereiften Möglichkeiten, Vektorformate sinnvoll anzuzeigen, ausser vielleicht mit Hilfe eines Java-Applets. Mittlerweile ist aber ein Trend in Richtung Vektorgrafiken entstanden. In diesem Zusammenhang ist hier sicher Macromedia's Flash [20] zu erwähnen. Für kartographische Zwecke eignet sich dieses Tool aber nur in begrenztem Masse. Ein grosse Innovation für die Kartographie und allgemein für Vektorgrafiken war aber die Realisierung des SVG-Formates (*scalable vector graphics*) [6] mit den dazugehörigen Browser Plugins[1] und auch den Exportfiltern für die Grafikprogramme Adobe Illustrator[1] und Corel Draw[2]. SVG ist ein Format, das auf XML basiert. Die grafischen Informationen werden in Text/Unicode gespeichert (Kapitel 2.1).

1.2 Ziele

Das Ziel dieser Diplomarbeit war das Erstellen einer dynamischen, interaktiven und SVG-basierten Internet-Karte zum Thema Migration von Europa bezüglich der Schweiz in den Jahren 1981 bis 1999. Der Schwerpunkt der Arbeit bestand darin, die Möglichkeiten von SVG in einem solchen Rahmen zu testen, aber auch Probleme und Grenzen dieser Technik zu erörtern. Dennoch galt es natürlich die Gestaltung der Karte auf das Thema abzustimmen und dieses möglichst optimal umzusetzen. Interessant ist auch die Fragestellung, wie eine Online-Karte sich von einer Papier-Karte zu unterscheiden hat, zum Beispiel bezüglich Anordnung der Elemente, der Gestaltung der Legende oder auch der benutzten Farben und Schriften. Da eine Online-Karte meist interaktiv ist, muss der Entwickler sich Gedanken über eine Benutzerführung machen. Das Ziel sollte sein, dass der Benutzer sich ohne grosse Mühe auf der Karte schnell und intuitiv zurechtfindet und die implementierten Features für seine Zwecke nützen kann. Schlussendlich war ein weiteres Ziel, dass die Karte auf den beiden meistbenützten Browsern, Microsofts Internet Explorer und Netscapes Navigator läuft.

1.3 zur Arbeit

Nach 8 Semestern Studium als Vermessungsingenieur an der ETH Zürich folgt zum Abschluss des Studiums das Diplomsemester. Den Diplomanden stehen für die Bearbeitung eines grösseren Projektes 12 Wochen zur Verfügung. Diese Diplomarbeit wurde vom Institut für Kartographie an der ETH Zürich unter der Leitung von Prof. Dr. Lorenz Hurni angeboten. Ich entschied mich für diese Arbeit, weil ich mich für viele Aspekte des Projekts interessierte. Zum Beispiel sind das der Bezug zum Internet, die Programmierung, die Entwicklung eines GUI und die kartographische Umsetzung/Visualisierung eines aktuellen Themas.

1.4 Dank

Ich danke meinem Betreuer Andreas Neumann, seine sehr guten Kenntnisse in den Bereichen Informatik und Kartographie halfen mir einige Male aus unliebsamen Situationen heraus. Mein Dank gilt auch André M. Winter, die von mir verwendeten grafischen Daten zu Europa stammen aus seinem SVG-Projekt. Die Diplomarbeit stand unter der Leitung von Prof. Dr. Lorenz Hurni, ich möchte auch ihm für die Unterstützung danken. Weiter gilt mein Dank auch René Sieber, Leiter des Atlas der Schweiz, er half mir bei der Suche nach geeigneten Daten und verschaffte mir die Kontakte zum Bundesamt für Statistik. Von dort stammen die verwendeten Daten. Darum möchte ich an dieser Stelle auch Herr Hans Steffen und Frau Esther Salvisberg vom Bundesamt für Statistik für ihre Bemühungen danken.

Kapitel 2

Grundlagen

2.1 XML und SVG

In einem ersten Schritt wird XML (*eXtensible Markup Language*) erläutert, XML ist die Grundlage für SVG (*scalable vector graphics*). Die Prinzipien von XML werden dann direkt anhand von Beispielen mit SVG gezeigt.

2.1.1 XML

XML ist die Abkürzung für *eXtensible Markup Language*. Es ist der universelle Standard für netzwerktransparente und plattformunabhängige, strukturierte Dokumente für das WWW und darüber hinaus als Austausch zwischen Applikationen aller Art. Die Entwicklung von XML begann 1996 und wurde vom W3C (*world wide web consortium*) [3] 1998 als Standard aufgenommen. XML enstand aus SGML (*Standard Generalized Markup Language, ISO 8879*), das schon in den 80-er Jahren als ISO-Standard für die Beschreibung von Struktur und Inhalt für elektronische Dokumente verschiedener Art entwickelt wurde. Das weit verbreitete HTML (*Hyper Text Markup Language*) [14, 4] ist zum Beispiel eine spezifische Anwendung von SGML. Auf der Grundlage von SGML und mit Berücksichtigung der Erfahrungen aus dem Gebrauch von HTML entstand schliesslich XML als Basis aller zukünftigen WWW-Dialekte.

Mit dem XML-Standard werden strukturierte Daten im Textformat gespeichert. Im Gegensatz zu binär gespeicherten Daten benötigen XML-Dateien mehr Speicherplatz. Dieser Nachteil wurde von den Entwicklern aber bewusst in Kauf genommen. Denn so können XML-Daten in jedem Texteditor betrachtet, editiert und korrigiert werden, unabhängig von der Anwendung, die das File kreierte. Ausserdem lassen sich XML-Dateien mit Kompressionsverfahren um ein Vielfaches komprimieren, zum Beispiel mit gzip [21]. Durch XML werden bekannte Probleme wie Erweiterbarkeit, Internationalisierung oder Plattformunabhängigkeit gelöst. Zusätzlich eignet sich XML für die langfristige Speicherung von Informationen, die Lesbarkeit von XML-Daten ist immer garantiert. Auf der Basis von XML können im Prinzip beliebig viele, den eigenen Bedürfnissen angepasste Standards entwickelt werden. Dies erlaubt zum Beispiel Organisationen oder Gruppen von Leuten mit ähnlichen Interessen ihre eigenen Standards zu entwickeln und mit speziellen Applikationen Informationen untereinander auszutauschen. Neue Standards können dem W3C vorgeschlagen und als offizielle

Standards abgenommen werden. Wichtige Vertreter dieser auf XML basierenden Standards sind zum Beispiel SVG (Vektorgrafiken), SMIL (Multimedia), X3D (3-D Grafik), MathML (Mathematische Sonderzeichen und Formatierungen), CML (Chemie) oder GML (Geographische Informationssysteme GIS).

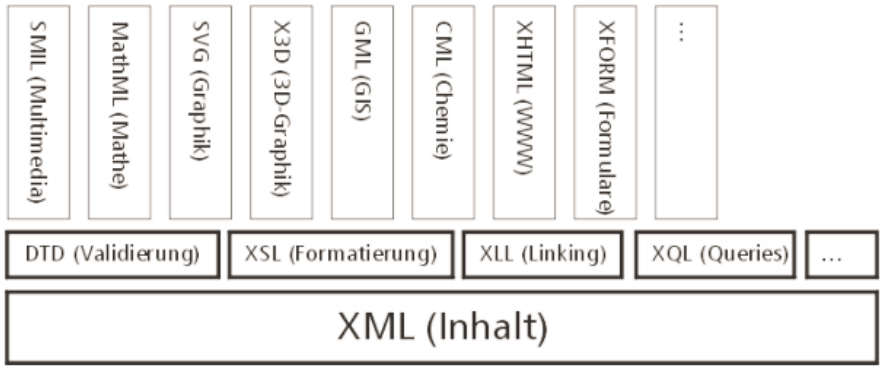


Abb. 2.1: Struktur von XML

Ein XML-File besteht aus einer Serie von *entities*, gemeint sind damit einzelne Objekte. Jedes dieser Objekte kann eines oder mehrere Elemente beinhalten. Alle Objekte und Elemente können mit Attributen ergänzt werden, die beschreiben wie die Informationen verarbeitet werden sollen. Um die XML-Anweisungen vom eigentlichen Inhalt zu unterscheiden, wird mit sogenannten *markup tags* gearbeitet. Die XML-Elemente und Attribute werden in spitzen Klammern notiert, wie man das zum Beispiel von HTML kennt. Im Gegensatz zu HTML muss XML-Code strengeren Regeln genügen, er muss zumindest *well-formed* sein. HTML-Code wird vom Browser auch dann interpretiert wenn einzelne Anweisungen fehlen. Zum Beispiel wenn eine Tabellencodierung nicht mit `</table>` abgeschlossen wird, kann es gut sein, dass die Tabelle trotzdem angezeigt wird. Bei XML hingegen werden solche Fehler nicht akzeptiert. XML-Code ist *well-formed*, wenn alle Tags geschlossen sind, die Verschachtelung stimmt und mindestens ein *root* -Element (z. B. `<svg>...</svg>`) vorhanden ist. Dies entspricht einer Syntax-Prüfung.

Die Validität des XML-Dokuments wird über die dazugehörige DTD (*document type definition*) oder ein XML-*Schema* [9] geprüft. XML-Schema ist erst eine *candidate recommendation* des W3C. Gegenüber einer DTD erlaubt ein *Schema* zusätzliche Datentypendeklarationen und weitergehende Kontrollmöglichkeiten. Eine DTD kann folgende Vorschriften beinhalten:

- Liste aller gültigen Elemente
- Liste aller gültigen Attribute
- gültige Notationsweise
- *Entity* Definitionen
- gültige Verschachtelung von Elementen
- Element verlangt, empfohlen oder optional

- wie oft darf Element oder Attribut vorkommen

Hat ein XML-Dokument alle Prüfungen der dazugehörigen DTD oder des XML-*Schemas* bestanden, erreicht es den Status *valid*. Im Gegensatz zu SGML muss XML nicht unbedingt Zugriff zu der entsprechenden DTD haben. Falls keine vorhanden ist, kreiert das XML-System eine Standard-DTD für die unbekannten Komponenten.
Laufend werden die Komponenten rund um XML erweitert, zum Beispiel gibt es bereits einen Nachfolger der CSS (*cascading style sheets* [10]), nämlich XSL (*XstyLe Sheets*). Weitere Stichworte in diesem Zusammenhang sind XML Protocol, XML Query, XPath, XPointer, XML Base, XLink, XML Signature, RDF [5, 13].

2.1.2 SVG

SVG ist eine XML-Variante für 2D-Vektordaten, Text, Grafik-Filter und die Einbindung von Rasterdaten. Basierend auf diesem Dateiformat können verschiedenste Applikationen grafische Information speichern und darstellen. Anfang des Jahres 1999 wurde die erste Arbeitsgrundlage (*working draft*) veröffentlicht. Das Ziel ist, dass sich SVG als globaler Standard etablieren kann. Mittlerweile hat SVG beim W3C mit der SVG 1.0 Spezifikation vom 2. August 2000 den Status einer *candidate recommendation*. Das W3C rechnet aber mit einer offiziellen Freigabe für SVG in den nächsten Monaten.

Mit SVG können grafische Objekte gruppiert, formatiert, transformiert und animiert werden. Die Gruppierung von Objekten entspricht etwa dem Arbeiten mit mehreren Layern in einem Grafikprogramm. Jede Gruppe ist über ein `id` -Attribut direkt ansprechbar. Die Formatierungen bieten sehr viele Möglichkeiten, es kann im Prinzip jedes Attribut aus den bekannten Grafikprogrammen implementiert werden. Erwähnenswert sind die Transparenzen und Filtereffekte, speziell ist sicher die Möglichkeit der Filterung von Vektorobjekten. Ausserdem können alle Formatierungen in einem CSS vordefiniert werden. Änderungen in der Formatierung sind auf diese Art sehr einfach und effizient möglich.

Für die Kartographie von speziellem Interesse sind sicher auch die Transformationsmöglichkeiten. SVG arbeitet mit einem eindeutigen Koordinatensystem. Grafische Objekte werden über ihre absoluten oder relativen x,y-Koordinaten plaziert und definiert. Jedem Objekt kann als Attribut zusätzlich eine eigene Transformation zugewiesen werden. Im Prinzip werden diese durch Translation, Skalierung und Rotation definiert, es gibt aber auch die Möglichkeit freie Transformationen mittels einer Matrix zu definieren.

Das Internet ist ein dynamisches Medium. Darum wurde in SVG auch die Möglichkeit implementiert, die grafischen Objekte zu animieren. Animationen in SVG sind folgendermassen möglich [6]:

- SVG bietet eine Vielzahl von eigenen Animations Elementen.
- Über das SVG DOM (*document object model*, Kapitel 2.2). SVG entspricht den wichtigsten Spezifikationen des DOM Level 1 und DOM Level 2. Jedes Attribut und Styleattribut kann über Scriptsprachen angesprochen und animiert werden.
- SVG wurde so designt, dass es mit zukünftigen Versionen von SMIL (*Synchronized Multimedia Integration Language*) kompatibel ist. SVG-Inhalte sollen in Zukunft auch als eigenständige Multimedia Komponenten gebraucht werden können.

SVG wurde darauf ausgerichtet dynamische und interaktive Inhalte zu präsentieren. Im SVG-Standard wurde deswegen das volle XML DOM (Kapitel 2.2) implementiert. Auf diese Weise können SVG-Objekte nicht nur animiert werden, sie können auch eine grosse Zahl von *event handlern* (z. B. *onmouseover*, *onclick*) auslösen. Diese können im Prinzip jedem SVG-Objekt zugewiesen werden und via der Skriptsprache kann auf diese Benutzeraktivitäten reagiert werden.

2.1.3 Aufbau von SVG

Der Aufbau einer SVG-Datei ist einfach und verständlich. Jede SVG-Datei beginnt mit einem Header. Zuerst wird der Inhalt der Datei als XML deklariert. Danach folgt die Angabe der XML-Applikation, in diesem Fall SVG, mit der dazugehörigen DTD. Eine Beschreibung des Inhalts kann innerhalb der `<desc>`-Tags erfolgen.

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN"
  "http://www.w3.org/TR/2000/
    CR-SVG-20001102/DTD/svg-20001102.dtd">
<desc> <!-- Beschreibung --> </desc>
```

Wie bei HTML folgt nun der einleitende Tag, bevor der eigentliche Inhalt beginnt. Innerhalb dieses `<svg>`-Tags können schon einige Attribute gesetzt werden, zum Beispiel die Breite und Höhe des Inhalts in beliebigen Masseinheiten, der sichtbare Ausschnitt oder ob die Zoom- und Pan-Elemente des Browser-Plugins verfügbar sein sollen.

```
<svg width="11425" height="8275" xml:space="preserve"
     viewBox="-6148 0 11425 8275"
     enableZoomAndPanControls="false">
```

Bevor die Definitionen der grafischen Objekte folgen, besteht die Möglichkeit globale Formatierungen vorzunehmen und Referenzen zu deklarieren. Zum Beispiel werden im nächsten Code-Ausschnitt CSS-Formatierungen definiert. Es ist auch möglich hier Filtereffekte zu definieren und Bilder, Symbole und Schriften zu referenzieren [7].

```
<defs>
  <style type="text/css">
   <![CDATA[
    .str1 {stroke:#131516;stroke-width:4}
    .str0 {stroke:#1F1A17;stroke-width:4}

    .fil2 {fill:#4180BC}
    .fil0 {fill:#C1B6A6; fill-opacity:0.8;}

    .fnt1 {font-weight:bold;font-size:500;
           font-family:"Humanist777BT-RomanB"}
    @font-face{font-family:"Humanist777BT-RomanB";
               src:url(humanist777bt.cef)}
   ]]>
  </style>
 </defs>}
```

Nach allen Headerinformationen und den Formatierungen folgen die grafischen Informationen. Über `<g>..</g>` Tags können die Objekte in Gruppen eingeteilt werden. Die Objekte werden in der Reihenfolge gezeichnet, wie sie im Text definiert wurden. Das zuletzt definierte Objekt überdeckt demzufolge alle anderen. Formatierungen und Transformationen können sowohl einem einzelnen Objekt, als auch einer ganzen Gruppe zugewiesen werden. Mit dem `class` -Attribut werden CSS-Formatierungen zugewiesen, das `style` -Attribut ermöglicht neue Formatierungen und mit dem `transform` -Attribut werden Transformationen definiert. Ausserdem kann dem Element oder der Gruppe mit einem `id` -Attribut eine eindeutige Identifikation zugewiesen werden. Über das DOM (Kapitel 2.2) kann dieses Objekt dann zum Beispiel mit JavaScript (Kapitel 2.3) angesprochen und beeinflusst werden. Ebenfalls als Attribut der Elemente und Gruppen werden eventuelle *event handler* (z. B. `onmouseover`) definiert. In der *SVG candidate recommendation* [7] im Anhang B ist die genaue Auflistung der unterstützten *event handler* zu finden.

```
<g id="land?20" class="fil2 str1"
   onmouseover="showLand('land?20')"
   transform="scale(2) rotate(30) translate(100 200)">
<path style="fill:#B539DF;"
      d="M-1107 5524l18 21 -3 28 -21 -4c-2,-18 2,-32 6,-45z"/>
<circle style="stroke-width:12;" cx="50" cy="50" r="25"/>
<text id="landName" x="-6070" y="8180" class="fnt1">
      Schweiz
</text>
</g>
</svg>
```

Im Beispiel oben wird zuerst eine Gruppe mit dem `id`-Attribut `land?20` und den CSS-Formatierungen `fil2` und `str1` definiert. Der Kontakt der Maus mit einem Element dieser Gruppe (`onmouseover`) bewirkt die Ausführung einer Skript-Funktion `showLand()`. Alle Elemente dieser Gruppe unterliegen einer Transformation, sie werden zuerst in x-Richtung um 100 Einheiten und in y-Richtung um 200 Einheiten verschoben, dann um 30 Grad rotiert und dann noch um den Faktor 2 skaliert.

Es folgen zwei SVG-Elemente, ein Pfad und ein Kreis. Beide haben ein zusätzliches `style` -Attribut. Diese haben eine höhere Priorität gegenüber den schon für die ganze Gruppe definierten Formatierungen. Das Attribut `d` im Pfad-Element leitet die eigentliche Definition des Pfades ein. An dieser Stelle werden nur die oben benutzten Pfadcodes erklärt, für weitere Information wird auf die SVG 1.0 Spezifikation [7] verwiesen. Dort werden alle gültigen Elemente ausführlich beschrieben. Für Pfadanweisungen gilt allgemein, dass Grossbuchstaben auf absolute Koordinatenangaben verweisen und Kleinbuchstaben auf relative.

Im Beispiel wird zuerst mit der **M**`ove` -Anweisung der Startpunkt bei (-1107, 5524) festgelegt. Mit **l**`ine` wird danach eine Folge von drei Linien gezeichnet. Die Anweisung **l**`ine` muss nur beim ersten Mal geschrieben werden. Die zweite Linie geht zum Beispiel vom Punkt (-1089,5545) zum Punkt (-1086, 5573). Nach diesen Linien folgt mit **c**`urve` eine kubische Bézierkurve vom aktuellen Punkt zu einem um 6 Einheiten in x-Richtung und -45 Einheiten in y-Richtung verschobenen Punkt. Die ersten vier Zahlen nach der **c**`urve` -Anweisung definieren die zum aktuellen Punkt relativen Kontrollpunkte für die Bézierkurve. Mit der **z** -Anweisung am Schluss der Pfaddaten wird errreicht, dass der Pfad automatisch geschlossen wird. Schliesslich wird das

Pfadelement noch mit einem Schrägstrich `/` abgeschlossen. Das gleiche könnte mit einem abschliessenden `</path>` -Tag erreicht werden. Das Abschliessen des Pfadelements ist aber unbedingt nötig, sonst verletzt das Element die XML-Regeln und der Code ist nicht mehr *well-formed*. Mit Pfadanweisungen könnten eigentlich alle möglichen Formen von grafischen Objekten beschrieben werden. In der SVG Spezifikation wurden dennoch die wichtigsten Grundformen (*basic shapes*) implementiert, nämlich Rechteck, Kreis, Ellipse, Linie, Polylinie und Polygon.

Die Definition eines Kreises ist verständlicher und simpler aufgebaut als die des Pfades. Dadurch lassen sich mit den *basic shapes* einfachere Grafiken auch problemlos direkt im SVG-Code erstellen. Mit den Attributen `cx`, `cy` und `r` wird ein Kreis mit dem Mittelpunkt (50,50) und dem Radius 25 Einheiten gezeichnet.

Der `<text>` -Tag definiert einen Text. Plaziert wird dieser über x- und y-Koordinaten. SVG bietet auch Möglichkeiten einzelne Abschnitte eines Textes gegenüber dem Rest zu verschieben (`<tspan>`) oder den Text entlang einem definierten Pfad laufen zu lassen.

Die Gruppe von Elementen muss natürlich mit einem `</g>` -Tag abgeschlossen werden. Um das ganze SVG-Dokument abzuschliessen wird am Schluss noch ein `</svg>` -Tag angefügt.

Die Erläuterungen anhand der gezeigten Beispiele sollten helfen, einige Prinzipien des SVG-Codes zu verstehen. Für weitere Informationen wird nochmals auf die SVG 1.0 Spezifikation [7] verwiesen. Sie ist gut strukturiert und die Grundlagen sind einfach zu verstehen. Für SVG-Einsteiger bietet ausserdem Adobe auf seiner SVG-Homepage ein gutes Tutorial an [1]. Auf carto:net [13] wird SVG sehr detailliert und in einem grösseren Rahmen erklärt, vor allem auch in Bezug auf die Kartographie. Auf dieser Seite führen Hyperlinks zu ähnlichen, auf SVG basierenden Projekten.

2.2 DOM

Das W3C DOM (*document object model*) ist eine plattform- und sprachneutrale Schnittstelle (*interface*) für den dynamischen Zugriff auf Inhalt, Struktur und Stil eines Dokuments. Das Dokument kann weiterverarbeitet und die Resultate des Prozesses auf der gleichen Seite wieder dargestellt werden. Das DOM stellt einen grundlegenden Satz von Objekten zur Verfügung, um XML und HTML hierarchisch gegliedert zu repräsentieren. Dazu gehört auch die Schnittstelle für den Zugriff und die Manipulation der Objekte. Entwickler von Web-Software können das DOM in ihre Produkte integrieren anstatt eigene Modelle zu entwickeln. Dadurch wird die Kompatibilität im Web gefördert [11].

Typischerweise steht zuoberst in der DOM Hierarchie das *window*-Objekt, es repräsentiert ein Fenster oder einen Frame. In der nächsten Hierarchie-Stufe folgen die Objekte *document*, *history*, *location*, *frames* und einige weitere. Das *document*-Objekt beinhaltet Informationen über das im Browser dargestellte Dokument, zum Beispiel die URL, das Datum der letzten Modifikation, die URL des Dokuments, das auf dieses Dokument verweist oder die Farbe des Hintergrunds. Das *document*-Objekt enthält weiter auch Methoden um dynamisch bestehende Dokumente mit Text zu erweitern oder auch neue Dokumente zu kreieren. Alle Elemente zur Darstellung des Dokuments sind im *document*-Objekt als Objekte der nächsten Stufe in der Hierarchie enthalten. Diese Objekte betreffen zum Beispiel Formulare, Bilder, Layer oder Applets. Interessant

ist sicher das *form*-Objekt, es ist das übergeordnete Objekt für alle gestalterischen Elemente in HTML, die zur Interaktion mit dem Benutzer gebraucht werden, zum Beispiel Buttons, Textfelder oder Auswahllisten [12].

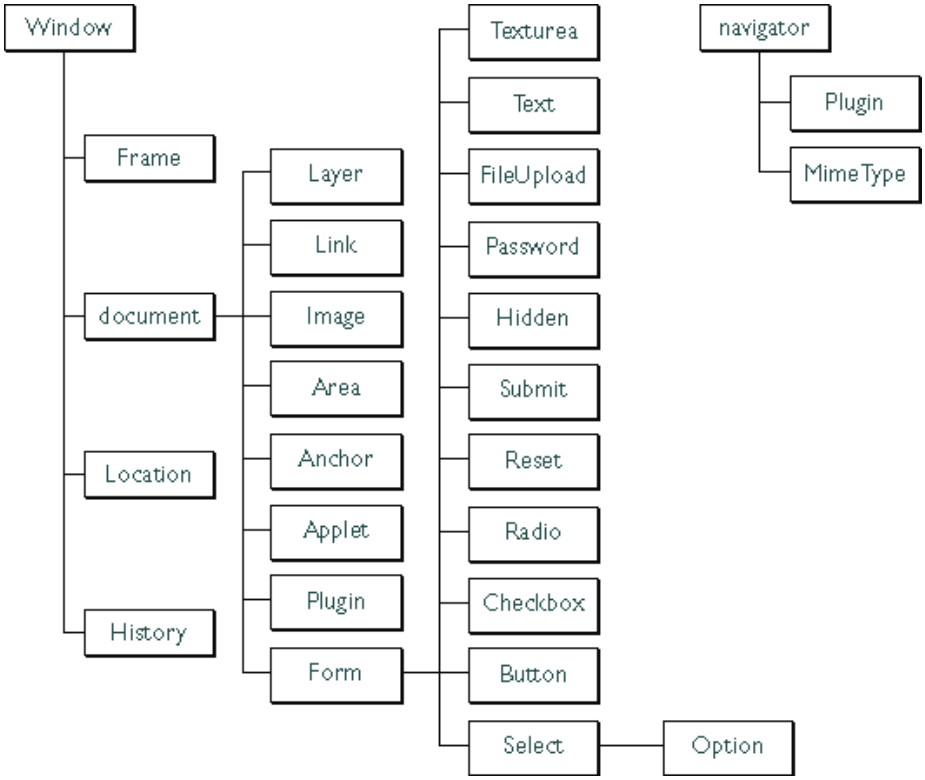


Abb. 2.2: Beispiel einer DOM-Hierarchie

Vor allem im Zusammenhang mit JavaScript (Kapitel 2.3) hat sich das DOM stark weiterentwickelt. Die Kombination der beiden ermöglicht neue Darstellungsmöglichkeiten im Internet. Die Entwicklungstufen des DOM werden als Level bezeichnet. In allen Browsern sollte mittlerweile das DOM Level 0 implementiert sein. Das nächste Ziel ist die Implementierung des DOM Level 1, wobei der Internet Explorer diesem Standard am nächsten kommt. Das DOM Level 1 ist bereits eine *W3C Recommendation*, das DOM Level 2 wird momentan noch getestet und liegt als *W3C Candidate Recommendation* vor, während bereits an einem DOM Level 3 gearbeitet wird.

2.3 JavaScript

JavaScript ist eine Skriptsprache, die im Zusammenhang mit CSS und HTML oft als *dynamic html* bezeichnet wird. Sie wurde von Netscape speziell für *web-programming* entwickelt. Wenn normalerweise von JavaScript geredet wird, ist *client-side* JavaScript gemeint. In dieser Version kann ausführbarer JavaScript-Code mit einer Webpage mitübertragen werden. Dadurch ergeben sich interessante Möglichkeiten des Webdesigns.

Webpages können dynamisch gestaltet oder sogar neu kreiert werden und verstärkt Interaktionen mit dem Benutzer stattfinden.

JavaScript ist nicht Java. Wegen der ähnlichen Namen wird oft ein Zusammenhang zwischen JavaScript und Java vermutet, diesen gibt es aber nicht. Die Skriptsprache JavaScript sollte ursprünglich LiveScript heissen, wurde dann aber kurzfristig aus markttechnischen Gründen mit dem jetzigen Namen benannt. Die einzige Beziehung zwischen den beiden Sprachen entsteht dadurch, dass Java-Applets mit JavaScript gesteuert werden können und umgekehrt.

JavaScript kann mit dem `<script>` -Tag in HTML eingebunden werden oder auch als externes File referenziert werden. Die Syntax der Sprache lehnt sich stark an C++ an, für Einsteiger ist sie dennoch relativ leicht zu lernen. JavaScript ist aber nicht eine einfache Sprache. Um die Sprache optimal zu nutzen, bedarf es vertiefter Kenntnisse.

Der Name "JavaScript" gehört der Firma "Netscape". Microsoft benannte demzufolge ihre Implementation der Sprache "JScript". Die beiden Sprachen sind mehr oder weniger kompatibel miteinander. Von einem einheitlichen Modell war aber nie die Rede, beide Konzerne wollten im Rahmen des sogenannten "Browserkriegs" den anderen mit neueren und besseren Features übertrumpfen. Für einen Entwickler hat das sehr unangenehme Folgen. Er muss nämlich für die verschiedenen Browser, zum Beispiel Netscape Navigator und Internet Explorer, zum Teil unterschiedlichen Code entwickeln. Bei jeder Anweisung, die nicht einheitlich für alle Browser ist, wird mit einer `if` -Anweisung bestimmt, welcher Browser vorhanden ist, dann wird der entsprechende Code ausgeführt. Diese Art von Codierung wurde *crossbrowsing* benannt. Ein weiteres Problem ist, dass man eine Idee für eine Webpage eventuell nicht auf allen Browsern gleich umsetzen kann. Dies führte zu den bekannten Vermerken im Internet z. B. *Site optimized for Internet Explorer*. Allerdings scheint sich eine Verbesserung dieser Verhältnisse abzuzeichnen. Zum Beispiel ist der erst kürzlich lancierte Browser Netscape 6 wieder mehr standard-konform.

Der komplette JavaScript Code meines Projektes ist im Anhang aufgelistet. Dies sind mittlerweile immerhin mehr als 2000 Zeilen. Leider existiert noch kein Entwicklungstool für JavaScript. Ausser das von *activestate* entwickelte Tool *Komodo*, diese Entwicklungs umgebung unterstützt JavaScript und steht in einer Beta-Version bereits zur Verfügung [22]. In einem normalen Texteditor geht die Übersicht nämlich relativ schnell verloren. Es stellt sich auch die Frage, ob JavaScript für grössere Projekte überhaupt geeignet ist.

Kapitel 3

Statistische Daten

3.1 Quelle

Die verwendeten Daten stammen vom Bundesamt für Statistik (BFS) [15]. Die Datengrundlage für die Jahre 1991 bis 1999 ist die *Statistik des jährlichen Bevölkerungsstandes* (ESPOP), für die Jahre 1981 bis 1990 ist es das *Zentrale Ausländerregister* (ZAR). Die Daten selber sind eine Spezialauswertung des Bundesamtes für Statistik. Ich erhielt die Daten zu Forschungszwecken zur Verfügung gestellt. Diese Arbeit ist ein Forschungsprojekt im Rahmen des Bundesprojektes Atlas der Schweiz [16].

> Die ESPOP ist eine Berechnung und Schätzung über die Struktur der ständigen Wohnbevölkerung nach Alter, Geschlecht, Heimat und Zivilstand. Sie werden jährlich veröffentlicht [15].

3.2 Inhalt

3.2.1 Beschreibung

Das Thema der Karte ist “Migration zwischen der Schweiz und Europa von 1981 bis 1999”. In der Datengrundlage sind aber die Migrationen der ganzen Welt enthalten. Für Länder ausserhalb der EU sind die Daten kontinentweise kombiniert worden. In der Karte sind darum auch die Daten zu Afrika, Asien und der Ex-Sowjetunion zusammmengefasst enthalten.

Die Auflösung der Daten in der Schweiz ist auf Stufe Grossregion. Eine kleinere Auflösung wäre zwar möglich gewesen, war aber gar nicht erwünscht. Erstens hätte das eine drei bis vier Mal so grosse Datenmenge mit sich gebracht und zweitens stellte sich die Frage, ob die 26 Einzelflächen anstatt der sieben nicht ein unruhiges Kartenbild zur Folge hätten.

Die Daten enthalten folgende Variablen: Ein- und Auswanderungen nach Heimatland (-kontinent) mal Grossregion, Geschlecht und Jahr. In den Einwanderungen sind die Statuswechsel von Saisonniers zu Jahresaufenthaltern und auch die Übertritte aus dem Asylbereich enthalten. Die sieben Grossregionen sind folgendermassen aus den entsprechenden Kantonen zusammengesetzt:

Genferseeregion: Genf, Wallis, Waadtland

Zentrales Mittelland: Bern, Fribourg, Jura, Neuchatel, Solothurn

Nordwestschweiz: Basel-Landschaft, Basel-Stadt, Aargau

Zürich: Zürich

Ostschweiz: Appenzell Ausser Rhoden, Appenzell Inner Rhoden, Glarus, Graubünden, St. Gallen, Schaffhausen, Thurgau

Zentralschweiz: Luzern, Nidwalden, Obwalden, Schwyz, Uri, Zug

Tessin: Tessin

Bezüglich der Heimatstaaten wurden nicht alle Daten des Bundesamtes für Statistik verwendet. Zuerst musste ein Kompromiss zwischen dem vorhandenen Kartenmaterial und den vorhandenen Daten gefunden werden. In der folgenden Liste sind alle Heimatstaaten und -staatengruppen aufgeführt deren Daten für die Karte verwendet wurden.

Afrika	Grossbritannien	Portugal
Albanien	Irland	Rumänien
Andorra	Island	Russland
Asien	Italien	Schweden
Belgien	Lettland	Slowakei
Bulgarien	Liechtenstein	Spanien
Dänemark	Litauen	Tschechien
Deutschland	Luxemburg	Türkei
Estland	Malta	Ukraine
Ex-Sowjetunion	Moldawien	Ungarn
Ex-Yugoslawien	Niederlande	Weissrussland
Finnland	Norwegen	Zypern
Frankreich	Österreich	
Griechenland	Polen	

In der Karte sind von Afrika, Asien (Nahost) und der Ex-Sowjetunion jeweils nur wenige Staaten zu sehen. Da für die Einzelstaaten dieser Gruppen keine Daten vorhanden sind, wurden sie zusammengefasst und repräsentieren nun die Daten für eine ganze Gruppe.

3.2.2 Probleme

Die Daten decken die Zeitspanne von 1981 bis 1999 ab. Während dieser Zeit gab es einige grundlegende Änderungen in der Struktur der Länder in Europa und den angrenzenden Staaten. Konkret sind folgende Ereignisse gemeint:

- 1991: Aufteilung der UdSSR in Russland, Ukraine, Lettland, Estland, Litauen, Weissrussland, Moldawien und weitere
- 1992: Aufteilung der Tschechoslowakei in die Tschechien und die Slowakei
- 1991: Auflösung von Yugoslawien

Es stellt sich die Frage, wie mit solchen Änderungen bei der Gestaltung der Karte umgegangen wird. Denn es ergeben sich einige grundlegende Probleme. Vor der Trennung zweier Länder, sind nur die Daten des ursprünglichen einzelnen Landes bekannt. Darstellungen vor und nach der Teilung der Länder müssten sich also topologisch unterscheiden und es könnte nicht mehr von einem statischen Datensatz für die geographischen Vektordaten ausgegangen werden. Dies gilt aber nur für Darstellungen, die ein bestimmtes Jahr betreffen. Für kumulierte Darstellungen über mehrere Jahre hinweg ergeben sich neue Darstellungs- und Berechnungsprobleme. Zum Beispiel möchte ein Benutzer eine Kartendarstellung von männlichen Immigranten aus der Ukraine betreffend der Genferseeregion zwischen 1985 und 1995. Diese Angaben stehen aber gar nicht zur Verfügung. 1991 wurde die Sowjetunion aufgelöst. Aus den Daten kann nur ermittelt werden, wieviele Männer von 1985 bis 1991 aus der Sowjetunion in die Genferseeregion eingewandert sind, und wieviele Männer von 1992 bis 1995 aus der Ukraine in die Genferseeregion eingewandert sind.

Für das Problem der Berechnung ergaben sich folgende Lösungsansätze:

1. Die Benutzereingaben werden kontrolliert und unmögliche Abfragen gar nicht akzeptiert.
2. Es werden bewusst falsche Kumulationen generiert, der Benutzer wird aber direkt in der Kartendarstellung darauf aufmerksam gemacht.

Die Entscheidung fiel zu Gunsten des Ansatzes zwei. Die Begründung für diese Entscheidung ist simpel und hängt auch mit der Entstehung dieser Arbeit zusammen. Ein grosser Teil der vorbereitenden Arbeiten für die Gestaltung und Programmierung der Karte wurde nämlich schon gemacht, als noch gar nicht klar war, welcher Struktur die bestellten Daten sein würden. Zum Beispiel war beim Eintreffen der Daten die Benutzerführung und die Struktur des Codes schon so weit fortgeschritten, dass sich eine ideale Anpassung der Programmierung auf die Daten als fataler Zeitverlust geäussert hätte. Der unter Punkt eins erwähnte Lösungsansatz hätte nämlich zur Folge, dass der Benutzer nicht nur in Bezug auf die drei Ex-Länder beschränkt würde, alle anderen Länder würden durch dieses Vorgehen auch blockiert. Dies gilt zwar nur für die eine Abfrageart, diese ist aber stark mit der anderen gekoppelt, so dass mit einer Umstellung die ganze Struktur des Codes hätte verändert werden müssen.

Für das Problem der Statik der Geometrie ergaben sich folgende Lösungsansätze:

1. Die Grenzverläufe werden dem Abfragejahr angepasst. Bei kumulierten Abfragen werden die Grenzverläufe gestrichelt dargestellt.
2. Die Geometrie der Grenzverläufe bleibt statisch. Es wird immer der aktuelle Zustand angezeigt. Änderungen zu diesem Zustand werden direkt in der Karte vermerkt.

Auch bei dieser Entscheidung wurde der Ansatz zwei gewählt, denn die Umsetzung des Lösungsansatzes eins hätte für die Programmierung einen grossen Aufwand mit sich gezogen und zudem schlecht in die bereits bestehende Struktur des Codes gepasst.

Die Lösung für beide beschriebenen Probleme ist demzufolge, dass grafisch der aktuelle Zustand angezeigt wird und auf der Seite der Berechnungen alle Benutzereingaben verarbeitet werden. Dadurch entstehen falsche Darstellungen und Angaben, diese

werden aber in der Grafik mit einem speziellen Symbol markiert und so der Benutzer auf das Problem hingewiesen. Betroffen sind die Staaten der Ex-Tschechoslowakei und der Ex-Sowjetunion. Hingegen sind die Länder von Ex-Jugoslawien nicht von dieser Einschränkung betroffen, da sie in der Datengrundlage sowieso zu einer Ländergruppe zusammengefasst sind und nicht einzeln vorhanden sind.

Die Symbole zur Markierung sind in der Version am Ende der Diplomarbeit noch nicht implementiert.

3.3 Einbindung der Daten

3.3.1 Zugriff

Für den Zugriff auf die Daten standen mehrere Möglichkeiten zur Auswahl. Die Entscheidung fiel auf einen Zugriff innerhalb des Java-Script-Codes. Denn die Datenbankanbindung mit *client-side* JavaScript ist nicht möglich und der Zugriff auf ein externes File auch nicht. Also wurden die Daten in einer eigenen Strukur in statischen Variablenfeldern gespeichert und vom Programm direkt darauf zugegriffen. In Zukunft sollte es allerdings möglich sein, mittels XML *Query* [5] auf ein externes XML zugreifen zu können.

3.3.2 Datenformat

Wie schon weiter oben erwähnt war zu Beginn der Entwicklung die Struktur der Daten nicht bekannt, nur die zu erwartende Art der Daten und welche Variablen dazugehörten. Für den Zugriff auf die Daten wurde also ein eigenes Datenformat entwickelt, mit der Absicht später die wirklichen Daten in das eigene Datenformat umzuformatieren. Diese Umformatierungen waren relativ umständlich, im Vergleich waren sie aber sicher einfacher als das Programm umzuschreiben.

Das programminterne Datenformat ist so aufgebaut, dass jedes Land, das Daten für die Karte liefert, zwei Felder zugewiesen erhält, ein Feld für alle Daten der Immigrationen und eines für alle Daten der Emmigrationen. Zu Beginn jedes Feldes werden die zwei ersten Feldzellen mit Koordinaten für die Plazierung von Diagrammen besetzt (Kapitel 6.12, 6.13). Danach folgen die Migrationsdaten strukturiert nach Grossregion, nach Geschlecht und nach Jahr. Zum Beispiel sind die ersten 19 Zahlen im Datenblock die Anzahl der männlichen Migranten bezüglich des Tessins in den Jahren 1981 bis 1999. Danach folgen die 19 Angaben zu den weiblichen Migranten bezüglich des Tessins. Nach dem Tessin folgen die 38 Daten für die Ostschweiz, und so weiter.

Die 38 Daten pro Region multipliziert mit den 7 Regionen ergibt pro Variablenfeld 266 benötigte Zellen. Das Projekt umfasst 41 Länder oder Ländergruppen. Alles in allem werden in diesen Variablenfeldern somit 21’812 Einzelinformationen gespeichert.

Im folgenden Beispiel wird schematisch dargestellt, wie das eigene Datenformat aufgebaut ist:

```
           Koordinatenangaben für Ex-Jugoslawien
var im1 = [200,6100, ->

           Tessin, männl. Immigranten aus Ex-Jugoslawien 1981-1999
```

```
        -> 76,114,66,78,112,177,168,205,298,456,564,964,1045,939,
           632,247,263,137,177, ->

           Tessin, weibl. Immigranten
        -> 82, 69,72,79,134,132,162,152,268,378,539,929,1030,913,
           651,232,228,187,195, ->

           Ostschweiz, männl. Immigranten
        -> 1346,1357,997,..
           ..];

           Koordinatenangaben für Ex-Jugoslawien
var em1 = [200,6100, ->

           Tessin, männl. Emmigranten aus Ex-Jugoslawien 1981-1999
        -> 28,30,22,24,22,32,32,42,48,50,65,61,85,163,265,323,152,
           170,103, ->

           Tessin, weibl. Emmigranten
        -> 26,26,24,31,32,28,39,39,36,54,45,51,83,133,279,310,105,
           113,103, ->

           Ostschweiz, männl. Emmigranten
        -> 494,569,528,..
           ..];
var im2 = [..];
var em2 = [..];
var im3 = [..];
..
```

Der dynamische Zugriff auf eine solche Mini-Datenbank ist logischerweise nicht sehr benutzerfreundlich. Dementsprechend komplex wurden auch die Zugriffsalgorithmen im Programm (Kapitel 6.7). Ein weiterer Nachteil eines solchen Vorgehens ist auch der grosse Aufwand, der betrieben werden muss, wenn das Programm oder die Daten erweitert oder aktualisiert werden sollen. Alternativen zu meiner Vorgehensweise sind sicher vorhanden, hätten aber meiner Meinung nach den Rahmen dieser Arbeit gesprengt. Die Grösse des Projekts bewegt sich am Limit punkto Daten, Erweiterbarkeit und zum Teil auch Geschwindigkeit. Für den Rahmen einer Diplomarbeit ist die Vorgehensweise aber vertretbar.

In Zukunft wäre es zum Beispiel denkbar, die Daten in einem externen XML-Dokument abzulegen und mit XML *Query* darauf zuzugreifen. Die Nachfrage nach einer Interaktion zwischen Internet und Datenbanken ist gross, es sollte darum in absehbarer Zeit eine vernünftige Lösung für dieses Problem geben.

3.4 Copyright und Datenschutz

Das Copyright der verwendeten Daten gehört dem Bundesamt für Statistik [15].

Die Daten stehen unter Datenschutz. Angaben zur Migration unter fünf Personen wurden mit einer Angabe von Null Personen ersetzt.

Kapitel 4

Grafische Daten

4.1 Europa und angrenzende Länder

Die grafischen Daten von Europa und den angrenzenden Ländern stammen aus einer Datengrundlage des BFS. Die Daten wurden in einem ersten Schritt schon von André M. Winter nachbearbeitet und als SVG-Code verwendet. Für die Migrations-Karte waren wiederum einige Anpassungen direkt am SVG-Code notwendig. Zum Beispiel wurden einzelne Länder zu Gruppen zusammengefügt. Ausserdem wurde die Karte natürlich auch ans Erscheinungsbild der ganzen Karte angepasst. Dank einer konsequenten Verwendung von CSS hielt sich der Aufwand für diese Einstellungen in Grenzen. Die wohl aufwendigsten Arbeiten direkt am SVG-Code hingen gar nicht mit grafischen Optionen zusammen, sondern betrafen vor allem die Anpassung der *event* -Auslöser und Identifikationen, um die Kommunikation mit JavaScript zu ermöglichen.

Das Copyright der grafischen Daten zu Europa gehört dem Atlas der Schweiz [16].

4.2 Schweiz

Auch die grafischen Daten zur Schweiz stammen aus einer Datengrundlage des BFS, sie stammen von der Swiss Publisher CD [19]. Dort fand sich ein Vektordatensatz der Schweiz mit ihren Kantonen. Diese Daten wurden mit dem Grafikprogramm Corel Draw 9 nachbearbeitet und dann als SVG exportiert [2]. Die Nachbearbeitung bestand vor allem darin, die Kantone zu den entsprechenden Grossregionen vektormässig zusammenzusetzen und kleine Anpassungen an den Vektoren vorzunehmen, zum Beispiel Elemente, die die Minimaldimensionen unterschreiten, zu löschen.

Das Copyright der grafischen Daten zu Europa gehört der Abteilung GEOSTAT des Bundesamtes für Statistik [15].

4.3 Bearbeitung von SVG-Files

In den meisten Fällen werden SVG Daten durch den Exportfilter eines Grafikprogramms generiert. Danach werden im SVG-Code direkt eventuell noch einige Änderungen und *event* -Auslöser hinzugefügt. Im Laufe eines grösseren Projektes, wie

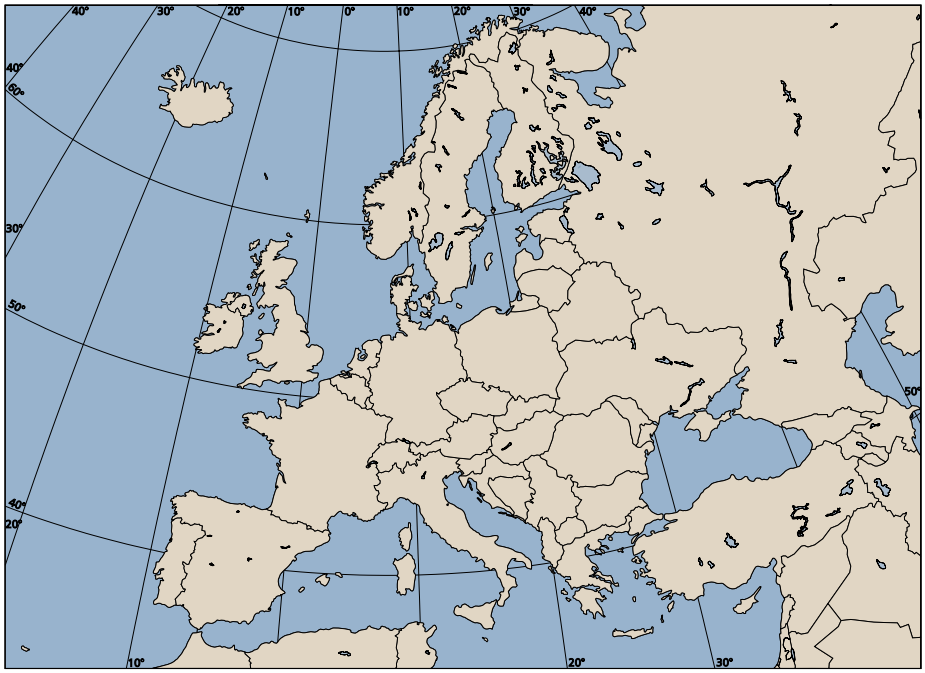


Abb. 4.1: Bearbeitete Vektordaten, Europa

der zu diesem Bericht gehörenden Migrations-Karte, kann es vorkommen, dass grössere Änderungen anstehen. Jetzt stellt sich die Frage, ob die modifizierten und mit fremden Code versehenen Dateien wieder in das Grafikprogramm importieren lassen. Ich arbeitete vor allem mit Corel Draw 9.0. Leider konnten SVG-Dateien nicht korrekt importiert werden. Dies hatte zur Folge, dass bei grösseren Änderungen die Grafiken immer neu exportiert werden mussten. Ein weiterer wünschenswerter Punkt wäre zum Schluss noch die Möglichkeit, die mit *event* -Auslösern versehenen Layer wieder korrekt zu exportieren. Dies würde eine schrittweise Entwicklung von JavaScript-gesteuerten SVG-Daten mit der Unterstützung des Grafikprogramms erlauben.

Das eben herausgekommene Corel Draw 10 erlaubt schon den korrekten Import von SVG-Dateien mit Fremdcode. Allgemein wird die Entwicklung der Grafikprogramme in Richtung Internetkompatibilität auch eine verbesserte SVG Unterstützung mit sich bringen.

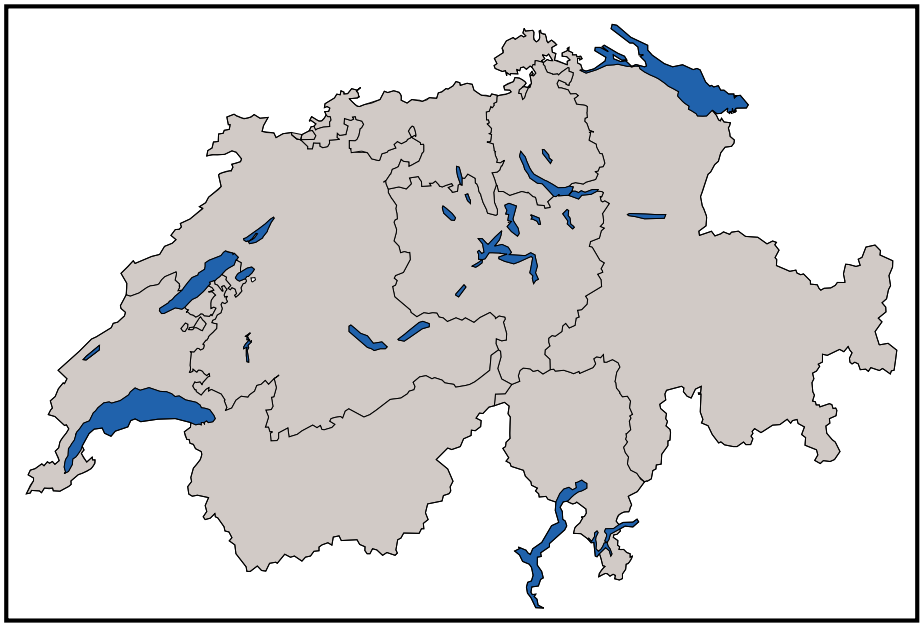


Abb. 4.2: Bearbeitete Vektordaten, Schweiz

Kapitel 5

Gestaltung

5.1 Idee

Für die Gestaltung der Karte gab es verschiedene Möglichkeiten. Ein grosser Teil des Bildschirms sollte aus der kombinierten Karte von Europa und der Schweiz für die grafische Darstellung der statistischen Daten bestehen. Die Einstellung aller Variablen und Darstellungsoptionen sollte über ein auf- und zuklappbares Menü erfolgen und die Legende in einem eigenen kleinen SVG-Dokument dargestellt werden. Als weitere Option für die Anzeige von Informationen standen auch pop-up Fenster zur Verfügung (pop-ups: Mit JavaScript neu erstellte Fenster). Der grösste Teil der Bildschirmfläche sollte der SVG-Karte zur Verfügung stehen.

5.2 Bildschirmaufbau

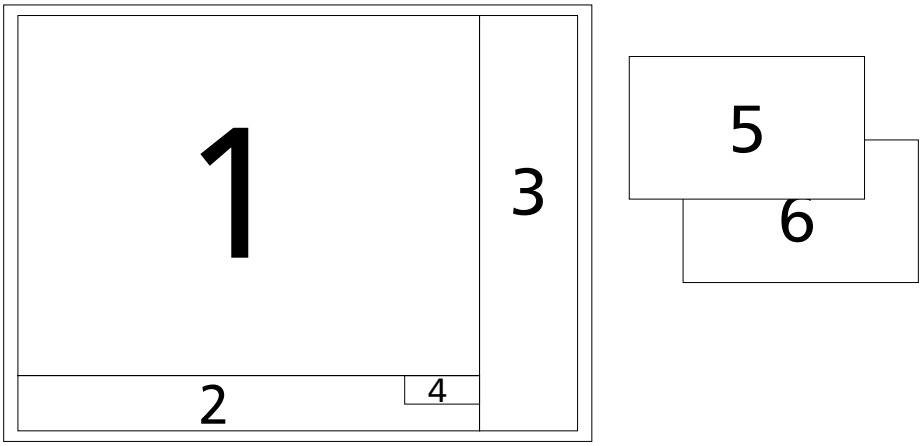


Abb. 5.1: Skizze für die Gestaltung der Bildschirmfläche

1: SVG-Karte kombiniert aus Schweiz und Europa

2: SVG-Legende für die Farbklassen

3: Menüleiste, auf- und zuklappbar

4: Symbolleiste für Zusatzfunktionen (SVG)

5: Pop-up Fenster für Datentabellen

6: Pop-up Fenster für die Darstellung der Histogramme (SVG)

Ausgehend von der Idee im Abschnitt 5.1 hat sich im Laufe der Arbeit die in Abbildung 5.1 skizzierte Bildschirmaufteilung für die verschiedenen Elemente ergeben. Neu hinzugekommen ist einzig noch eine kleine Symbolleiste. Durch die technischen Randbedingungen musste das grösste Element, die SVG-Karte, bündig mit der linken und oberen Begrenzung der Anzeigefläche plaziert werden. Diese Randbedingungen haben folgenden Ursprung; bevor die Karte im Browserfenster aufgebaut wird, ermittelt JavaScript die Grösse der Anzeigefläche des Fensters. Entsprechend dieser Informationen werden dann die einzelnen SVG-Bildschirmelemente skaliert und ihre Positionen so angepasst, dass das Erscheinungsbild der Karte unabhängig von der Fenstergrösse immer gleich aussieht. Das Problem ist, dass die Karte, die Legendenleiste und die Symbolleiste im Gegensatz zum Menü, das vor allem aus Text besteht, sehr einfach skaliert werden können. Darum wird das Menü am rechten Rand der Karte plaziert und hat falls nötig freien Auslauf nach rechts, und es ist garantiert, dass die Karte immer komplett angezeigt wird.

Die Legende und die Symbolleiste wurden direkt unterhalb der Karte mit der gleichen horizontalen Ausdehnung plaziert. Der zur Verfügung stehende Platz reicht gut aus um die maximal sieben Legendenelemente für die Klassendarstellung und die Symbolleiste darzustellen. Der Bildschirm wird durch diese Anordnung in zwei durch eine Vertikale getrennte Bereiche geteilt. Im linken Bereich sind alle SVG-Elemente zu finden, der rechte beinhaltet das Menü. Die Anpassung des Kartenbildes auf verschiedene Fenstergrössen wird so erleichtert. Ausserdem kann das Menü beliebig erweitert werden, ohne dass Konflikte mit den SVG-Elementen durch Überdeckungen entstehen. Falls noch weitere Darstellungselemente gebraucht werden sollten, können diese in dynamisch kreierten pop-up Fenstern untergebracht werden.

5.3 SVG-Karten

5.3.1 Bestandteile

Zuerst wurde versucht zwei eigenständige SVG-Karten für die Schweiz und Europa zu überlagern, dies führte vor allem beim Netscape Navigator zu grossen Darstellungsproblemen. Also wurden die Karten in einem Dokument kombiniert. Obwohl die SVG-Karte für die Schweiz über ein eigenständiges Koordinatensystem verfügte, war die Integration in die Europa-Karte relativ einfach. Anstatt die Koordinaten an das System der Europa-Karte anzupassen, wurde mit dem `transform` -Attribut das Schweizer System dem von Europa angepasst, bis die Darstellung stimmte. Das `transform` -Attribut kann auf jede Gruppe und jedes Element angewendet werden. Grafische Element können so unabhängig von ihrem inneren Koordinatensystem exakt plaziert werden.

```
<g id="schweiz"
transform="translate(-1448,0) scale(0.45) translate(-5750,0)">
```

Das `transform` -Attribut erlaubt sechs Transformationen, Translation (*translate*), Rotation (*rotate*), Skalierung (*scale*), Verdrehungen in X und Y (*skewX, skewY*) und Matrixtransformationen (*matrix*). Die Koordinaten der Schweiz werden zuerst in den Ursprung der Europa-Karte verschoben, auf die richtige Grösse skaliert und dann an die richtige Stelle innerhalb der Karte gesetzt. Die Anweisungen werden von rechts nach links ausgeführt. Für weitere Informationen wird auf die SVG 1.0 Spezifikation [7], Kapitel 7 verwiesen. Durch die Plazierung der Schweizer Karte im linken oberen Bereich, und durch das Bestreben die Schweiz möglichst gross darzustellen, musste Island von seiner geographisch korrekten Position weichen und wurde unterhalb der Schweiz neu positioniert. Die Darstellung mit Rahmen und eigenen Koordinatenlinien soll garantieren, dass die verschobene geographische Position Islands erkannt wird.

5.3.2 Farben und Schriften

In der Karte soll Information grafisch dargestellt werden. Die neutralen Farben der Karte, wenn keine Informationen dargestellt werden, dürfen also keine Konflikte mit den verwendeten Farben für die Darstellung der Daten verursachen. Die Karte ist darum im neutralen Zustand in Grau- und Beigetönen gehalten. Die HTML-Seite, Plattform für alle anderen Elemente, ist mit einem mittleren Grau eingefärbt. Für das Meer in der Karte wurde Pastellblau, für die Länder zur Unterscheidung ein Pastellbeige verwendet. Alle Ränder, Schriften und Symbole sind schwarz.

Die Farben sind so aufeinander abgestimmt, dass ein ruhiges Kartenbild entsteht. Für die Darstellung von Informationen stehen genügend Farben und Farbreihen zur Verfügung, die sich vom neutralen Kartenbild abheben und das Auge auf sich ziehen.

Als Kartenschrift wurde einheitlich die serifenlose Schriftart *Humanist 777 BT* verwendet. Um zu gewährleisten, dass diese Schrift auf jedem Computer korrekt dargestellt wird, ist die Definition der Schrift als CEF-File im SVG-Code referenziert. CEF bedeutet *Compact Embedded Font*; der SVG-Exportfilter für das Programm Illustrator von Adobe speichert Schriftarten standardmässigs in diesem Format. Es besteht die Möglichkeit die Schriftdefinition direkt im SVG-Dokument oder als externes File zu speichern. CEF-Fonts sind in der SVG-Spezifikation [7] nicht speziell definiert, aber die Spezifikation verlangt, dass Viewer die das CEF-Format nicht unterstützen, auf dem System des Benutzers nach einer ähnlichen Schriftart suchen und diese benutzen [1].

Für das Menü und den Titel wurde ebenfalls eine serifenlose Schrift vorgeschrieben. Dies geschieht über einen Eintrag im *style sheet*. Falls die Schrift Helvetica auf dem Computer des Benutzers vorhanden ist, wird diese verwendet. Sonst wird einfach eine andere serifenlose Schrift benutzt.

5.3.3 dynamische Elemente

Die dynamischen Elemente sind grafische Symbole oder Schriften, die je nach Aktion des Benutzers, einen speziellen Zustand darstellen.

Ländernamen: Fährt der Benutzer mit dem Mauszeiger über ein Land oder eine Ländergruppe, erscheint in einem Feld links unten in der Karte die Bezeichnung dieses Elements.

Namen der Grossregionen: Analog dem dynamischen Textelement für die Länder und Ländergruppen werden die Namen der Grossregionen links oben in der Ecke beim Überfahren mit der Maus angezeigt.

Variablensymbole: Mit diesen Symbolen werden die vom Benutzer eingestellten Variablen aus dem Variablenmenü visualisiert. Dadurch kann der Benutzer auf einen Blick die Einstellungen des Variablenmenüs bei geschlossener Menüleiste erkennen. Positioniert sind die Symbole in den noch ungenutzten Ecken der Schweizer Karte. Dargestellt werden:

- das Jahr oder der Bereich von Jahren, rechts unten
- Immigration, Emmigration oder Saldo, rechts oben
- Mann, Frau oder beides, links unten

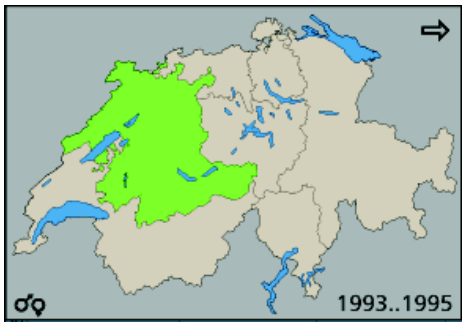


Abb. 5.2: Menüsymbole in der Schweizerkarte

In der Abbildung 5.2 sind zum Beispiel folgende Einstellungen zu erkennen:

Zeitbereich von 1993 bis 1995

Wanderungsrichtung Emmigration

Geschlecht Frauen und Männer

Natürlich sind auch die einzelnen Länder selber dynamische Elemente, da sie je nach Abfrage anders eingefärbt werden.

5.4 Menüleiste

Zuoberst in der Menüleiste (Abb. 5.3) steht ein abgekürzter Titel zur Karte. Der komplett ausformulierte Titel würde zu viel Platz auf der Bildschirmfläche wegnehmen und vor allem bei der dynamischen Anpassung des Fensterinhalts an die Fenstergrösse Probleme bereiten.

Das Menü (Abb. 5.3) an sich wurde dynamisch aufgebaut und in drei Bereiche gegliedert. Die einzelnen Bereiche können per Mausklick auf- und zugeklappt werden, dadurch steht mehr Platz für die Menüpunkte zur Verfügung. Zudem kann das Menüsystem relativ einfach mit weiteren Menüpunkten ausgebaut werden. Die Menüpunkte sind als Hyperlinks getarnt, so dass der Benutzer ein Feedback erhält. Unterstützt wird dieser Hinweis durch die Dreieck-Symbole zu Beginn eines Menüpunkts. Falls das

Menü aufgeklappt wird, drehen sich diese Dreiecke nach unten (funktioniert nur im Internet Explorer). Realisiert wird das dynamische Menü mit *css-layern*. Die Umsetzung wird genauer im Kapitel 6.3 beschrieben.

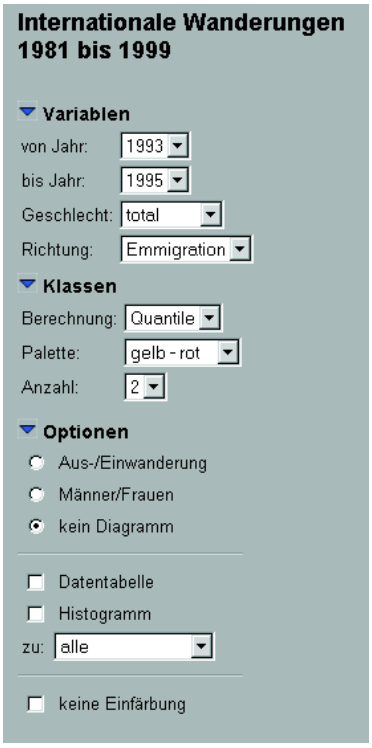


Abb. 5.3: dynamisches Menü

Im ersten Menübereich Variablen können die gewünschten Einstellungen betreffend der Daten vorgenommen werden. Die Details zur klassenweisen Einfärbung werden im zweiten Menübereich Klassen manipuliert. Alle zusätzlichen Optionen sind im dritten Menübereich Optionen untergebracht. Diagramme, Histogramme und Detaildatenfenster können dort aktiviert werden.

Menübaum:

Variablen

- von Jahr: Beginn des darzustellenden Zeitbereichs [1981 . . . 1999]
- bis Jahr: Ende des darzustellenden Zeitbereichs [1981 . . . 1999]
- Geschlecht: Geschlecht der Migranten [Mann, Frau, Beide]
- Wanderungsrichtung: Auswahl der Wanderungsrichtung [Immigration, Emmigration, Saldo]

Klassen

- Berechnung: Art der Berechnung der Klassen [Quantile, Stufen, manuell]

Palette: Farbpalette für die Einfärbung der Länder nach den berechneten Klassen [gelb-rot, blau, grün, rosa-blau, rot, grün2, grau, manuell]

Anzahl: Anzahl der darzustellenden Klassen [2 . . . 7]

Optionen

Balkendiagramme: Die Anzahl der Migranten wird durch Balkendiagramme dargestellt

Kreisdiagramme: Das Verhältnis zwischen Männer und Frauen und die Anzahl Migranten wird durch Kreisdiagramme dargestellt

kein Diagramm: Es wird kein Diagramm angezeigt, Standardeinstellung

Datentabelle: Anzeige eines pop-up Fenster mit einer Datentabelle zu den aktuellen Einstellungen

Histogramm: Anzeige eines pop-up Fensters mit einem Histogramm über den ganzen Zeitbereich hinweg betreffend des im nächsten Menüpunkt ausgewählten Landes

zu: Auswahl des Landes oder der Ländergruppe für die Histogrammdarstellung

Animation: Einfärbung der Länder läuft von 1981 bis 1999 im Sekundentakt

keine Einfärbung: Falls Diagramme angezeigt werden, kann mit diesem Punkt eine Einfärbung der Länder und Ländergruppen verhindert werden

Bei der Auswahl des Zeitbereichs kann theoretisch ein negativer Zeitbereich verursacht werden. Ist dies der Fall (von Jahr grösser als bis Jahr), dann wird bei einer Aktivierung der Karte (Kapitel 6.2) bis Jahr automatisch auf den gleichen Wert wie von Jahr gesetzt. Die Option Saldo im Menüpunkt Wanderungsrichtung hat eine Darstellung der Differenz zwischen Immigrierten und Emmigrierten Wanderern zur Folge. Ein negativer Saldo bedeutet mehr Abwanderer als Zuwanderer.

Für die Berechnung der Klassen sind bis jetzt drei Optionen vorhanden. Bei der Berechnung nach Quantilen erhält jede Klasse gleich viele Länder, also auch gleichviele Klassenfarben. Dies ergibt ein ausgewogeneres Kartenbild als mit der Berechnung der Klassen nach festen Stufenwerten. Bei der Berechnung nach Stufen werden die Klassengrenzen linear nach der Anzahl der gewanderten Personen zwischen dem Minimum und dem Maximum aufgeteilt. Dabei kann der Fall auftreten, dass einzelne Klassen viel mehr Länder enthalten als andere. Diese Klasseneinteilung macht Extrema gut sichtbar. Die letzte Option, manuelle Berechnung, überlässt dem Benutzer die Möglichkeit anhand des angegebenen Minimum, Maximum und des Bereichs die Klassengrenzen manuell einzugeben. Kapitel 6.8.

Die vorgegebenen Farb-Paletten sind Vorschläge für den Benutzer. Eine Palette wird nur durch ihre Anfangs- und Endfarbe definiert. Falls mehr als zwei Klassen dargestellt werden müssen, werden die Farbabstufungen dazwischen für jeden Fall neu berechnet. Der Benutzer kann mit dem Menüpunkt manuell durch die Angabe einer Anfangs- und Endfarbe im RGB-Raum (Rot, Grün, Blau) eigene Paletten erzeugen. Die Farbinterpolation erfolgt im Moment noch über eine lineare Interpolation der Rot-, Grün- und Blauwerte. Eine interessante Alternative wäre eine Interpolation im HLS-Raum oder eine Auswahl der Farbe über ein grafisches User-Interface. Diese Optionen könnten in einer späteren Version realisiert werden. Kapitel 6.9.

Die Balkendiagramme funktionieren für alle drei Optionen der Wanderungsrichtung. Interessant ist dabei vor allem die Darstellung der Saldowerte, weil sehr schnell sichtbar wird, wo negative und wo positive Saldi vorherrschen. Ausserdem werden mit den Balken Absolutwerte dargestellt, die Höhe der Balken entspricht exakt der Anzahl gewanderter Personen. Diese Information kann nur aus der Einfärbung der Länder nach Klassen nicht aus der Karte gelesen werden. Im linken unteren Bereich der Karte wird eine Legende zu den Balkendiagrammen eingeblendet, an einem Referenz-Balken werden Minimum und Maximum angegeben. Kapitel 6.12.

Eine Aktivierung der Kreisdiagramme hat zur Folge, dass im Menü Variablen der Menüpunkt Geschlecht auf Beide umgestellt wird. Die Einfärbung nach Klassen wird so der Aussage der Kreisdiagramme angepasst. Falls im Menüpunkt Richtung die Option Saldo aktiviert ist, wird auf Immigration gewechselt. Denn mit den Kreisdiagrammen werden erstens die Anzahl Wanderer entsprechend der Kreisfläche und zweiten das Verhältnis zwischen Männern und Frauen mit zwei Kreissektoren visualisiert. Eine zusätzlich Anzeige negativer und positiver Saldi würde zu einem überladenen Kartenbild führen. In der Skala für die Darstellung der Kreisdiagramme wurde ein Minimumkreis eingeführt. Dieser enthält alle Werte die unter einer bestimmten Grenze liegen, denn es soll immer garantiert sein, dass das Verhältnis zwischen Frauen und Männern eindeutig erkennbar ist. Wiederum wird bei der Aktivierung der Kreisdiagramme eine Legende im linken unteren Bereich eingeblendet. Kapitel 6.13.

Die Option Datentabelle verursacht, dass bei einer Aktivierung der Karte (Kapitel 6.2) die entsprechenden Daten in einem neuen Fenster in Tabellenform dargestellt werden. Das Fenster schliesst sich von alleine wieder, wenn es den Fokus verliert, zum Beispiel wenn irgendwo im Mutterfenster ein Mausklick detektiert wird. Im Fenster kann aber, falls nötig, gescrollt werden und das Fenster kann auch verschoben werden. Kapitel 6.14.

Ein Histogramm kann über den nächsten Menüpunkt Histogramm erzeugt werden. Wie bei der Datentabelle erscheint das Histogramm nach Aktivierung der Karte (Kapitel 6.2) in einem neuen Fenster. Das angezeigte Histogramm bezieht sich auf das Land oder die Ländergruppe, die in der Auswahlliste direkt unter dem Menüpunkt Histogramm eingestellt wurde. Egal welcher Zeitbereich im Menü Variablen definiert ist, es werden immer alle Daten von 1981 bis 1999 angezeigt. Das Fenster mit dem Histogramm schliesst sich ebenfalls selber, sobald es den Fokus verliert. Eine Einschränkung der Histogramm-Option gegenüber allen anderen Anzeigen ist, dass es nicht in beiden Szenarien der Kartenaktivierung funktioniert, sondern nur nach einem Klick auf eine der Grossregionen der Schweiz. Die Implementierung der zweiten Variante wäre zwar möglich, bringt aber keine nennenswerte Verbesserung für den Benutzer. Kapitel 6.15.

Durch die Auswahl der Option Animation wird bei der nächsten Aktivierung der Karte eine Animation gestartet. Anhand der ausgewählten Variablen und Klasseneinstellungen wird die Karte im Sekundentakt für die Jahre 1981 bis 1999 eingefärbt. Um die Animation nicht zu verzögern, werden dabei keine Diagramme angezeigt und keine pop-up Fenster generiert. Die aktuellen Einstellungen bleiben aber alle vorhanden. Kapitel 6.16.

5.5 Legende

Die Legende ist wie die Karte mit SVG realisiert. Dadurch können schnell grafische Elemente dynamisch kreiert und auch wieder gelöscht werden. Die Legende wird mit jeder Aktivierung der Karte (Kapitel 6.2) neu aufgebaut. Unabhängig von der Anzahl darzustellender Klassen wird die Legende immer am am linken Rand orientiert. Die Legende besteht aus sovielen Elementen wie Klassen ausgewählt wurden. Ein Element setzt sich aus einem der Klassenfarbe entsprechenden farbigem Rechteck und der Beschriftung mit dem der Klasse entsprechenden Bereich zusammen.

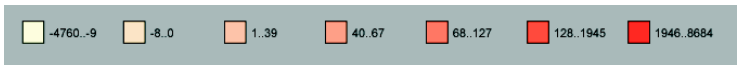


Abb. 5.4: Beispiel einer Menüanzeige

Bei einem Neuladen der Karte, wird der Bereich der Legende zusätzlich für einen *scroll* -Text gebraucht, der dem Benutzer den ersten Einstieg in die Benutzerführung der Karte geben soll. Mit dem Text wird der Benutzer dazu aufgefordert, auf die Karte zu klicken. Kapitel 6.10.

5.6 Symbolleiste

Abb. 5.5: Symbolleiste

Mit den Icons in der Symbolleist am rechten unteren Rand der Karte kann der Benutzer vier Aktionen auslösen.

1. Zugang zur online-Hilfe: Die online-Hilfe soll dem Benutzer helfen alle Optionen der Karte optimal nutzen zu können und Antworten auf alle möglichen Fragen zu liefern.

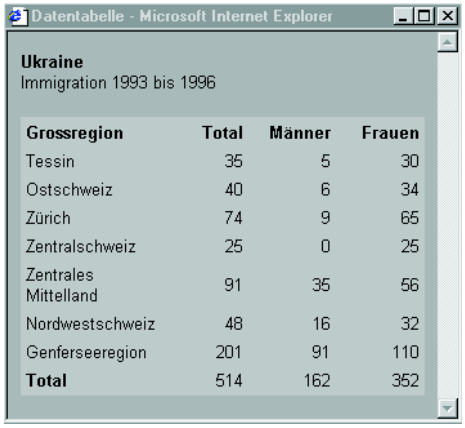
2. Anzeigen von Informationen über die Karte: Mit dem about-Icon wird in der Karte ein zusätzlicher Layer eingeblendet, in dem allgemeine Informationen über die Karte in englischer Sprache geschrieben sind. Mit einem weiteren Klick auf das Icon wird der Layer wieder ausgeblendet.

3. e-mail: Ein Klick auf dieses Icon öffnet ein Mail-Fenster des Standard-Mailprogramms, mit der e-mail Addresse des Autors und dem Vermerk *Migrations-Karte*.

4. Zugang zu online-Ressourcen bezüglich SVG: Diese Webpage soll SVG-interessierten Benutzern helfen im Internet SVG-relevante Webpages zu finden.

5.7 Pop-up Fenster

Pop-up Fenster werden gebraucht, falls eine Datentabelle oder ein Histogramm angezeigt werden soll.



Datentabelle - Microsoft Internet Explorer

Ukraine
Immigration 1993 bis 1996

Grossregion	Total	Männer	Frauen
Tessin	35	5	30
Ostschweiz	40	6	34
Zürich	74	9	65
Zentralschweiz	25	0	25
Zentrales Mittelland	91	35	56
Nordwestschweiz	48	16	32
Genferseeregion	201	91	110
Total	514	162	352

Abb. 5.6: Beispiel einer Datentabelle in einem dynamisch kreierten pop-up Fenster

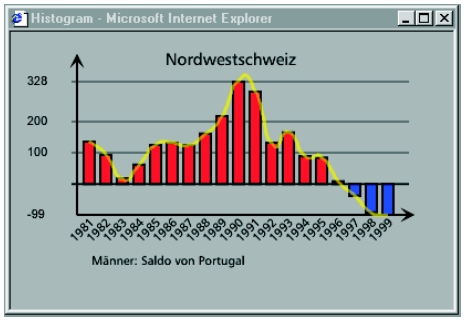


Abb. 5.7: Beispiel einer Datentabelle in einem dynamisch kreierten pop-up Fenster

Kapitel 6

Umsetzung, Programmierung

6.1 Struktur

Alle Komponenten werden in einem HTML-File eingebettet. Die eigentliche Steuerung der Karte erfolgt durch JavaScript-Code. SVG ist für die Anzeige der Karte, der Legende, der Symbolleiste und des Histogramms zuständig.

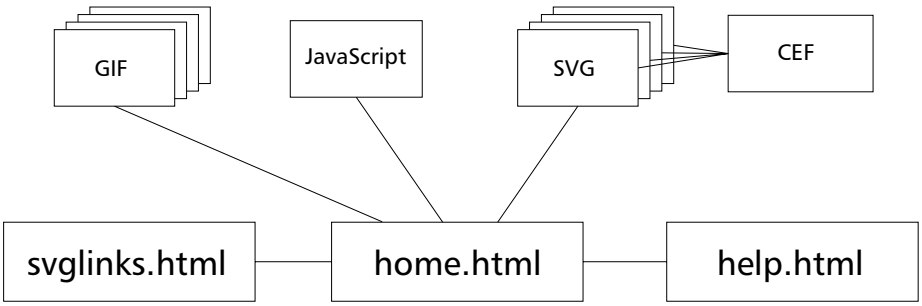


Abb. 6.1: Struktur

Das ganze Projekt umfasst 13 Dateien:

- 3 html-Files
  - home.html: html-Seite für die Kartendarstellung
  - help.html: Online-Hilfe
  - svglinks.html: SVG-Links Seite
- 1 JavaScript-File
  - scripts.js: JavaScript-Datei für die Verarbeitung und Umsetzung der Benutzereingaben, die grafischen Darstellungen sowie die Steuerung des Menüs
- 4 SVG-Files
  - europa2.svg: SVG-Karte Schweiz und Europa
  - legend.svg: SVG-Legende

  - iconRow.svg: SVG-Symbolleiste
  - histo.svg: SVG-Datei für die Histogrammdarstellung

- 1 .cef-File
  - humanist777bt.cef: Adobe Illustrator Schriftdefinition für die Schriftart Humanist777BT

- 4 Bilddateien für die dynamischen Menüs
  - pfeil_offen.gif
  - pfeil_offen_over.gif
  - pfeil_zu.gif
  - pfeil_zu_over.gif

Mit GNU Zip [21] komprimierte SVG-Dokumente weisen die Endung .svgz auf.

6.2 Benutzerführung, Aktivierung der Karte, Szenarien

Die Idee der Benutzerführung für die Karte ist, dass der Benutzer auf ein Land klicken muss, um die Karte zu aktivieren. Eine Aktivierung der Karte resultiert in der darstellerischen Umsetzung der eingestellten Variablen und Optionen. Für die Aktivierung gibt es zwei Szenarien. Im Szenario eins klickt der Benutzer auf eine Grossregion der Schweiz. Dies hat zur Folge, dass aus der Datenbank alle Migrationsdaten der Länder aus der Europakarte bezüglich der ausgewählten Grossregion aufbereitet und mit den eingestellten Optionen dargestellt werden. Im Szenario zwei klickt der Benutzer auf ein Land der Europakarte und es werden dementsprechend alle Daten der sieben Grossregionen der Schweiz bezüglich des ausgewählten Landes ausgewertet und dargestellt. Pro Grossregion und Land gibt es somit zwei Möglichkeiten die dazugehörigen Daten darzustellen, in der Schweizerkarte und in der Europakarte.

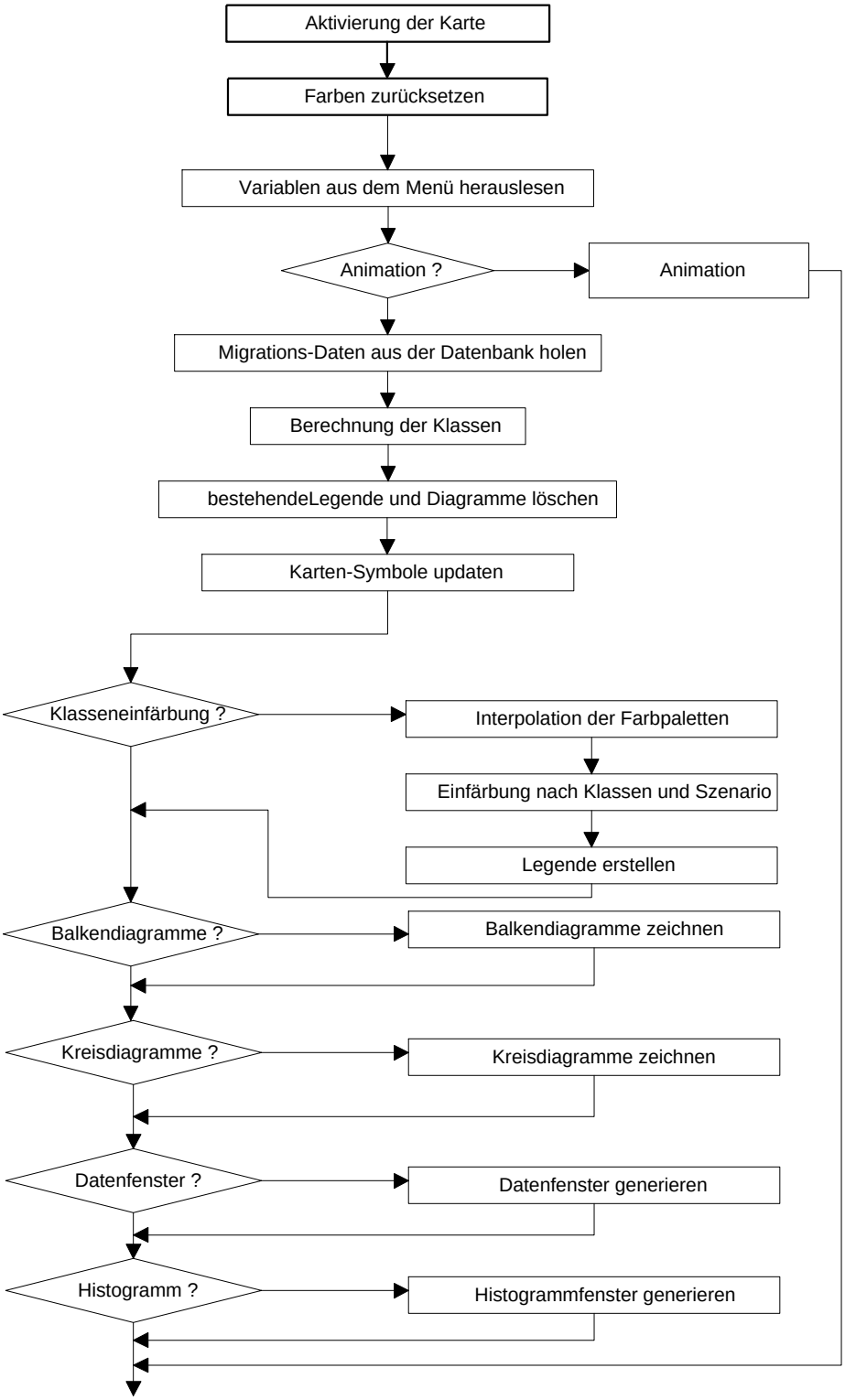


Abb. 6.2: Ablauf nach Aktivierung der Karte

6.3 Menü

Das Menü besteht aus sechs *css-layern*, jeder der drei Menüabschnitte benötigt einen *layer* für den Menükopf und einen für den Menüinhalt. Diese Layer können mit JavaScript unter anderem dynamisch plaziert und verborgen/angezeigt werden. Im Ausgangszustand sind alle Menüinhalte verborgen, nur die Titel der drei Abschnitte sind sichtbar. Mit einem Klick auf einen Menütitel lässt sich der Menüinhalt anzeigen. Zuerst werden alle unterhalb liegenden Menüs, egal ob offen oder geschlossen, so weit nach unten geschoben, dass der Inhalt des zu öffnenden Menüs genügend Platz hat. Danach wird der Inhalt des angeklickten Menüs sichtbar gemacht.

css-layer werden mit JavaScript über das DOM (Kapitel 2.2) des Browsers angesprochen. Dies erfolgt getrennt für Internet Explorer und Netscape Navigator. Im folgenden Beispiel wird ein Layer `menu` an die Position (140,210) verschoben und dann sichtbar gemacht.

Internet Explorer:

```
menu.style.left = 140;
menu.style.top = 210;
menu.style.visibility = 'visible';
```

Netscape Navigator:

```
window.document.menu.left = 140;
window.document.menu.top = 210;
window.document.menu.visibility = 'show';
```

6.4 Kommunikation JavaScript und SVG

Mit JavaScript kann über das SVG DOM direkt auf die SVG Elemente zugegriffen werden und es besteht die Möglichkeit innerhalb des bestehenden SVG-Dokuments neue Elemente zu kreieren. SVG hingegen kann nur über *event-handler* wie *onmouseover*, *onclick* usw. Funktionen im JavaScript aufrufen. Die Interaktion von JavaScript und SVG beruht also auf der Auslösung eines *events* im SVG-Dokument und der Reaktion darauf mit JavaScript.

Um mit JavaScript ein SVG-Element beeinflussen zu können muss eine Referenz auf das SVG-Dokument vorhanden sein und das Element muss über ein `id` -Attribut eindeutig identifizierbar sein.

Beispiel aus der Migrationskarte (vereinfacht), sobald der Benutzer mit der Maus über das Land Andorra fährt, wird die Füllung des Landes mit einer Transparenz von 50% versehen:

Andorra in SVG mit *event-handle*:

```
<g id="andorra" class="land"
   onmouseover="showLand('andorra')"
   <path  d="M-2337 6531l4 -22 0 -4 -33 -14 -15
             14 -1 29 9 2 29 -7 7 2z"/>
</g>
```

JavaScript-Code für die Reaktion auf den *onmouseover-event*:

```
//Referenz auf SVG, Internet Explorer
svgDoc = layer1.getSVGDocument();
//Referenz auf SVG, Netscape
svgDoc = document.layer1.document.embeds[0].getSVGDocument();

function showLand(id){
   myLand = svgEuropaDoc.getElementById(id);
   myLandStyle = myLand.getStyle();
   myLandStyle.setProperty('fill-opacity','0.5');
}
```

In der Migrationskarte verfügt jede Grossregion und jedes Land über ein `id`-Attribut und drei *event-handler* (*nmouseover*, *onmouseout* und *onclick*). Programmintern sind die Grossregionen den Zahlen 101 bis 107 und die Länder den Zahlen 1 bis 41 zugeordnet (Anhang A.3, A.4). Diese Nummern sind ein Bestandteil des `id`-Attributs im SVG-Code und werden bei jeder Auslösung eines *events* als Variable der aufgerufenen Funktion übergeben. Diese filtert zuerst die Identifikationsnummer heraus um damit zum Beispiel auf die richtigen Migrationsdatenfelder zuzugreifen oder den Region-/Landesnamen aus einem Feld zu lesen.

6.5 Dynamisch kreierte SVG-Objekte

Die Erstellung von neuen SVG-Elementen mit JavaScript ist sehr einfach umzusetzen. Es muss aber mindestens eine Objektreferenz, zum Beispiel eine Gruppe, im SVG vorhanden sein, wo die neuen Elemente angehängt werden können. Es lassen sich alle Attribute und Eigenschaften von SVG verwenden.

Folgenden Beispiele sollen das veranschaulichen:

```
// Rechteck
myElement = svgLegDoc.createElement('rect');
myElement.setAttribute('x',myX);
myElement.setAttribute('y','15');
myElement.setAttribute('width','20');
myElement.setAttribute('height','20');
myStyle = myElement.getStyle();
myStyle.setProperty('fill',classColors[i]);
myStyle.setProperty('stroke', '#000000');
myStyle.setProperty('stroke-width','1');
svgLegDoc.getElementById('quadrate').appendChild(myElement);

// Pfad
var myElement = svgEuropaDoc.createElement('path');
myElement.setAttribute('d',mensPath);
myElement.setAttribute("onmouseover","showLand('land?"+i+"')");
myElement.setAttribute("onmouseout","hideLand('land?"+i+"')");
var myStyle = myElement.getStyle();
myStyle.setProperty('fill','#0000FF');
```

```
myStyle.setProperty('fill-opacity','0.5');
myStyle.setProperty('stroke', '#000000');
myStyle.setProperty('stroke-width','5');
svgEuroPiles.appendChild(myElement);
```

Das Entfernen von dynamisch kreierten SVG-Elementen geschieht am besten mit einer `while` -Anweisung, die einfach alle Elemente einer bestimmten Gruppe, an die die Elemente angehängt wurden, wieder entfernt.

```
while((svgEuropaDoc.getElementById('piles').hasChildNodes())){
  svgEuropaDoc.getElementById('piles') ->
  ->.removeChild(svgEuropaDoc.getElementById('piles') ->
  ->.getFirstChild());
}
```

6.6 Symbolleiste

Die Symbolleiste ist ein SVG-Dokument. Jedes Icon (online-Hilfe, About-Seite, Mail schicken, SVG-Links) wurde mit *onmouseover-*, *onmouseout-* und *onclick-events* versehen. Die drei JavaScript-Funktionen `showIcon()`, `hideIcon()` und `clickIcon()` sind für die Verarbeitung dieser Aktionen zuständig. Für die online-Hilfe, das Mailicon und die SVG-Links wird mit dem JavaScript-Befehl `open()` ein bestehendes html-Dokument geöffnet oder der Standard Mailclient aufgerufen. Die About-Seite ist ein zusätzlicher versteckter Layer in der SVG-Karte. Dieser wird über den Klick auf das About-Symbol halb-transparent vor die Karte gelegt. Im Prinzip kann mit der Karte immer noch weiter gearbeitet werden, da das Kartenbild immer noch sichtbar und eine Aktivierung der Karte immer noch möglich ist. Über einen weiteren Klick auf das About-Symbol wird der Layer wieder ausgeblendet.

6.7 Aufbereitung der Migrationsdaten

Nach einer Aktivierung der Karte (Kapitel 6.2) werden zuerst die Farben aller Länder oder Regionen auf ihre Ausgangsfarbe zurückgesetzt. Danach werden mit der Funktion `varUpdate();` alle Einstellungen im Menü ausgelesen und in globale Variablen geschrieben. Dies ermöglicht nun die Aufbereitung der aktuell abgefragten Migrationsdaten. Die Daten werden über spezielle Algorithmen aus den Variablenfeldern gelesen und in zwei globale Felder geschrieben; ein Feld (`data1/data2`) enthält die totale Anzahl Migranten pro Region/Land und das andere (`men`) nur die Anzahl der männlichen Migranten pro Region/Land. Die Abfrage erfolgt nach Szenarien getrennt mit den zwei Funktionen `getData1(index);` und `getData2(index);`. Die Funktionen erhalten als Parameter den Index des angeklickten Elements (Region/Land).

6.7.1 Abfrage im Szenario eins

Im Szenario eins werden für jedes Land die Daten bezüglich der angeklickten Region abgefragt. Dies sind im ganzen 40 Informationen, doppelt so viele, falls noch nach Frauen und Männer unterschieden wird. Durch die in Kapitel 3.3.2 beschriebene Datenstruktur ergibt sich für dieses Szenario eine Abfrage innerhalb aller 80 Feldvariablen, falls nicht das Migrationssaldo verlangt ist innerhalb 40 Feldvariablen (vertikale Abfrage).

Der Abfragealgorithmus besteht aus einer dreifachen `for` -Schleife. Die erste Schleife wird für jedes Land einmal durchlaufen, die zweite Schleife wird abhängig von der Anzahl abgefragter Jahre ausgeführt und die dritte Schleife wird ein oder zwei Mal ausgeführt, je nachdem, ob Männer, Frauen oder Männer und Frauen dargestellt werden sollen. Der ganze Algorithmus wurde einmal für eine Abfrage nach Immigration oder Emmigration, und einmal für eine Abfrage nach Migrationssaldo programmiert.

Folgender JavaScript-Code ist für die Abfrage nach Immigration oder Emmigration zuständig (vereinfacht, vollständig auf der beiliegenden CD unter Code/JavaScript/getData1.txt):

```
for(i = 1; i < (landZahl+1); i++){
  gotThis = 0;
  dummy = prefix + i;
  pos = myJahr - 1979;
  for(k = 0; k < posBis; k++){
    for(j = 0; j < step; j++){
      gotThis += eval(dummy)[pos + jump + j * 19 + index * 38];
    }
    pos++;
  }
  data1[i-1] = gotThis;
}
```

Die Variable `gotThis` ist ein Zwischenspeicher. Erst wenn die beiden inneren `for` -Schleifen durchlaufen sind, wird der kumulierte Wert im Feld `data1` an der richtigen Position gespeichert.

6.7.2 Abfrage im Szenario zwei

Im Szenario zwei werden für jede Region die Daten bezüglich des angeklickten Landes abgefragt. Dies sind im ganzen 7 Informationen, doppelt so viele, falls noch nach Männern und Frauen unterschieden wird. Durch die in Kapitel 3.3.2 beschriebene Datenstruktur ergibt sich für dieses Szenario eine Abfrage innerhalb einer Feldvariablen, falls der Migrationssaldo verlangt ist innerhalb von zwei Feldvariablen (horizonale Abfrage).

Ein vereinfachtes Beispiel für die Abfrage im Szenario zwei (vollständig auf der CD unter Code/JavaScript/getData2txt):

```
for(i = 1; i < 8; i++){
  gotThis = 0;
  dummy = prefix + index;
  pos = myJahr - 1979;
  for(k = 0; k < posBis; k++){
    for(j = 0; j < step; j++){
      gotThis += eval(dummy)[pos + jump + j * 19 + (i-1) * 38];
    }
    pos++;
  }
  data2[i-1] = gotThis;
}
```

Die Algorithmen für die beiden Szenarien sind sich sehr ähnlich. Auffallend ist, dass die beiden Variablen `i` und `index` gerade vertauscht sind. Dies ist auch der entscheidende Unterschied zwischen der horizontalen und der vertikalen Abfrage.

6.8 Berechnung der Klassen

Die drei implementierten Methoden zur Berechnung der Klassen reichen in den meisten Fällen aus, um die Darstellung den eigenen Wünschen anzupassen.

6.8.1 nach Quantilen

Bei der Berechnung nach Quantilen werden jeder Klasse gleich viele Elemente zugeordnet, das Histogramm zu den Klassen wird ausgeglichen. Dadurch entsteht ein ausgewogenes Kartenbild, da jede Farbe gleich oft vorkommt.

Der Algorithmus zur Berechnung der Klassen ist relativ simpel. Das Variablenfeld mit den Daten zur aktuellen Abfrage wird zuerst sortiert. Danach wird die Anzahl Elemente des Feldes durch die Anzahl der gewünschten Klasse geteilt. Das Resultat ist eine Zahl, die die Anzahl Werte pro Klasse definiert. Nun können die Werte zum Beispiel von unten nach oben ihrer Klasse zugewiesen werden, beziehungsweise die Klassengrenzen in einem Variablenfeld gespeichert werden.

Ein Problem bei der Berechnung nach Quantilen ergibt sich, wenn viele gleichgrosse Werte im Datensatz vorhanden sind. Die Folge davon sind mehrere Klassen mit der gleichen Unter- und Obergrenze. Zum Beispiel kommt der Wert 30 in einem Datensatz mit 20 Elementen 10 Mal vor und es sind 5 Klassen vorgegeben. Daraus ergibt sich folgendes Resultat für die Klassenberechnung:

5, 7, 9, 15, 17, 30,. . . , 30, 31, 40, 41, 53, 67

Klasse 1	Klasse 2	Klasse 3	Klasse 4	Klasse 5
5, 7, 9, 15	17, 30, 30, 30	30, 30, 30, 30	30, 30, 30, 31	40, 41, 53, 67

Die Klasse 3 hat als Unter- und Obergrenze den Wert 30. Diese Klasseneinteilung scheint nur bedingt sinnvoll. Zum Beispiel könnte dem Benutzer vorgeschlagen werden, die Anzahl der Klassen zu reduzieren. Dies sollte auch in einer späteren Version der Karte implementiert werden.

6.8.2 nach gleichmässigen Stufen

Die Berechnung nach Stufen ergibt ein quantitativ korrektes Kartenbild, das heisst man erkennt schnell, wo sehr viele oder wo sehr wenige Leute migriert sind, im Verhältnis zu den anderen Ländern oder Regionen.

Die Berechnung für diese Methode ist auch relativ einfach. Das Datenfeld wird wiederum zuerst sortiert. Danach wird die Differenz zwischen dem grössten und dem kleinsten Datenelement gebildet und dieser Wert durch die Anzahl Klassen dividiert. Das Resultat ist der relative Stufenwert mit welchem die Klassengrenzen zum Beispiel durch Addition ausgehend vom kleinsten Wert bestimmt werden können.

Beispiel für die Einteilung der folgenden neun Werte in drei Klassen nach den Methoden Quantile und lineare Stufen:

5, 7, 9, 15, 17, 24, 30, 31, 90

	Klasse 1	Klasse 2	Klasse 3
Quantile	5, 7, 9	15, 17, 24	30, 31, 90
Stufen	5, 7, 9, 15, 17, 24	30, 31	90

6.8.3 manuelle Werte

Ist diese Option angewählt erscheint nach der Aktivierung der Karte ein JavaScript-Prompt mit der Aufforderung Werte für die Klassengrenzen einzugeben. Dem Benutzer werden der kleinste Wert, der grösste Wert und der Bereich angegeben. Ausserdem werden ihm die zuletzt gespeicherten Klassengrenzenwerte vorgeschlagen. Natürlich können diese Werte völlig ohne Zusammenhang mit der aktuellen Kartendarstellung sein. Aber falls der Benutzer iterativ sich an ein Kartenbild heranarbeiten will, kann er dies tun, indem er wiederholt auf die gleiche Region oder das gleiche Land klickt; in der Eingabemaske des JavaScript-Prompt findet er dann seine zuletzt eingegebenen Werte und kann diese wunschgemäss abändern.

Ein ungewünschter Nebeneffekt der manuellen Eingabe ist, dass wenn sich der Benutzer nicht an das angegebene Minimum und Maximum hält, die Diagramme falsch dargestellt werden. Diese beziehen sich nämlich auf das Variablenfeld, in dem die Klassengrenzen gespeichert sind. In einer nächsten Version der Karte sollte dieser Umstand allerdings eliminiert sein.

Die Lösung mit dem JavaScript-Prompt ist aber keineswegs befriedigend. Wünschenswert wäre eine grafisches Interface für alle drei Berechnungsmethoden mit einer dynamischen Histogrammdarstellung, wie es zum Beispiel im interaktiven Atlas der Schweiz [16] implementiert wurde. Der Aufwand um dieses grafische Interface umzusetzen war aber zu gross, für eine zukünftige Version der Karte aber sicher denkbar.

6.8.4 Spezialfall: Migrationssaldo

Ein Spezialfall für die Berechnungen ist, wenn das Migrationssaldo dargestellt werden soll, weil im Datenfeld dann auch negative Werte stehen. Eine Klasse, die zum Beispiel durch den unteren Wert -50 und den oberen Wert 27 definiert ist, macht nicht viel Sinn. Für ein Migrationssaldo ist nur interessant, wo es Abwanderungen gab und wo Zuwanderungen. Darum wurde eine zusätzliche Funktion eingeführt, die die kleinste absolute Klassengrenze auf Null setzt.

6.9 Farbinterpolation

Für die Einfärbung der Klassen werden die Farbwerte für die Klassen nach jeder Aktivierung der Karte abhängig von der Anzahl Klassen neu berechnet. Die vordefinierten Farbpaletten sind somit nur nurch durch eine Start- und eine Endfarbe definiert.

6.9.1 RGB-Interpolation

In der aktuellen Version der Karte werden die Farbwerte im RGB-Raum interpoliert. RGB steht für Rot-Grün-Blau und ist ein oft verwendetes Farbformat, zum Beispiel in HTML, CSS und SVG (als hexadezimale Zahlen). Für die drei Komponenten können jeweils Werte zwischen 0 und 255 angegeben werden. Somit können mehr als 16 Mio. Farbtöne definiert werden.

Die Farbwerte werden seperat für die drei Komponenten linear interpoliert. Zum Beispiel wird zuerst die Differenz zwischen dem Rotanteil der Endfarbe und dem der Startfarbe durch die Anzahl Klassen geteilt und so die Rotanteile für die Zwischenfarben bestimmt. Danach folgen die Interpolationen für die Grün- und Blauanteile.

Mit der Funktion `decToHex()` werden die berechneten Farbwerte in das hexadezimale Format umgerechnet. Die Farbwerte können erst in diesem Format den SVG-Elementen zugewiesen werden.

6.9.2 Erweiterungen

Die Eingabe von RGB-Werten ist nicht sehr benutzerfreundlich, es wäre wünschenswert die Farben über ein grafisches Interface einstellen zu können. Das Resultat der Interpolation könnte in diesem Interface auch immer direkt betrachtet werden, bevor die Karte eingefärbt würde. Ausserdem wäre eine Kombination mit dem Interface für die manuelle Klasseneinteilung denkbar. Im interaktiven Atlas der Schweiz [16] wurde das in dieser Weise implementiert.

Der RGB-Raum an sich ist nicht sehr intuitiv, da eine Farbe aus einer Mischung von drei Anteilen zusammengesetzt ist. Eine Alternative dafür wäre die Farbauswahl im HLS-Raum. HLS bedeutet Hue (Farbton), Lightness (Aufhellung) und Saturation (Sättigung). Dabei wird die Farbe nur mit dem H-Wert definiert, mit den L- und S-Werten wird die ausgwählte Farbe selber nicht mehr verändert, nur in der Aufhellung und der Sättigung angepasst.

6.10 Legende erstellen

Die Legende wird aus den berechneten Klassen und Farbwerten aufgebaut. Die Darstellung erfolgt durch dynamisch kreierte Rechtecke, in der entsprechenden Klassenfarbe eingefärbt und mit dem entsprechenden Legendentext kommentiert. Die Erstellung dynamischer SVG-Elemente wird im Kapitel 6.5 beschrieben.

6.11 dynamische Symbole

Mit den dynamischen Symbolen werden die Zustände der Menüpunkte von Jahr und bis Jahr, Geschlecht sowie Richtung des Variablen-Menüs in der Karte direkt durch Symbole dargestellt. Die Jahreszahlen werden direkt als Text einem Textfeld angehängt. Die Symbole für das Geschlecht und die Wanderungsrichtung sind versteckt schon im SVG-Dokument vorhanden und werden bei Bedarf eingeblendet.

6.12 Balkendiagramme

Die Balkendiagramme zeigen direkt die Verhältnisse zwischen der Anzahl Migranten für die einzelnen Länder. Nur aus der klassenweisen Einfärbung der Länder und Regionen können diese Verhältnisse nicht erkannt werden. Ausserdem kann mit den Balkendiagrammen auch schnell erkannt werden, wo posititve und wo negative Migrationssaldi vorkommen. Die Balkendiagramme sind somit eine gute Ergänzung zu der klassenweisen Einfärbung.

Die Balkendiagramme werden mit dynamischen SVG-Elementen aufgebaut. Ein Balken besteht aus einer Vorderseite und einer rechten Seite und einem Deckel, um

einen gewissen 3D-Eindruck in die Karte zu bringen. Aus den darzustellenden Migrationsdaten wird das Maximum und das Minimum bestimmt und der ganze darzustellende Bereich auf eine Länge von 1000 Bildschirmeinheiten für das Balkendiagramm skaliert. Bei der Darstellung des Migrationssaldos, werden die negativen Werte mit einer anderen Farbe dargestellt. Es wäre denkbar einen Minimumbalken einzuführen, falls die Anzahl der Migranten für ein Land oder eine Region einen gewissen Prozentsatz des Maximums unterschreitet.

Alle dynamisch erstellten Balkendiagramme werden mit den *event handlern* `onmouseover` und `onmouseout` des entsprechenden Landes oder der Region versehen. Somit wird die dynamische Anzeige der Länder- und Regionennamen durch die Darstellung der Diagramme nicht behindert.

Die Legende wird links unten in der SVG-Karte erstellt. Dargestellt wird ein Balken, der den ganzen Bereich vom Minimal- bis zum Maximalwert repräsentiert. Eine Erweiterung für die Legende wäre die Einführung einer mittleren Grössenangabe, zum Beispiel einer runden Zahl.

6.13 Kreisdiagramme

Durch die Kreisdiagramme soll in erster Linie das Verhältnis zwischen männlichen Migranten und weiblichen Migranten dargestellt werden. Dies geschieht durch zwei Kreissektoren, die blaue Fläche entspricht den männlichen Migranten, die rote Fläche den weiblichen Migranten. Zudem entspricht die gesamte Kreisfläche der totalen Anzahl Migranten aus der entsprechenden Region oder des entsprechenden Landes. Um zu garantieren, dass das Verhältnis zwischen der blauen und der roten Fläche immer erkennbar ist, darf der Kreis nicht beliebig klein werden. Alle Werte die kleiner als 5% des Maximalwertes sind, werden in der Grösse von 5% des Maximalwertes dargestellt.

Die Kreisdiagramme werden, wie die Balkendiagramme, mit dynamischen SVG-Elementen aufgebaut. Leider lässt es sich nicht vermeiden, dass sich die Kreise zum Teil überschneiden, darum werden sie nur halb-transparent gezeichnet. Dadurch können alle Kreise trotz Überschneidungen erkannt werden. Auch die Länder und Regionen werden so durch die Kreise nicht komplett verdeckt.

Alle dynamisch erstellten Kreisdiagramme werden mit den *event handlern* `onmouseover` und `onmouseout` des entsprechenden Landes oder der Region versehen. Somit wird die dynamische Anzeige der Länder- und Regionennamen durch die Darstellung der Diagramme nicht behindert.

Die Legende zu den Kreisdiagrammen wird links unten angezeigt, neben dem Minimal- und dem Maximalkreis wird noch ein mittlerer Kreis gezeichnet, dessen Fläche einer runden Anzahl Migranten entspricht.

6.14 Datenfenster

Das Datenfenster ist ein pop-up Fenster, das eine Tabelle mit allen Daten zur aktuellen Abfrage enthält. Bei einem Klick auf die Schweiz erscheint also eine Tabelle mit den Angaben zur Migration bezüglich aller 40 Länder. Bei einem Klick auf ein Land enthält die Tabelle die Migrations-Angaben zu den sieben Grossregionen. Das Tabellenfenster schliesst sich wieder, sobald es den Fokus verliert, zum Beispiel wenn eine Änderung im Menu gemacht wird.

Da im Datenfenster die Angaben zu den männlichen und den weiblichen Migrationen erscheinen, ist unter Umständen nochmals eine Aufbereitung der Daten notwendig. Die Tabelle wird mit JavaScript (`document.write()`-Methode) direkt ins pop-up Fenster geschrieben. Um zu verhindern, dass Netscape abstürzt, muss nach der Initialisierung des Fensters, eine kurze Zeit gewartet werden, bevor ins Dokument geschrieben wird. Darum sind zwei Funktionen für den Aufruf des Datenfensters nötig, eine zum Initialisieren des Fensters (`dataLos()`) und eine zum verzögerten Schreiben der Tabelle (`dataLosJetzt()`). Die Verzögerung erfolgt mit einem Aufruf der zweiten Funktion aus der ersten mit einer `window.setTimeOut()`-Anweisung.

6.15 Histogramm

Mit der Histogramm-Option kann in einem pop-up Fenster ein Histogramm der Migration eines oder aller Länder Europas bezüglich einer Grossregion der Schweiz von 1981 bis 1999 angezeigt werden. Dazu muss im Menü das gewünschte Land eingestellt werden und dann auf eine Region der Schweiz geklickt werden. Das Fenster schliesst sich, sobald es den Fokus verliert. Das Histogramm-Feature wurde nicht für beide Szenarien entwickelt.

Dem Histogramm-Fenster wird bei jedem Aufruf ein SVG-Dokument (histo.svg) zugewiesen, danach werden die neu aufbereiteten Migrationsdaten mit dynamischen SVG-Elementen dargestellt. Analog zum Datenfenster muss das Histogrammfenster zuerst initialisiert werden, nach einer kurzen Zeit kann das SVG-Dokument geladen werden, und nochmals nach einer Verzögerung dürfen erst die dynamischen SVG-Elemente in das SVG-Dokument geschrieben werden, sonst ist mit einem Absturz des Netscape Navigator zu rechnen. Diese Verzögerungen äussern sich durch den sichtbaren Aufbau der dynamischen SVG-Elemente.

Mit dem kürzlich erschienen SVG Browser-Plugin 2.0beta werden leider die einzelnen Balken nicht angezeigt. Die interpolierte Kurve erscheint allerdings trotzdem. Die Ursache für diesen Effekt konnte aus Zeitgründen nicht mehr bestimmt werden.

6.16 Animation

Die Option Animation löst nach einer Aktivierung der Karte eine animierte Darstellung der Karte aus. Im Sekundentakt wird die Karte für die Jahre von 1981 bis 1999 eingefärbt, abhängig von den Menüeinstellungen und dem angegklickten Land oder der angeklickten Region. Die Jahreszahlen werden während der Animation in grosser Schrift links unten in der Karte angezeigt.

Realisiert wird die Animation einzig mit JavaScript, und zwar mit der Methode `setTimeOut()` des `window`-Objekts. Die Funktionen `move1()` und `move2()` rufen sich mit einer Verzögerung von einer Sekunde solange selber wieder auf, bis die 19 Jahre von 1981 bis 1999 durchlaufen sind. Dabei wird für jedes Jahr eine neue Klasseneinteilung berechnet. Um die verschiedenen Zustände besser vergleichen zu können wäre eine gesamthafte Klassenberechnung, die alle Jahre zusammen berücksichtigt, besser.

SVG bietet auch einige Animationsmöglichkeiten, für diesen speziellen Fall eignete sich eine reine JavaScript-Animation aber besser.

Kapitel 7

Schlussbemerkungen

7.1 Ziele

Die zu Beginn des Projekts gesteckten Ziele wurden grösstenteils erreicht. Die Karte funktioniert und verfügt über zahlreiche Darstellungsmöglichkeiten für die Migrationsdaten. Der Zeitplan für die Fertigstellung der Diplomarbeit konnte gut eingehalten werden. Es gab nur geringe Verzögerungen durch die Beschaffung und Anpassung der Grundlagendaten für das Projekt, nämlich der Geometriedaten (Schweiz und Europa) und der Migrationsdaten.

Die Geometriedaten standen schon bald nach Beginn der Arbeit zur Verfügung und konnten relativ schnell auf das Projekt angepasst werden. Die Umwandlung in SVG-Code und die Einbindung ins System verlief auch ohne Probleme. Aber da die Migrationsdaten erst ein paar Wochen nach den Geometriedaten zur Verfügung standen, waren im Nachhinein trotzdem noch einige grössere Änderungen nötig.

Die Migrationsdaten wurden vom Bundesamt für Statistik [15] in zwei Etappen geliefert. Sehr bald nach der Anfrage stand schon ein erster Teil mit den Daten von 1991 bis 1999 zur Verfügung. Dadurch konnte das Projekt auf diesen Daten aufgebaut werden. Der zweite Teil wurde kurz vor Weihnachten geliefert und konnte Dank der schon für den ersten Teil entwickelten Umwandlungstools sehr schnell ins Projekt eingebaut werden.

Ausserdem verlief die Umsetzung der Karte mit JavaScript relativ gut und im Zeitplan. Die grundlegenden Funktionen der Karte waren schon nach wenigen Wochen umgesetzt. Auf diesem Grundgerüst konnten die weiteren Elemente einzeln aufgebaut werden. Probleme entstanden vor allem durch die Implementierung der Karte für beide vorgegebenen Browser, den Internet Explorer und den Netscape Navigator. Schwierigkeiten gab es mit dem Navigator Version 4.7. Schlussendlich konnten aber fast alle Funktionen der Karte für diese zwei Browser implementiert werden.

Die SVG-Karte könnte an verschiedenen Stellen noch optimiert und verbessert werden. Folgende Liste enthält einige Anregungen und Vorschläge:

Klasseneinteilung Einteilung der Klassen anhand eines Histogramms für die Verteilung der Daten, grafisches User Interface, am besten kombiniert mit Farbeinstellungen.

Farbauswahl grafisches User Interface, am besten kombiniert mit Einteilung der Klassen. Erweiterung von RGB- auf HLS-Farbraum.

Migrationssaldi Verwendung einer bipolaren Farbskala.

Animationen Berechnung einer einheitlichen Klasseneinteilung für alle Jahre, Veränderungen könnten qualitativ besser verfolgt werden.

Daten Verwendung von *XML Query* für den Datenzugriff, Daten in XML File gespeichert.

Warn-Symbole Die Warn-Symbole für den Fall falscher Darstellungen (beschrieben im Kapitel 3.2.2) sind noch nicht implementiert.

7.2 Schlussfolgerungen und Ausblick

Die Kombination von SVG und JavaScript eignet sich sehr gut für die Erstellung interaktiver und dynamischer Darstellungen. Den Gestaltungs-Möglichkeiten sind fast keine Grenzen gesetzt, SVG kann fast alles darstellen, was ein professionelles Grafikprogramm auch kann. In SVG können die Grafikelemente aber zusätzlich interaktiv und dynamisch sein. Dies eröffnet nicht nur für die Kartographie enorme Möglichkeiten, sonder allgemein für die Visualisierung von Daten oder natürlich auch das Webdesign.

Probleme ergaben sich vor allem durch die Anpassung des Projekts auf verschieden Browser-Plattformen, speziell beim Netscape Navigator Version 4.xx. Das erst kürzlich in Vollversion erschienene Netscape 6 soll in dieser Beziehung allerdings einige Fortschritte gegenüber seinem Vorgänger gemacht haben.

Die Arbeit mit SVG und JavaScript macht Spass, man erzielt schnell und ohne grossen Aufwand die ersten Resultate und kann das System dann Stück für Stück erweitern. Allerdings stosst man mit JavaScript zum Teil an technische Grenzen. Zum Beispiel ist der Datenzugriff alles andere als ideal oder es existiert noch keine gute Entwicklungsumgebung. Auch bezüglich der Erweiterbarkeit des Projekts gäbe es sicher Alternativen. Für den Rahmen dieser Diplomarbeit haben sich SVG und JavaScript bewährt und es konnten zufriedenstellende Resultate erreicht werden. Ob sich für grössere Projekte andere Techniken besser eignen ist noch abzuklären.

SVG wird bald ein offizieller Standard des W3C und wird sich als Grafikstandard etablieren. SVG ist relativ einfach zu verstehen und kann in verschiedensten Bereichen verwendet werden. Die Arbeit mit SVG im Rahmen einer Diplomarbeit oder eines Vertiefungsblockes kann daher nur weiterempfohlen werden.

Anhang A

JavaScript

A.1 Code

JavaScript-Code wurde zum Teil direkt im HTML-Code integriert. Alle Funktionen für die Steuerung der Karte wurden im File scripts.js zusammengefasst. Dieses ist mit dem `src`-Attribut im HTML-Code referenziert. Der gesamte Code ist auf der beigelegten CD unter Code/JavaScript/ zu finden.

A.2 Liste aller Funktionen

function clickIcon(id) *onclick-events* für die einzelnen Elemente der SVG-Symbolleiste. Es wird eine Aktion entsprechend dem betroffenen Symbol ausgelöst. Kapitel 6.6.

function clickLand(id) Wird bei einem *onclick-event* für ein Land Europas aufgerufen. Dies entspricht einer Aktivierung der Karte mit dem Szenario zwei. Der Ablauf nach einer Aktivierung der Karte ist ausführlich im Kapitel 6 beschrieben und in der Abbildung 6.2 dargestellt.

function dataLos(index,szenario) Funktion für die Initialisierung des Datenfensters, danach Aufruf der Funktion `dataLosJetzt()`, in der das Datenfenster mit Inhalten gefüllt wird. Diese Vor-Funktion ist notwendig, um einen Absturz von Netscape Navigator zu verhindern. Kapitel 6.14.

function dataLosJetzt(index,szenario) Generierung einer Datentabelle zur aktuellen Abfrage in einem pop-up Fenster. Kapitel 6.14.

function decToHex(dec) Routine für die Umwandlung von dezimalen RGB-Werten in hexadezimale Farbwerte. Kapitel 6.9.1.

function defineColors(nullValue) Funktion für die Interpolation der Farbwerte abhängig von der Anzahl Klassen und der ausgewählten Farbpalette. Der Parameter `nullValue` wird momentan nicht benutzt. Kapitel 6.9.1.

function diaOver() *onmouseover-event* für die Titelzeile des Optionen -Menüs. Die Schrift wird rot eingefärbt und das Dreieck-Symbol mit einem roten Symbol ausgetauscht.

function diaOut() *onmouseout-event* für die Titelzeile des Optionen -Menüs. Die Schrift und das Dreieck-Symbol werden in den Ausgangszustand zurückversetzt.

function diaClick() *onclick-event* für die Titelzeile des Optionen -Menüs. Je nach Status des Menüs wird es auf- oder zugeklappt.

function evaluate(id) Wird bei einem *onclick-event* für eine Region der Schweiz aufgerufen. Dies entspricht einer Aktivierung der Karte mit dem Szenario eins. Der Ablauf nach einer Aktivierung der Karte ist ausführlich im Kapitel 6 beschrieben und in der Abbildung 6.2 dargestellt.

function fixVal(szenario) Berechnet die Stufenwerte für die Klasseneinteilung nach linearen Abstufungen abhängig vom Szenario, das über den Parameter `szenario` bekannt ist. Die Berechnung der linearen Abstufung wird ausführlich im Kapitel 6.8.2 beschrieben.

function getData1(index) Startet eine Abfrage der Migrationsdaten im Szenario eins für die Region mit der Identifikationsnummer (Anhang A.3), die im Parameter `index` übergeben wurde. Kapitel 6.7.1

function getData2(index) Startet eine Abfrage der Migrationsdaten im Szenario zwei für das Land mit der Identifikationsnummer (Anhang A.4), die im Parameter `index` übergeben wurde. Kapitel 6.7.2

function getUserColor() Von dieser Funktion aus wird ein JavaScript-Prompt aufgerufen, falls der Benutzer manuelle Farbpaletten generieren will. Die eingegebenen Werte werden in das entsprechende Feld für die Farbpaletten geschrieben.

function hideIcon(id) *onmouseout-events* für die einzelnen Elemente der SVG-Symbolleiste. Die Linien des betroffenen Symbols werden wieder schwarz eigefärbt. Kapitel 6.6.

function hideLand(id) Wird aufgerufen bei einem *onmouseout-event* für ein Land Europas. Als Parameter wird der Text `land?` und daran angehängt die Identifikationsnummer des Landes (Kap. A.4) übergeben. Die Füllung des Landes wird auf Opazität 100% zurückgesetzt und das Textfeld für die Anzeige der Namen der Länder wieder gelöscht.

function hideRegio(id) Wird aufgerufen bei einem *onmouseout-event* für eine Region der Schweiz. Als Parameter wird der Text `regio?` und daran angehängt die Identifikationsnummer der Region (Kap. A.3) übergeben. Die Füllung der Region wird auf Opazität 100% zurückgesetzt und das Textfeld für die Anzeige der Namen der Regionen wieder gelöscht.

function histoLos(index,szenario,item) Funktion für die Initialisierung des Histogrammfensters, danach Aufruf der Funktion `histoLosJetzt()`, in der das Fenster mit Inhalten gefüllt wird. Diese Vor-Funktion ist notwendig, um einen Absturz von Netscape Navigator zu verhindern. Kapitel 6.15.

function histoLosJetzt(index,szenario,item) Generierung eines Histogramms zu dem im Menü eingestellten Element (Parameter `item`) in einem pop-up Fenster. Kapitel 6.15.

function init() Wird aufgerufen, sobald die HTML-Seite geladen ist. Schreibt Referenzen auf die verschiedenen SVG-Dokumente und auf oft benützte SVG-Elemente in globale Variablen. Die Referenzierung muss jeweils getrennt nach Internet Explorer und Netscape Navigator erfolgen.

function klasOver() *onmouseover-event* für die Titelzeile des Klassen -Menüs. Die Schrift wird rot eingefärbt und das Dreieck-Symbol mit einem roten Symbol ausgetauscht.

function klasOut() *onmouseout-event* für die Titelzeile des Klassen -Menüs. Die Schrift und das Dreieck-Symbol werden in den Ausgangszustand zurückversetzt.

function klasClick() *onclick-event* für die Titelzeile des Klassen -Menüs. Je nach Status des Menüs wird es auf- oder zugeklappt und das sich weiter unten befindende Menü dem neuen Zustand angepasst.

function manuClass(szenario) Über einen JavaScript-Prompt (Eingabemaske) kann der Benutzer eigene Klassengrenzen eingeben. Diese Funktion wird im Kapitel 6.8.3 ausführlich beschrieben.

function numerisch(a,b) Notwendige Funktion für die *sort* -Methode, aufgerufen aus den Funktionen `quantile()`, `fixVal()` und `manuClass()`. Kapitel 6.8.

function makeLeg() Erstellt die Legende zur aktuellen Klasseneinteilung. Kapitel 6.10.

function move1(index,szenario) Funktion für die Animation im Szenario eins, die Funktion ruft sich nach jeweils einer Sekunde selber wieder auf, bis alle Jahre durchlaufen sind. Kapitel 6.16.

function move2(index,szenario) Funktion für die Animation im Szenario zwei, die Funktion ruft sich nach jeweils einer Sekunde selber wieder auf, bis alle Jahre durchlaufen sind. Kapitel 6.16.

function pileEm(szenario,nullValue) Erstellt die Balkendiagramme sowie die dazugehörige Legende in der SVG-Karte. Kapitel 6.12.

function quantile(szenario) Berechnet die Stufenwerte für die Klassen nach Quantilen abhängig vom Szenario, das über den Parameter `szenario` bekannt ist. Die Berechnung nach Quantilen wird ausführlich im Kapitel 6.8.1 beschrieben.

function removePiles() Löscht alle dynamisch kreierten Balken- und Kreisdiagramme sowie die dazugehörige Legende.

function removeLeg() Löscht alle dynamisch kreierten Elemente im SVG-Dokument der Legende.

function reset(szenario) Setzt alle Flächen der Länder oder Regionen, je nach Szenario, auf ihre Ausgangfarbe zurück.

function roundEm(szenario,nullValue) Erstellt die Kreisdiagramme und die dazugehörige Legende in der SVG-Karte. Kapitel 6.13.

function setNull() Wird von den Funktionen `quantile()` und `fixVal()` aufgerufen, um im Falle einer Abfrage nach dem Migrationssaldo, die Klassenabstufung so zu verändern, dass der kleinste Absolutwert auf Null gesetzt wird. Kapitel 6.8.4.

function showIcon(id) *onmouseover-events* für die einzelnen Elemente der SVG-Symbolleiste. Die Linien des betroffenen Symbols werden rot eigefärbt. Kapitel 6.6.

function showLand(id) Wird aufgerufen bei einem *onmouseover-event* für ein Land Europas. Als Parameter wird der Text `land?` und daran angehängt die Identifikationsnummer des Landes (Anhang A.4) übergeben. Die Nummer wird aus der Variable `id` herausgefiltert und damit der Name des Landes aus dem Feld `landNamen` bestimmt. Die Füllung des Landes wird auf Opazität 50% gesetzt und der Name des Landes im Textfeld links oben in der Karte angezeigt.

function showRegio(id) Wird aufgerufen bei einem *onmouseover-event* für eine Region der Schweiz. Als Parameter wird der Text `regio?` und daran angehängt die Identifikationsnummer der Region (Anhang A.3) übergeben. Die Nummer wird aus der Variable `id` herausgefiltert und damit der Name der Region aus dem Feld `regioNamen` bestimmt. Die Füllung der Region wird auf Opazität 50% gesetzt und der Name der Region im Textfeld links oben in der Karte angezeigt.

function symbols() Passt die dynamischen Symbole in den Ecken der Schweizer Karte den Werten der herausgelesenen Variablen an. Kapitel 6.11.

function tintCh() Routine für die Einfärbung der Schweizer Grossregionen nach den berechneten Klassen.

function tintEu() Routine für die Einfärbung der Länder Europas nach den berechneten Klassen.

function varOver() *onmouseover-event* für die Titelzeile des Variablen -Menüs. Die Schrift wird rot eingefärbt und das Dreieck-Symbol mit einem roten Symbol ausgetauscht.

function varOut() *onmouseout-event* für die Titelzeile des Variablen -Menüs. Die Schrift und das Dreieck-Symbol werden in den Ausgangszustand zurückversetzt.

function varClick() *onclick-event* für die Titelzeile des Variablen -Menüs. Je nach Status des Menüs wird es auf- oder zugeklappt und die sich weiter unten befindenden Menüs werden dem neuen Zustand angepasst.

function varUpdate() Nach einer Aktivierung der Karte wird diese Funktion aufgerufen, um die eingestellten Werte aus dem Menü zu lesen und in globalen Variablen zu speichern. Die Abfrage der Werte muss jeweils getrennt nach Internet Explorer und Netscape Navigator erfolgen.

A.3 Identifikationsnummern Grossregionen

Nr.	Land
101	Tessin
102	Ostschweiz
103	Zürich
104	Zentralscheiz
105	Zentrales Mittelland
106	Nordwestschweiz
107	Genferseeregion

A.4 Identifikationsnummern Länder

Nr.	Land	Nr.	Land
1	Ex-Yugoslawien	22	Italien
2	Grossbritannien	23	Irland
3	Ukraine	24	Island
4	Türkei	25	Ungarn
5	Afrika	26	Griechenland
6	Asien	27	Finnland
7	Slowakei	28	Frankreich
8	Schweden	29	Spanien
9	Russland	30	Estland
10	Rumänien	31	Dänemark
11	Portugal	32	Deutschland
12	Polen	33	Tschechien
13	Norwegen	34	Zypern
14	Niederlande	35	Schweiz
15	Malta	36	Weissrussland
16	Moldawien	37	Bulgarien
17	Lettland	38	Belgien
18	Luxemburg	39	Oesterreich
19	Litauen	40	Albanien
20	Liechtenstein	41	Andorra
21	Ex-Sowjetunion		

A.5 Entwicklungsumgebung

Der JavaScript-Code wurde mit dem Texteditor UltraEdit 32 Version 7.00a entwickelt.

Anhang B

SVG *scalable vector graphics*

B.1 Code

Für das Projekt wurden vier SVG-Dokumente verwendet. Der SVG-Code der einzelnen Dokumente ist auf der beigelegten CD unter Code/SVG/ zu finden.

B.2 europa2.svg

Dieses Dokument enthält alle grafischen Daten für Europa und die Schweiz inkl. eventhandler. Dynamische Symbole, Textanker und Anker für die Erstellung dynamischer Elemente sind ebenfalls Teil dieses Dokuments. Für die einheitliche Formatierung der Elemente wurden zu Beginn des Dokuments einige CSS-Angaben definiert.

B.3 legend.svg

Das Ausgangsdokument für die dynamische Legende enthält vor allem das Grundgerüst für ein SVG-Dokument und die Textanimation (bei Neuladen der Karte). Zusätzlich muss der sichtbare Hintergrund mit einer Fläche mit der Farbe des Hintergrundes des HTML-Dokuments ausgefüllt werden. Sonst füllt der Netscape Browser diese Fläche weiss, im Internet Explorer würde der Hintergrund auch ohne diese Fläche gleich wie der Hintergrund des HTML-Dokuments erscheinen.

B.4 iconRow.svg

Die Definition der vier Icons der Symbolleiste sind der Hauptbestandteil dieses SVG-Dokumentes. Wiederum wird für Netscape zu Beginn der Hintergrund in Form eines Rechtecks definiert.

B.5 histo.svg

Dieses SVG-Dokument bildet das Grundgerüst für die Histogramm-Darstellung. Das beinhaltet Koordinatenlinien und die horizontale Skala für die Jahre 1981 bis 1999

sowie zwei Textanker und einen Anker für die Balken und die interpolierte Linie. Wiederum wird für Netscape zu Beginn der Hintergrund in Form eines Rechtecks definiert.

B.6 Entwicklungsumgebung

Der SVG-Code wurde mit dem Texteditor UltraEdit 32 Version 7.00a zum Teil entwickelt und bearbeitet. Zum Teil wurden auch SVG-Dokumente aus Corel Draw 9 oder Adobe Illustrator mit dem entsprechenden Exportfilter generiert (Anhang D.2).

Anhang C

HTML

C.1 Code

Das Projekt umfasst drei HTML-Dokumente. Der HTML-Code ist auf der beigelegten CD unter Code/HTML/ zu finden.

C.2 home.html

Dies ist das Basis-HTML-Dokument. Es enthält den Aufbau des Dokuments und das Menü. Darin eingebettet sind die SVG-Dokumente und das JavaScript-File.

C.3 help.html

Die online-Hilfe ist im Prinzip eine HTML-Version dieses Berichts.

C.4 svglinks.html

Der SVG-interessierte User findet hier Hyperlinks zu anderen SVG-relevanten HTML-Seiten.

C.5 Entwicklungsumgebung

Der HTML-Code wurde direkt im dem Texteditor UltraEdit 32 Version 7.00a entwickelt.

Anhang D

Software

D.1 online-Ressourcen

Die Karte ist online unter folgender URL zu betrachten:

http://www.karto.ethz.ch/~jorayc/migration/

Dieser Bericht ist im pdf-Format auf der Homepage des Instituts für Kartographie online verfügbar:

http://www.karto.ethz.ch/

Unter der gleichen URL können auch alle zum Projekt gehörenden Dateien in einem zip-File heruntergeladen werden.

D.2 Software

SVG Browser-Plugin
Für Windows und Mac (IE 4.0 - 5.0 und NN 4.0 - 4.73)
Download unter *http://www.adobe.com/svg* [1]

SVG Export-Filter für Corel Draw 9 & Adobe Illustrator 8
Corel Draw: *http://venus.corel.com/nasapps/DrawSVGDownload/index.html* [2]
Adobe Illustrator: *http://www.adobe.com/svg* [1]

UltraEdit-32 Version 7.00a
Editor für JavaScript, SVG und Latex

CorelDraw 9
Erstellung und Anpassung der grafischen Elemente, sowie SVG-Export

Adobe Illustrator 9.0
Erstellung und Anpassung der grafischen Elemente, sowie SVG-Export

Internet Explorer 5.0
Plattform für die der SVG-Karte

Netscape Navigator 4.7
Plattform für die der SVG-Karte

Micrografx Flowcharter
Erstellung von Flussdiagrammen für die Visualisierung der Programmabläufe

Windows NT 4.0
Betriebsystem

D.3 Systemanforderungen

Windows
Die SVG-Karte sollte auf allen Windows-Systemen funktionieren, falls das SVG-Plugin installiert ist. Ältere Browser-Versionen (IE unter 4.xx, Netscape unter 4.xx) werden durch den SVG-Viewer nicht unterstützt.

Mac
Das SVG-Plugin stellt die Karte im Netscape Navigator korrekt dar, einzig die Histogramme funktionieren nicht. Es muss darauf geachtet werden, dass dem Navigator vor dem Aufruf der Karte genügend Speicher zur Verfügung gestellt wird. Im Internet Explorer wird die Karte zwar korrekt dargestellt, aber JavaScript wird im Plugin nicht unterstützt. Es kann darum keine Interaktion mit der Karte stattfinden.

Linux
Leider existiert noch kein SVG-Viewer für Linux.

Literaturverzeichnis

[1] Adobes SVG-Page *http://www.adobe.com/svg*

[2] Corels SVG-Page *http://www.corel.com*

[3] W3C *World Wide Web Consortium http://www.w3.org/*

[4] W3C, HTML Seite *http://www.w3.org/MarkUp/*

[5] W3C, XML Seite *http://www.w3.org/XML*

[6] W3C, SVG Seite *http://www.w3.org/SVG*

[7] W3C, SVG 1.0 Spezifiktation
http://www.w3.org/TR/2000/CR-SVG-20001102/index.html

[8] W3C, SVG DTD
http://www.w3.org/TR/2000/CR-SVG-20001102/svgdtd.html

[9] W3C, XML-Schema *http://www.w3.org/XML/Schema*

[10] W3C, CSS *http://www.w3.org/Style/CSS/*

[11] W3C, DOM *http://www.w3.org/Style/DOM/*

[12] David Flanagan: JavaScript, The Definitive Guide, Third Edition, O'Reilly, 1998

[13] carto:net *http://www.carto.net*

[14] HTML-Einführung *http://www.teamone.de/selfhtml/*

[15] Bundesamt für Statistik *http://www.statistik.admin.ch*

[16] Atlas der Schweiz *http://www.atlasderschweiz.ch*

[17] André M. Winter, OECD-Atlas Europa (SVG)
http://www.carto.net/papers/svg/eu/oecdatlas.html

[18] Andreas Neumann, Kartogramm
http://www.karto.ethz.ch/ an/cartography/vienna/

[19] Swiss Publisher CD, copyright 1996, Digipress GmbH, Zurich, ISBN 3-905390-01-9

[20] Macromedia Homepage *http://www.macromedia.com*

[21] GNU Zip Homepage *http://www.gzip.org/*

[22] Komodo by activestate *http://www.activestate.com*