Dynamische Landkarten auf Webseiten

Franz Fiala

Wer geografische Zusammenhänge darstellen möchte, benötigt Landkarten und muss in diesen Landkarten geografische Bereiche verschieden einfärben.

Statische Grafiken

Am Anfang stehen normalerweise Excel-Tabellen, deren Zahlenwerte in verschieden Farben umgerechnet werden, mit denen dann in einem Grafikprogramm händisch eine Landkarte eingefärbt wird. Diese Landkarte wird als feste Grafik in Internet-Seiten eingebunden.

Jede Änderung der Daten erfordert eine Wiederholung dieser Arbeitsschritte; mühsam.

Ein Beispiel für eine solche statische Darstellung ist diese Grafik über die Internet-Nutzung aus dem Jahr 2003 **(Bild unten links)**.

Dynamische Grafiken

An den statistischen Auswertungen von Eurostat (*http://epp.eurostat.ec.europa.eu/*) kann man sehen, wie solche geografischen Zusammenhänge auch dynamisch hergestellt werden können. Daten betreffend die „Informationsgesellschaft“ findet man hier: *http://epp.eurostat.ec.europa.eu/portal/page/portal/information_society/data/main_tables*

Das Beispiel **(Bild rechts unten)** ist auch gleichzeitig ein Browsertest. Firefox und Google Chrome können die Karten darstellen, Explorer aber nicht. Der Grund: es handelt sich um SVG-Grafiken, die man bei Microsoft nicht mag.

Ziel

In diesem Artikel wird gezeigt, wie man mit wenig Aufwand solche dynamischen Landkarten in der eigenen Webseite darstellen kann und wie man mit etwas mehr Aufwand auch ganz individuelle Landkarten herstellt.

Zum Beispiel soll die Verteilung der Clubmitglieder und PCNEWS-Leser auf die Bundesländer und auf die Wiener Bezirke dargestellt werden.

Google Charts

Wie kann man nun geografische Daten browserunabhängig darstellen?

Neben programm– und kostenintensiven Lösungen mit Flash-Animationen bieten sich als einfachste und kostenlose Lösung die Google-Charts an.

http://code.google.com/intl/de-DE/apis/chart/

Die Google-Charts zeigen prinzipiell eine Weltkarte, die als Ganzes oder durch eine Parametrierung auf einzelne Regionen reduzierbar ist.

http://code.google.com/intl/de-DE/apis/chart/docs/gallery/new_map_charts.html

Weltkarte **(Bild rechts oben)**

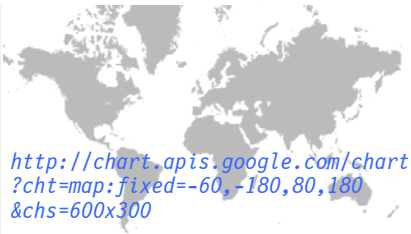
Die Parameter von `fixed` geben die Längen- (-180..180) und Breitegrade (-90..90) des Kartenausschnitts an. Damit die Karte „vernünftig“ ausschaut, darf man nicht bei den Polen (90,-90) beginnen.

Weltkarte mit Deutschland und Österreich **(Bild rechts mitte)**

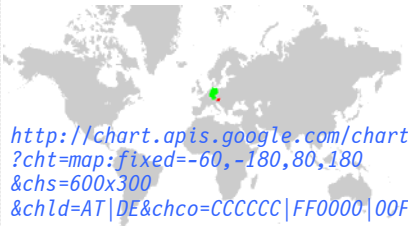
Ein zweibuchstabiger Kode in `chld` markiert Länder, `chco` gibt die Farben an, der erste Wert ist die neutrale Farbe.

Deutschland uns Österreich mit Rand **(Bild rechts unten)**

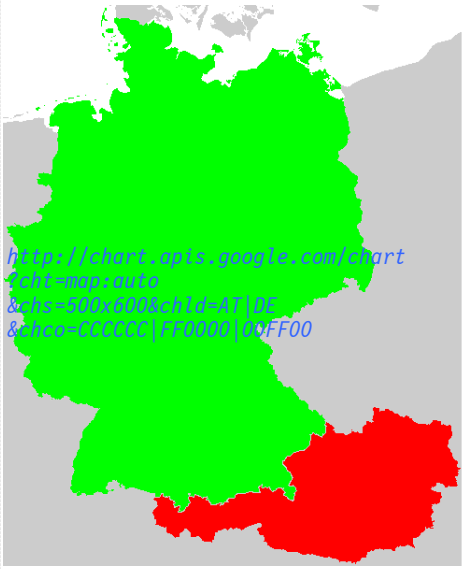
Ersetzt man `fixed` durch `auto`, reduziert sich die Karte auf die markierten Länder. `auto=30,30,30,30` erzeugt einen allseitigen Rand mit 30 Pixel Breite.



http://chart.apis.google.com/chart?cht=map:fixed=-60,-180,80,180&chs=600x300

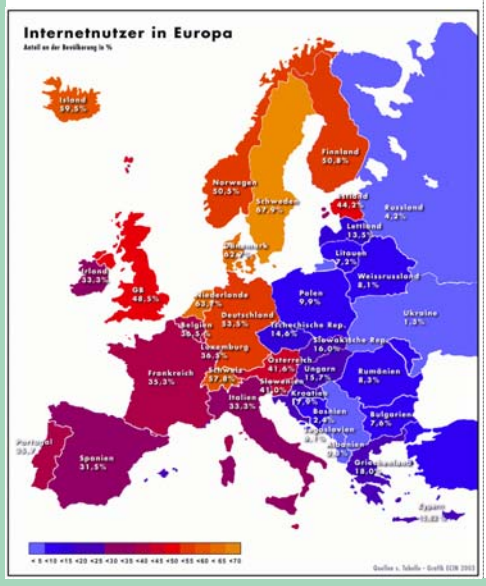


http://chart.apis.google.com/chart?cht=map:fixed=-60,-180,80,180&chs=600x300&chld=AT|DE&chco=CCCCCC|FF0000|00FF00

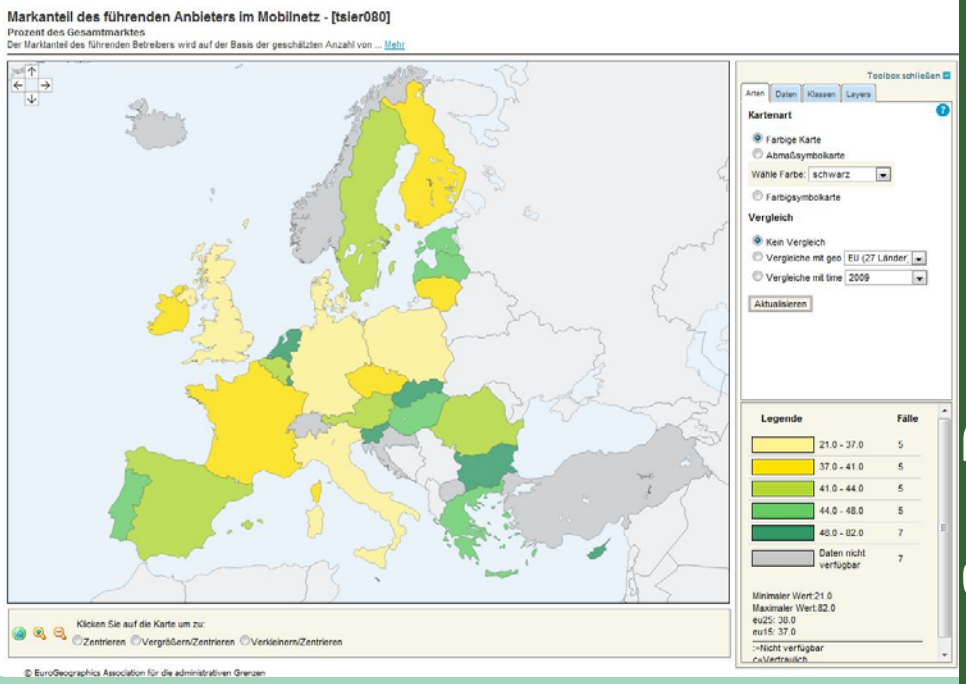


http://chart.apis.google.com/chart?cht=map:auto&chs=500x600&chld=AT|DE&chco=CCCCCC|FF0000|00FF00

Statische Grafik als gif-Bild von ECIN – Electronic Commerce Info Net, Seite *http://www.ecin.de/marktbarometer/europa3/*



Dynamische Grafik von Eurostat, die auf SVG-Grafik-Dateien aufbaut und die durch umfangreiche JavaScript-Elemente am Client konfigurierbar ist. Beispiel „Marktanteil des führenden Anbieters im Mobilnetz“: *http://epp.eurostat.ec.europa.eu/tgm/mapToolClosed.do?tab=map&init=1&plugin=1&language=de&pcode=tsier080&toolbox=types#*



Aufrufsyntax für Google-Landkarten *http://chart.apis.google.com/chart*

Param	Wert	Beispiel	Beschreibung
cht=	auto		auto wählt den Ausschnitt automatisch, entsprechend der durch chld angegebenen Länder
	auto:<r>,<r>,<r>,<r>	auto:30,30,30,30	Vier optionale und dem auto nachgestellte Parameter bestimmen einen Randabstand in Pixel
	fixed:<lat1>,<lng1>,<lat2>,<lng2>	fixed:-60,-180,80,180	fixed erfordert die nachgestellte Angabe der linken unteren und rechten oberen Koordinate der Landkarte. lat=geografische Breite {-90..90}, lng=geografische Länge {-180..180}
chs=	<bb>x<hh>	600x300	Breite bb und Höhe hh der Landkarte in Pixel, Maximum 600 Pixel oder 300000 Pixel für das gesamte Bild Achtung: wenn diese Zahlen nicht genau dem Verhältnis der realen Seitenverhältnisse des Bildes entsprechen wird ein der beiden Dimensionen voll ausgenutzt, die zweite bleibt teilweise weiß
chld=	<LL>|<LL>|<LL>	DE|AT|CH BE|BG|DK|DE|EE|FI|FR|GR|GB|IE|IT|LV|LT|LU|MT|NL|AT|PL|PT|RO|SE|SK|SI|ES|CZ|HU|CY (EU-Länder)	Liste der farblich hervorgehobenen Länder, LL ist durch den großbuchstabigen, zweistelligen ISO-Kode zu ersetzen. Die Länderkodes werden durch einen senkrechten Strich getrennt. Wenn cht=auto angegeben ist, wird der Kartenausschnitt automatisch auf diesen Kartenbereich eingestellt. Je nach Land sind auch Regionen oder Städte vordefiniert; dem Länderkode wird ein Bindestrich nachgestellt, gefolgt von dem Kode für die Region. Details dazu in der englischen ISO-Kodetabelle. *http://en.wikipedia.org/wiki/ISO_3166-2* *http://en.wikipedia.org/wiki/ISO_3166-2:AT*
chco=	<RRGGBB>|<RRGGBB>|...|<RRGGBB>	CCCCCC|FF0000|00FF00|0000FF	ohne chd: erster Farbwert neutrale Hintergrundfarbe, danach pro Land in chld eine durch senkrechten Strich getrennte Farbwerte
	<RRGGBB>,<RRGGBB>,...,<RRGGBB>		mit chd: erster Farbwert neutrale Hintergrundfarbe, zweiter Farbwert Anfangsfarbe, dritter Farbwert Zielfarbe
chd=	t:<VL>,<VL>,...<VL>		chd enthält Daten, die den einzelnen Ländern zugeordnet werden. Das vorangestellte t bedeutet, dass die Zahlen in gewöhnlichem Textformat vorliegen. Wenn chd angegeben ist, geben diese Werte die Reihenfolge beim Farbübergang an, der mit chco angegeben wird.
chtt=	<titletext>	Oesterreich	Titeltext (Achtung: Umlaute funktionieren auch bei korrekter Url-Kodierung nicht). Space ist ein Pluszeichen.
chts=	<RRGGBB,fontsize>	0000FF,24	Titelfarbe und Titelgröße in Points
chm=	<type><text>,<color>,<index>,<point>,<size>,<z_order>,<placement>	FText,000088,0,3,12	Marker: type {f,t,A}, f rahmt den Text ein, t ist ein einfacher Text, A erzwingt, dass die Texte nicht überlappen können, color ist die Farbe, index ist 0 und wird nur in anderen Diagrammtypen verwendet, point ist die Nummer des beschrifteten Landes, size ist die Größe der Schrift Mehrere Beschriftungen werden durch | getrennt.
chma=	<links>,<rechts>,<oben>,<unten>|<legendenbreite>,<legendenhöhe>		Ränder um die Landkarte sowie die Abmessungen der Legende in Pixel
chf=	=bg,s,RRGGBB		Hintergrundfarbe
chdl=			Legende
chof=	=validate		Überprüft die Syntax und zeigt Fehlermeldungen bei falsch formatieren Kommandozeilen

map_europe_0.htm

```
<html>

<head>
    <meta name="GENERATOR" content="Visual Studio 2008" />
    <meta http-equiv="Content-Type" content="text/html; charset=windows-1252" />
    <title>Europa-Landkarte</title>
</head>

<body>
    <script language="javascript" type="text/javascript">
        Debug = false
        Bild = true
        Title = "EU-Laender"
        TitleColor = "0000FF"
        TitleSize = 24
        Rand = 10
        Background = 'EEFFFF'
        Legendenbreite = 0
        Legendenhoehe = 0
        Laender = new Array(
            Array('Belgien', 'BE', '99CCCC'),
            Array('Bulgarien', 'BG', 'CC99CC'),
            Array('Daenemark', 'DK', 'CCCC99'),
            Array('Deutschland', 'DE', '9999CC'),
            Array('Estland', 'EE', 'CC9999'),
            Array('Finnland', 'FI', '99CC99'),
            Array('Frankreich', 'FR', '99CC99'),
            Array('Griechenland', 'GR', 'CC9999'),
            // Array('Grossbritannien', 'GB', 'CCCC99'),
            Array('England', 'GB-ENG', 'CCCC99'),
            Array('Wales', 'GB-WLS', 'CCCC99'),
            Array('Schottland', 'GB-SCT', 'CCCC99'),
            Array('Nordirland', 'GB-NIR', 'CCCC99'),
            Array('Island', 'IS', 'CCCC99'),
            Array('Faeroeer', 'FO', 'CCCC99'),
            Array('Irland', 'IE', '9999CC'),
            Array('Italien', 'IT', 'CC99CC'),
            Array('Lettland', 'LV', '99CC99'),
            Array('Litauen', 'LT', '99CCCC'),
            Array('Luxemburg', 'LU', 'CC99CC'),
            Array('Malta', 'MT', 'CCCC99'),
            Array('Niederlande', 'NL', 'CCCC99'),
            Array('Oesterreich', 'AT', 'CC9999'),
            Array('Polen', 'PL', '99CC99'),
            Array('Portugal', 'PT', '99CCCC'),
            Array('Rumaenien', 'RO', 'CCCC99'),
            Array('Schweden', 'SE', 'CC99CC'),
            Array('Slowakei', 'SK', '9999CC'),
            Array('Slowenien', 'SI', 'CCCC99'),
            Array('Spanien', 'ES', '9999CC'),
            Array('TschechischeRep', 'CZ', '99CCCC'),
            Array('Ungarn', 'HU', 'CC99CC'),
            Array('Zypern', 'CY', 'CCCC99')
        )

        var s = GetEuropeMap(false)
        document.write(s.replace(/&/gi, '<br/>&'))
        document.write('<br/>' + s)
        if (Bild) document.write('<img src="' + s + '">');

        function GetEuropeMap(Debug) {
            var chtVal = 'map:auto=' + Rand + ',' + Rand + ',' + Rand + ',' + Rand
            var chsVal = '500x550'
            var chttVal = 'EU-Laender'
            var chtsVal = '000088,24'
            var chfVal = 'bg,s,' + Background
            var chmaVal = '10,10,10,10|0,0'
            var chldVal = ''
            var chcoVal = '444444'
            var chmVal = ''
            for (i = 0; i < Laender.length; i++) {
                chldVal += (chldVal == '' ? '' : '|') + Laender[i][1];
                chcoVal += '|' + Laender[i][2]
                chmVal += (chmVal == '' ? '' : '|') +
                          'f' + escape(Laender[i][0]) + ',000088,0,' + i + ',8'
            }
            return 'http://chart.apis.google.com/chart'
                + '?cht=' + chtVal
                + '&chs=' + chsVal
                + '&chld=' + chldVal
                + '&chco=' + chcoVal
                + '&chm=' + chmVal
                + '&chtt=' + chttVal
                + '&chts=' + chtsVal
                + '&chf=' + chfVal
                + '&chma=' + chmaVal
                + (Debug ? '&chof=validate' : '')
        }
    </script>
</body>
</html>
```

Darstellung der EU-Länder

Die Darstellung aller 27 EU-Staaten erfolgt mit folgendem Aufruf:

```
http://chart.apis.google.com/chart?
cht=map:auto=10,10,10,10&chs=500x600&chld=BE|BG|DK|DE|EE|FI|FR|GR|GB
-ENG|GB-WLS|GB-SCT|GB-
NIR|IE|IT|LV|LT|LU|MT|NL|AT|PL|PT|RO|SE|SK|SI|ES|CZ|HU|CY&chco=44444
4|99CCCC|CC99CC|CCCC99|9999CC|CC9999|99CC99|99CC99|CC9999|CCCC99|CCC
C99|CCCC99|CCCC99|9999CC|CC99CC|99CC99|99CCCC|CC99CC|CCCC99|CCCC99|C
C9999|99CC99|99CCCC|CCCC99|CC99CC|9999CC|CCCC99|9999CC|99CCCC|CC99CC
|CCCC99&chm=fBelgien,000088,0,0,8|fBulgarien,000088,0,1,8|fDaenemark
,000088,0,2,8|fDeutschland,000088,0,3,8|fEstland,000088,0,4,8|fFinnl
and,000088,0,5,8|fFrankreich,000088,0,6,8|fGriechenland,000088,0,7,8
|fEngland,000088,0,8,8|fWales,000088,0,9,8|fSchottland,000088,0,10,8
|fNordir-
land,000088,0,11,8|fIrland,000088,0,12,8|fItalien,000088,0,13,8|fLet
tland,000088,0,14,8|fLitauen,000088,0,15,8|fLuxemburg,000088,0,16,8|
fMal-
ta,000088,0,17,8|fNiederlande,000088,0,18,8|fOesterreich,000088,0,19
,8|fPolen,000088,0,20,8|fPortugal,000088,0,21,8|fRumaenien,000088,0,
22,8|fSchweden,000088,0,23,8|fSlowakei,000088,0,24,8|fSlowenien,0000
88,0,25,8|fSpanien,000088,0,26,8|fTschechischeRep,000088,0,27,8|fUng
arn,000088,0,28,8|fZypern,000088,0,29,8&chtt=EU-
Laender&chts=0000FF,24&chf=bg,s,EFFFFF&chma=10,10,10,10|100,20
```

Die Parameterliste ist im Prinzip nicht sehr kompliziert, doch mit zunehmender Zahl anzuzeigender Länder steigt auch die Fehlerwahrscheinlichkeit und es empfiehlt sich, die Kommandozeile nicht händisch zu erstellen sondern das einem JavaScript-Programm zu überlassen, sodass die Daten leicht bearbeitbar bleiben und die korrekte Syntax immer gegeben ist.

Das nebenstehende Programm zeigt, wie man das bewerkstelligen kann. Die Parameter werden am Anfang des Javascript-Abschnitts eingestellt, danach folgt die Liste der Länder im Array `Laender`. Bei jedem Land ist der Name, der 2-stellige ISO-Kode und die Farbe für die Darstellung angegeben. Leider kann der Google-Server keine Zeichen außerhalb der ASCII-Zeichen darstellen, auch dann nicht, wenn man diese Zeichen richtig Url-kodiert, daher müssen alle Umlaute ersetzt werden.

Am Beispiel von Großbritanien wird gezeigt, wie man statt des ganzen Landes auch nur einzelne Regionen (Schottland, England, Wales, Nordirland) anzeigen kann.

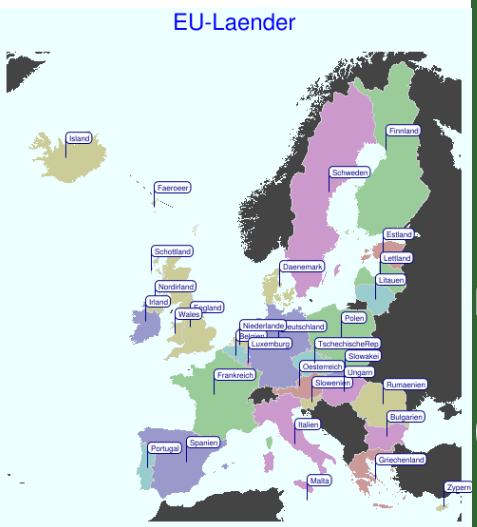
Die Variable `Debug` bewirkt mit `Debug=true`, dass der Kode nur zur Prüfung an den Google-Server gesendet wird (das geschieht durch Einfügung des Parameters chof=validate. Man erhält vom Google-Server entweder eine OK-Meldung oder eine detaillierte Fehlerliste.

Das Programm hat drei Ausgaben:

- die Kommandozeile in mehreren Zeilen
- Die Kommandozeile in einer einzigen Zeile und
- Das von Google erzeugte Bild.

Das Bild entsteht durch ein `img`-Tag, in das statt einem Bildnamen genau dieser lange Aufruf des Google-Servers eigegeben wird:

```
<img src="http://chart.apis.google.com/chart?cht=map...">
```



Dynamische Karte von Europa

Das erste Beispiel vereinfacht die Herstellung der Kommandozeile für das Bild. Es wird aber nur ein Bild erzeugt. Um mehrere Ansichten von Europa zu zeigen, benötigt man mehrere Versionen des vorigen Programms. Aber diese Programme wären sehr ähnlich. Der Unterschied wäre nur, welche Länder und mit welcher Farbe und Beschriftung anzuzeigen wären. Gefragt ist daher ein Programm, das eine europäische Landkarte mit verschiedenen Inhalten und Farben gesteuert über Parameter anzeigen kann.

Wenn man dieses Programm realisiert, bekommt man einmal eine Fehlermeldung vom Google-Server, die besagt, dass der angeforderte URL die maximal zulässige Größe von 2k übersteigt.

Um aber dennoch Darstellungen dieses Umfangs zeigen zu können, muss man die Parameterwerte nicht in einem GET-Request (also über die Kommandozeile) sondern über einen POST-Request (also über ein Formular) absetzen. Das hat aber den Nachteil, dass man den Kode nicht im Rahmen eines Bildes (`<img>`-Tag) verwenden kann sondern nur im Rahmen einer eigenen Seite.

Damit nun verschiedene Bilder auf einer Seite gezeigt werden können, benötigt man für jedes Bild ein `iframe` und in dieses `iframe` muss dann der Kode für das Bild eingefügt werden.

Damit hätte man eine Datei für die Bilder und für jedes Bild eine Kodedatei mit dem JavaScript-Kode. Das Kodebeispiel map_europe_eu.js zeigt, wie man das auf eine einzige Datei vereinfachen kann.

Ein zweites Problem ist, dass die Europa-Karte immer anders aussieht, je nachdem, welche Länder ausgewählt sind. Der Parameter `auto` bewirkt nämlich, dass der Kartenausschnitt sich automatisch auf die ausgewählten Länder ausrichtet. Daher wird stattdessen der Parameter `fixed` mit Angaben der Länge und Breite der diagonalen Eckpunkte des darzustellenden Kartenausschnitts gewählt.

```
map:fixed=30,-20,65,60
```

Anfangskoordinate: 20° Nord, 20° West
Endkoordinate 65° Nord, 60° Ost

Um die unten abgebildeten vier Europakarten zu zeichnen, benötigt man eine Html-Seite mit folgendem Inhalt:

map_europe_eu.htm

```
<script src="map_color.js"
  type="text/javascript"></script>
<script src="map_europe_eu.js"
  type="text/javascript"></script>
<script language="JavaScript1.2" type="text/
javascript">
  WriteEuropeMap('all', false, '', false)
  WriteEuropeMap('eu', false, '', false)
  WriteEuropeMap('euro', false, '', false)
  WriteEuropeMap('uefa', false, '', false)
</script>
```

Der eigentliche Kode befindet sich in der Skriptdatei `map_europe_eu.js`, die Bibliothek `map_color.js` unterstützt das Arbeiten mit den Farbmodellen RGB und HSV. Die vier übergebenen Parameter sind:

`Chart`: Name der Karte; `Inverse`: Darstellung der ausgewählten Länder oder der durch die Auswahl nicht ausgewählten Länder; `Color`: Farbe für die ausgewählten Länder; `Debug`: Debugging einschalten

Ein bestimmtes Land wird so beschrieben:

```
new Array('Oesterreich', 'AT', 'Oesterreich',
true, true, true, true),
```

Die Parameter bedeuten der Reihe nach: Name des Landes, ISO-Kode, Text der Beschriftung, europe, eu, euro, uefa

Die vier Bool-Felder geben an:

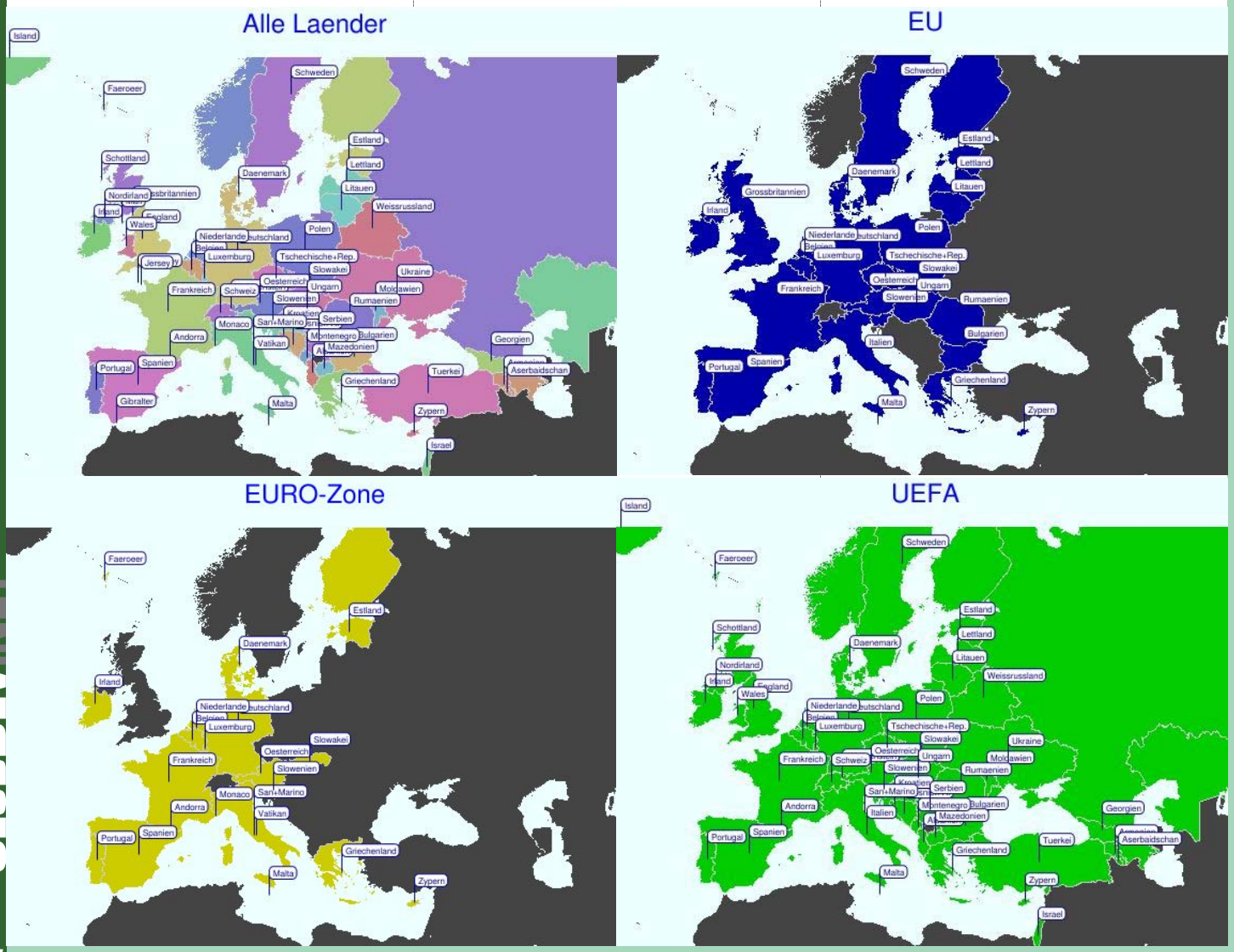
Europe: dieses Land ist ein europäisches Land ist (Türkei, Israel zählen hier nicht mit); **Eu**: dieses Land ist ein EU-Land (nicht Andorra, San Marino oder Vatikan); **Euro**: in diesem Land gilt der Euro (auch Andorra, Monaco und Vatikan); **Uefa**: Land ist Mitglied bei den (Fußball-)Konföderation Uefa (mit Israel, Türkei, Aserbaidschan, Georgien, Armenien, Färöer, Island, Schottland, England und Wales aber nicht Großbritanien)

Titel und Farben werden hier festgelegt:

```
Title = new Array('Europa', 'EU', 'EURO-Zone', 'UEFA')
Farbe = new Array('OOOO44', 'OOOOAA', 'CCCCOO', 'OOCCOO')
```

Interessant ist die Farbzuteilung bei der Europa-Karte, denn es ist nicht ganz trivial, unterschiedliche Farben automatisch zuzuteilen. Die Farbintensität und Helligkeit werden mit 40 und 80 festgelegt, Die Farbe als eine von 27 Farben (=Anzahl der Länder) aufgeteilt auf 360 Grad, dann Umrechnung auf RGB:

```
HSV = new HSVobject((360 / Laender.length) * i, 40, 80)
RGB = new RGBobject(0, 0, 0)
HSV2RGB(HSV, RGB)
```



Thematische Landkarten

Die bisher gezeigten Kartenbeispiele beschränken sich auf die Kennzeichnung der Länder durch verschiedene Farben. Vielfach besteht aber der Wunsch, ein Land oder eine Region mit einem Zahlenwert zu charakterisieren, der sich dann in einem Farbwert ausdrückt, wie zum Beispiel Klimabedingungen, Arbeitslosenzahlen und andere.

Man benötigt daher ein Programm, welches eine vorgegebene Datenreihe in Farbwerte umrechnet.

In dem folgenden Beispiel wird wieder ein JavaScript-Programm mit einer Ländertabelle verwendet, aber die Spalten stellen nicht ja/nein-Felder dar sondern eben darzustellende Zahlenwerte. Dargestellt werden die Auslandsspiele des Fußballklubs SK Rapid. In der ersten Darstellung sieht man die Anzahl der Spiele und in der zweiten die jeweiligen Erfolge. Während die Anzahl der Spiele durch einen mehr oder weniger intensiven Grünwert charakterisiert werden, wobei mehr Spiele auch ein satteres Grün bedeuten, erfolgt die Darstellung des Erfolgs durch einen Farbübergang von Grün über Gelb nach Rot, wobei Grün dem Punktemaximum 3, Gelb einer ausgeglichenen Bilanz von 1.5 Punkten und Rot null Punkten - bezogen auf alle Spiele - entspricht.

Wie werden nun die Farben für die jeweiligen Zahlenwerte bestimmt?

Das RGB-Farbmodell könnte man zur Not für die Anzahl der Spiele verwenden, besser ist es aber, das HSV-Farbmodell zu verwenden, bei dem Farbton, Sättigung und Intensität eingestellt werden.

Anzahl der Spiele

```
if (Laender[i][1] == 'AT')
    MyColorRGB = new RGBobject(0, 128, 0)
else {
    var MyColorSat =
      Laender[i][Index] * 100 * 1.0 / 60
    var MyColorHSV =
      new HSVobject(100, MyColorSat, 100)
    MyColorRGB = new RGBobject(0, 0, 0)
    HSV2RGB(MyColorHSV, MyColorRGB)
}
```

Wegen des großen Unterschieds nationaler und internationaler Spiele wird der Farbwert für Österreich aus dem Berechnungsverfahren ausgenommen. Tut man das nicht, wären die Farben für andere Länder viel zu hell. Die für Österreich gewählte Farbe ist ein dunkles Grün: #008000 (0,128,0).

Die maximale Farbsättigung für 60 Spiele ist 100. Die eigentliche Berechnung der Farbe erfolgt, durch Anlegen eines HSV-Objekts mit dem Farbton 100 (grün), der berechneten Sättigung und der Intensität 100. Dieser Farbwert wird in eine RGB-Objekt umgerechnet, das zur Anzeige verwendet werden kann.

Spielerfolg

```
if (Laender[i][2] == 0) // keine Spiele
    MyColorRGB =
      new RGBobject(255, 255, 255)
else {
    var MyColorHue =
      (Laender[i][Index]) * 100 * 1.0 / 3
    var MyColorHSV =
      new HSVobject(MyColorHue, 100, 100)
    MyColorRGB = new RGBobject(0, 0, 0)
    HSV2RGB(MyColorHSV, MyColorRGB)
}
```

Beim Spielerfolg wird nicht die Sättigung sondern der Farbton moduliert. Das Maximum ist wieder 100, bedeutet aber die Farbe Grün, das Minimum ist 0 und bedeutet die Farbe Rot.

Damit man kein Legende zeichnen muss, wurde der konkrete Datenwert bei jedem Land in Form einer Textmarke angefügt.

Das gesamte JavaScript-Programm ist in der Datei `map_europe_rapid.js` enthalten, die in einer Skript-Zeile im Kopf der Datei `map_europe_rapid.htm` eingebunden wird.

Die Datentabelle hat vier Spalten, jedes Land wir in einer Zeile definiert:

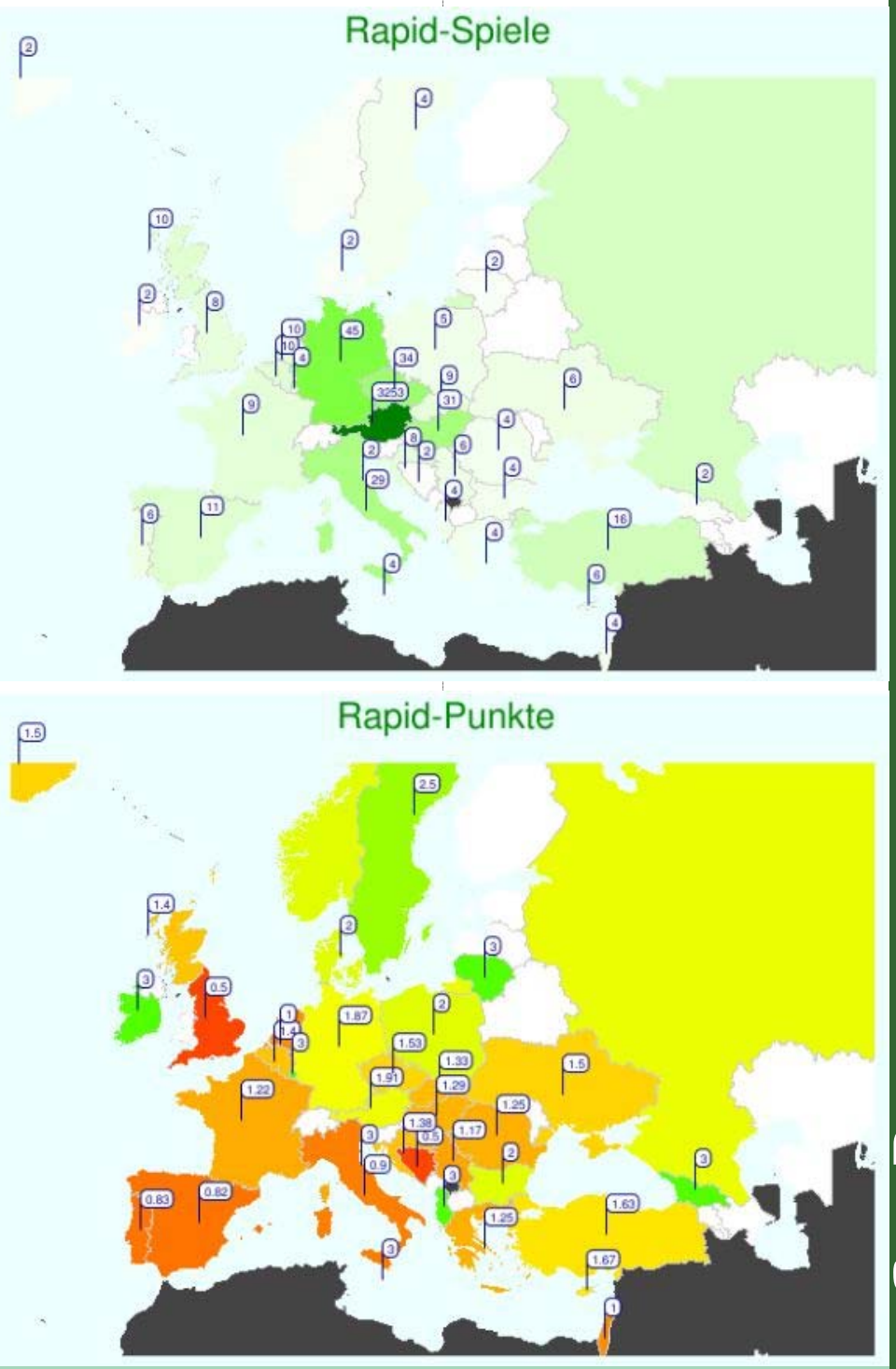
```
Array('Oesterreich', 'AT', 3253, 1.91),
```

Spalte 0: Laendername, Spalte 1: Kurzbezeichnung, Spalte 2: Anzahl der Spiele, Spalte 3: erreichte Punktezahl

map_europe_rapid.htm

Die Html-Datei, die die beiden Bilder anzeigen soll, benötigt nur einige Zeilen:

```
<html>
<head>
  <meta name="GENERATOR"
    content="Visual Studio 2008" />
  <meta http-equiv="Content-Type"
    content="text/html; charset=windows-1252" />
  <title>Rapid in Europa</title>
  <script src="map_color.js"
    type="text/javascript">
  <script src="map_europe_rapid.js"
    type="text/javascript">
</head>
<body>
  <script type="text/javascript">
    WriteEuropeMap('games')
    WriteEuropeMap('points')
    </script>
</body>
```



Regional-Karten

Bis hierher leisten die Google-Karten gute Dienste. Wenn es aber darum geht, regionale Verhältnisse darzustellen, gibt es keine Möglichkeit, diese Regionen in den Google-Karten abzugrenzen.

Beispielsweise könnte es von Interesse sein, die Verhältnisse in Wiener Bezirken darzustellen.

Was tun?

Hier ist es notwendig, eigene Karten zu verwenden und nach kurzer Suche findet man diese Karten fertig gezeichnet in den Kartenarchiven der Wikipedia.

Dank dem User „Rosso Robot“ konnten die Karten der österreichischen Bundesländer mit den jeweiligen Bezirksgrenzen nach einiger Modifikation für diese Anwendung weiter verwendet werden.

Die nebenstehenden Abbildungen zeigen die Schritte, die dazu notwendig sind:

Ausgangpunkt war die Karte von Wien von User „Rosso Robot“ mit den eingezeichneten Flussläufen und den Grenzen der Nachbarbezirke. (Bild rechts oben).

Diese Karte liegt im Format SVG vor und kann daher (theoretisch) mit Vektorgrafikprogrammen bearbeitet werden. Wirklich gut gelingt die Bearbeitung nur mit dem dafür spezialisierten Programm Inkscape. Der Grund ist, dass Inkscape für alle Formatierungen direkt das SVG-Format manipuliert. Die Grafikdateien sind daher extrem kompakt. Andere Vektorprogramme arbeiten immer mit einem proprietären internen Format, welches nur auf Wunsch in das SVG-Format umgewandelt wird. Das Resultat sind unlesbarere und sehr große SVG-Dateien.

Svg-Grafiken können auf drei Arten bearbeitet werden:

- Interaktiv in Inkscape
- Mit einem hierarchischen Xml-Text-Editor in Inkscape
- Direkt durch Änderung des Html-ähnlichen Svg-Kode

Wie immer man es auch tut, jede Änderung wirkt sich unmittelbar in der Anzeige der Grafik aus; sehr praktisch und lehrreich.

Interaktiv wurden aus dem Wien-Plan die Flussläufe und benachbarten Bezirksgrenzen entfernt (Bild mitte links) und danach die Flächen der Bezirke einheitlich eingefärbt (Bild mitte rechts).

Kodeänderungen

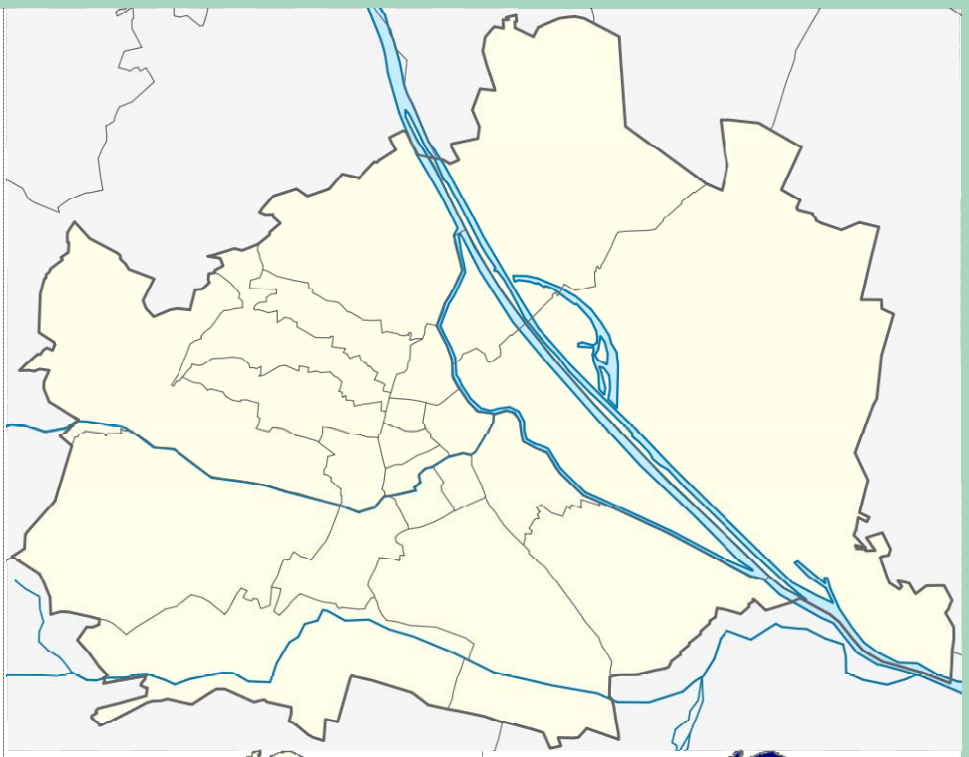
Damit man per Programm die verschiedenen Parameter des Bildes auch verändern kann, muss man die entstandene SVG-Datei in ihre Bestandteile zerlegen und einzelne Passagen daraus mit einem Programm veränderbar machen.

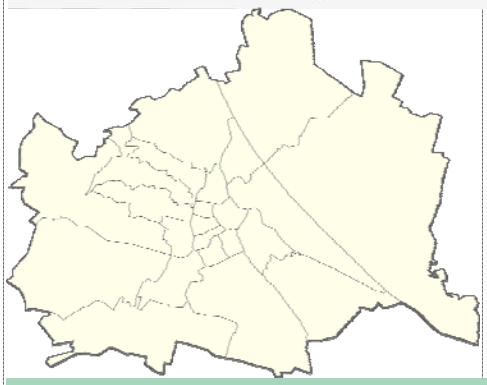
Dieser sehr aufwändige Vorgang wird hier nicht im Detail beschrieben aber eine Zusammenfassung ist auf der folgenden Seite zusammengefasst. Interessierte können die Dateien bei der Online-Version dieses Artikels downloaden.

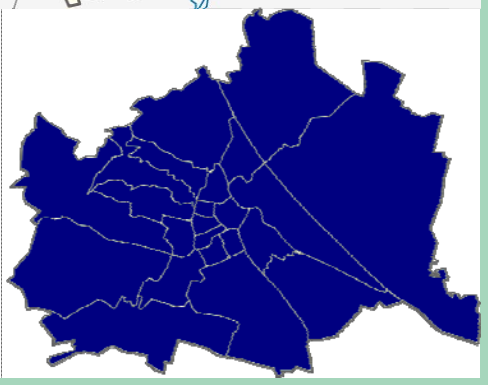
Man kann die so entstandene Anwendung entweder direkt im Internet nutzten oder am eigenen Server installieren.

Google Apis

Es gibt auch Webdienste, die das für uns erledigen. Ein Beispiel dazu findet sich hier:







Die Google-Falle

Gerald Reischl beschreibt in seinem Buch „Die Google-Falle“ die zahlreichen Gefahren beim Umgang mit dem Giganten. Mir schien Vieles in dem Buch überzeichnet, bis ich bei diesem Landkartenprojekt auch in eine dieser Fallen getappt bin.

Ich war einige Zeit Nutzer der kostenlosen Anwendung Mgmaps (*http://www.mgmaps.com/*) auf meinem PocketPC. Der Autor Cristian Streng bietet mit Mgmaps ein kostenloses Navigationssystem an, bei dem Kartenmaterial verschiedener Online-Anbieter ausgewählt werden kann. Als Kartenmaterial kann man wählen: Google Maps, Yahoo! Maps, Windows Live Local (MSN Virtual Earth), Ask.com, Open Street Map. Ich testete diese Software eine Zeitlang mit den Google-Karten erfolgreich bis auf einmal die Karten verschwanden.

Die Fehlersuche ergab, dass nicht die Anwendung fehlerhaft war, sondern das Kartenmaterial verschwunden war. Google hat kurzerhand diesem Anwender den Zugriff auf das Kartenmaterial verwehrt. Ob es tatsächlich eine Verletzung der Nutzungsrechte war, kann ich als Nicht-Jurist nicht beurteilen. Die anderen Kartenanbieter können ja noch verwendet werden. Auf der Homepage von Cristian Streng erscheint jetzt der Schriftzug Google durchgestrichen, um auf diesen Umstand besonders hinzuweisen.

Die Vorgansweise von Google zeigt aber, in welche Abhängigkeit man sich begibt, wenn man scheinbar frei zugängliche Dienste für die eigene Anwendung nutzt. Auch wenn etwas derzeit kostenlos verfügbar ist, kann sich das über Nacht ändern. Es ist daher vorteilhaft, freies oder selbst erstelltes Kartenmaterial zu verwenden und eher in eigene Programme zu investieren.

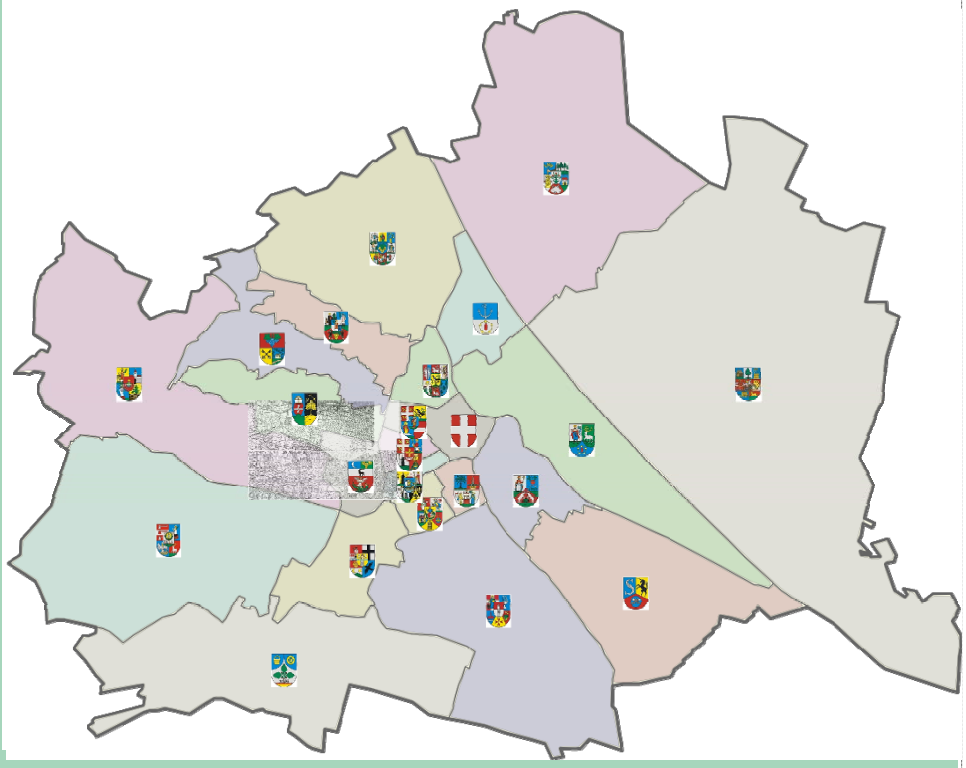
Daher wurde für das vorliegende Kartenprojekt mit Hilfe der Daten von Wikipedia am eigenen Server gehostet. Alle Anwendungen, die die Landkarten benutzen wollen, können auf die Adresse *http://iam.at/map* zugreifen.

http://code.google.com/intl/de-DE/apis/visualization/interactive_charts.html und *http://code.google.com/intl/de-DE/apis/visualization/documentation/gallery/intensitymap.html*

Man übergibt dem Google-Server die Daten und bekommt die Grafik als Antwort; die ganze Datenübergabe versteckt sich in einem eingebetteten Javascript-Modul.

Der Nachteil dieses Webdienstes ist aber seine viel geringere Konfigurierbarkeit. Man kann zum Beispiel nicht den Kartenausschnitt wählen, denn es gibt nur die Auswahl zwischen den Kontinenten. Auch die Farbe wird automatisch vergeben. Will man daher eine universellere Darstellung, muss man eine eigene Anwendung programmieren.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<svg
    xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    version="1.1"
    width="210mm" height="297mm" id="svg2">
  <g id="Map_Wien" transform="translate(-4.8777085,16.655673)">
    <path id="Outline" d="m 60.865983,284.96302 4.00606,0.61091 3.40657,-1.45191 … z"
    style="fill:#fefee9;fill-opacity:1;fill-rule:evenodd;stroke:#646464;stroke-
width:2;stroke-miterlimit:4;stroke-opacity:1;stroke-dasharray:none;stroke-
linecap:round;stroke-linejoin:round;" />
    <path id="Border_17" d="m 189.32927,216.64204 3.52773,1.49838 7.4079,5.72954
4.52689,4.93636 … " style="fill:none;fill-opacity:0;fill-
rule:evenodd;stroke:#646464;stroke-width:0.8;stroke-miterlimit:4;stroke-opacity:1;stroke
-dasharray:none;stroke-linecap:round;stroke-linejoin:round;" />
... hier folgen alle anderen Grenzlinien
    <path id="Region_01" d="m 353.0643,343.0033 c 0,-0.35754 -2.80347,-3.09962 -6.22994,
-6.09352 … z" style="fill:#AAAAAA;fill-opacity:0.6;fill-
rule:evenodd;stroke:#000000;stroke-width:0.1;stroke-miterlimit:4;stroke-opacity:1;stroke
-dasharray:none;stroke-linecap:round;stroke-linejoin:round;" />
... hier folgen alle anderen Bezirke
    <image id="Overlay_0" x="209.58153" y="320.02563" width="103.58086"
height="68.826752" transform="" style="opacity:0.80134678" xlink:href="http://
rapid.iam.at/d/kmlo/WienHistorisch2.jpg" />
... hier folgen weitere Overlays
    <circle id="Bezirksmitte_01" cx="-319" cy="81" r="5" transform="matrix
(1.3285961,0,0,1.3285961,788,213)" style="fill:#56d500;fill-
opacity:0.58823529;stroke:#6a0000;stroke-width:0.17299999;" />
    <image id="Wappen_01" x="23" y="-57" width="17.7" height="23" transform="translate
(330,370)" style="" xlink:href="http://rapid.iam.at/d/i/c/bezirk01-kl.jpg" />
... hier folgen die Bezirksmitten und die Wappen aller anderen Bezirke
```



Projekt im Entstehen

Folgende Merkmale fehlen und werden in den nächsten Wochen ergänzt:

- Übergabe der Parameter in einem Formular: (die Länge der Kommandozeile ist begrenzt und ihr Aufbau ziemlich unübersichtlich)
- JavaScript-Ansprache wie bei der Google-Anwendung gezeigt wurde.
- Geokodierung der Landkarte: damit können Overlays nicht nur über Pixel-Maße sondern auch über die Angabe der geografischen Länge und Breite positioniert werden.
- Regionen als Hyperlink ausführen damit man von Landkarte zu Landkarte oder zu anderen Anwendungen springen kann
- Erweiterung des Landkartensatzes auf alle österreichischen Bundesländer
- Automatische Erstellung einer Legende, Verbesserungen bei der Beschriftung
- Aussagekräftige Fehlermeldungen bei Syntaxfehlern in der Parameterübergabe

Achtung: die hier dargestellten Aufrufe können sich im Laufe des Projektfortschritts noch verändern; in diesem Fall die Angaben unter http://iam.at/map benutzen.

Bild und Svg-Kode

Die nebenstehende SVG-Datei (dargestellt ist nur ein Kodefragment des Bildes darunter) wurde mit Inscape händisch erstellt. Die Datei besteht aus:

`Svg`-Tag mit Kopfangaben wie der Sprachumfang der verwendeten Xml-Elemente, Version und Abmessungen des Ausgabemediums (A4).

`Map_Wien` gruppiert alle nachfolgenden Zeichnungselemente. Gleichzeitig wird mit `translate` eine Parallelverschiebung vorgenommen, damit die Zeichnung genau auf das Blatt passt.

`Outline` ist der Umriss des Stadtgebiets. Da es ein geschlossener Polygonzug ist, werden die Wertepaare der Punkte mit einem `z` beendet.

`Border_17` ist eine Bezirksgrenzlinie des 17. Bezirks. Der Linienzug ist nicht geschlossen.

`Region_01` ist die Umrisslinie des ersten Bezirks, die Füllfarbe des `style`-Attributs ist von Bedeutung.

`Overlay_0` ist eine Landkarte, die über die Vektorkarte gelegt wird. Die Skalierung wird mit `width` und `height`, die Positionierung mit `x` und `y` vorgenommen.

`Bezirksmitte_01` ist ein Kreis, der interaktiv auf die Mitte des Bezirks positioniert wurde.

`Wappen_01` ist das Wappen des Bezirks.

http://iam.at/map

Damit nun diese Karte parametriert werden kann, müssen die festen Elemente (Umriss, Grenzen und Regionsflächen) als Konstanten in einer Serversprache abgelegt und die variablen Elemente wie Farben, Strichstärken, Bilder durch Kommandozeilenangaben des Benutzers ersetzt werden. Diese Aufgabe übernimmt das Programm `map.aspx`.

Für interessierte Leser steht der komplette Source-Kode dieser Anwendung bei der Webversion dieses Artikels oder auf `iam.at/map` zum Download bereit.

Server Microsoft Server 2008

Programmierumgebung Visual Studio 2008, Sprache C#

Dateien

`Map.aspx, map.aspx.cs:` Aufruf durch den Benutzer (Startdatei) und erste Analyse der Kommandozeile; ruft `GeoMap.cs`

`GeoMap.cs`: parametrierte Ausgabe einer Landkarte. Funktion GetSvgMap konstruiert aus den Angaben der Kommandozeile die SVG-Datei und übergibt als Stream an die rufende Funktion. Alle folgenden anderen Dateien enthalten Hilfsklassen zur Generierung der SVG-Datei.

`GeoMapBorder.cs`: beschreibt eine Grenzlinie

`GeoMap.Coordinates.cs`: enthält ein Koordinatenpaar

`GeoMapOverlay.cs`: verwaltet ein Overlay

`GeoMapRegion.cs`: beschreibt die Fläche einer Region

`GeoMaps.cs`: Koordinaten von Umriss, Grenzlinien und Flächen aller gespeicherten Karten

`GeoMapStatic.cs`: Hilfsfunktionen

`SvgStyle.cs`: beschreibt einen Svg-Stil und einen Svg-Text-Stil

`ColorMode.cs`: Konversionsfunktionen für das RGB- und HSV-Farb-Modell

Kartendarstellungen

Die Anwendung

http://iam.at/map

erlaubt die Anzeige von Landkarten in der eigenen Webseite. Die Landkarte wird als SVG-Grafik angezeigt. Das Aussehen der Landkarte kann flexibel angepasst werden.

Aufruf im Browser durch einen direkten Link

Die Landkarte wir über Kommandozeilenparameter ausgewählt und parametriert. Die folgende Version zeigt eine Karte von Wien. **(1)**

http://iam.at/map?map=at-9

Aufruf in der eigenen Website

Die Einbindung in der eigenen Seite erfolgt derzeit über ein iFrame:

```
<iframe width="300" height="300"
frameborder="0" scrolling="no"
src="http://iam.at/map?map=at-9|||300,300">
```

Bei der Einbindung ist zu beachten, dass die Parameter `width` und `height` des `iframe` mit der Zielgröße im Aufrufparameter `map` abzustimmen sind. Die Einbindung funktioniert in Firefox und Chrome, im Internet-Explorer nur mit einem PlugIn von Adobe (siehe PCNEWS-118, Seite 24).

Map

Das map-Kommando hat folgende Parameter:

`map=<iso>|<margin>|<viewport>|<scale>`

`iso`	Kartenbezeichnung nach dem zweistelligen ISO-Kode. `AT` für Österreich, `AT-9` für Wien. Voreinstellung: `AT-9`
`margin`	definiert einen Rand um die Landkarte
`viewport`	definiert einen anzuzeigenden Kartenausschnitt
`scale`	skaliert die Karte auf eine gewünschte Größe (0...1)

Kartenauswahl, Ränder

`map=<iso>|<margin>`
`map=<iso>|<margin_left>,<margin_upper>,<margin_right>,<margin_lower>`

`map` wählt eine Landkarte aus. Derzeit definiert sind die Karten `AT` (Österreich) und `AT-9` (Wien). Der Parameter `margin` ist eine Gleitkommazahl und definiert einen Rand um die Landkarte. `map`, `iso` und `border` sind optional aber es muss wenigstens ein Parameter in der Kommandozeile angegeben werden. Gültig sind `map=`, oder `map=,10` oder `map=at-9`. Wenn sonst keine weiteren Parameter angezeigt werden, sieht man die Bezirke mit Pastellfarben eingefärbt.

`map=at-9|10`

Zeigt die Landkarte von Wien mit einem 10 Pixel breiten Rand.

1

`map=at-9|10,100,10,10`

Zeigt die Landkarte von Wien mit einem 10 Pixel breiten Rand, der obere Rand ist 100 Pixel breit.

Kartenausschnitt

`map=<iso>|<margin>|<x>,<y>,<width>,<height>`

Ein Ausschnitt der Karte kann dargestellt werden, indem einem zweiten Parametersatz die Abmessungen des Kartenausschnitts angezeigt werden (`x,y`: linke obere Ecke; `width` und `height`: Länge und Breite) Der Parameter `margin` kann auch leer bleiben.

Verkleinern

`map=<iso>|<margin>|<x>,<y>,<width>,<height>|<sizex>,<sizey>`

Für die Einbindung in eine eigene Seite kann mit `sizex` und `sizey` die Landkarte verkleinert werden.

Einfärben der Regionen

Die Regionen können über den Parameter `color` eingefärbt werden. Das Verhalten von `color` ist aber davon abhängig, ob Daten über den zusätzlichen Parameter `data` angezeigt werden sollen. Wenn daher `data` definiert ist, dann gelten für `color` andere Regeln (siehe unter "Datensteuerung"). Wenn `color` nicht angegeben ist, werden die Bezirke mit vordefinierten Farben initialisiert (siehe unter „Farbsteuerung durch Daten“).

`color=all,<rrggbb>`

Färbt alle Bezirke mit einer einheitlichen Farbe.

`color=all,44cc88`

Färbt alle Bezirke grün. **(2)**

`color=random`

Erzeugt zufällige Farben. **(3)**

`color=<region>,<rrggbb>|...`

Färbt einzelne Regionen der Karte mit der nachgestellten Farbe ein.

`color=all,ffddff|21,ff8888|10,ff0000`

Das Beispiel **(4)** kombiniert das Attribut `all` und färbt zusätzlich die Bezirke 21 hellrot und 10 rot.

`color=<rrggbb>|...`

Fortlaufende Zuweisung einer Farbe zu einer Region, beginnend bei der ersten Region.

`color=888888|884488|888844|448888|444488|884444|448844|888888|884488|888844|448888|444488|884444|448844|888888|884488|888844|448888|444488|884444|448844|444444|444444` **(5)**

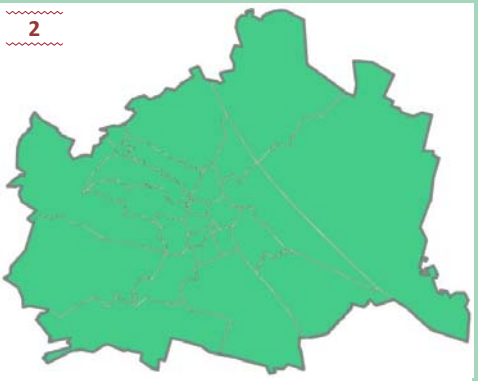
`color=none`

Mit `color=none` **(6)** wird die Anzeige der Regionen vollständig unterdrückt und man sieht nur die Farbe der Landkarte, den Umriss und die Grenzlinien zwischen den Regionen. Deren Charakteristik kann über den Parameter `style` eingestellt werden.

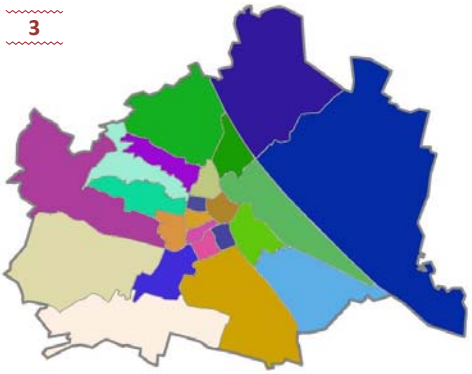
Steuerung der Umrisscharakteristik

Die Landkarte besteht aus einem normalerweise unsichtbaren rechteckigen Untergrund, der Umrisslinie und den Grenzen zwischen den Regionen und den darüber gelegten Flächen für jede einzelne Region. Die Darstellung dieser Linien/Flächen kann einheitlich über den Parameter `style` gesteuert werden. `style-background` steuert den Hintergrund, `style-outline` den Umriss der Karte, `style-border` die Grenzlinien zwischen den Regionen und `style-region` die Region. `style-text` formatiert die Beschriftungen und hat einen eigenen Parametersatz. Die Stile müssen nicht angege-

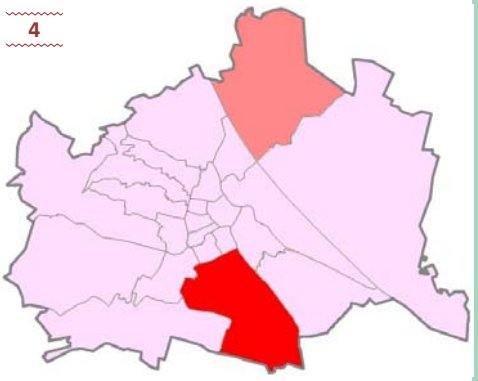
2



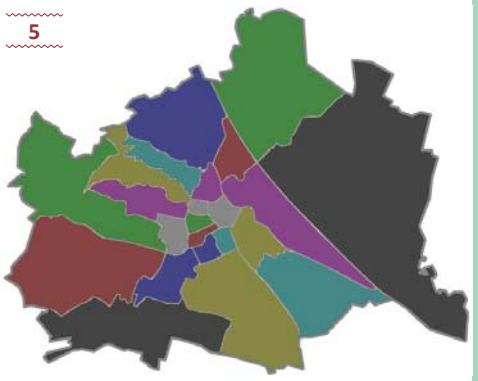
3



4



5



6

ben alle werden, die senkrechten Trennstriche sind aber erforderlich.

```
style=<style-background>|<style-outline>|<style-border>|<style-region>|<style-text>
<style-~>=<stroke-width>,<stroke>,<fill-opacity>,<fill>
```

Zuweisung eines generellen Stils zu einer Region.

`stroke-width` ist die Linienstärke der Umrisslinie in Points, `stroke` ist die Linienfarbe, `fill-opacity` ist die Deckkraft (0..1), `fill` ist die Füllfarbe.

```
style=|||5,666688,1,ccccff&color=10,ff0000
```

Regionen werden einer dunkelblauen 5-pt-Linie eingerahmt, die Füllung ist ein helles Blau. Zusätzlich wird mit `color` die Region 10 rot eingefärbt. **(7)**

Hintergrund

Größe und Farbe der Fläche hinter der Landkarte kann gewählt werden. Die Größenangabe erfolgt im Rahmen des `map`-Attributs, die Farbgebung erfolgt im Rahmen des `style`-Attributs. Der Hintergrund wird gleich im ersten Abschnitt definiert: **(8)**

```
style=0,666688,1,ccffcc
```

Farbsteuerung durch Daten

Ohne Angabe des Attributs `color` wird als Anfangsfarbe weiß und als Endfarbe rot gewählt. Das Minimum der Zahlenreihe wird der Farbe weiß, das Maximum der Farbe rot zugeordnet. Die anderen Zahlenwerte werden proportional zu ihrer Größe mehr oder weniger rot gefärbt sein.

Fortlaufende Datenreihen

```
data=<d>|...
```

`d` sind die Zahlenwerte pro Region als ganze Zahl oder als Gleikommazahl (Achtung: Dezimalpunkt verwenden). Werden nicht alle Regionen angegeben, wird für den Rest die neutrale Farbe verwendet.

```
data=1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23
```

Jeder Region wird zur einfacheren Kontrolle als darzustellendes Datum die jeweilige Regionsnummer zugeordnet. **(9)**

Datenwerte pro Region

```
data=<b>,<d>|...
```

`b` ist Nummer der betreffenden Region, `d` ist der darzustellende Zahlenwert für diese Region. Für nicht angeführte Regionen wird die neutrale Farbe verwendet. Die Zählung der Bezirke beginnt bei 1. **(10)**

```
data=1,1.5|2,1|3,3.5|10,1.5|11,2.5|12,3.5|17,0.5|20,2|21,1|22,-1
```

Beschriftung

```
data=<b>,<d>,<text>,<dx>,<dy>|...
```

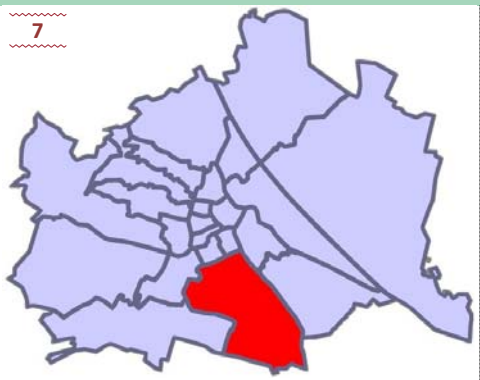
`text` ist der anzuzeigende Text. Der Text wir normalerweise zentriert in die Mitte der Region geschrieben. Wenn der Text aber mit anderen Texten überlappt, kann er mit den Zahlenangaben `dx` und `dy` verschoben werden (positiv nach rechts und nach unten).

Maximal- Minimalwerte

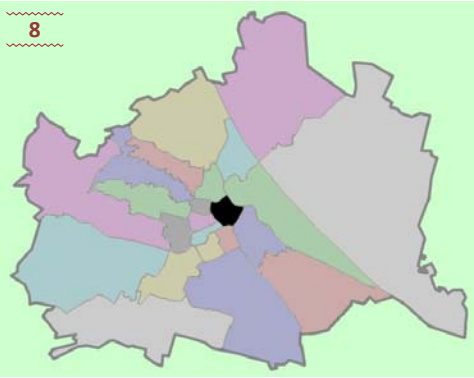
```
data=m<min>,<max>|<d>|...
```

Maximum und Minimum der Datenfolge wird durch das Programm automatisch bestimmt. Dem Minimum wird die Farbe weiß und dem Maximum die Farbe rot zugeordnet. Diese Darstellung ist aber oft irreführend, weil mit weiß

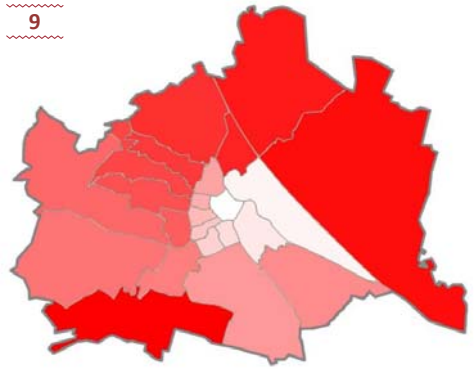
7



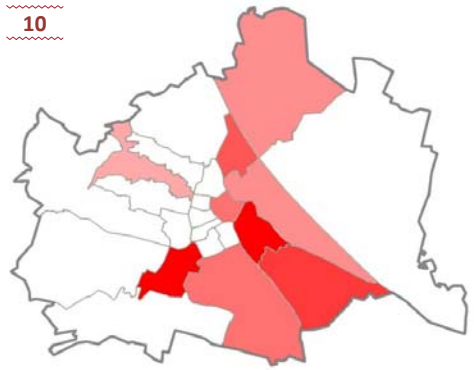
8



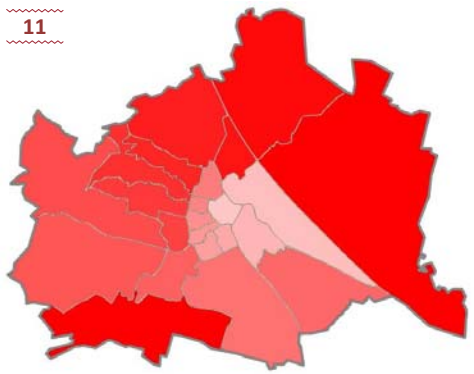
9



10



11



Null assoziiert wird. Weiters können extreme Maximalwerte die Schattierungen der Rot-Werte wenig sichtbar machen. Es besteht daher die Möglichkeit, ein Minimum und ein Maximum für die Datenfolge fest einzustellen. Wählt man daher dieses Minimum unterhalb des Minimums der Datenreihe, dann wird auch dem Minimum ein Rot-Wert und nicht weiß zugeordnet. Wählt man ein kleineres Maximum als dem automatisch bestimmten, kann man die Schattierungen besser sichtbar machen.

```
data=m-5,22|1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23
```

Der Minimalwert `-5` macht die Rot-Schattierung der ersten Region sichtbar; der Maximalwert `22` bewirkt, dass die Regionen `22` und `23` in derselben Intensität dargestellt werden. **(11)**

Visualisierung durch Intensitätssteuerung

Die Datenwerte werden durch die Intensität einer einigen Farbe visualisiert. Um die voreingestellte Farbe rot und die neutrale Farbe ändern zu können, benutzt man den Parameter `color`.

```
color=<rrggbb0>|<rrggbb>
```

Die einzelne Farbe `rrggbb` wird als Maximalwert interpretiert, die Anfangsfarbe ist weiß; `rrggbb0` ist die neutrale Farbe für nicht mit Daten belegten Regionen.

```
data=1|2|3|4|5|6|7|8|9|10|11&color=ffffdd|000044
```

Den Regionen 1..11 wird die jeweilige Regionsnummer zugeordnet; die Datenfarbe ist dunkelblau, die neutrale Farbe ist Creme. **(12)**

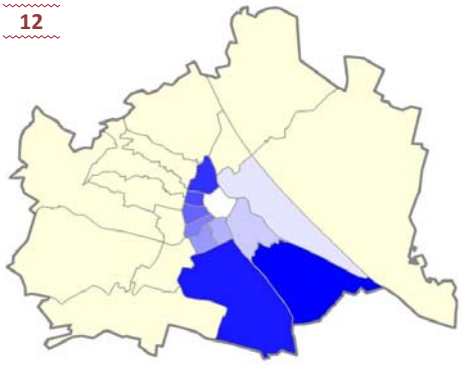
Umkehr der Intensitätsrichtung

```
inv=1
```

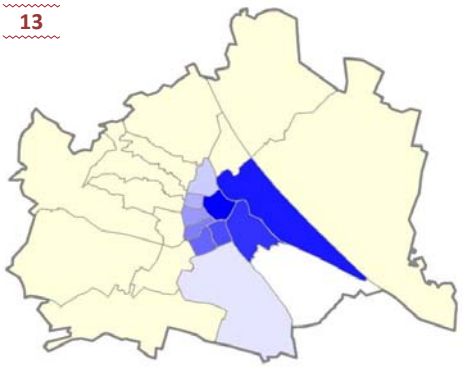
Der Parameter `inv` kehrt die Intensitätsrichtung um. **(13)**

```
data=1|2|3|4|5|6|7|8|9|10|11&color=ffffdd|-000044&inv=1
```

12



13



Ausgangsfarbe schwarz statt weiß

`color=<rrggbb0>|-<rrggbb>`

Die einzelne Farbe `rrggbb` wird als Maximalwert interpretiert. Bei einem vorangestellten Minuszeichen ist die Anfangsfarbe ist schwarz.

`data=1|2|3|4|5|6|7|8|9|10|11&color=ffffdd|-000044` (14)

`inv=1` kehrt die Intensitätsrichtung um

Farbwertsteuerung

`color=<rrggbb0>|<rrggbb1>|<rrggbb2>`

Statt der Farbintensität kann auch der Farbwert die Daten visualisieren, was gerne bei klimatischen Darstellungen verwendet wird. Der kleinste Datenwert entspricht der Farbe `rrggbb1`, der größte Datenwert der Farbe `rrggbb2`. `inv=1` kehrt die Farbrichtung um

`data=1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23&color=ffffdd|ff0000|ffff00` (15)
`data=1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|21|22|23&color=ffffdd|ff0000|ffff00&inv=1` (16)

Vorgespeicherte Darstellungen

Bei jeder Region sind die Fläche und die Einwohnerzahl gespeichert. Diese Daten können direkt visualisiert werden oder man kann auch die eigenen Daten auf eine dieser Größen beziehen.

Größe der Region

`data=area`

Bevölkerungszahl

`data=population`

Bevölkerungsdichte

`data=density`

Beispiele

`data=area&color=000000|003300` (17)
`data=population&color=000000|000033` (18)
`data=density&color=000000|003344` (19)

Eigene Daten auf die vorgespeicherten Größen beziehen

Man kann die vorgespeicherten Daten mit eigenen Daten kombinieren. Jede Datenfolge kann auf die Fläche, die Einwohnerzahl oder die Einwohnerdichte bezogen werden. Gleichzeitig kann auch ein Minimum oder Maximum der Datenwerte voreingestellt werden.

Bezug auf die Größe der Region

`data=area|<d>|...`
`data=area|m<min>,<max>|<d>|...`

Bezug auf die Bevölkerungszahl

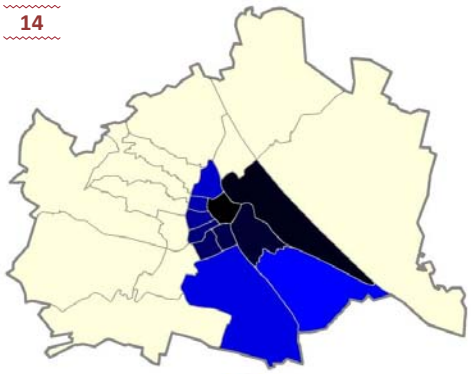
`data=population|<d>|...`
`data=population|m<min>,<max>|<d>|...`

Bezug auf die Bevölkerungsdichte

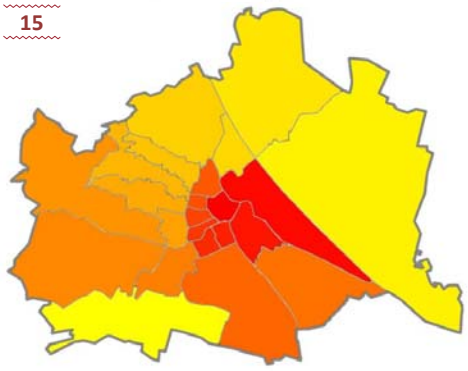
`data=density|<d>|...`
`data=density|m<min>,<max>|<d>|...`

Zur Demonstration wird die Leserzahl der PCNEWS in absoluten Zahlen pro Region dargestellt und danach im Vergleich bezogen auf die Bevölkerungszahl.

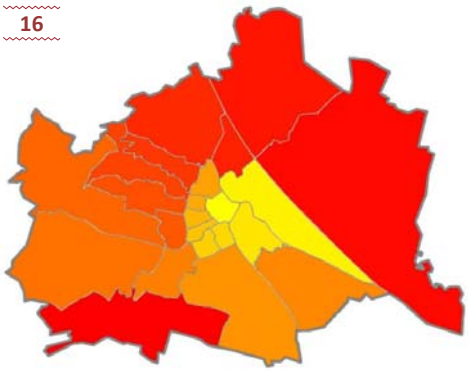
14



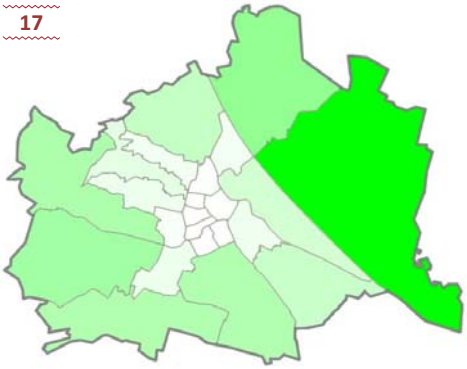
15



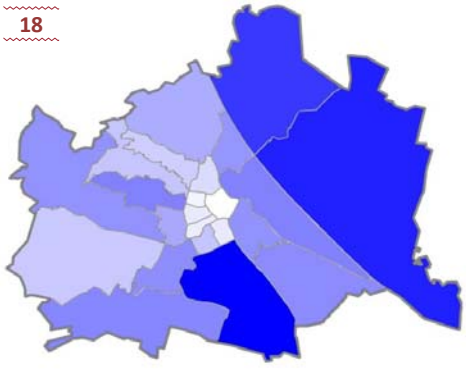
16



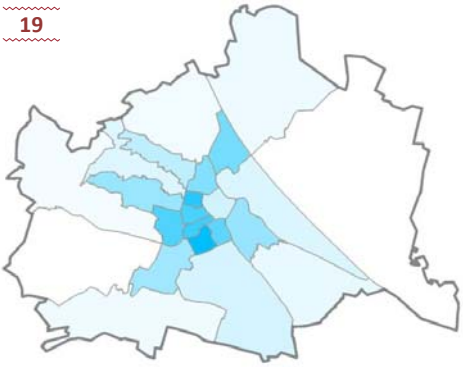
17



18



19



Leserzahlen absolut (Wien)

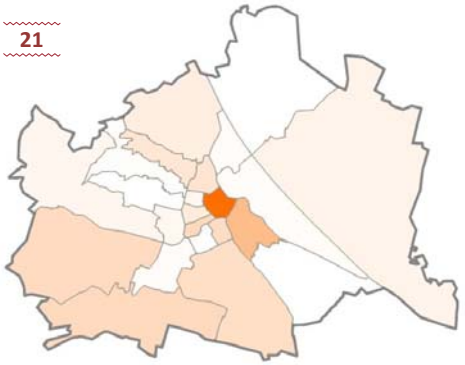
`data=25|25|67|16|12|16|12|7|18|89|20|22|27|26|21|22|13|21|25|21|32|52|50&color=000000|886040` (20)

20

Leser bezogen auf die Bevölkerungszahl

`data=population|25|25|67|16|12|16|12|7|18|89|20|22|27|26|21|22|13|21|25|21|32|52|50&color=000000|886040` (21)

21



Leserzahlen absolut (Österreich)

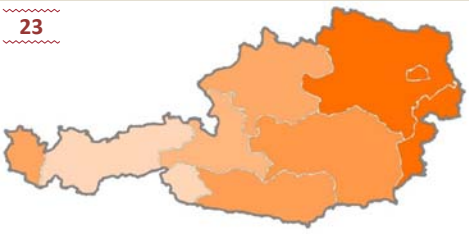
`map=at|20&data=m-100,300|32|37|340|84|27|87|20|23|639&color=000000|886040` (22)

22

Leser bezogen auf die Bevölkerungszahl

`map=at|20&data=population|m0,0.0001|32|37|340|84|27|87|20|23|639&color=000000|886040` (23)

23



Overlays

Overlays sind Grafikelemente die auf der Landkarte platziert werden können.

Vordefinierte Anzeigen

Mittelpunkt der Region zeigen (24)

`show=center`

Flagge der Region zeigen (25)

`show=flag`

Namen der Region anzeigen (26)

`show=name`

Texformatierung

`style=||||<font-family>,<font-size>,<font-weight>,<fill>,<text-align>,<text-anchor>`

Wenn dieser 5. Bestandteil von `style` nicht angegeben wird, werden folgenden Anfangswerte gewählt:

`style=||||Tahoma,10,bold,000000,center,middle`

`font-weight {normal | bold | bolder | lighter}, text-align {left | right | center}, text-anchor {start | middle | end}`

Text verschieben

Wenn sich die Texte überlappen, kann man Beschriftungen einzelner Regionen verschieben.

`data=<b>,<d>,<text>,<dx>,<dy>|...`

Externe Bilder

Das Bild muss durch eine Internetadresse angegeben sein. Man gibt die Position und die Größe an sowie die Deckkraft. Bei mehreren Bildern desselben Verzeichnisses genügt es, die Adresse nur beim ersten Bild anzugeben.

`ovl=<caption>,<url>,<img>,<x>,<y>,<width>,<height>,<opacity>,<resize>|...`

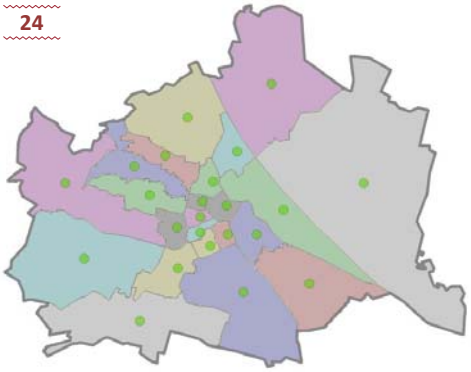
`caption` ist der Name des Bildes, `url` die Adresse ohne den Dateinamen aber mit dem letzten Slash, `img` ist der Dateiname. Der Koordinatenursprung der Landkarte ist die linke obere Ecke, die positive x-Richtung läuft nach rechts, die positive y-Richtung nach unten. `x` und `y` sind die Koordinaten der linken oberen Ecke des überlagerten Bildes; `width` und `heigth` sind Breite und Höhe des Bildes in Pixel. `opacity` ist die Deckkraft (0 ist durchsichtig, 1 ist undurchsichtig).

Wenn die wirkliche Pixelzahl des Bildes größer ist als die Pixelzahl auf der Landkarte, dann verhalten sich die Browser leider nicht gleich. Der Firefox-Browser lässt das Bild unverändert und benutzt die übliche Html-Skalierung. Der Chome-Browser reduziert aber automatisch die Pixelgröße auf den neuen Wert. Bei Betrachtung der Landkarte ergibt das zunächst keinen Unterschied, denn man kann ohnehin nicht weniger als ein Pixel auflösen. Wenn es aber darum geht, einen Landkartenausschnitt vergrößert zu zeigen, dann kann Firefox das Bild noch korrekt auflösen, Chrome aber nicht. Für diesen Fall ist der zusätzliche Parameter `resize` gedacht, der mit dem Wert `resize=0` und einem Svg-Trick auch bei Chrome die volle Auflösung ermöglicht. Man braucht den Parameter nicht, wenn die überlagerten Bilder dieselbe Auflösung haben wie die darunter liegende Landkarte.

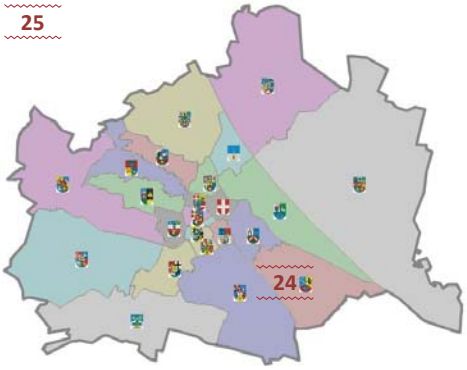
`grid=<n>`

Mit dem Parameter `grid` wird ein Gitter über Landkarte gelegt, damit man die genaue Position für die Bilder besser abschätzen kann. `n` ist die Anzahl der Pixel pro Linie.

24

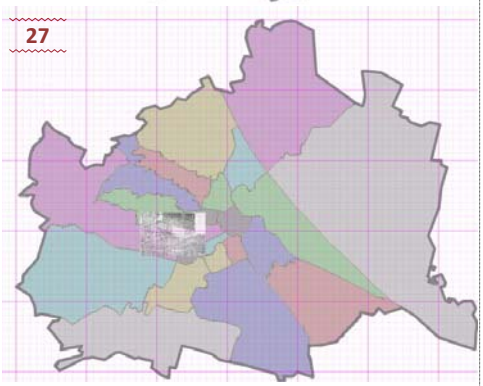


25



26

27



Historische Landkarte am Stadtplan von Wien

Das Bild zeigt den um 20 Pixel verschobenen Raster, und ein überlagertes Bild. (27)

`http://iam.at/map/map.aspx?map=at-9|20,20,20,20&ovl=Schmelz,http://rapid.iam.at/d/kmlo/,WienHistorisch2.jpg,176,269,93,71,0.8,1&grid=10`

Ausschnittsvergrößerung

In dieser Form ist das überlagert Bild nicht gut sichtbar. Man kann aber durch die Angabe eines Ausschnitts im Atttribut `map` den dargestellten Kartenausschnitt wählen. (28)

`http://iam.at/map/map.aspx?map=at-9|20,20,20,20|176,269,93,71&ovl=Schmelz,http://rapid.iam.at/d/kmlo/,WienHistorisch2.jpg,176,269,93,71,0.8,0|athletik,,athletik.jpg,190,280,5,5,1,0|rapid,,rapid.jpg,225,295,5,5,1,0|rapid1,,rapid.jpg,235,310,5,5,1,0&grid=10`

Ausgabeformat

`out=<format>`

Das Ausgabeformat kann zwischen SVG, Kode, JPG und vordefinierten Daten gewählt werden.

`out=svg`

Ausgabe als SVG-Datei (optimale Darstellung in Google-Chrome und Firefox. keine Darstellung in Internet-Explorer)

`out=txt`

Ausgabe des SVG-Kodes als Text

Beispiel: Kodefragment

```
?>xml version="1.0" encoding="utf-8"?>
<svg xmlns:xlink="http://
www.w3.org/1999/xlink" version="1.1"
width="654.88721" height="514.11169"
id="svg_AT-9" viewBox="176,269,93,71"
xmlns="http://www.w3.org/2000/svg">
  <g id="Map_AT-9" transform="translate(-
48.862683,-35.793523)">
    <rect id="area" x="48.862683"
y="35.793523" width="654.88721"
height="514.11169" transform="" sty-
le="fill:#ffffff;fill-opacity:1;fill-
rule:evenodd;stroke:#ffffff;stroke-
width:0.0;stroke-miterlimit:4;stroke-
opacity:1;stroke-dasharray:none;stroke-
linecap:round;stroke-linejoin:round;" />
...
```

`out=jpg`

Ausgabe der Landkarte als JPG-Bild. Achtung: bei JPG-Format funktionieren keine Overlays mit Bildern.

`out=data`

Ausgabe der Zusatzdaten (Fläche, Bevölkerung...) für die ganze Landkarte und pro Region: Name, Landkartenindex, Index, Kfz-Kennzeichen, Fläche, Bevölkerung. Dazu kommt der optionale über das Attribut `data` eingegebene Datenwert und der berechnete Wert (bei Verwendung der Fläche, Einwohnerzahl und Bevölkerungsdichte als Bezug).

Wien

Map-Id	AT-9
Id	9
Kfz	W
Fläche	415
Einwohner	1698822
Dichte	4093,54698795181
Datum	none

28