# SVG − Scalable Vector Graphics

**Ein zukünfiger Eckstein der WWW−Infrastruktur[1]**


## 1. Was ist SVG?

»Scalable Vector Graphics« (Acronym ist SVG) ist der neue offizielle Vektorgraphikstandard des W3C (World Wide Web Consortium). Wie alle neuen W3C Standards ist er XML−basiert und arbeitet mit anderen XML−Basistechnologien wie DTD, Schema, XSL, XQL, SMIL, XHTML, XFORMS, XQL, RDF, Namespaces, etc. zusammen. SVG ist auch DOM[2]−transparent und erlaubt den Einsatz von ECMA−Javascript. Dies bedeutet, dass jedes einzelne SVG−Element und deren Attribute zur Laufzeit mit Hilfe von Scripten verändert werden kann. SVG erlaubt erstmals einen offenen Standard zu benutzen, um hoch−qualitative vektorbasierte und interaktive Graphik innerhalb von Web−Applikationen zu verwenden. Vorher konnte man lediglich proprietäre Standards verwenden.

## 2. Wer entwickelt SVG?

Der SVG−Standard wird von einem Konsortium, bestehend aus Mitgliedern des W3C, Forschungsinstituten und Firmen aus dem Graphik−, Multimedia− und Netzwerk−Umfeld entwickelt. Zu den Firmen, die in den Standardisierungsprozess involviert sind, gehören Adobe, Apple, Autodesk, Bitflash, Canon, Corel, HP, IBM, Kodak, Macromedia, Microsoft, Netscape, Quark, Sun, Visio. SVG wird als offenes Format auch von der Open−Source Community unterstützt. Das W3C Konsortium und die SVG−Arbeitsgruppe ist offen für Vorschläge von Institutionen und Individuen.

## 3. Einsatzbereiche von SVG

Als offener Standard, der hochqualtiative Vektorgraphik, Rastergraphik, Animation und Interaktivität unterstützt, kann SVG für alle Webseiten mit dynamischen Inhalten und der Forderung nach hochqualtiativer Graphik eingesetzt werden. Mögliche Anwendungsbereiche sind:

- Webdesign generell
- Technische Illustrationen
- Wissenschaftliche Datenvisualisierung (Chemie, Physik, explorative Visualisierung, etc.)
- Webmapping und Online GIS−Dienste, Navigation und Wegefindung.

---

1 Dieser Artikel ist die leicht modifizierte deutschsprachige Übersetzung der Originalfassung in Englisch (»SVG − a future cornerstone of the WWW−Infrastructure«), die im vierteljährlich herausgegebenen Interchange Newsletter der internationalen SGML−Usergroup (http://www.isgmlug.org/) erscheinen wird.

2 DOM ist die Abkürzung für »Document Object Model« und regelt den Zugriff auf alle Elemente einer Webseite und deren Attribute.

- Location based services (SVG Mobile)
- Online−Kataloge and E−Commerce Projekte
- Online Lern−Systeme
- Multimedia und Unterhaltung
- User−Interfaces für Webseiten und Webapplikationen
- SVG als offenes Graphik−Austauschformat (zwischen versch. Graphik−Software und versch. Plattformen)
- SVG Viewer als Rendering−Komponente von Offline−Applikationen (ActiveX, COM, Java Beans, etc.)

# 4. SVG's Architektur und Dokumentenstruktur

Wie jedes XML−File, besteht auch SVG im Groben aus einem »header«, einem »root−element« (svg−element) und verschiedenen Kind−Elementen und deren Attribute. SVG macht extensiven Gebrauch von XML und CSS[3] Attributen. SVG−Entwickler können »Entitäten« definieren, die später im File wiederverwendet werden können. Diese Technik erleichtert beispielsweise CSS−Definitionen und den Einsatz von wiederverwendbarer Graphik. Eine weitere Möglichkeit der Wiederverwendung ist der Einsatz des »<use xlink:href="">−elements« wobei auf eine eindeutige Objekt−ID verwiesen wird. Zentrale Definitionen innerhalb der »<defs>−section« (z.B. Symbole, Farbverlaufsdefinitionen, Füllungen, etc.), werden nicht gerendert, können aber im weiteren Verlauf des Files instantiert werden.

SVG kennt schliesslich auch Gruppierungselemente (ähnlich dem Einsatz von Layern) und sog. »Switch−statements«, die einfache Kontrollstrukturen ermöglichen. So kann beispielsweise die Sprache des Clients abgefragt werden und dementsprechend unterschiedliche Sprachversionen ausgeliefert werden. Die Elemente im SVG−Dokument haben eine implizite Zeichnungs−Reihenfolge: Elemente am Beginn der Datei werden zuerst gezeichnet, folgende Elemente werden über die vorangehenden Elemente gezeichnet, unter Berücksichtigung der Opazitätswerte.

```
<?xml version="1.0" standalone="no"?>
   <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
   "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
    <svg width="5cm" height="4cm" xmlns="http://www.w3.org/2000/svg">
       <desc>Four separate rectangles</desc>
       <rect x="0.5cm" y="0.5cm" width="2cm" height="1cm"/>
       <rect x="0.5cm" y="2cm" width="1cm" height="1.5cm"/>
       <rect x="3cm" y="0.5cm" width="1.5cm" height="2cm"/>
       <rect x="3.5cm" y="3cm" width="1cm" height="0.5cm"/>
    </svg>
```

Obiger SVG−Code zeigt ein »well−formed« und »valid« SVG−Dokument, das vier Rechtecke darstellt. Der Code ist sehr einfach zu lesen, editieren und interpretieren.

# 5. SVG's graphische Möglichkeiten

SVG kennt prinzipiell 3 Arten von Graphik−Objekten: Vektorbasierte Geometrie, Rasterbilder und Texte. SVG−Viewer stellen die Graphik üblicherweise in hoher Qualität dar, unter Zuhilfenahme von Antialiasing−Techniken. Dabei wird intern eine hierarchische

3 CSS ist das Acronym für »Cascading Style Sheets« und dient zur zentralen Definition von Text− und Graphik−Formaten

DOM−Struktur aufgebaut, die einen schnellen Zugriff auf die einzelnen Elemente und deren Attribute erlaubt. Elemente können hinzugefügt, gelöscht und neu innerhalb der Hierarchie angeordnet werden. Jegliche Graphik, die mit Graphik− oder DTP−Software erzeugt wurde, kann auch mit Hilfe von SVG abgespeichert und angezeigt werden. SVG kennt die folgenden Basis−Geometrie−Typen.

- Rechtecke
- Kreise
- Ellipsen
- Linien
- Polylinien
- Polygone
- Symbole
- Pfad−Elemente

Der Code unterhalb zeigt einige Basis−Elemente. Sie können ihn in ihren Lieblings−Texteditor kopieren, Elemente und Attribute ändern und in ihrem SVG−Viewer ansehen. Source: [14, WINTER, 2001].

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
   "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xml:space="preserve" width="225px" height="300px" viewBox="0 0 300 400">
 <defs>
  <style type="text/css">
   <![CDATA[
    .str {stroke:black;stroke-width:1}
    .fnt {font-weight:normal;font-size:20;font-family:'Arial, sans-serif'}
    .red {fill:red}
    .blu {fill:blue}
    .yel {fill:yellow}
    .gre {fill:green}
    .frn {fill-rule:nonzero}
   ]]>
  </style>
 </defs>
 <g id="mainlayer">
  <text class="fnt" x="44" y="88">rect</text>
  <rect class="blu str" x="150" y="15" width="100" height="50" rx="12" ry="18" />
  <text class="fnt" x="140" y="88">rect (rounded)</text>
  <text class="fnt" x="36" y="180">circle</text>
  <text class="fnt" x="36" y="263">line</text>
  <text class="fnt" x="170" y="180">ellipse</text>
  <text class="fnt" x="140" y="363">path:<tspan x="140" y="383">simple +
    bezier</tspan></text>
  <text class="fnt" x="16" y="363">polygon</text>
  <rect class="red str" x="15" y="15" width="100" height="50" />
  <text class="fnt" x="156" y="255">polyline</text>
  <circle class="yel str" cx="62" cy="135" r="20" />
  <ellipse class="gre str" cx="200" cy="135" rx="50" ry="20" />
  <g style="fill:none; stroke:green">
    <line x1="15" y1="240" x2="30" y2="200" style="stroke-width:2" />
    <line x1="30" y1="240" x2="45" y2="200" style="stroke-width:4" />
    <line x1="45" y1="240" x2="60" y2="200" style="stroke-width:8" />
    <line x1="60" y1="240" x2="75" y2="200" style="stroke-width:10" />
    <line x1="75" y1="240" x2="90" y2="200" style="stroke-width:12" />
  </g>
  <polyline style="fill:none; stroke:red; stroke-width:2" points="160,200 180,230
     200,210 234,220"/>
  <polygon class="yel" transform="scale(.3),translate(-188,830)"
     style="stroke-width:3;stroke:black;" points="350,75 379,161 569,111 397,215
     423,301 350,250 277,301 303,215 231,161 321,161" />
  <path class="str" style="fill:none;stroke-width:3" d="M150 280l19,10 -22,33
     40,3c12,43 44,-83 83,20" />
 </g>
</svg>
```

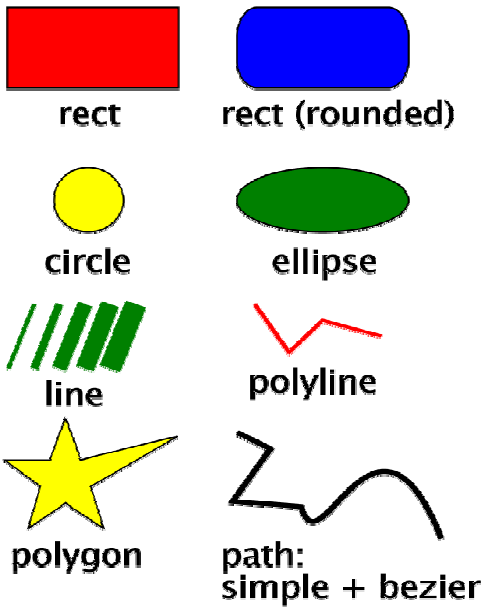


*Figure 1: Basic Shapes.*

Das Symbol−Element kann zur Symbolisierung von Punkt−Objekten verwendet werden. Symbole definiert man am Beginn des Files in der »<defs>−section«. Die Symboldefinition kann beliebigen SVG−Code enthalten (z.B. andere Basis−Geometrie). Das »Pfad−Objekt« ist der mächtigste Geometrie−Typ: es kann Linien−Segmente, quadratische und kubische Bezier−Kurven und Kreisbogensegmente beinhalten. SVG−Pfade können Löcher beinhalten und können aus mehreren disjunkten Objekten bestehen (z.B. kann ein Land mit Inseln oder Exklaven als eine Einheit betrachtet werden.). Marker−Symbole können entlang der Stützpunkte einer Linie plaziert werden und Pfeile an den Endpunkten.

Bei den Linien−Typen unterstützt SVG die »Linienbreite«, »Linienfarbe«, »Opazität«, »Strichlierungsart«, »Linienendentypen« (butt, round und square) und »Linienverbindungstypen« (miter, round, bevel). Zum Füllen der SVG−Objekte wird das Prinzip des »paint−servers« eingeführt. Dabei wird die Methodik (Logik, Kontrollstrukturen) beschrieben, wie die Objekte gefüllt werden sollen. Derzeit gibt es den »solid fill« (uniforme Farbwerte), »Farbverläufe« (linear und radial) und »Musterfüllungen« (Raster− und Vektor, gekachelt oder nicht gekachtelt). Farbverläufe können verschiedene »Stop−Werte« und verschiedene »Ausbreitungsmethoden« haben. Alle Füllungen und Striche können auch Opazitätswerte entgegennehmen. Für komplexe Pfad−Elemente (z.B. bei Löchern) muss auch die »Füllungsregel« (nonzero−winding, evenodd) definiert werden.

SVG erlaubt den Einsatz verschiedener Längeneinheiten, wie es auch bei der CSS−Spezifikation der Fall ist. Derzeit können »em«, »ex«, »px«, »mm«, »pt«, »pc«, »cm« und »in« verwendet werden. SVG erlaubt die Definition zweier separater kartesischer Koordinatensysteme: das Geräte−Koordinatensystem (oder viewport Koordinatensystem) definiert mit dem »width« und »height«−Attribut innerhalb des root−elements, und das »user Koordinatensystem« definiert im »viewBox«−Attribut. Der Ursprung ist wie bei CSS in der linken oberen Ecke der Zeichenfläche, mit der positiven x−Achse nach rechts und der positiven Y−Achse nach unten. »Viewport«− und »User«−Koordinatensysteme können unterschiedlich definiert werden, damit globale Translations− und Skalierungsfaktoren ohne grosse Berechnungen realisiert werden können. Dies ist von Vorteil für skalierte technische Illustrationen und für die Präsentation von Landkarten.

SVG kennt verschiedene Transformationsarten: Skalierung, Translation, Rotation, Schrägstellen und Matrix−Operationen. Jedes einzelne Element oder Gruppe kann sein eigenes lokales Koordinatensystem haben, entsprechend der jeweiligen Transformationsparameter. Transformationen können auch verschachtelt werden. Beinahe jedes SVG−Objekt kann beschnitten werden oder als Beschneidungspfad dienen, inkl. Textelemente.

SVG ist bezüglich der Verwendung von Text− und Graphikformate CSS und XSL transparent. Formate können zentral im Header des SVG−files definiert werden, in externen Stylesheet−Files (in diesem Fall kann es die gleichen Files verwenden wie HTML/XHTML/XML), sowie direkt als Attribut innerhalb des Geometrie− oder Text−Elementes. Es ist auch üblich, die Styles als Entität innerhalb der »doctype−section« des SVG−Header zu definieren. Styles sind ein mächtiges Instrument um ein ganzes grösseres Webprojekt bezüglich des Aussehens (»look and feel«) zentral zu verwalten.

## 6. SVG's Text−Fähigkeiten

Text nimmt innerhalb von SVG eine ähnliche Stellung ein wie alle anderen Basisgeometrie−Elemente − d.h. Text−Elemente können gefüllt werden, Strichdefinitionen erhalten bleiben und Opazitätswerte können zur Anwendung kommen. Text kann beschnitten werden oder als Beschneidungspfad dienen, kann transformiert und animiert werden. SVG unterstützt Internationalisierung (Unicode), bidirektionalen und rechts−nach−links Text. Das »switch−element« kann für die Unterstützung mehrerer Sprachen verwendet werden (ähnlich dem switch/case−statement in Programmiersprachen).

Laut der SVG−Textspezifikation werden sämtliche in der DTP−Welt gebräuchlichen Textformatierungsmöglichkeiten unterstützt, wie z.B. Schriftart, Gewicht, Schriftschnitte, Schriftlage, Schriftgrösse, Sperren, Kerning, usw. In der aktuellen SVG−Spezifikation (1.0) ist jedoch kein mehrzeiliger Text vorgesehen. Aufgrund der Entwicklerwünsche wird dieses Feature jedoch in die nächste Spezifikationsversion aufgenommen. Derzeit muss man entweder einzelne Zeilen verwenden oder fremde XML−Namespaces/Objekte für Textblöcke verwenden, z.B. XHTML. Sie können einzelne Glyphen oder ganze Schriftsätze einbetten/verknüpfen, um sicherzugehen, dass der Benutzer auch die Schriftart zu sehen bekommt, die Sie als Webautor definiert haben, selbst wenn diese nicht auf dem Client−System installiert ist. Einzelne Zeichen und Wörter können unterschiedliche Baselines haben und einzelne Zeichen können rotiert werden. Ein nützliches Feature ist die Möglichkeit Text an Pfad−Objekten auszurichten. Dies kann beispielsweise in der Kartographie zur Beschriftung von Linienobjekten zur Anwendung kommen.

## 7. SVG und Filter−Effekte

Ein spezielles, doch recht brauchbares, Feature von SVG ist der Einsatz von Filter−Effekten. Jedes Objekt, nicht nur Rasterbilder, kann gefiltert werden. Filter sind in der »Rendering−Pipeline« zwischen der Rasterisierung der Objekte und der Darstellung am Ausgabegerät (das ja in aller Regel rasterbasiert arbeitet) angeordnet. Derzeit können »Lichteffekte«, »Schärfe/Unschärfe−Filter«, »Überblenden«, »Kombinieren«, »Fluten«, »Zusammenfügen«, »Morphologie«, »Turbulenz«. Filter können auch kombiniert werden oder nur auf eine Region des graphischen Objekts angewandt werden.

Der Code und das Bild unterhalb zeigen zwei Pfad−Elemente und ein Text−Element mit kombinierten Filtern darauf angewandt: »Gaussian Blur«, »Specular Lighting«, »PointLight«, »Compositing« and »merging«. Quelle: SVG−Spezifikation unter [4,

FERRAIOLO et.al, 2001].

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
    "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="7.5cm" height="5cm" viewBox="0 0 200 120"
     xmlns="http://www.w3.org/2000/svg">
  <title>Example filters01.svg - introducing filter effects</title>
  <desc>An example which combines multiple filter primitives
        to produce a 3D lighting effect on a graphic consisting
        of the string "SVG" sitting on top of oval filled in red
        and surrounded by an oval outlined in red.</desc>
  <defs>
  <filter id="MyFilter" filterUnits="userSpaceOnUse" x="0" y="0" width="200"
       height="120">
      <feGaussianBlur in="SourceAlpha" stdDeviation="4" result="blur"/>
      <feOffset in="blur" dx="4" dy="4" result="offsetBlur"/>
      <feSpecularLighting in="blur" surfaceScale="5" specularConstant=".75"
                          specularExponent="20" lighting-color="#bbbbbb"
                          result="specOut">
        <fePointLight x="-5000" y="-10000" z="20000"/>
      </feSpecularLighting>
      <feComposite in="specOut" in2="SourceAlpha" operator="in"
         result="specOut"/>
      <feComposite in="SourceGraphic" in2="specOut" operator="arithmetic"
                   k1="0" k2="1" k3="1" k4="0" result="litPaint"/>
      <feMerge>
        <feMergeNode in="offsetBlur"/>
        <feMergeNode in="litPaint"/>
      </feMerge>
    </filter>
  </defs>
  <rect x="1" y="1" width="198" height="118" fill="#888888" stroke="blue" />
  <g filter="url(#MyFilter)" >
          <g>
      <path fill="none" stroke="#D90000" stroke-width="10"
            d="M50,90 C0,90 0,30 50,30 L150,30 C200,30 200,90 150,90 z" />
      <path fill="#D90000"
            d="M60,80 C30,80 30,40 60,40 L140,40 C170,40 170,80 140,80 z" />
      <g fill="#FFFFFF" stroke="black" font-size="45" font-family="Verdana" >
        <text x="52" y="76">SVG</text>
      </g>
    </g>
  </g>
</svg>
```


*Abbildung 2: SVG Filter Effekt*

# 8. SVG und beschreibende Animation

Innerhalb SVG kann beinahe jedes Element und v.a. dessen Attribute animiert werden. Die Syntax und Spezifikation wird dabei von SMIL[4] ausgeliehen. Farbe, Geometrie, Tranformation und Lokalität sind nur ein paar der Eigenschaften die animiert werden können. Speziell interessant ist auch die Möglichkeit, Objekte entlang eines Pfades zu bewegen. Das Objekt kann dabei entsprechend der Orientierung des jeweiligen Pfad-

4 SMIL ist die Abkürzung von »Synchronized Multimedia Integration Language« und dient zur Spezifikation von Multimedia-Elementen und Animation

Segments ausgerichtet werden. Animationen können auch kombiniert und verschachtelt werden. SVG unterstützt »diskrete«, »lineare«, »schrittweise« und »spline«−basierte Interpolation. »Diskret« bedeutet eine sprunghafte Animation von einer Lokalität zur nächsten, »schrittweise« (paced) garantiert eine kontinuierliche gleichbleibend schnelle Bewegung und »spline−basiert« ist die flexibelste, wenn auch komplexeste Interpolationsart, die etwa auch Beschleunigungseffekte simulieren kann. Um die Variable »Zeit« zu kontrollieren, gibt es das »duration«−Attribut. Die Kombination von »keyTimes« und »values« (relative Zeit/Werte Paare) hat sich bei Animationen als besonders nützlich herausgestellt. Mit Hilfe von »keyTimes« kann man Zeitpunkte relativ zur Gesamtdauer der Animation definieren, »values« definiert den jeweiligen Attributwert zum Zeitpunkt X. Dies entspricht in etwa der »Zeitlinien−Metapher« wie sie auch von Macromedias Autorensystemen geprägt wurde. Der Start und das Ende einer Animation kann auch relativ zu einem anderen Animationsevent festgelegt werden. Alternativ zur beschreibenden Animation kann auch Javascript mit Hilfe von Timern und DOM−Zugriffen für die Animation von Elementen angewandt werden.

Das folgende Beispiel zeigt die Verwendung von SVG−Animationen und insbesondere des »animateMotion«−Elements, das erlaubt, ein Objekt (in unserem Fall ein Flugzeug) entlang eines Pfades zu bewegen. Unser Beispiel kombiniert 3 Animationen: Animation entlang eines Pfades, eine Skalierungs−Animation für Start und Landung des Flugzeuges und eine Animation der Opazität, um Beschriftungen der wichtigsten Städte ein/auszublenden während das Flugzeug »vorbeifliegt«.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
"http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-20000303-stylable.dtd" [
        <!ENTITY stKanton "stroke:#3984FF;stroke-width:150;fill-rule:nonzero;
          fill:#C5FFE8;">
        <!ENTITY flightRoute "stroke:orange;stroke-width:2000;fill:none;">
        <!ENTITY cityText "font-size:8000;font-family:'sans-serif';">
        <!ENTITY animDuration "10s">
]>
<svg xml:space="preserve" width="900" height="600" viewBox="480000 0 360000
240000">
<defs>
  <symbol id="symbolRect" overflow="visible">
    <rect x="-3000" y="-3000" width="6000" height="6000" style="fill:
      rgb(240,65,25); fill-opacity: 0.8; stroke: rgb(0,0,0); stroke-width:300" />
  </symbol>
  <symbol id="symbolCirc" overflow="visible">
    <circle cx="0" cy="0" r="3000" style="fill: rgb(12,166,107); fill-opacity:
      0.8; stroke: rgb(0,0,0); stroke-width:300" />
  </symbol>
  <symbol id="airplane" overflow="visible">
    <path style="stroke:blue;stroke-width:100;fill:lightgray;" d="M-4000,0
      a1000,300 0 0,1 1000,-300 H-1000 L1500,-3000 h400 L0,-300 h2000 L3000,-1500
      h500 L2500,-50 V100 L3500,1500 h-500 L2000,300 h-2000 L1900,3000 h-400 L-
      1000,300 H-3000 a1000,300 0 0,1 -1000,-300" />
  </symbol>
</defs>
<g id="Kantone">
<path id="Zuerich" style="&stKanton;" d="M673825,63480 l1077,-820 ... some
path_coordinates ... z"/>
<!-- more path elements cut off ..., holds geometry of swiss cantons -->
</g>

<g id="AnimationPaths">
        <path id="Zuerich_Geneva" style="&flightRoute;" d="M682500,53500
          C632500,53500 549500,80000 499500,181000" />
        <use id="AirplaneZurichGeneva" xlink:href="#airplane">
                <animateMotion id="animMotionZurGen" dur="&animDuration;"
                   repeatCount="indefinite" rotate="auto-reverse">
                      <mpath xlink:href="#Zuerich_Geneva"/>
                </animateMotion>
                <animateTransform id="animTransZurGen" attributeName="transform"
                  attributeType="XML" type="scale" keyTimes="0;0.2;0.8;1"
                  values="1.5;4;4;1.5" dur="&animDuration;" additive="replace"
                  fill="freeze" repeatCount="indefinite"/>
```

```
        </use>
</g>
<text id="labelZurich" x="687000" y="50000" style="&cityText;">Zurich
        <animate id="textAnimZur" attributeType="CSS" attributeName="opacity"
          keyTimes="0;0.15;0.3;1" values="1;1;0;0" dur="&animDuration;"
          repeatCount="indefinite" />
</text>
<text id="labelBern" x="608500" y="102500" style="&cityText;">Bern
        <animate id="textAnimBer" attributeType="CSS" attributeName="opacity"
          keyTimes="0;0.15;0.3;0.5;0.65;1" values="0;0;1;1;0;0"
          dur="&animDuration;" repeatCount="indefinite" />
</text>
<text id="labelGeneva" x="504000" y="187000" style="&cityText;">Geneva
        <animate id="textAnimGen" attributeType="CSS" attributeName="opacity"
          keyTimes="0;0.65;0.8;1" values="0;0;1;1" dur="&animDuration;"
          repeatCount="indefinite" />
</text>
<g id="staedteRect">
        <use id="Zurich" x="682500" y="53500" xlink:href="#symbolRect"/>
        <use id="Basel" x="612000" y="33500" xlink:href="#symbolRect"/>
        <use id="Genf" x="499500" y="181000" xlink:href="#symbolRect"/>
        <use id="Bern" x="604500" y="106000" xlink:href="#symbolRect"/>
</g>
<g id="staedteCirc">
        <use id="Luzern" x="665000" y="89000" xlink:href="#symbolCirc"/>
        <use id="St. Gallen" x="748000" y="46500" xlink:href="#symbolCirc"/>
        <use id="Chur" x="759500" y="109000" xlink:href="#symbolCirc"/>
        <use id="Lugano" x="718500" y="206500" xlink:href="#symbolCirc"/>
</g>
</svg>
```

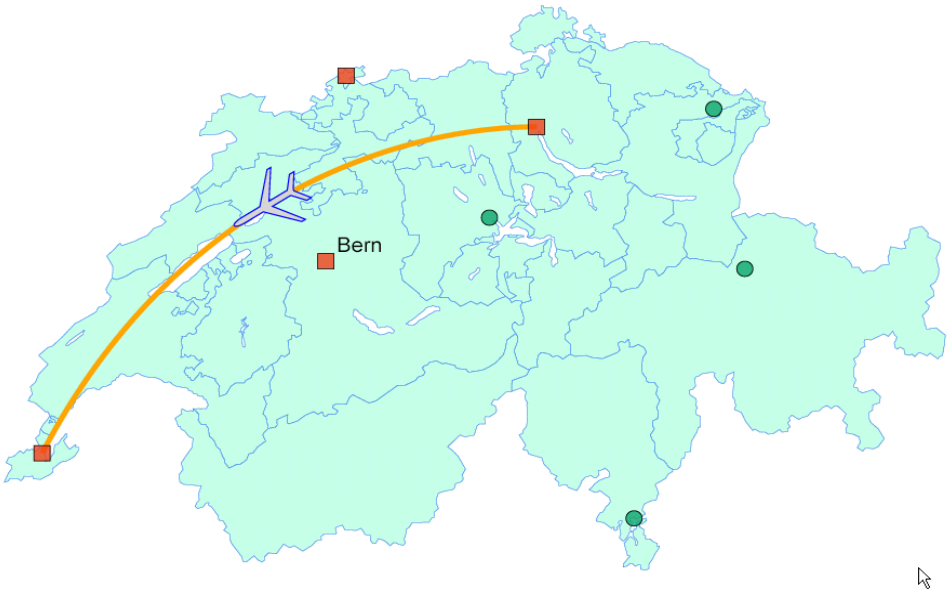


*Abbildung 3: SVG und Animation. Quelle: [9, NEUMANN, 2001]*

# 9. Interaktives SVG

Interaktivität ist ohne Zweifel eine der Stärken, die SVG interessant machen für dynamische Webseiten mit graphischen Inhalten. Interaktivität kann im Client−System und am Server implementiert werden − mit verschiedenen Antwortszeiten natürlich. In der Praxis wird man die beiden Ansätze häufig kombinieren. Serverseitige Datenbanken und Applikationen kann man zum Selektieren und Berechnen von Daten verwenden, während man am Client Maus−, Keyboard und Status−Events dazu verwenden kann, um mit Hilfe eines Subsets der extrahierten Daten schnell auf Anfragen des Benutzers reagieren zu können. Serverseitig sind Scripting−Sprachen und Techniken wie PHP, PERL, JSP/Servlets und ASP häufig im Einsatz, Clientseitig sind Scriptingsprachen wie Javascript, VBScript, Python, Perl und C# populär. So kann Javascript beispielsweise dazu verwendet werden, um den SVG−Szenegraphen (DOM−Tree) zu bearbeiten, Elemente zu erzeugen, zu löschen und deren

Attribute zu verändern. Neben dem Reagieren und Auswerten von Events können SVG−Entwickler auch die Maus−Cursor ändern und Hyperlinks gemäss der XLINK/XPOINTER Spezifikation einsetzen, um auf Objekte innerhalb des DOMs aber auch in externen Dokumenten zu verweisen.

Eine der aussergewöhnlichen interaktiven SVG−Beispiele ist Adobes »SVG Draw«−Anwendung. Sie erlaubt dem Benutzer interaktiv und online Zeichnungen zu erstellen: Elemente können kreiert und transformiert werden, deren graphische Attribute wie Füllung, Farbe, Strich−Typ und Transparenz können interaktiv angepasst werden. Andere interessante interaktive Anwendungen und Tutorials können unter http://www.kevlindev.com/ und http://www.carto.net/ betrachtet werden.

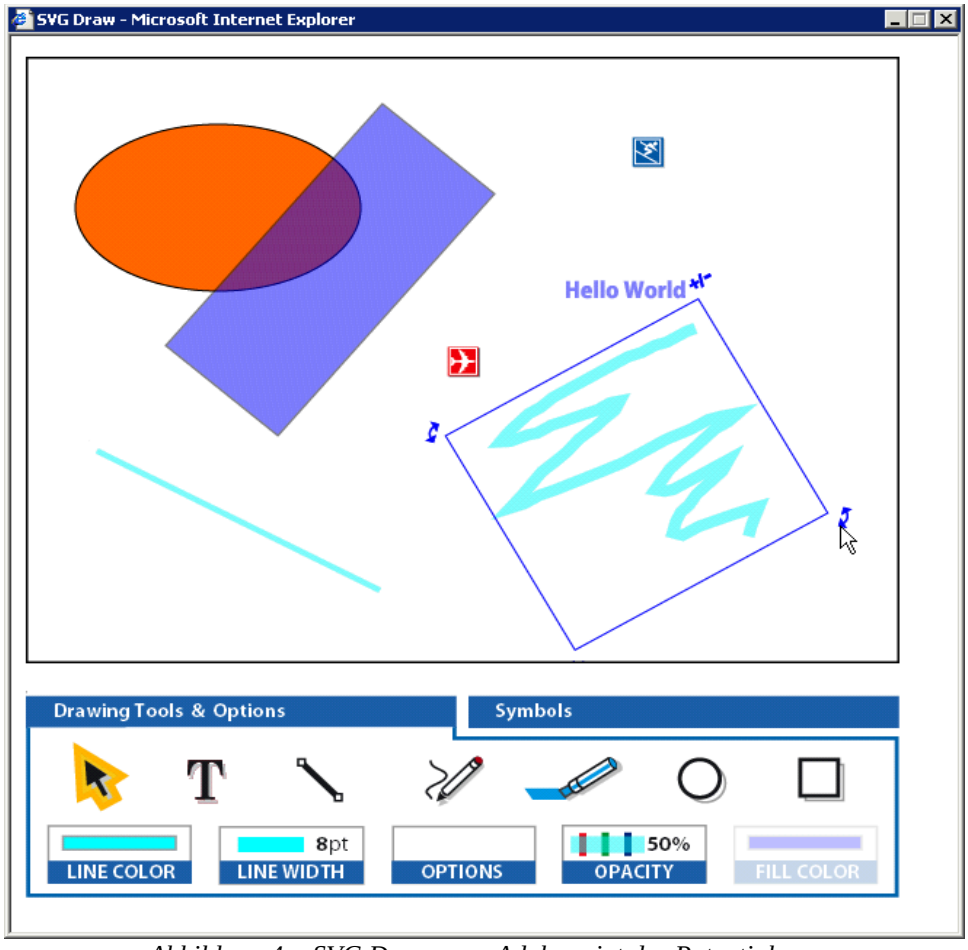


*Abbildung 4: »SVG Draw« von Adobe zeigt das Potential von interaktivem SVG auf dem Client*
*Quelle: [1, ADOBE, 2001]*

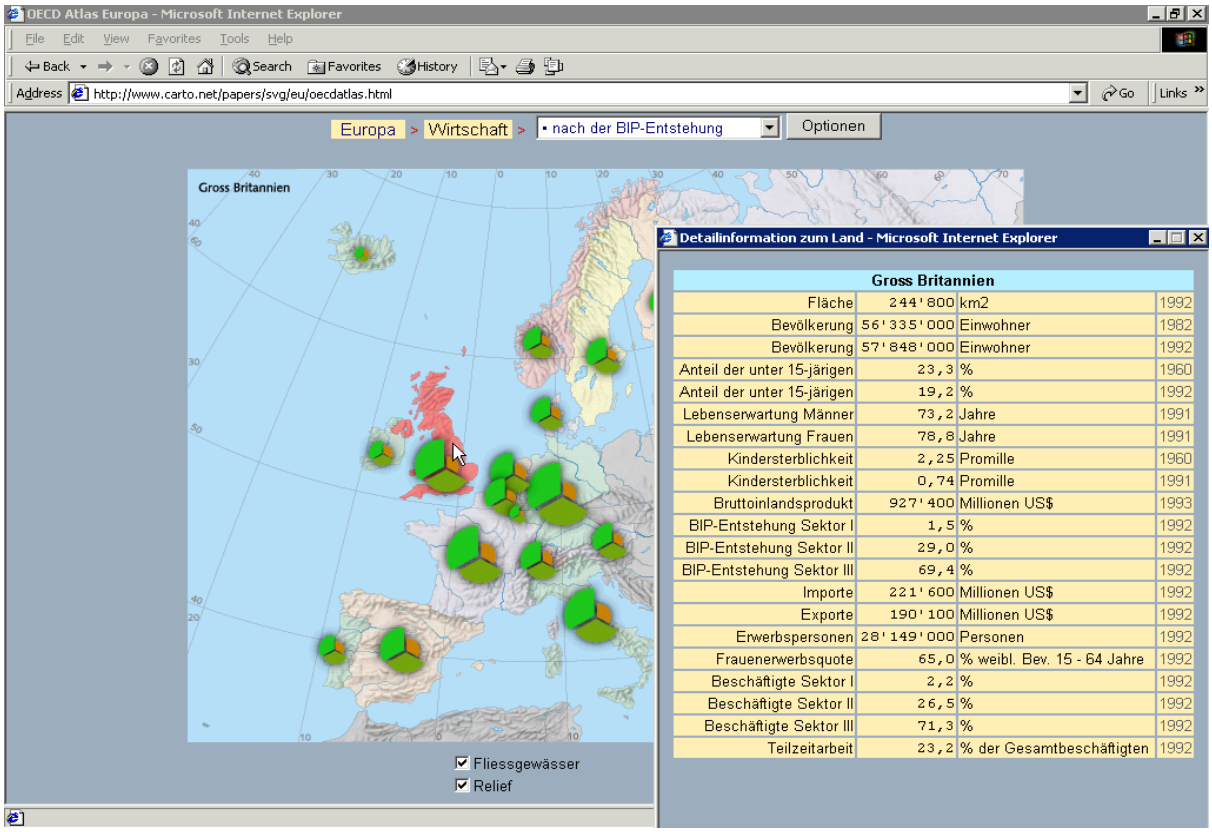


*Abbildung 5: Interaktive Karte von Europa (Wirtschaftliche Situation) − zeigt die Verwendung von SVG für die Kartographie und die Präsentation geographischer Daten. Die Diagramme werden zur Laufzeit direkt aus den statistischen Daten heraus generiert.*
*Quelle: [13, WINTER, 2000]*

# 10. Erweiterbarkeit und Metadaten

SVG erlaubt das Einbetten von fremden XML−Namespaces. Diese können entweder von zentralen Organisationen/Interessensvertretungen definiert werden, oder von Individuen. Sie können ihre eigenen Attribute innerhalb SVG verwenden, solange diese nicht mit den offiziellen kollidieren. Scripte können diese anschliessend über das DOM anfragen. Eigene Attribute müssen Sie jedoch selbst abfragen und auswerten − SVG−Viewer werden diese zunächst einfach ignorieren. Zum Einbetten von Metadaten hat das W3C Richtlinien herausgegeben, die in der RDF[5] beschrieben sind. Diese sollten innerhalb des »metadata«−Tags verwendet werden.

VRML hatte ein interessante Technik zur Erweiterung von Datentypen und Objekten eingeführt: der »Prototyp−Knoten«. Er erlaubt das Definieren neuer individueller Knotentypen (mit eigener Geometrie, Variablen, Logik und Scripten), die später im File mit eigenen Parametern instantiert werden können. Leider kennt SVG derzeit keine derartigen Mechanismen, obwohl dies sicherlich sehr nützlich wäre.

# 11. SVG Workflows; wie erstellt man SVG−Files?

Es gibt viele Wege um SVG−Dateien zu erstellen, zu konvertieren oder zu editieren. Chris Lilley zeigt unter [7, LILLEY, 2001] eine aktuelle Liste von SVG−Implementierungen.

5 RDF ist Acronym für »Resource Description Framework«

- **Text−und XML−Editoren**

Da SVG−Dateien aus lesbarem Text bestehen, können diese mit einem beliebigen Text−Editor bearbeitet werden. Spezielle Editoren mit Programmier−Unterstützung offerieren zusäztlich Syntax−Highlighting, Klammern−Matching, erweiterte Selektions− wie auch »Finden und Ersetzen«−Mechanismen. XML−Editoren ermöglichen zusätzlich das Kontrollieren von »wellformedness« und »validity« (Gültigkeit) entsprechend der DTD[6]. Sie helfen auch bei der Auswahl der gültigen Elemente und Attribute (ähnlich der Eigenschaftsliste in visuellen Programmierumgebungen). Obwohl Text− und XML−Editoren sicherlich nicht zur Erstellung komplexer Geometrie geeignet sind, können diese helfen, Fehler aufzuspüren, zu korrigieren, Ergänzungen vorzunehmen, sowie SVG Code aus anderen Applikationen zu optimieren.

- **Export aus Graphik−Software**

Als dieser Artikel geschrieben wurde, offerierten die folgenden Software−Produkte SVG−Export− und/oder Import−Filter: Adobe Illustrator, Corel Draw, Mayura Draw, Sphinx Open Editor (In−GmbH), Sketch (Open−Source) und Kontour (Open−Source, KOffice). Nicht alle Filter unterstützen »high−level« SVG−Objekte, einige Filter reduzieren alle Objekte zu Pfad−Elementen. Der Export von Layer−Namen und Objekt−Ids ist leider noch nicht in allen Produkten selbstverständlich. Einige Export−Filter benötigen eine nachträgliche manuelle oder automatisierte Nachbearbeitung/Optimierung ... Mehr Unterstützung aus der Graphik−Industrie ist also erwünscht, wird aber auch ziemlich sicher kommen.

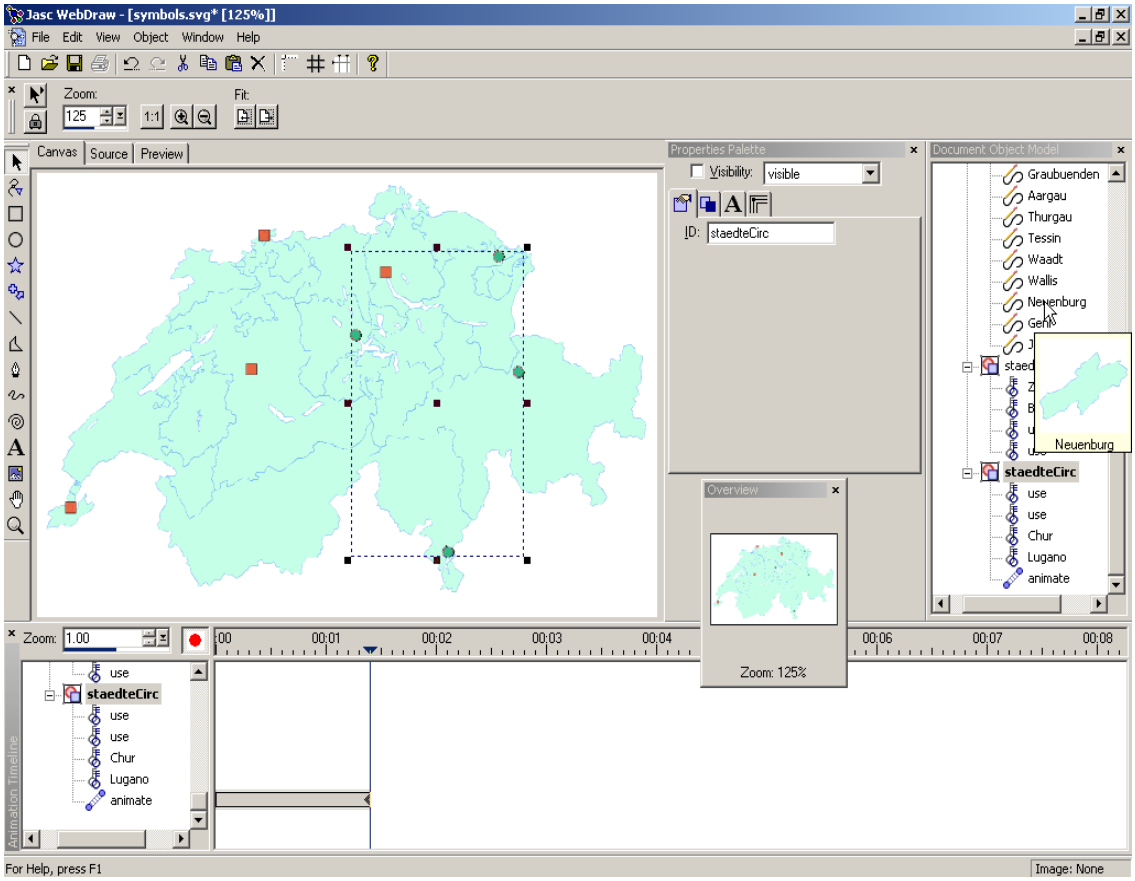


*Figure 6: JASC WebDRAW, a specialized SVG authoring system with animation features.*

6 Die DTD (Document Type Definition) gibt an, welche Elemente und Attribute vorkommen dürfen, welche optional sind und welche zwingend angegeben werden müssen.

- **SVG Editoren und Autoren Systeme**

Derzeit gibt es erst wenige spezialisierte SVG−Autorensysteme. Das derzeit wohl am weitesten fortgeschrittene Produkt ist Jasc »WebDraw« (von der Firma, die PaintShopPro herstellt). Mit WebDraw kann man die Basis−Geometrie interaktiv erstellen, Pfad−Elemente und Symbole hinzufügen. Der SVG Entwickler/Webdesigner kann im WYSIWYG−Modus arbeiten oder im Quellcode − beide Modi führen den jeweils anderen Modus nach. Zusätzlich ist Adobes SVG−Viewer als Komponente für Vorschauzwecke eingebettet. Fehlerhafter Quellcode wird nach der Eingabe validiert und auf mögliche Fehler hingewiesen. Zu den weiteren Highlights zählen ein hierarchischer DOM−Viewer (mit Icon−Vorschau der jeweils selektierten Elemente), einem Navigationstool und kontext−sensitiven Werkzeugleisten, entsprechend dem selektierten Zeichentool. WebDraw hat ausserdem einen integrierten Filter−Editor und eine Zeitlinien−Palette, die zur Erstellung und Kontrolle von Animationen dient. Obwohl das Produkt derzeit noch in der Beta−Phase ist, erscheint es bereits sehr vielversprechend. Es ist auch sehr wahrscheinlich, dass Adobe und Corel SVG−Unterstützung für ihre jetzigen Animationstools und Zeichentools »Adobe LiveMotion«, »Adobe GoLive«, »Adobe Illustrator«, »Corel Draw« und »Corel Rave« hinzufügen werden.

- **SVG Konverter**

Es existieren auch spezialisierte SVG−Konverter (kommerziell und als Open Source) um andere Graphikformate nach SVG zu konvertieren. Es verwundert kaum, dass offene, gut−dokumentierte und strukturierte Formate besser, schneller und einfacher nach SVG konvertiert werden können als undokumentierte und/oder proprietäre Formate. Uns sind Konvertierungstools von SWF (Flash), pdf, ps/eps und wmf bekannt. Zusätzlich gibt es für Java2D, C und Perl−Programmierer libraries, die beim Lesen und Schreiben von SVG−Dokumenten helfen. Mit Scriptingsprachen (wie Perl, Python, PHP, Ruby, etc.), und deren Stärken beim Manipulieren von Text−Strings (Pattern Matching) sind sog. »Quick Hacks« besonders einfach. Das SVG−Format ist sehr gut dokumentiert und es existieren auch XML−Parser und Bibliotheken. Wenn Sie Ihren eigenen Konverter schreiben wollen, können Sie diesen auf ihre eigenen Bedürfnisse optimieren und Features, die sie nicht benötigen einfach weglassen, resp. ihren Konverter ausbauen wenn Sie diese benötigen.

- **SVG Drucker−Treiber**

»Software Mechanics« hat einen SVG Druckertreiber (SVGMaker) entwickelt, der ähnlich wie Adobe Distiller funktioniert, und Applikationen erlaubt, direkt in SVG−Files zu drucken. Dies ist prinzipiell eine sehr gute Idee um SVG−Unterstützung für Applikationen hinzuzufügen, die SVG nicht direkt unterstützen. In der Praxis hat sich jedoch gezeigt, dass dadurch die Dokumentenstruktur verloren geht, und viele Optimierungen und erweiterte Funktionalität, die selber geschriebene Konverter durchführen können, bei einem Druckertreiber leider nicht zur Anwendung kommen können. So dürfte dies v.a. eine Alternative für statische SVG−Graphiken sein.

- **Serverseitige SVG Erzeugung**

Serverseitige Generierung ist eigentlich ein Muss für alle dynamischen Seiten, die stets aktuelle Informationen liefern sollen. Derartige Sites sind meist sehr stark datenbank−gestützt, resp. beziehen ihre Informationen aus grösseren Datensätzen. Vergleicht man serverseitige SVG−Generierung mit serverseitiger Flash−Generierung, so zeigt sich, dass die Integration von serverseitigen Techniken mit text/XML basierten Formaten wesentlich leichter vonstatten geht, als bei binären File−Formaten. Man kann wie bei den oben beschriebenen Konvertern Scriptingsprachen oder richtige Programmiersprachen nutzen. Häufig kommen Techniken wie Perl/CGI, PHP, JSP/Java Servlets und ASP zum Einsatz. Für

den Webmapping−Bereich kann man SVG mit Open−Source−Datenbanken verknüpfen (z.B. PostgresGIS) oder mit kommerziellen Datenbanken (wie z.B. Oracle Spatial oder IBM DB2). Aber auch mehr und mehr kommerzielle GIS−Hersteller bieten SVG−Support an. Schliesslich kann man SVG serverseitig auch mit Hilfe von XSLT−Technologie aus anderen XML−Daten konvertieren. XSLT (Stylesheet−Language) definiert dabei einen Satz von Templates (Vorlagen) und Regeln um strukturierte Daten in andere strukturierte Formate (z.B. SVG) umzusetzen.

- **Client−Seitige SVG Erzeugung**

XSL wird durch zukünftige Browser−Versionen auch direkt auf dem Client unterstützt werden. Bereits heute kann Javascript (oder andere clientseitige Programmiersprachen) dazu benutzt werden um SVG−Code »on the fly« aus Datenfeldern heraus zu generieren. Sind die numerischen Daten bereits zusammen mit den SVG−Files und den Skripten übertragen, so können diese ohne Rückfrage mit dem Server halten zu müssen auf schnelle Art und Weise in SVG umgesetzt werden. Die Daten können dabei in eigenen XML−Dateien, Javascript−Arrays oder versteckten Feldern übertragen werden. Adobes SVG−Viewer bietet zudem auch IDL−Bindings für richtige Programmiersprachen, wie z.B. Java. Beispiele zur Clientseitigen SVG−Erstellung und DOM−Manipulation können unter [8, LINDSEY, 2001] und [11, NEUMANN/WINTER, 2001] eingesehen werden.

- **Optimieren von SVG Graphiken**

Häufig produzieren universelle SVG−Konverter und Export−Filter suboptimalen SVG−Code. Die folgenden Richtlinien können helfen um Dateigrössen zu minimieren und die Wartung der Daten zu vereinfachen.

- Verwenden Sie Stylesheets (CSS oder XSL)
- Verwenden Sie Entities
- Verwenden Sie Symbole
- Verwenden Sie Referenzen/Verweise auf existierende Geometrie und Attribute mit Hilfe von XLINK/XPointer
- Verwenden Sie relative Koordinaten für grössere Geometrie in Pfad−Elementen
- Verwenden Sie Bezier−Kurven anstelle vieler kleiner Liniensegmente in Pfad−Elementen
- Komprimieren Sie die Pfad−Geometrie, indem sie unnötige Leerzeichen weglassen
- Verwenden Sie nur die für Ihren Zweck notwendige Präzision (Anzahl Nachkommastellen), v.a. wenn Sie mit räumlichen Daten arbeiten.
- Komprimieren Sie Ihre SVG, CSS und Javascript−Files mit gzip

- **SVG−Browser/Viewer Implementationen**

Da SVG eine relativ junge Technologie darstellt, gibt es kaum nativen SVG Support in den grossen Browsern. Adobe offeriert das derzeit kompletteste Plugin, das übrigens auch als Komponente in Ihren eigenen Programmen verwendet werden kann. Offiziell werden derzeit die aktuellen Versionen von Internet−Explorer und der Netscape 4.7 Zweig unter Windows und Macintosh unterstützt. Das Mozilla−Team arbeitet an nativem Support für Mozilla 1.0 (Multiplattform) und das KDE/Konqueror Team arbeitet an Support innerhalb ihres exzellenten Browsers für Linux/Unix Systeme. Das W3C fügte ihrem Amaya−Browser Support für statisches SVG hinzu. Daneben gibt es »standalone« Implementationen, die meist in Java2 geschrieben werden: Batik ([2, APACHE.ORG, 2001] − der derzeit populärste standalone SVG−viewer der auch im Quellcode erhältlich ist), Csiro SVG Viewer und IBM SVG View sind nur ein paar der derzeit existierenden SVG/Java2 Implementationen. Keiner davon unterstützt bereits die ganze SVG Spezifikation, Batik kommt dieser jedoch am

nächsten, was statischen SVG−Support betrifft. Java2 und SVG sind recht verwandte Technologien (beide wurden massgeblich von Adobe und Sun entwickelt): Das Java2D API erfüllt alle Voraussetzungen eines 2D Graphik−APIs um damit SVG Viewer implementieren zu können.

# 12. Vergleich zwischen SVG und anderen Vektorformaten für das WWW

Derzeit gibt es v.a. zwei sinnvolle Alternativen neben SVG um Vektorgraphik im WWW zu präsentieren: Macromedia Flash (proprietär) und WebCGM (eine W3C Empfehlung). Nach fast zweijähriger Erfahrung beim Arbeiten mit vektorbasierter 2D−Webgraphik (insbesondere SVG) hat sich herauskristallisiert, dass SVG ein Superset der beiden Formate Flash und WebCGM darstellt. Dies v.a. betreffend Funktionalität und Flexibilität. Weiters ist SVG ein offizieller W3C−Standard und basiert auf XML. XML Technik wird heute und in Zukunft die Basis für alle weiteren Web−Formate liefern. Deshalb können Sie ihre einmal erworbenen XML−Kenntnisse für die verschiedensten Web−Formate und Techniken einsetzen. Technologien wie XSL, DTD/Schema, XQuery, etc. lassen sich ebenso universell für beinahe alle Zwecke einsetzen. Dies ist ein wichtiges Argument für Leute, die sich über hohe Lernschwellen und Hürden beim Erlernen neuer Software−Technologien beklagen. Da Sie Ihr erworbenes XML−Wissen breit einsetzen können, haben Sie mehr Zeit, sich auf die tatsächlichen Inhalte zu konzentrieren, anstatt ständig neue Formate und Syntaxen lernen zu müssen. Auf diese Art− und Weise bilden SVG und XML generell eine Art Symbiose, die schnellen Erfolg für SVG und kurze Entwicklungszyklen garantieren kann.

Für einen vertieften Vergleich zwischen Macromedias .SWF[7] Format (derzeit wohl der direkteste Konkurrent von SVG) und SVG können Sie eine Tabelle unter [10, NEUMANN, 2001] einsehen. Bitte beachten Sie, dass wir mit diesem Vergleich nicht Gelegenheits−Webdesigner ansprechen wollen, sondern technisch interessierte Entwickler und Webdesigner, die mit komplexen dynamischen Inhalten zutun haben und ihre Systeme daher flexibel gestalten müssen um sie schnell an die rasch ändernden eigenen Bedürfnisse (und die der Kunden) anpassen zu können. Das Macromedia Flash−Autorensystem setzt viele der höherentwickelten Features die wir im .SWF−Format vermissen auf einer einfacheren Ebene im endgültigen .SWF−Format über das Autorensystem um. Macromedia unterscheidet zwischen dem undokumentierten .FLA Arbeitsformat und dem weniger kräftigen (aber dokumentierten) .SWF−Format für die Dokumentendistribution (vergl. [12, OPENSWF.ORG, 2001]). Macromedia hat ohne Zweifel SVG hinsichtlich der Animations−Möglichkeiten beeinflusst, wird aber heute in Funktionalität und Flexibilität klar von SVG überboten. Da Macromedia Flash−Plugins jedoch eine hohe Markt−Penetration aufweisen, sich bereits wesentlich länger in Entwicklung befindet und in weiten Teilen auch für Linux verfügbar ist werden SVG und Flash sicherlich für einige Zeit koexistieren. Als dieser Artikel geschrieben wurde, hatten Flash−Autoren auch die ausgereifteren Autorensysteme für die nicht−technischen Benutzer zur Verfügung − ein Umstand, der sicherlich schnell ändern wird.
WebCGM ist der Nachfolger und eine Weiterentwicklung des recht populären CGM (Computer Graphics Metafile)−Formats. Es ist normalerweise ein binäres Format, erlaubt aber die Verwendung von Klartext und Zeichencodierungen. Der CGM−Ansatz ist wesentlich einfacher, als der SVG−Ansatz, kann daher aber auch nur eingeschränkte Interaktivität offerieren und keine Animations−Features. Der CGM−Standard ist daher einfacher zu implementieren, umsomehr als schon beträchtliches CGM Know−How in der Graphik− und CAD−Industrie vorhanden ist. WebCGM erlaubt graphische Objekte, Layer

7 SWF ist ein Acronym für »Shockwave Flash«

und Text−Absätze/Subparagraphen. Die Basis−Graphik−Objekte sind im Groben die gleichen wie in SVG; Transparenz wird ebenso unterstützt wie auch das Einbetten von Rastergraphiken/−bildern. Externe Attribut−Daten können über eine korrespondierende ID mit externen Datenquellen verknüpft werden. CGM erlaubt das Definieren externer Symbol−Bibliotheken und auch den limiterten Einsatz von Graphikformaten. Bezüglich Interaktivität stehen den Entwicklern Hyperlinks, vordefinierte Ansichten, »high−lighted objekts« und »Tool−Tips« zur Verfügung. Die WebCGM Spezifikation kann unter [3, CRUIKSHANK et.al, 2001] eingesehen werden. Es scheint uns zusammenfassend, dass der Einsatz von WebCGM vorerst recht zurückhaltend erfolgt. Es scheint, abgesehen von der Einfachheit der Implementation und der Möglichkeit, sowohl mit binären als auch mit textbasierten CGM−Dateien zu arbeiten, kein wesentlicher Vorteil gegenüber SVG zu existieren. Eine Verwendung im grösseren Stil ist v.a. von zukünftigen Software−Implementierungen und der Unterstützung in den wesentlichen Browser−Projekten abhängig.

# 13. Die Zukunft von SVG

SVG ist ein »heisses« Thema für Web−Developer mit Schwerpunkt auf hochqualitativem graphischem Inhalt und dynamischer Generierung. Es ist derzeit der kräftigste und eleganteste Ansatz um interaktive und animierte Graphik ins Web zu bringen. Der breite Support vom W3C, Forschungsinstituten und Firmen wird eine schnelle Weiterentwicklung der Werkzeuge, Viewer, Autorensysteme und serverseitiger Generatoren garantieren. Es existiern bereits zahreiche Entwickler−Tutorials, Artikel und auch erste Fachbücher sind bereits erschienen. XML bietet weiters eine solide Basis und gute Integration in zukünftige Web−Architekturen. Die Markt−Penetration der SVG−Plugins ist derzeit noch relativ gering und keiner der SVG Viewer unterstützt bereits die gesamte Spezifikation. Leider ist Adobes SVG Viewer (sicherlich der populärste und am weitesten entwickelte Viewer) noch nicht für die Linux/Unix Plattform erhältlich. Es ist zu hoffen, dass die klar wachsende Popularität von Linux und der Kundenbedarf Adobe zu einer Entwicklung für diese zukunftsweisende Open−Source Plattform drängen wird. Der Erfolg und die Zukunft von SVG wird von der breiten Verfügbarkeit der Werkzeuge und Viewer auf allen wichtigen Plattformen abhängen, wie auch vom hochqualtiativen Support der SVG Developer Community, der derzeit sicherlich gegeben ist.

Zwei W3C Arbeitsgruppen sind derzeit in die Weiterentwicklung von SVG involviert:

- **SVG Mobile Working Group, SVG 1.1.**

Diese neu etablierte Arbeitsgruppe diskutiert die Anforderungen für SVG auf mobilen Geräten, wie Handys und PDA’s. Die Gruppe schlägt vor, zwei separate Profile einzuführen, die ein Subset der derzeitigen SVG 1.0 Spezifikation darstellen: SVGB (SVG Basic) für stärkere PDA’s und Mini−Computer und SVGT (SVG Tiny) für »low−level« Geräte mit noch geringerer Rechenleistung, Hauptspeicher und Netzwerkbandbreite. Der derzeit diskutierte Vorschlag (requirement analysis) mit einer Auswahl der Features, Elemente und Attribute für SVG Mobile kann unter [5, GRAHAM/CAPIN, 2001] nachgelesen werden. Die Gruppe sieht für SVG Mobile die folgenden Anwendungsbereiche:

- Location−Based Services
- Kartographie und Verortung
- Versand animierter Botschaften und Grüsse
- Multimedia Nachrichten
- Unterhaltung
- Industrielle Anwendungen mit Bedarf für mobile Geräte

- eCommerce
- Benutzerschnittstellen

- **SVG 1.1/2.0 Working Group**

Obwohl die SVG 1.0 Spezifikation erst kürzlich eine W3C−Empfehlung wurde, diskutiert eine weitere neue Arbeitsgruppe bereits mögliche Erweiterungen für das nächste kleinere Release (SVG 1.1) und den grösseren Upgrade der Spezifikation hin zur Version 2.0 (siehe [6, JACKSON, 2001]). Für SVG 1.1 werden im wesentlichen die Bedürfnisse von SVG für mobile Geräte eingearbeitet. Es wird die modulare Struktur der Profile eingeführt, ein Weg der bereits auch bei der X3D Spezifikation begangen wurde. Zusätzlich zum SVGT und SVGB Profil wird wahrscheinlich auch ein SVG Printing Profil ausgearbeitet. SVG 2.0 wird ein grösseres Upgrade. Da die Resultate der W3C Arbeitsgruppen (Bedarfsanalyse, Entwurf und Spezifikation) öffentlich diskutiert werden, wird das Feedback von der Öffentlichkeit wie auch von eingeladenen Spezialisten, wie auch Mitgliedern der W3C Arbeitsgruppen gerne entgegengenommen und nach Möglichkeit eingearbeitet.

Unter den Zielen der Arbeitsgruppe ist auch die Plazierung von SVG als Standard−Applikation auf den Desktops der Benutzer (innerhalb von Web−Browsern, Graphik−Applikationen, Autorensysteme und auch als Austauschfomat), mobilen Geräten (in deren Browsern, als User−Interface und auch in der Autonavigation), in Druckern und industriellen Geräten mit graphischen Anwendungen. Obwohl dieses Ziel ehrgeizig klingen mag, bietet es eine grosse Chance für die Software−Industrie: sämtliche graphikorientierten Applikationen können erstmals auf einem durchdachten Standard aufbauen, der bereits die meisten Anforderungen an strukturierte interaktive Graphik erfüllt, bei Bedarf aber sehr leicht erweiterbar ist. Die Arbeitsgruppe will SVG vor allem aber auch als Austauchformat für die verschiedensten vektorbasierten Graphik−Applikationen etablieren. Das W3C wird zwar die Aufsplitterung in verschiedene Profile wegen der unterschiedlichsten Bedürfnisse der Ausgabegeräte zulassen, will aber für sämtliche Profile umfassende Testsuiten bereitstellen um zukünftige SVG−Software auf deren Kompatibilität und Standard−Konformität zu testen.

Es ist wohl wenig sinnvoll an dieser Stelle sämtliche Vorschläge im Detail zu diskutieren. Die Verbesserungen/Erweiterungen sollen v.a. verschiedene Anwendungsbereiche aus dem Kartographie, GIS, CAD und Design−Umfeld betreffen. Sie können die Bedarfsanalyse unter [6, JACKSON, 2001] studieren und selber sinnvolle Vorschläge der W3C−Arbeitsgruppe übermitteln. Wir denken, dass die Chance, an der Verbesserung offener Standards mitzuwirken von der interessierten Öffentlichkeit auch wahrgenommen werden sollte!

# 14. SVG Ressourcen

**SVG Spezifikation:**
http://www.w3.org/TR/SVG/
http://www.w3.org/TR/SVGMobileReqs (Working Draft!)
http://www.w3.org/TR/SVG2Reqs (Working Draft!)

**Ausgewählte SVG News−, Tutorial−, Links− und Demo−Seiten:**
http://www.w3.org/Graphics/SVG/Overview.htm8 (News und Links)
http://www.adobe.com/svg/community/external.html (Links)
http://www.adobe.com/svg/demos/main.html (Dynamische SVG Beispiele)
http://www.adobe.com/svg/basics/intro.html (SVG Entwickler tutorial)
http://www.kevlindev.com/ (SVG tutorial und Beispiele, SVG und GUI)
http://www.carto.net/papers/svg/ (SVG tutorial, Beispiele, Artikel)
http://www.pinkjuice.com/SVG/SVGlinks.htm (SVG links)

http://www.sun.com/software/xml/developers/svg/ (SVG bei Sun)
http://www.svgopen.org/ (Erste internationale SVG Entwickler−Konferenz, Zürich 2002)

**Ausgewählte SVG Implementationen:**
http://www.adobe.com/svg/ (Heimat des Adobe SVG Viewer)
http://xml.apache.org/batik/ (Heimat des Batik, Java2 basierten SVG Viewer)
http://www.alphaworks.ibm.com/tech/svgview (IBM SVG Viewer)
http://sis.cmis.csiro.au/svg/ (CSIRO, SVG toolkit)
http://www.croczilla.com/svg/index.html (Mozilla SVG)
http://www.in−gmbh.de/en/Products/in−gmbh/sphinxopen/editor.htm (SVG Editor)
http://www.jasc.com/products/webdraw/ (Kräftiges SVG Autorensystem)
http://broadway.cs.nott.ac.uk/projects/SVG/svgpl/ (SVG Perl Module)
http://sketch.sourceforge.net/ (Veltorgraphik−Software für Linux, Open Source, SVG exp.)
http://koffice.kde.org/kontour/ (Vektorgraphiksoftware f. Linux, Open Source, SVG im/exp.)
http://www.svgmaker.com/ (SVG Druckertreiber, ähnlich Acrobat Distiller)
http://www.mayura.com/ (Graphik Software mit SVG Export)

**SVG Mailinglists und Discussions−Gruppen:**
http://groups.yahoo.com/group/svg−developers
http://www.carto.net/papers/svg/mail_e.html (Kartographie spezifisch)
http://www.adobe.com/support/forums/main.html (siehe SVG Forum)
http://lists.w3.org/Archives/Public/www−svg/

# 15. Literatur


[1] ADOBE Systems (2001): »Adobe SVG DRAW«, http://www.adobe.com/svg/demos/main.html

[2] APACHE.ORG (2001): »Batik SVG toolkit«, http://xml.apache.org/batik/index.html

[3] CRUIKSHANK David et.al. (2001) »WebCGM Profile«, http://www.w3.org/TR/REC−WebCGM/

[4] FERRAIOLO Jon, et.al. (2001): »Scalable Vector Graphics (SVG) 1.0 Specification«, http://www.w3.org/TR/SVG/

[5] GRAHAM Rick and Tolga CAPIN (2001): »SVG Mobile Requirements«, W3C Working Draft 3 August 2001, http://www.w3.org/TR/SVGMobileReqs

[6] JACKSON, Dean (2001): »SVG 1.1/2.0 Requirements«, W3C Working Draft 3 August 2001, http://www.w3.org/TR/SVG2Reqs

[7] LILLEY, Chris (2001): »SVG implementations«, http://www.w3.org/Graphics/SVG/SVG−Implementations.htm8

[8] LINDSEY, Kevin (2001): »KevLinDev − resources for SVG, Javascript, Perl, PHP, C«, http://www.kevlindev.com/

[9] NEUMANN, Andreas (2001): »Example for Animation along a path«, http://www.carto.net/papers/svg/path_animation_e.html

[10] NEUMANN, Andreas (2001): »Comparing .SVG and .SWF file−format«, http://www.carto.net/papers/svg/comparison_flash_svg.html

[11] NEUMANN, Andreas and Andréas M. WINTER (2001): »SVG webmapping examples«, http://www.carto.net/papers/svg/first_e.html

[12] OPENSWF.ORG (2001): »The Source for Flash File Format Information«,

http://www.openswf.org/spec.html

[13] WINTER, Andréas M. (2000): »OECD Atlas Europa«, http://www.carto.net/papers/svg/eu/oecdatlas.html

[14] WINTER, Andréas M. (2001): »SVG basic shapes«, http://www.carto.net/papers/svg/shapes_e.html

**Über die Autoren:**

Andreas Neumann
Institut für Kartographie der ETH Zürich
ETH Hoenggerberg
CH−8093 Zürich
Email: neumann@karto.baug.ethz.ch
Web: http://www.karto.ethz.ch/neumann/

Andréas M. Winter
Freytag & Berndt
Brunnerstrasse 69
A−1231 Wien
Email: andre.mw@gmx.net
Web: http://www.carto.net/

Beide Autoren studierten Geographie/Kartographie an der Universiät Wien und begannen ihre Arbeit an interaktivem Webmapping mit Hilfe von SVG in 1999/2000. Ihr gemeinsames Projekt »carto.net« (http://www.carto.net/) offeriert ein Forum für Kartographen, um ihre Arbeiten, mit Schwerpunkt auf Webmapping, präsentieren zu können. Carto.net bietet Tutorials, eine Diskussions−Mailingliste, dokumentierte Beispiele und Implementationen, sowie Artikel, um den Kartographen und anderen interessierten Graphikern den Einstieg in SVG zu erleichtern. Die Beispiele zeigen, zusammen mit vielen anderen interaktiven SVG−Seiten in aller Welt, das enorme Potential von SVG für interaktive Webgraphik und Web−Kartographie Zwecke. Andreas Neumann startet gerade seine Arbeiten für die Dissertation am kartographischen Institut der ETH Zürich als Teil des ETH−Projektes »Distributed Mapping on Demand«, während Andréas M. Winter bei einer österreichischen Kartographie−Firma mit Sitz in Wien arbeitet.