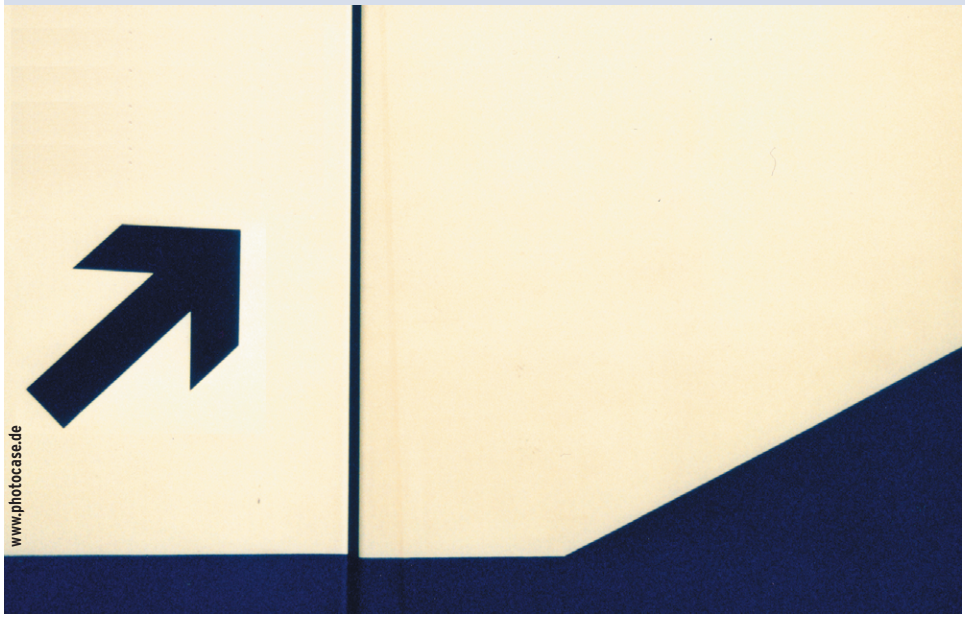
Das Grafikformat SVG - Struktur und Praxis

Mit Vektoren zeichnen

Bilder im Format Scalable Vector Graphics (SVG) brauchen weniger Speicherplatz und lassen sich ohne Qualitätsverlust skalieren. Selbst Animationen und Programmcode mit Javascript sind möglich, was den Standard fürs Web und den Desktop interessant macht. Stephan Siemen



Im grafischen Web dominieren die Rasterformate Gif, Jpeg und PNG. Mit dem SVG-Format (Scalable Vector Graphics) sollen Vektorgrafiken auch dort Einzug halten. Das Vektorformat basiert auf XML, einer Beschreibungssprache, die auch andere Webformate erfolgreich nutzen. Der Standard wird vom World Wide Web Consortium (W3C) [1] verwaltet und ist zurzeit bei Version 1.1 angelangt. Version 1.2 liegt als Draft vor.

Vektor- oder Rastergrafik

Die kleinste Einheit von Rastergrafiken ist das so genannte Pixel (Picture Element, Bildpunkt). Ein Bild von 400 mal 300 Punkten besteht im Normalfall aus 120000 solcher Pixel, abgesehen von speziellen Methoden zur Kompression wie Run-Length-Encoding. Die Dateigröße ist im Wesentlichen unabhängig vom Bildinhalt. Diese Art der Bildbeschreibung ist nützlich für fotorealistische Darstellungen, aber weniger geeignet für einfache Grafiken, die nur geometrische Objekte enthalten (Kreise, Dreiecke und so weiter).

Solche Grafiken lassen sich platzsparender speichern, wenn man nur die Koordinaten jener Punkte erfasst, die einzelne Geometrien beschreiben. Dadurch werden Bilddateien kleiner und sind im Internet schneller zu übertragen. Das Anwendungsprogramm muss das Bild für die Anzeige erst berechnen (rastern), da nur eine Beschreibung vorliegt. Mit aktuellen Computern ist dies aber kein Problem mehr.

Das Abspeichern der Kurvenzüge einer Grafik bringt einen weiteren Vorteil: Bei einem Vektorformat kann der Benutzer einzelne Objekte direkt bearbeiten. Mit Hilfe einer Koordinate, die dem Betrachter zum Beispiel durch eine Maus mitgeteilt wird, lässt sich berechnen, welches Objekt an diesem Punkt liegt. Soll ein Objekt verschoben werden, sind nur seine Koordinaten zu transformieren. Einzelne Elemente einer Rastergrafik verändern ist viel aufwändiger.

Im Bereich des Computer Aided Design (CAD) werden aus diesem Grund fast nur Vektorformate genutzt. Wenn der Benutzer in eine Bitmap-Datei zoomt, vergrößert das entsprechende Programm nur die Pixel, mehr Details gibt es nicht zu sehen. Vektorgrafiken bleiben dagegen beim Vergrößern scharf, da Linienzüge für jede Zoomstufe neu gerastert werden, siehe Abbildung 1. Daher die Bezeichnung Scalable (skalierbare) Vector Graphics.

Erste Schritte

Eine SVG-Datei hat die Endung ».svg« oder, wenn sie mit Gzip gepackt ist, ».svgz«. Die meisten SVG-Browser unterstützen solche gepackten Dateien. Das Koordinatensystem hat seinen Nullpunkt (Ursprung) in der oberen linken Ecke der Zeichenfläche, die x-Achse verläuft horizontal (rechts positiv), die y-Achse von oben nach unten. Listing 1 zeigt als einfaches Beispiel ein SVG-Dokument.

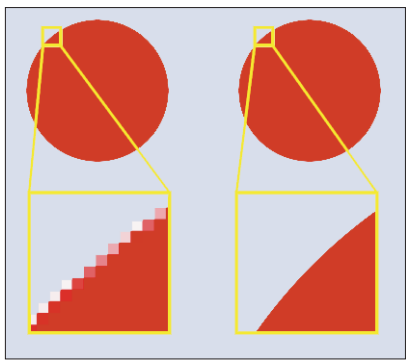


Abbildung 1: Vergrößert man den gefüllten Kreis im Pixelformat, erscheinen die Bildpunkte größer (links). Bei der Vektor-Variante bleibt der Rand scharf.

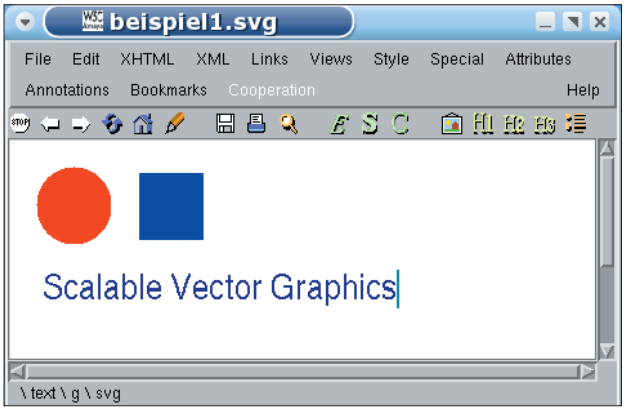


Abbildung 2: Der W3C-Webeditor Amaya zeigt das SVG-Dokument aus Listing 1.

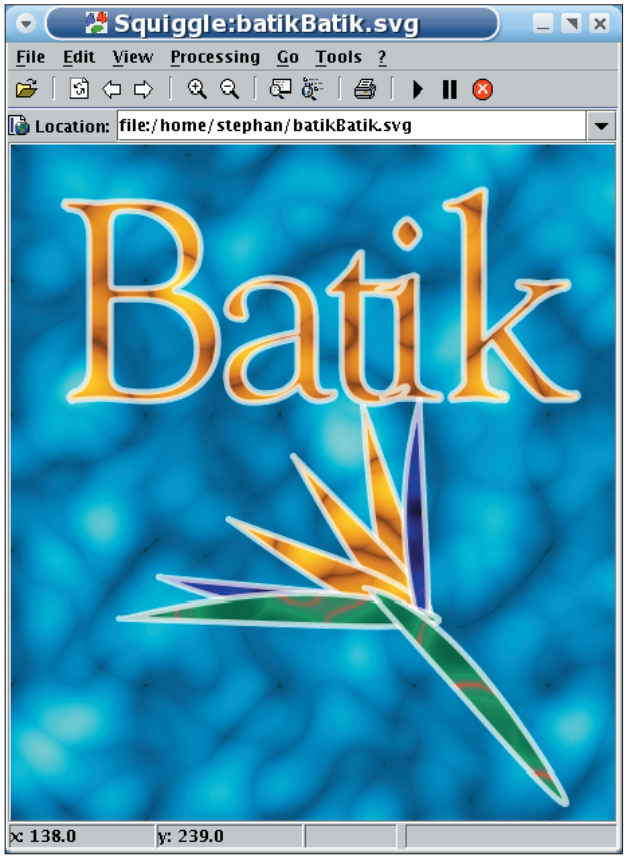


Abbildung 3: Ein SVG-Beispiel des Batik-Projekts, angezeigt in Batiks Squiggle.

Wer sich schon einmal mit einem XML-Format beschäftigt hat, wird sich gleich zurechtfinden.

Ein SVG-Dokument ist immer im XML-Format abgefasst, wobei die erste Zeile XML-Version und -Zeichensatz angibt. Die zweite Zeile spezifiziert mit dem »!DOCTYPE«-Tag den Dokumententyp SVG. Dann folgt der so genannte Public Identifier »PUBLIC „-//W3C//DTD SVG 1.0/EN“«. Der System-Identifier in Zeile 3 ist der Name einer SVG-DTD-Datei, die zulässige Elemente und Attribute bestimmt. Die DTD wird übrigens vom jeweiligen SVG-Browser bereitgestellt und normalerweise nicht von dieser Internetadresse geladen.

Struktur durch XML

Nach einem XML-Kommentar beginnt in Zeile 4 von Listing 1 die Beschreibung der Grafik mit einem »svg«-Tag. Mögliche Attribute sind die Abmessungen und die Betrachtungsfläche (Viewport). Die Tags »title« und »desc« geben der Grafik einen Titel und eine Beschreibung, die ein SVG-Browser anzeigt, siehe Tabelle 1. Der KDE-Browser Konqueror stellt zum Beispiel den Titel in der Titelzeile dar und die Beschreibung in der Statuszeile.

Ein SVG-Dokument besteht aus mehreren Objekten, die jeweils durch zwei Tags eingeschlossen sind. Die Objekte repräsentieren sichtbare (»circle«) oder unsichtbare (»path«) Teile einer Grafik. Die Darstellung hängt von den Objektattributen ab: In Listing 1 besitzen zum Beispiel die Objekte »circle« (Kreis), »rect« (Rechteck) und »text« unterschiedliche Attribute, um Position (x, y), Größe, Farbe und Schriftart festzulegen. Eine Liste aller SVG-Objekte und ihrer Attribute findet sich unter [1].

Die meisten SVG-Primitive besitzen das Farbenattribut »color«, das auf vier verschiedene Arten belegt werden kann. Drei davon nutzen das RGB-Farbsystem (Rot, Grün, Blau), in dem je ein Wert für die Intensität einer Farbkomponente steht: von 0 bis 255, Prozentwerte oder Hexadezimalzahlen von 00 bis FF. Für einige vordefinierte Farben kann auch einfach nur deren (englischer) Name angegeben werden. So lässt sich zum Bei-

Tabelle 1: Wichtige SVG-Tags

Tag	Bedeutung
<svg>	Umschließt ein SVG-Dokument
<g>	Gruppiert Elemente
<a>	Umschließt einen Link (ähnlich wie in HTML)
<text>	Definiert ein Textelement
<line>	Beschreibt eine Linie
<rect>	Beschreibt ein Rechteck
<path>	Erzeugt einen Pfad (sichtbar oder unsichtbar)
<use>	Benutzt ein schon definiertes Element nochmals

Listing 1: Einfaches SVG-Dokument

```
01 <?xml version="1.0" encoding="iso-8859-1"?>
02 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
03           "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd" >
04 <!-- Ein Kommentar -->
05 <svg width="10cm" height="10cm" viewBox="0 0 250 150" >
06   <title>Mein erstes SVG Dokument</title>
07   <desc>Eine einfache SVG Grafik, die als Vorlage dienen kann.</desc>
08   <g>
09     <circle cx="50" cy="50" r="30" style="fill:rgb(255,0,0)"/>
10     <rect x="100" y="25" width="50" height="50" style="fill:blue"/>
11     <text x="25" y="120" style="font-family:Arial; font-size:20; fill:darkblue">
12       Scalable Vector Graphics
13     </text>
14   </g>
15 </svg>
```

spiel die rote Füllung des Kreises in Listing 1 auf unterschiedliche Arten beschreiben:

```
fill="rgb(255,0,0)"
fill="rgb(100%,0%,0%)"
fill="#FF0000"
fill="red"
```

Ansichtssache

Bis vor kurzem unterstützte Linux SVG-Grafiken mehr schlecht als recht. Der SVG-Viewer von Adobe [2], der unter Windows das Standard-SVG-Plugin ist, existiert für Linux immer noch als Beta einer Uraltversion. Seit letztem Jahr gibt es aber reichlich Alternativen. Amaya ist W3Cs eigene Implementierung eines Browsers und Webeditors [3]. Er ist als Beispielanwendung für alle W3C-Standards gedacht. Mit Amaya lassen sich daher nicht nur HTML und SVG, sondern auch MathML, XHTML und andere Dokumenttypen darstellen und bearbeiten. Amaya ist zwar nicht sehr benutzerfreundlich, aber für viele Plattformen (natürlich auch Linux) verfügbar. In Abbildung 2 zeigt Amaya die SVG-Datei aus Listing 1.

Schon frühzeitig hat das Apache-Webserver-Projekt SVG ernst genommen. Deshalb startete es das Projekt Batik, das ein Java-Toolkit zur Bearbeitung von SVG-Dokumenten ist und eine der besten Implementierungen des Standards bietet. Auf der Webseite des Projekts [4] findet sich ein Zip-Archiv des Toolkits, das mindestens Java 1.3 erfordert.

Programme des Batik-Projekts sind in Jar-Dateien (Java-Archive) aufgeteilt. Unter anderem bringt es den SVG-Betrachter Squiggle mit, den man so startet (siehe Abbildung 3):

```
java -jar batik-squiggle.jar
```

Ein weiteres interessantes Programm ist der Rasterizer. Er wandelt SVG-Grafiken in Rasterformate wie Jpeg und PNG um. Dieser Aufruf erzeugt eine PNG-Datei namens »simple.png«:

```
java -jar batik-rasterizer.jar
  -m image/png simple.svg
```

Hinter dem Schalter »-m« folgt der Mime-Typ, der noch das Ausgabeformat festlegt.

Seit einiger Zeit arbeiten Mozilla-Entwickler an direkter SVG-Unterstützung, also ohne auf zusätzliche Plugins zurückzugreifen. Leider ist das Projekt noch in der Betaphase. Der Code ist zwar schon Teil des Mozilla-Projekts, aber noch nicht standardmäßig aktiviert. Auf der Webseite des Projekts [5] ist eine Version mit aktivierter SVG-Unterstützung zu finden.

SVG auf dem Desktop

Der wirkliche Durchbruch von SVG unter Linux kam erst mit KDE und Gnome. Seit Version 3.1 beherrscht KDE das SVG-Format, zunächst nur, um entsprechende Icons anzuzeigen. Seit Version 3.2 können Konqueror und andere KDE-Programmen SVG-Grafiken auch direkt darstellen. Möglich machte dies das KSVG-Projekt [6], das sich in KDE um das SVG-Rendering kümmert.

KDE 3.2 unterstützt auch einfache Skripte durch SVG-DOM (siehe Kasten „SVG und DOM“). Die Implementierung ist noch nicht vollständig, funktioniert jedoch schon ganz gut.

Auch die Gnome-Leute waren nicht untätig. Die Entwicklung für volle SVG-Unterstützung geht voran und ist mit Gnome 2.8 geplant. Die Librsvg [7] ist eine der führenden Implementierungen des SVG-Standards. Die Bibliothek hat die Version 2.7 erreicht und enthält versuchsweise den eigenen Viewer namens »rsvg-view« sowie ein Plugin für Mozilla. Die Zeichenprogramme Sodipodi [8] und Incscape [9] nutzen SVG sogar als bevorzugtes Speicherformat, siehe Abbildung 4. Beide Projekte verwenden die Librsvg.

Software-Unterstützung

SVG-Grafiken lassen sich sogar, wie in Listing 1 zu sehen ist, in einem einfachen Texteditor schreiben. Auch Skriptsprachen eignen sich gut dafür. So gibt es beispielsweise für Perl ein SVG-Modul, das die Aufgabe noch einmal vereinfacht. Grafikprogramme wie Open Office Draw und Gimp unterstützen mittlerweile ebenfalls das SVG-Format. Auch das unter Unix beliebte Grafikpaket Image Magick [10], das unter anderem die Programme »display« und »convert« enthält, hat seine SVG-Unterstützung wesentlich verbessert.

XML + XSLT = SVG

Immer mehr Datensätze werden in XML-Formaten gespeichert, die Daten besser strukturieren. Mit XSLT (extensible Stylesheet Language Transformations)

Listing 2: XML-Beispieldatei

```
01 <?xml version="1.0" ?>
02 <!DOCTYPE nations SYSTEM "nations.dtd">
03 <nations>
04   <caption>
05     <heading>Population</heading>
06     <subheading>In millions, broken down by
   nation</subheading>
07   </caption>
08   <nation>
09     <name>USA</name>
10     <population>287.7</population>
11   </nation>
12   <nation>
13     <name>Indonesia</name>
14     <population>213.6</population>
15   </nation>
16   <nation>
17     <name>Japan</name>
18     <population>127.1</population>
19   </nation>
20   <nation>
21     <name>Germany</name>
22     <population>82.5</population>
23   </nation>
24   <nation>
25     <name>United Kingdom</name>
26     <population>59.9</population>
27   </nation>
28 </nations>
```

SVG und DOM

DOM (Document Object Model) bezeichnet eine Hierarchie von Objekten eines Dokuments. Darüber lassen sich Elemente einer Grafik eindeutig ansprechen. DOM ist nicht auf SVG oder XML beschränkt, sondern findet sich in vielen verwandten Formaten, zum Beispiel in HTML.

Das SVG-DOM ermöglicht es, die Grafik per Skript oder Programm zu verändern, aber auch die Reaktion auf Ereignisse wie zum Beispiel Mausklicks. Einige der Ereignisse, die unter SVG unterstützt werden, fasst die Tabelle 2 zusammen. Das DOM bietet außerdem Methoden und Objekte, die Elemente zur Laufzeit erzeugen.

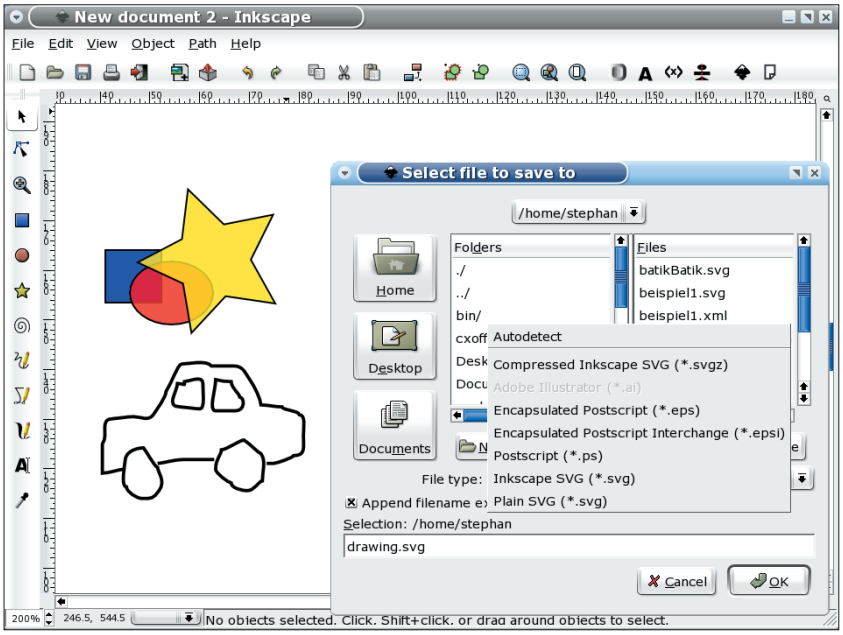


Abbildung 4: Inkscape bietet viele Funktionen, die Benutzer anderer Zeichenprogramme kennen. Neben (gezippten) SVGs speichert Inkscape auch Postscript und EPS.

lässt sich XML in andere Formate umwandeln. Damit werden auch aus XML-Daten SVG-Grafiken. Listing 2 zeigt eine XML-Datei, die Bevölkerungszahlen verschiedener Länder angibt. Über eine XSLT-Datei (»nations.xslt«, zu finden auf dem Server des Linux-Magazins [13]) wandelt ein XSLT-Prozessor die Rohdaten in ein SVG-Spaltendiagramm um. Folgender Aufruf benutzt dafür das Programm Saxon [11]:

```
saxon nations.xml nations_populations.xslt ➥
  > populations.svg
```

Die erzeugte Grafik in Konqueror zeigt Abbildung 5. Unter [12] ist ein weiteres Beispiel mit SVG und XSLT mehr im Detail beschrieben.

Eine SVG-Grafik kann einen oder mehrere Pfade enthalten, die nicht direkt sichtbar sind. Sie legen stattdessen den Weg anderer Objekte wie Texte oder Animationen fest.

Zeichenpfade

Ein Pfad wird durch das »path«-Tag definiert, mit den Wegkoordinaten im Attribut »d«. Die Koordinatenpaare beginnen jeweils mit einem Buchstaben. Ein M steht für Move und verschiebt den aktuellen Punkt, ohne dass dieser Weg Teil des Pfades wird. Ein L fügt in absoluten Koordinaten einen unsichtbaren Linienzug vom letzten gesetzten Punkt zum Pfad hinzu. Der Buchstabe l macht dasselbe mit relativen Koordinaten. Das folgende Element legt einen Pfad vom Ursprung zum Punkt (100, 100) und dann zu (105, 200) an:

```
<path d="M0 0 L100 100 l5,100" />
```

Interaktionen verlangen, dass einzelne Elemente der Grafik ansprechbar sind. Deshalb braucht jedes Element einen eindeutige Namen, den das Attribut »id« spezifiziert. Auch unsichtbare Grafikelemente wie Gruppen oder Animationen können Namen tragen.

Das ist recht praktisch für Grafiken, in denen ein Element mehrfach erscheinen soll – die SVG-Dateien werden kleiner. Dazu definiert man in einem »defs«-Abschnitt einen Pfad und vergibt einen Namen. Mit »defs« definierte Tags werden erst bei einem späteren Aufruf durch »use« oder »xlink:href« gezeichnet, nicht schon bei der Definition:

```
<use xlink:href="#TextPath" fill="none"➥
  stroke="blue" />
```

Das »use«-Tag erlaubt es, mit zusätzlichen Attributen das Aussehen des Pfads zu ändern oder ihn zum Beispiel mit »transform« zu verschieben. Als Attribut eines »textPath«-Tags steuert der Pfad auch das Textlayout:

```
<textPath xlink:href="#textPath">➥
  Entlang des Pfads geschrieben</textPath>
```

Bewegte Bilder

SVG ist nicht nur ein einfaches Grafikformat, es bietet auch verschiedene Wege an, um Bilder interaktiv zu gestalten. Neben dem »a«-Tag für normale Links unterstützt SVG einfache Anima-

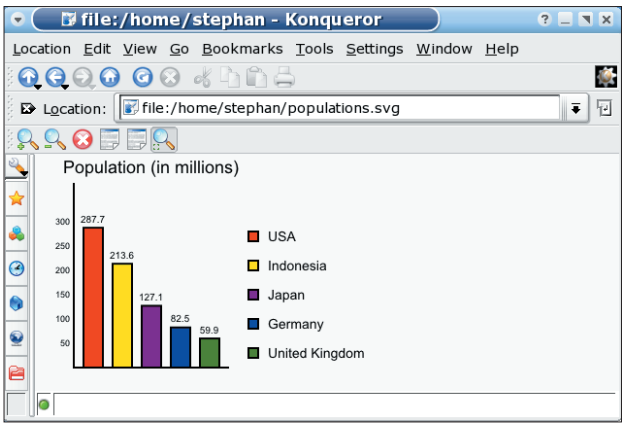


Abbildung 5: Konqueror zeigt das Ergebnis der Umwandlung der XML-Daten aus Listing 2 über XSLT ins SVG-Format.

Abbildung 6: Listing 3 im Konqueror. Ein gehaltener Mausklick blendet das linke Quadrat aus. Lässt man die Taste wieder los, verschwindet auch das rechte.

tionen mit »animate« und verwandten Tags. Das Beispiel spezifiziert einen Animationspfad für ein Rechteck:

```
<rect ...>
  <animateMotion path="M0 0 Q10 10 20 20" />
</rect>
```

Richtig mächtig wird SVG aber erst durch seine Skript-Unterstützung. Mit DOM (siehe Kasten „SVG und DOM“) lassen sich Skripte schreiben, die auf Interaktion des Anwenders reagieren.

Programmieren mit ECMA-Script

In SVG-Grafiken integrierte Skripte sind in ECMA-Script abgefasst, der standardisierten Variante von Javascript. Der Code steht innerhalb von »script«-Tags.

```
<script type="text/ecmascript"> <![CDATA[
   ... Skript-Code ...
]]> </script>
```

Zu beachten ist dabei die Kennzeichnung »text/ecmascript«, die die Art des Skripts angibt.

Listing 3 zeigt eine SVG-Grafik, die auf zwei Mausereignisse reagiert: gedrückte (»onmousedown«) und losgelassene (»onmouseup«) Maustaste. Sobald eines der beiden Ereignisse über dem grünen Kreis (»circle«, Zeile 25) eintritt, wird die Funktion »onEvent()« ausgeführt, die zwei Parameter besitzt (Zeile 10). Für die Ausführung sorgt die Gruppierung mit dem »g«-Tag, das in seinen Attributen die Event-Handler festlegt, siehe Zeilen 23 und 24. Die Variable »evt« steht für das ausgelöste Ereignis. Sie wird benötigt, um ein Objekt des gesamten SVG-Dokuments zu erhalten:

```
var doc = evt.getTarget().
  getOwnerDocument();
```

Das zweite Argument ist die ID des zu verändernden Objekts: Abhängig vom Ereignis wird das linke oder rechte Rechteck (»rect«, Zeilen 27 und 28) sichtbar (»visible«) oder unsichtbar (»hidden«). Zum Auslesen von Attributen eines DOM-Objekts steht die Methode »getAttribute()« bereit (Zeile 14), zum Schreiben das Gegenstück »setAttribute()« (Zeilen 16 und 18).

Tabelle 2: SVG-Ereignisse

Ereignis	Ursache
onclick	Mausklick
onmousedown	Maustaste gedrückt
onmouseup	Maustaste losgelassen
onfocusin	Element wird selektiert
onfocusout	Element wird deselektiert
onmouseover	Zeiger bewegt sich über dem Element

Die Zukunft von SVG

SVG bietet sich für automatisch erzeugte Webgrafiken an, ist aber auch für Desktop-Anwendungen interessant. Linux-Desktops wie KDE und Gnome sind dabei führend. Doch SVG ist nicht länger nur auf PCs zu finden. Nach den Firmen Sony, Ericsson und Siemens hat auch Nokia im Juni erklärt, dass ihre Mobiltelefone in Zukunft den SVG-Standard unterstützen werden. Das W3C setzte von Anfang an auf diesen Markt und hat mit SVG 1.1 zwei neue Varianten des Standards veröffentlicht. SVG-Tiny ist eine reduzierte Version für Mobiltelefone, während SVG-Basic auf PDAs zugeschnitten ist.

Wer sich nach diesem Artikel näher mit SVG auseinander setzen will, findet ein reiches Anwendungsfeld mit Zukunftspotenzial – auch außerhalb der Linux-Welt. *(ofr/jcb)* ■

Infos

[1] SVG-Spezifikation: [http://www.w3.org/graphics/svg/]
[2] Adobes SVG-Viewer: [http://www.adobe.com/svg/]
[3] W3Cs Amaya: [http://www.w3.org]
[4] Batik XML-Viewer: [http://xml.apache.org/batik/]
[5] Mozilla SVG-Project: [http://www.mozilla.org/projects/svg/]
[6] KSVG: [http://www.kde.org/ksvg/]
[7] Librsvg: [http://librsvg.sourceforge.net]
[8] Sodipodi: [http://www.sodipodi.com]
[9] Inkscape: [http://www.inkscape.org]
[10] Image Magick: [http://www.imagemagick.org]
[11] Saxon: [http://saxon.sourceforge.net]
[12] XSL-SVG-Transformationen: [http://www-106.ibm.com/developerworks/education/transforming-xml/xmltosvg/]
[13] Listings zum Artikel: [http://www.linux-magazin.de/Service/Listings/2004/11/SVG]

Der Autor

Stephan Siemen beschäftigt sich seit einigen Jahren mit dem Programmieren von wissenschaftlichen Grafiken in 2D und 3D unter Linux.

Listing 3: DOM und Scripting

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1 Basic//EN"
   "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-basic.dtd">
03
04 <svg version="1.1" baseProfile="basic" xmlns=
   "http://www.w3.org/2000/svg"
05     xmlns:xlink="http://www.w3.org/1999/xlink" id="svg-root"
06     width="100%" height="100%" viewBox="0 0 400 400">
07
08   <title>Scripting Beispiel</title>
09   <script type="text/ecmascript"><![CDATA[
10     function onEvent(evt, indicatorId){
11       var doc = evt.getTarget().getOwnerDocument();
12
13       var element = doc.getElementById(indicatorId);
14       var sichtbar  = element.getAttribute("visibility");
15       if(sichtbar == 'hidden')
16         element.setAttribute('visibility','visible');
17       else
18         element.setAttribute('visibility','hidden');
19     }
20   ]]></script>
21
22   <text x="100" y="100" font-size="24">Klick den Kreis ;-)</text>
23   <g onmousedown="onEvent(evt, 'links')"
24      onmouseup="onEvent(evt, 'rechts')">
25     <circle cx="50" cy="100" r="40" fill="green" stroke="black" />
26   </g>
27   <rect id="links"   x="10" y="150" width="30" height="30"
   fill="blue" />
28   <rect id="rechts" x="60" y="150" width="30" height="30" fill="red" />
29 </svg>
```