Verwendung von Scalable Vector Graphics und MathML in web-basierten Lernumgebungen

Martin Rotard, Waltraud Schweikhardt, Thomas Ertl
Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart

Zusammenfassung

In web-basierten Lernumgebungen werden Rastergraphiken eingesetzt, um Skizzen und Formeln darzustellen. Dies erschwert den Austausch und die Wiederverwendung von Lehrmaterialien unter Lernenden und Lehrenden. Dieser Beitrag stellt das Vektorgraphikformat SVG (Scalable Vector Graphics) und MathML, das Format zur Beschreibung mathematischer Ausdrücke, vor. Die Möglichkeiten des SVG-Formats reichen von der Darstellung einfacher Graphikelemente bis hin zu komplexen Filtern und Beleuchtungseffekten. In MathML lässt sich der inhaltliche Zusammenhang oder die Darstellung von mathematischen Ausdrücken beschreiben. Das letzte Kapitel gibt einen Ausblick auf Anwendungsfelder von web-basierten Lernumgebungen mit SVG und MathML.

1 Einleitung

In virtuellen Universitäten und virtuellen Unternehmen sind Lehrende und Lernende räumlich getrennt. Das Kommunikationsmedium Internet und speziell das Web spielen dabei eine zentrale Rolle. Der Lehrstoff ist über viele Dokumente verteilt und kann von mehreren Lehrenden entwickelt werden. Dieses Lernszenario dient dazu, eine Gruppe von verteilten Lernenden, Lehrenden und Experten über weite Distanzen so zusammenzuschalten, dass sie ihr Wissen problemlos austauschen können [Schulmeister 2001]. „Wissen“ wird dabei im Allgemeinen repräsentiert durch Texte, Graphiken, mathematische Ausdrücke, Videos, etc.

Der Austausch und die Verwendung der nativen Quelldateien der Dokumentenwerkzeuge ist dadurch erschwert, dass sich die Formate meistens nicht direkt im Web darstellen lassen. Hier einige Beispiele zu Text und Graphikformaten:

- Dokumentenformate von Adobe Framemaker (FM), Microsoft Word (DOC), etc.
- Graphikformate wie das Tagged Image File Format (TIFF), Windows Meta File (WMF) etc.

Mathematische Ausdrücke werden im Web heute fast ausschließlich als Rastergraphik dargestellt und damit nicht im ursprünglichen nativen Quellformat jeweiligen Werkzeuges.

Speziell bei Graphiken und mathematischen Ausdrücken ist der Austausch der Quellen und die gleichzeitige Verwendung im Web von Nachteil. Heutige Autorenwerkzeuge verfügen über einen HTML-Export, der alle im Dokument enthaltenen Graphiken und Formeln als Rastergraphiken

speichert. Die darin enthaltenen Informationen können nur durch Neuerstellung oder Manipulation auf Pixelebene verändert werden. Um Bandbreite zu sparen wird zusätzlich die Farbanzahl reduziert (GIF) oder die Datenmenge durch verlustbehaftete Quantisierung verringert (JPEG). Dies mindert die Qualitiät für die Wiederverwendung zusätzlich.

Im folgenden Kapitel wird das Vektorgraphikformat SVG vorgestellt. Dort lassen sich Graphikelemente explizit speichern. Anschließend wird ein Überblick über MathML gegeben. Mit diesem Format lassen sich mathematische Ausdrücke einfach beschreiben. SVG und MathML erleichtern die Bearbeitung und fördern damit die Wiederverwendung und den Austausch von Lehrmaterialien. Das letzte Kapitel gibt einen Ausblick auf die Möglichkeiten für web-basierte Lernumgebungen mit SVG und MathML.

1 SVG

Das World Wide Web Consortium [W3C] hat im September 2001 als Empfehlung die Spezifikation des Formats *Scalable Vector Graphics* [SVG] verabschiedet. Dieses Format erlaubt die Beschreibung zweidimensionaler Vektorgraphik als Anwendung von XML (Extensible Markup Language [XML]). An der Spezifikation des Formates (Version 1.0 [SVG Spec 1.0]) waren viele bekannte Unternehmen beteiligt, unter anderem Adobe, Microsoft, Canon, Quark, Kodak, IBM, Sun Microsystems, Netscape, Macromedia, Apple, Xerox, Hewlett-Packard etc.

In den folgenden Abschnitten werden die Eigenschaften von SVG-Vektorgraphik erläutert. Anschließend werden elementare SVG-Formen für Graphik und Text vorgestellt. Der nächste Abschnitt geht auf erweiterte Konzepte wie Animationen, Filter und Effekte ein. Daraufhin werden Hinweise zu Werkzeugen für die Erzeugung und Darstellung des SVG-Formats gegeben. Anschließend werden noch die Vorteile des SVG-Formats für sensorisch Behinderte vorgestellt.

1.1 Eigenschaften von SVG

SVG-Vektorgraphik hat gegenüber Rastergraphik viele Vorteile. Die verwendeten Graphikelemente werden als Objekte abgespeichert und können nachträglich wieder verändert werden. Die Graphiken sind beliebig vergrößerbar, ohne dass Rasterstrukturen sichtbar werden. Die Dateien sind in der Regel kleiner als die von Rastergraphiken, da mit wenigen Tags schon komplexe Graphiken entstehen können und sich diese "Textdateien" zusätzlich komprimieren lassen (im bereits vorgesehenen SVG-Format SVGZ). Herkömmliche Rastergraphikformate wie PNG, JPEG und GIF können in SVG eingebettet werden. In SVG-Dateien kann auf einfache Weise Text gesucht und extrahiert werden. SVG-Graphiken sind mit anderen XML-Anwendungen konform. Deshalb ist die Einflussnahme von Aktionen innerhalb der SVG-Graphik auf andere Elemente der Seite und umgekehrt möglich. Beispielsweise könnte die Interaktion des Benutzers mit einer zweidimensionalen SVG-Graphik die Darstellung einer dreidimensionalen X3D-Graphik [X3D] ändern, die sich auf derselben Webseite befindet. SVG unterstützt einen standardisierten Farbraum, bei dem Profile des International Color Consortium [ICC] verwendet werden können. Mit diesem Farbmanagement ist die Farbdarstellung auf verschiedenen Ausgabegeräten gesichert.

Rastergraphikformate haben auch Vorteile gegenüber Vektorgraphikformaten. Die Zeit die benötigt wird, eine Rastergraphik anzuzeigen, ist unabhängig von deren Komplexität. Der Aufbau einer komplexen Vektorgraphik kann daher langsamer sein, als bei der entsprechenden Rastergraphik. Dieser Vorteil relativiert sich aber durch die steigende Leistung der heutigen Rechner. Ein Anwärter der W3C-Empfehlung ist der SVG-Mobile Standard [SVG Mobile]. Dieser ist für mobile Geräte mit geringerer Rechenleistung vorgesehen. Durch die geringere Komplexität des Standards ist eine schnellere Darstellung möglich.

Bereits heute hat sich das vektorgraphikbasiertes Format Flash von Macromedia [Flash] etabliert. Flash liegt im Gegensatz zu SVG in einem Binärformat vor und ist damit nicht XML-basiert. Damit ist die Generierung von Graphiken aus einer Skriptsprache sehr erschwert oder nicht möglich. Rastergraphiken werden bei Flash in das Binärformat eingebettet. Bei SVG können diese von einer beliebigen erreichbaren Stelle im Web referenziert werden oder auch in einem CDATA-Abschnitt eingebettet werden. Für die Erzeugung von Flash werden ausschließlich proprietäre Tools benötigt. Für SVG reicht als minimale Ausstattung ein Texteditor. Flash unterstützt keine ICC-Profile für das Farbmanagement. Bei Animationen ist Flash frameorientiert. Das bedeutet, der Autor legt wie bei einem Film fest, was auf jedem Bild zu sehen ist. Dies ist bei SVG auch möglich und wird erweitert um ein Animationskonzept, bei dem nur Anfangs- und Endzustand angegeben werden. Die interpolierten Zwischenbilder errechnet der SVG-Viewer automatisch. Flash kann Audio-Dateien einbetten, dies ist in SVG nicht vorgesehen. Es ist jedoch ein Verweis auf eine Audio-Datei möglich, die dann von einem anderen Player abgespielt wird. Der W3C Standard SMIL (Synchronized Multimedia [SMIL]) ermöglicht dabei eine Integration von Audio, Video, Graphiken und Text. Auch eine Konvertierung von Flash in SVG ist möglich [Flash2SVG].

1.2 Elementare SVG-Formen

Der SVG-Standard sieht einfache geometrische Formen wie Rechtecke, Ellipsen, Linien, Polygone etc. vor (siehe Abbildung 1). Dabei können die Attribute der Formen wie Füllfarbe, Deckkraft, Strichstärke, etc. in Parametern angegeben werden.

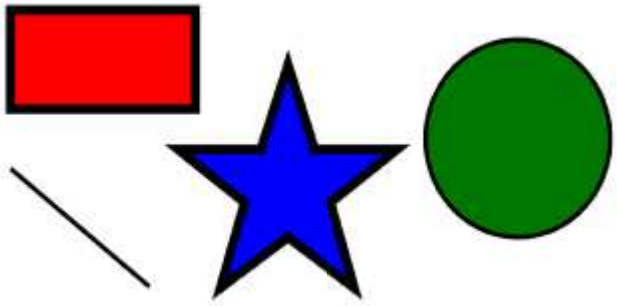

Abbildung 1: Elementare SVG-Formen

Der Quelltext dieser Graphik ist unten abgebildet und einfach zu interpretieren. Nach der xml-Auszeichnung in der ersten Zeile folgt im Tag "DOCTYPE" der Verweis auf die zugehörige Docu-

ment Type Definition (DTD) von SVG. Anschließend wird mit dem <svg>-Tag der Inhalt der SVG-Graphik eingeleitet. In diesem Tag kann als Attribut die Größe des Graphik und mittels *viewBox* die Auflösung des Koordinatensystems angegeben werden. Dabei werden die unterschiedlichsten Maß-Einheiten unterstützt, wie mm, cm, inch, punkt, pixel etc. SVG verwendet kein kartesisches Koordinatensystem, sondern der Nullpunkt befindet sich in der linken oberen Ecke der Graphik. Die positive X-Richtung geht somit nach rechts und die positive Y-Richtung nach unten.

Anschließend werden die elementaren SVG-Formen in der Graphik definiert. Beim Rechteck wird dazu der Bezugspunkt links oben und die Breite und Höhe angegeben. Der Kreis hat in der Definition einen Mittelpunkt als Bezugspunkt und einen Radius. Die Linie dagegen benötigt zwei Punkte p1 und p2 mit jeweils x- und y- Koordinaten. Beim Polygon werden die einzelnen Punkte nicht explizit, sondern in einer Punktliste als "x,y"-Koordinatenpaar angegeben.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<desc>Beispiel für elementare Formen (Rechteck, Kreis, Linie und Polygon)</desc>

<svg width="10cm" height="5cm" viewBox="0 0 800 400"
     xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">

  <!-- Rotes Rechteck -->
  <rect x="50" y="20" width="200" height="100"
        fill="red" stroke="black" stroke-width="10" />

  <!-- Grüner Kreis -->
  <circle cx="600" cy="150" r="100"
        fill="green" stroke="black" stroke-width="5" />

  <!-- Schwarze Linie -->
  <line x1="50" y1="180" x2="200" y2="300"
            stroke="black" stroke-width="5" />

  <!-- Blauer Stern -->
  <polygon fill="blue" stroke="black" stroke-width="10"
            points="350,75  379,161 469,161 397,215
                    423,301 350,250 277,301 303,215
                    231,161 321,161" />
</svg>
```

Den Formen können Attribute hinzugefügt werden. Alle Formen im Beispiel werden mit der Linienfarbe schwarz (*stroke="black"*) gezeichnet. Über das Attribut *stroke-width* kann jeder Linie eine Strichstärke zugewiesen werden. Über das Attribut *fill* werden im Beispiel allen geschlossenen Formen eine Füllfarbe zugeteilt. Wichtig für die Graphik ist die Reihenfolge, in der die Formen gezeichnet werden. SVG-Formen werden sequenziell gezeichnet, d.h. die erste Form im Quelltext wird zuerst gezeichnet und anschließend die zweite definierte Form usw. Dies hat zur Folge, dass Formen durch andere Formen, die weiter unten definiert werden, überdeckt werden können. Auch komplexere Formen können aus den elementaren Formen kombiniert werden. Dabei können in SVG Gruppen gebildet werden, denen man wiederum Attribute zuweisen kann.

Abbildung 2: Text in SVG

Text kann in SVG sehr flexibel gehandhabt werden (siehe Abbildung 2). Bei der Definition von Text in der Graphik, bleibt der Text an sich im Quelltext erhalten. Dies ist ein großer Vorteil gegenüber Rasterformaten. Damit ist es möglich, textuelle Inhalte in SVG-Graphiken mit Suchalgorithmen zu indizieren.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
  "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">

<svg width="10cm" height="5cm" viewBox="0 0 800 400"
     xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
<desc>Beispiel für Text in SVG (normal und als TextPfad)</desc>

  <defs>
    <path id="Wave"
          d="M 100,250
             C 200,100 300,0   400,100
             C 500,200 600,300 700,200
             C 800,100 900,100 900,100" />
  </defs>

  <!-- Schwarzer Text -->
  <text x="80" y="320"
        font-family="Verdana" font-size="62" fill="black" >
    Universität Stuttgart
  </text>

  <!-- Blauer Text entlang des Pfades "Wave" -->
  <text font-family="Verdana" font-size="30" fill="blue" >
    <textPath xlink:href="#Wave">
      Institut für Visualisierung und Interaktive Systeme
    </textPath>
  </text>
</svg>
```

Wie man diese Graphik erstellt, ist im obigen Quelltext dargestellt. Einfacher Text lässt sich innerhalb des <text>-Tags darstellen. Dabei können weitere Attribute wie Position, Schriftart,

Schriftgrad, Farbe etc. angegeben werden. Der zweite Text wird entlang eines Pfades gezeichnet. Dieser kann eine Kurve oder einen Linienzug definieren. Dabei lassen sich Punkte absolut und relativ zum davor definierten Punkt positionieren. SVG unterstützt kubische und quadratische Bézier-Kurven, bei denen die Stützstellen in einer Punktliste angegeben werden. Auf die genaue Notation soll an dieser Stelle nicht eingegangen werden, sondern auf die SVG-Spezifikation [SVG Spec 1.0] verwiesen werden. Im <text>-Tag wird dann innerhalb des <textpath>-Tags der Pfad für die Ausrichtung des Textes mittels eines Verweises, einem so genannten XLink [XLink], zugewiesen und der darzustellende Text angegeben. Diese wenigen Zeilen genügen, um einen geschwungenen Schriftzug zu erzeugen.

1.3 Erweiterte SVG-Konzepte

SVG bietet weit mehr Funktionalität als nur die oben vorgestellten elementaren Formen. In SVG lassen sich nicht nur statische Graphiken, sondern auch dynamische Effekte definieren. Dazu werden Animationsknoten eingefügt, die Zeit- oder Ereignis-getriggert sind. Neben den Attributen von Formen lassen sich Bewegungen, Farben und Transformationen animieren. Das in SVG integrierte Animationsmodell ist an den W3C Standard SMIL angelehnt. Um erweiterte Funktionalitäten zuzulassen, die über Animation und Eventhandling hinaus gehen, können SVG-Graphiken per Skript gesteuert werden (über [ECMAScript, JavaScript] etc.). Alle Elemente der SVG-Graphik lassen sich dann über das Document Object Model [DOM] adressieren.

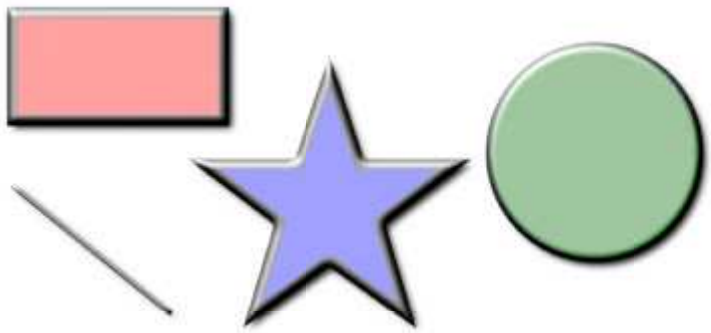

Abbildung 3: Effekte bei elementare SVG-Formen

Über einfache Vektorgraphik hinaus sieht der SVG-Standard Filter-Effekte vor. Damit können unter anderem Farb-, Überblend-, Wisch- und Mischeffekte berechnet werden. Auch Beleuchtungseffekte sind über die Definition von Lichtquellen möglich. Diese Filter können beliebig kombiniert und vielseitig konfiguriert werden. In Abbildung 3 ist das Ergebnis der Filterung der elementaren SVG-Formen aus Abbildung 1 dargestellt. Dazu wurden Beleuchtungs- Wisch- und Mischeffekte verwendet. An dieser Stelle soll auf die genauere Erläuterung von SVG-Filtern verzichtet werden. Eine gute Einführung findet sich in der SVG-Spezifikation [SVG Spec 1.0].

1.4 Erzeugung und Darstellung des SVG-Formats

Um SVG-Graphiken anzuzeigen, muss ein Viewer installiert sein, der den XML-Code verarbeitet und eine Graphik rendert. Ein SVG-Plugin ist von Adobe [Adobe SVG] und unterstützt alle gängigen Browser und Betriebssysteme. Ein weiterer Viewer mit dem Namen Squiggle ist Teil des Batik-Toolkits [Batik] und ist aus dem Apache-XML-Projekt [Apache XML] entstanden. Dieser Standalone-Viewer basiert auf Java und steht damit auf allen gängigen Betriebssystemen zur Verfügung. Im Mozilla Projekt [Mozilla] wird für den Mozilla-Browser ein nativer SVG-Viewer [Croczilla] entwickelt. Weiter Aktivitäten im Bereich SVG-Viewer gibt es in den Projekten KDE [KSVG] und Gnome [Gnome]. Um eine größere Verbreitung zu erlangen, ist es für das SVG-Format unabdingbar, native Viewer in die gängigen Browser und Anwendungssoftware zu integrieren.

Einfache SVG-Graphiken könne mit einem normalen Texteditor erstellt werden. Für komplexere Graphiken mit Pfaden, Animationen, Filtern etc. empfiehlt es sich, die Graphik mit einem Anwendungsprogramm zu generieren. Graphikprogramme wie Adobe Illustrator 10.0 [Illustrator] und Corel CorelDraw! 10.0 [CorelDraw] importieren und exportieren SVG-Graphiken. Auch Native SVG-Editoren wie Jasc WebDraw [WebDraw] und W3C Amaya [Amaya] sind bereits vorhanden. Neben Editoren gibt es Konverter von diversen Graphik-Formaten in das SVG-Format [SVG Converters]. Eine Besonderheit stellt der SVGMaker dar [SVGMaker]. Diese Software installiert sich unter Windows als Drucker und kann damit jede Druckausgabe als SVG-Datei speichern. Mehrseitige Dokumente bekommen eine Navigation, mit der die einzelnen Seiten angewählt und vergrößert werden können.

1.5 Vorteile für sensorisch Behinderte

Blinden Menschen sind elektronische Texte dank Braillezeile und Screenreader zugänglich. Jedoch sind Graphiken zu komplex, um sie von einer elektronischen Stimme beschreiben oder interpretieren zu lassen. Die Abbildung von Rastergraphik auf ein taktiles Ausgabemedium ist schwierig, da die Auflösung und Farben sehr stark reduziert werden müssen. SVG-Graphiken lassen sich auch auf Ausgabemedien mit geringer Auflösung darstellen. Neben der graphischen Ausgabe auf einem taktilen Ausgabemedium ist für den Blinden die Information verfügbar, aus welchen Elementen die Graphik aufgebaut ist. Zusätzlich können Attribute der Graphikelemente angezeigt werden. Die selektive Darstellung von Farben und Gruppen kann einem blinden Benutzer weiter unterstützen. Wenn Graphikelemente mit Gradienten oder Texturen gefüllt sind, können diese automatisch ausgeblendet werden. Aus den SVG-Dateien lassen sich Texte einfach extrahieren. Diese können dann auf einer Braillezeile dargestellt oder von einem Screenreader vorgelesen werden.

Die Spezifikation von SVG sieht vor, dass einer Graphik ein Titel und eine Beschreibung von Elementen hinzugefügt werden kann (mit dem <desc>-Tag vergleiche Quelltext oben). Für Sehende kann dieser Text als "ToolTip" angezeigt werden. Für Blinde ist er eine Hilfestellung, da er direkt vom Autor geschrieben wurde. Weitere Möglichkeiten werden in den *Accessibility Features* aufgeführt [SVG Accessibility].

2 MathML

Mathematische Ausdrücke werden im Web heutzutage als Rastergraphiken dargestellt. Dies hat bei der Wiederverwendung große Nachteile. Das W3C hat MathML [MathML] als Empfehlung für die Beschreibung von mathematischen Ausdrücken als Anwendung von XML verabschiedet. Die aktuelle Spezifikation liegt in der Version 2.0 vor [MathML Spec 2.0].

Die folgenden Abschnitte erläutern die Eigenschaften von MathML und stellen die Beschreibung von mathematischen Ausdrücken in MathML vor. Danach werden Hinweise zu Werkzeugen für die Erzeugung und Darstellung von MathML gegeben

2.1 Eigenschaften von MathML

MathML ist eine Beschreibungssprache für mathematische Ausdrücke. Anwendungsbereiche sind vor allem Lehrmaterialien und wissenschaftliche Dokumente. Dabei kann in MathML die Darstellung für mathematische Ausdrücke und auch der mathematische Zusammenhang beschrieben werden. Bereits existierende mathematische Ausdrücke können wiederverwendet und in verschiedene Autorenwerkzeuge importiert und exportiert werden. MathML ist wie alle XML-Anwendungen ein "Textformat" und kein Binärformat. Dadurch lassen sich mathematische Ausdrücke mit einem Texteditor erstellen. Für größere Ausdrücke bietet sich ein Formeleditor an, da die Ausdrücke sehr komplex werden können und damit nur schwer zu überblicken sind.

2.2 Beschreibung von mathematischen Ausdrücken in MathML

Für die Beschreibung von mathematischen Ausdrücke sind in MathML zwei Ebenen vorgesehen. Mathematische Ausdrücke können durch ihre mathematischen Zusammenhänge auf der semantischen Ebene oder durch ihre Darstellung beschrieben werden. Für mathematische Zusammenhänge ist in MathML die *Content Markup* und für die Darstellung die *Presentation Markup* vorgesehen. Um den Unterschied zu verdeutlichen, werden beide in den folgenden Abschnitten erläutert. Dazu wird der mathematische Ausdruck $(a + b)^2$ in beiden Beschreibungen vorgestellt. Anschließend wird eine Kombination der Content Markup und Presentation Markup als sinnvolle Beschreibung für die Praxis begründet.

Content Markup

In MathML wird der Inhalt eines mathematischen Ausdruckes im *Content Markup* beschrieben. Damit ist der Zusammenhang zwischen Zahlen, Bezeichnern (bzw. Variablen wie x), Operatoren etc. gemeint.

Das <apply>-Tag weist einem Operator seine Argumente zu. In der syntaktischen Reihenfolge steht der Operator vor den Operanden: <apply> operator Arg1 Arg2 ... </apply>

Um diesen mathematischen Ausdruck einfacher erklären zu können substituieren wir $a + b$ in $(a + b)^2$ durch x und erhalten x^2 und erklären diese beiden Ausdrücke getrennt voneinander.

Das Substitut $a + b$ enthält zwei Bezeichner und einen Operator. Damit ist in der oben vorgestellten <apply>-Syntax der Operator <plus/> und die Argumente sind <ci> a </ci> und <ci> b </ci>. Dabei ist für das Argument die Kurzform <plus/> verwendet worden und steht für <plus>...</plus>. Für jeden Argument-Typ gibt es verschiedene Tags: Zahlen wird <cn> und Bezeichnern <ci> zugewiesen.

```
<apply>
  <plus/>
  <ci>a</ci>
  <ci>b</ci>
</apply>
```

Der zweite Ausdrucke x^2 wird auf die gleiche Weise beschreiben. Dazu werden der Operand <power> und die Argumente x und 2 verwendet:

```
<apply>
 <power/>
     <ci> x </ci>
     <cn> 2 </cn>
</apply>
```

Fügt man nun beide Beispiele zusammen erhält man die Beschreibung für den mathematischen Ausdruck $(a + b)^2$:

```
<apply>
  <power/>
  <apply>
    <plus/>
    <ci>a</ci>
    <ci>b</ci>
  </apply>
  <cn>2</cn>
</apply>
```

Bei diesem Ausdruck sind die Klammern in $(a + b)^2$ inhärent gegeben und müssen deshalb nicht explizit angegeben werden.

Presentation Markup

Die Darstellung von mathematischen Ausdrücken wird in der Presentation Markup beschreiben. Alle darin vorkommenden Tags haben das Präfix m. Als Komplement zum obigen Beispiel wird auch hier der mathematische Ausdruck $(a + b)^2$ beschrieben:

Für die Darstellung werden mathematische Ausdrücke in horizontale Blöcke mit dem Tag <mrow> unterteilt und separiert damit Unterausdrücke voneinander. Um den Exponent des Ausdrucks darstellen zu können, wird eine Superscript-Notation verwendet: <msup> Basis Superscript </msup>. Für die Basis wird nun der Unterausdruck $(a + b)$ beschrieben. Die Klammern dieses Ausdrucks werden als Operatoren mit <mo> erzeugt. Anschließend folgt noch die Darstellung für den + Operator ebenfalls mit <mo> und die Bezeichner mit <mi>:

<mi> a </mi> <mo> + </mo> <mi> b </mi>

Schließlich wird noch der Exponent des Ausdrucks hinzugefügt als Zahl mit <mn>:

<mn> 2 </mn>

```
<mrow>
  <msup>
    <mrow>
      <mo>(</mo>
      <mi>a</mi>
      <mo>+</mo>
      <mi>b</mi>
      <mo>)</mo>
    </mrow>
    <mn>2</mn>
  </msup>
</mrow>
```

Content Markup und Presentation Markup kombinieren

Oben wurden die beiden Ebenen von MathML, die Content Markup und die Presentation Markup, vorgestellt. Wichtig ist es für einen Autor, entscheiden zu können, auf welcher Ebene er seine mathematischen Ausdrücke beschreibt. Die Content Markup ist für den Austausch von mathematischen Inhalten gedacht. Im Vordergrund stehen dabei Formeleditoren und so genannte Computer Algebra Systeme. Die Presentation Markup beschreibt die Darstellung von mathematischen Ausdrücken. Sie kann beispielsweise in Webbrowsern oder Textverarbeitungssystemen verwendet werden. In diesen Anwendungsprogrammen ist oft noch kein Formelinterpreter integriert. Sinnvoll ist deshalb eine Kombination von Content Markup und der Presentation Markup. Dies kann über das <semantic>-Tag geschehen, mit dem die Content Markup in die Presentation Markup integriert werden kann. Damit ist die Darstellung und Weiterverarbeitung vom mathematischen Ausdrücken gesichert. Die Content Markup kann in die Presentation Markup umgeformt werden, jedoch ist der umgekehrte Weg nicht immer eindeutig und deshalb nur über Heuristiken möglich. Das Ziel für die Zukunft ist, mathematische Ausdrücke in der Content Markup zu beschreiben, um diese für die Darstellung und auch für die Berechnung verwenden zu können.

Erweiterte mathematische Ausdrücke in MathML

In MathML lassen sich auch erweitere mathematische Ausdrücke aus den Disziplinen Arithmetik, Algebra, Logik, Mengenlehre, Lineare Algebra, Statistik, etc. beschreiben. In [MathML Spec 2.0] sind im Anhang über einhundert Seiten diesem Thema gewidmet. Abbildung 4 zeigt einige erweiterte mathematische Ausdrücke, die mit dem Browser Mozilla 1.0 gerendert wurden und dem Mozilla MathML Torture Test entnommen sind [Mozilla Math Test].

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{a_4}}}} \qquad \sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+x}}}}}}}$$

$$\sum_{i=1}^{p}\sum_{j=1}^{q}\sum_{k=1}^{r} a_{i\,j}\, b_{j\,k}\, c_{k\,i} \qquad \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) \left|\varphi(x + \mathrm{i}\, y)\right|^2 = 0$$

Abbildung 4: Erweiterte mathematische Ausdrücke in MathML

2.3 Erzeugung und Darstellung von MathML

MathML kann bereits von einigen Webbrowsern direkt angezeigt werden [Mozilla, Netscape, Amaya]. Für den Internet Explorer [Internet Explorer] ist dagegen noch ein Plugin notwendig (beispielsweise [MathPlayer]). Einige Viewer können allerdings nur die Presentation Markup darstellen.

Um mathematische Ausdrücke zu erstellen oder zu bearbeiten, gibt es viele Anwendungsprogramme, die MathML importieren und exportieren können. Wenn die Ausdrücke für Berechnungen verwendet werden sollen (Content Markup), können Computer Algebra Systeme verwendet werden [Maple, Mathcad, Mathematica]. Reine Formeleditoren, die nur die Presentation Markup verarbeiten können, sind beispielsweise Amaya, Open Office, Star Office, MathType, Publicon, etc. [Open Office, Star Office, MathType, Publicon]. Außerdem besteht die Möglichkeit mathematische Ausdrücke von LaTeX nach MathML zu konvertieren [Tex4Moz, Tex4ht] und vice versa [XSLT MathML Lib]. Dadurch ist es auch für Blinde möglich, die mathematischen Ausdrücke zu erfassen und zu formulieren, da LaTeX sehr häufig von Blinden am Computer eingesetzt wird [Aldridge]. Die Transformation mathematischer Ausdrücke von MathML in Mathematik-Schriften für Blinde [Schweikhardt] kann mittels XSLT-Stylesheets [XSLT] geschehen [Bosse].

3 Ausblick: SVG und MathML für web-basierte Lernumgebungen

SVG und MathML erlauben eine neuartige Form der Kommunikation auf Lernplattformen. Es wird ermöglicht in Lehrmaterialien, in Diskussionsforen, in Chats, in E-Mails etc. Vektorgraphiken und mathematische Ausdrücke zu verwenden. Dadurch ist eine intensivere und mit weniger Aufwand verbundene Kommunikation möglich, die zudem blinde Benutzer nicht ausschließt.

SVG und MathML bieten in web-basierte Lernumgebungen viele Vereinfachungen beim Austausch von Lehrmaterialien und in der Kommunikation. SVG-Graphiken erleichtern die Handhabung und speziell den Austausch von graphischen Informationen für Lernende und Lehrende. Das Verwenden von Vektorgraphik ermöglicht die Überarbeitung von bereits existierenden Graphiken. Dadurch sinkt die Hemmschwelle, eine Graphik als Skizze anzufertigen und diese dann später zu korrigieren oder in eine saubere Form zu bringen. Auch Prozesse können einfach in SVG dargestellt werden. Der Aufwand, diese Animationen zu erstellen, kann durch Autorenwerkzeuge verringert werden. Durch SVG ist es aufgrund der einfachen Skalierbarkeit möglich, komplexe Graphiken auf kleinen Ausgabegeräten wie Personal Digital Assistants (PDAs) und Handys darzustellen. Das W3C hat dazu einen Entwurf für die SVG Version 1.1 veröffentlicht [SVG Mobile]. Analog hierzu können mathematische Ausdrücke mit MathML einfacher als bisher ausgetauscht und bearbeitet werden. Durch die Skalierbarkeit sind auch diese für kleine Ausgabegeräte geignet.

Bei SVG sind schon viele Erweiterungen angedacht. Eine Erweiterung ist Constraint SVG [CSVG]. Mit CSVG können Attribute wie Größe, Position, etc. der einzelne elementaren Formen relativ zu den anderen beschrieben werden. Dadurch ist die Anordnung der einzelnen Elemente auf einer Seite festgelegt und kann damit auch besser an die Restriktionen des Ausgabegeräts angepasst werden. Andere Erweiterungen von SVG sind bereits in Planung. Es gibt die ersten Entwürfe von Benutzungsoberflächen-Widgets in SVG [KevLinDev]. Dadurch wird es möglich, direkt im Browser graphische Benutzungoberflächen anzuzeigen. Die Einsatzmöglichkeiten wie beispielsweise die Simulationen, Tele-Experimente etc. sind vielfältig.

Für Benutzer, deren Browser SVG und MathML nicht unterstützt, kann auf dem Server eine Rastergraphik für die Darstellung erzeugt werden. Diese kann dann problemlos angezeigt werden. Die eigentliche SVG- und MathML-Datei kann zusätzlich zum Download angeboten werden. Dies ermöglicht die Integration dieser neuen Formate bereits in einer frühen Phase.

Kontaktadresse

Martin Rotard
Universität Stuttgart
Institut für Visualisierung und Interaktive Systeme
Breitwiesenstraße 20-22
70565 Stuttgart
martin.rotard@informatik.uni-stuttgart.de

Quellen

[Adobe SVG] Adobe, *SVG Zone*, http://www.adobe.com/svg/, 17.07.2002

[Aldridge] Aldridge, Vivian: *Bericht zur Marburger LaTeX-Tagung im März 2002: Mathematikschrift am Computer - LaTeX setzt sich durch* http://www.braille.ch/mathe/tagu1ber.htm, 17.07.2002

[Amaya] W3C, *Amaya*, http://www.w3.org/Amaya/, 17.07.2002

[Apache XML] Apache XML Project, http://xml.apache.org/, 17.07.2002

[Batik] Apache XML Project, Batik, http://xml.apache.org/batik/, 17.07.2002

[Bosse] Bosse, Klaus: *Tansformation von mathematischen Dokumenten, die Teile in MathML enthalten, in eine in Stuttgarter Mathematikschrift für Blinde (SMFB) formulierte Darstellung*, Institut für Informatik, Universität Stuttgart, 2002

[CorelDraw] Corel Corporation, *CorelDraw*, http://www.corel.com/, 17.07.2002

[Croczilla] Mozilla, *SVG Samples*, http://www.croczilla.com/svg/, 17.07.2002

[CSVG] Greg J. Badros, Will Portnoy, Jeff Nichols, Alan Borning, *CSVG: Constraint Scalable Vector Graphics*, http://www.cs.washington.edu/homes/gjb/CSVG/, 17.07.2002

[DOM] W3C, *Document Object Model*, http://www.w3.org/DOM/, 08.07.2002

[ECMAScript] Mozilla.org, JavaScript, http://www.mozilla.org/js/, 08.07.2002

[Flash] Macromedia, *Flash*, http://www.macromedia.com/software/flash/, 11.07.2002

[Flash2SVG] Steve Probets, *Convert Flash into SVG*, http://www.ep.cs.nott.ac.uk/~sgp/swf2svg.html, 11.07.2002

[Gnome] Gnome project, Gnome, http://www.gnome.org/, 16.07.2002

[ICC] International Color Consortium, *ICC Homepage*, http://www.color.org/, 11.07.2002

[Illustrator] Adobe, *Illustrator*, http://www.adobe.com/products/illustrator/, 17.07.2002

[Internet Explorer] Microsoft, *Internet Explorer*, www.microsoft.com/windows/ie/, 17.07.2002

[JavaScript] Mozilla.org, *JavaScript*, http://www.mozilla.org/js/, 08.07.2002

[KevLinDev] Kevin Lindsey, *KevLinDev*, http://www.kevlindev.com/, 16.07.2002

[KSVG] KDE.org, *KSVG*, http://svg.kde.org/, 16.07.2002

[Maple] Waterloo Maple Inc., *Maple*, http://www.maplesoft.com, 16.07.2002

[Mathcad] MathSoft, *MatCad*, http://www.mathcad.com/, 16.07.2002

[Mathematica] Wolfram Research, *Mathematika*, http://www.wolfram.com/products/mathematica/, 16.07.2002

[MathML] W3C, *MathML- Math Home*, http://www.w3.org/Math/, 17.07.2002

[MathML Spec 2.0] W3C, *Mathematical Markup Language (MathML) Version 2.0*, http://www.w3.org/TR/MathML2/, 17.07.2002

[MathPlayer] Design Science, *MathPlayer*, http://www.dessci.com/webmath/mathplayer/, 15.07.2002

[MathType] WebEQ, *MathType*, http://www.dessci.com/, 16.07.2002

[Mozilla] Mozilla.org, *Mozilla.org*, http://www.mozilla.org/, 08.07.2002

[Mozilla Math Test] Mozilla.org, *MathML Torture Test,* http://www.mozilla.org/projects/mathml/demo/texvsmml.xml, 17.07.2002

[Netscape] Netscape, *Browser Central*, http://wp.netscape.com/browsers/, 17.07.2002

[Open Office] OpenOffice.org Source Project, *OpenOffice*, http://www.openoffice.org/, 17.07.2002

[Publicon] Wolfram Research, *Publicon*, http://www.wolfram.com/products/publicon/, 16.07.2002

[Schulmeister 2001] Schulmeister, Rolf: *Virtuelle Universität - virtuelles Lernen,* Oldenbourg Verlag, München, Wien, 2001

[Schweikhardt] Schweikhardt, Waltraud; Weicker, Nicole: *Mathematik am Computer für Blinde*, In: Horst Oberquelle and Reinhard Oppermann (ed.), Mensch und Computer 2001, Stuttgart, Teubner, 2001

[SMIL] W3C, *Synchronized Multimedia*, http://www.w3.org/AudioVideo/, 17.07.2002

[Star Office] Sun Microsystems, *Star Office*, www.sun.com/staroffice/, 16.07.2002

[SVG] W3C, *Scalable Vector Graphics (SVG)*, http://www.w3.org/Graphics/SVG/, 16.07.2002

[SVG Accessibility] W3C, *Accessibility Features of SVG*, http://www.w3.org/TR/SVG-access/, 23.07.2002

[SVG Converters] W3C, *SVG Implementations - SVG Converters*, http://www.w3.org/Graphics/SVG/SVG-Implementations.htm8#convert, 08.07.2002

[SVG Mobile] W3C, *Mobile SVG Profiles: SVG Tiny and SVG Basic,* http://www.w3.org/TR/SVGMobile/, 12.07.2002

[SVG Spec 1.0] W3C, *Scalable Vector Graphics (SVG) 1.0 Specification*, http://www.w3.org/TR/SVG/, 17.07.2002

[SVGMaker] Software Mechanics Pty Ltd, *SVGmaker*, http://www.svgmaker.com/, 08.07.2002

[Techexplorer] IBM, *Techexplorer*, http://www.software.ibm.com/network/techexplorer, 16.07.2002

[Tex4ht] Eitan M. Gurari, *TeX4ht*, http://www.cis.ohio-state.edu/~gurari/TeX4ht/, 16.07.2002

[Tex4Moz] MathZilla, *Tex4Moz*, http://pear.math.pitt.edu/mathzilla/tex4moz.html, 16.07.2002

[W3C] W3C, *World Wide Web Consortium,* http://www.w3.org/, 17.07.2002

[WebDraw] Jasc Software, *WebDraw* , http://www.jasc.com/products/webdraw/, 17.07.2002

[X3D] Web3D Consortium, *Extensible 3D Graphics Working Group*, http://www.web3d.org/x3d.html, 17.07.2002

[XLink] W3C, *XML Pointer, XML Base and XML Linking*, http://www.w3.org/XML/Linking, 17.07.2002

[XML] W3C, *Extensible Markup Language (XML)*, http://www.w3.org/XML/, 17.07.2002

[XSLT] W3C, *XSL Transformations*, http://www.w3.org/TR/xslt, 23.07.2002

[XSLT MathML Lib] Yaroshevich, Vasil I.: *XSLT MathML Library - MathML to LaTeX translator*, http://www.raleigh.ru/MathML/mmltex/index.php?lang=en, 23.07.2002