SVG

Wikibooks.org

20. Juni 2012

Inhaltsverzeichnis

Abb. 1 SVG - Das Buch

SVG (Skalierbare Vektorgraphik, englisch: Scalable Vector Graphics) ist der internationale, vom W3C empfohlene Standard für Vektorgraphik. SVG ist ein Format der Sprachfamilie XML[1] und beschreibt zweidimensionale Vektorgraphiken.

SVG fügt sich gut in andere bestehende Standards ein, ergänzt sie und wird von ihnen ergänzt. SVG-Dokumente können wie (X)HTML-Dokumente alleine verwendet werden, um Information zugänglich zu vermitteln. Es lassen sich aber auch SVG-Fragmente direkt in beliebige andere XML-basierte Dokumente (zum Beispiel XHTML) einfügen. Andersherum lassen sich auch andere XML-Formate in SVG einfügen, üblich sind insbesondere Metainformationen per RDF und Dublin-Core im Element ***metadata*** und direkt anzeigbare Informationen im Element ***foreignObject***.

Die in SVG enthaltenen Fähigkeiten zur deklarativen Animation basieren auf dem XML-Format SMIL, ebenso wie die neuen Multimediaelemente in SVG tiny 1.2. Graphiken können auch über das DOM (Dokument-Objekt-Modell) zum Beispiel per ecmascript manipuliert werden, was neben der deklarativen Animation, die interaktiv begonnen und beendet werden kann, eine andere Möglichkeit eröffnet, auf Ereignisse zu reagieren und mit dem Benutzer zu interagieren. Graphikelemente lassen sich als Verweise verwenden (keine Verwendung eines Elements wie ***map*** in (X)HTML notwendig), Filter und Maskierungen können vorgenommen werden, Schriftarten definiert werden und vieles mehr.

Ein aktueller SVG-Standard liegt seit 2003-01-14 in der Version 1.1 als Empfehlung des W3C vor. Eine überarbeitete zweite Auflage mit der Korrektur bekannter Fehler ist 2011-08-16 erschienen. SVG tiny 1.2 liegt seit 2008-12-22 als Spezifikation vor und ist zum einen auf den mobilen Markt ausgerichtet, wie auch die älteren mobilen Varianten von SVG 1.1, zum anderen wird es als Grundmodul für die in Arbeit befindliche Version 2 dienen, für welche auch schon für einige Module Arbeitsentwürfe veröffentlicht wurden. Das Buch beschreibt Version 1.1 und tiny 1.2.

Dieses Buch soll sowohl Einsteigern als Tutorium als auch fortgeschrittenen Anwendern als Referenz dienen, deshalb werden die Elemente in einer (aus Sicht der Autoren, Diskussionen dazu sind gerne willkommen) sinnvollen Reihenfolge eingeführt (auch die Reihenfolge der Attribute folgt dieser Richtlinie so gut wie möglich), und werden zum Nachschlagen in der Referenz[2] referenziert.

Anmerkung zu Vorschaubildern: Da leider die Qualität des Programmes deutlich begrenzt ist, welches aus den SVG-Beispieldateien Vorschaubilder erstellt, sind diese Bilder kein Hinweis auf eine korrekte Interpretation des jeweiligen SVG-Dokumentes. Im Begleittext im SVG-Dokument und im laufenden Text des Buches wird hingegen erläutert, wie die entsprechenden Bestandteile von SVG korrekt zu interpretieren sind. Als zusätzlicher Hinweis ist zumeist für einige Darstellungsprogramme angegeben, ob für diese mit einer korrekten Darstellung des entsprechenden Bestandteils zu rechnen ist. Durch Verbesserung dieser Programme wird sich in Zukunft sicherlich die Zuverlässigkeit der Darstellung erhöhen.

Anmerkung zur PDF-Version: Diese ist mehr oder weniger automatisch erstellt, für die meisten Beispiele gibt es darin nur Vorschaubilder und Verweise auf die relevanten SVG-Dateien, die dann jeweils von wikibooks geladen werden müssen. Auch die direkt dargestellten Beispiele sollten direkt von wikibooks geladen werden, sonst kann man sich in der PDF-Version auch bei diesen Beispielen den Quelltext nicht ansehen.

1 `http://de.wikibooks.org/wiki/XML`
2 Kapitel 23 auf Seite 757

Anmerkung zur Druckversion: Auch hier ist es natürlich notwendig, die meisten Beispiele am Computer nachzuvollziehen, den Quelltext anzusehen. Auch hier gilt, auf die meisten Beispiele wird nur verwiesen beziehungsweise es werden nur Vorschaubilder im laufenden Text angeboten. Die SVG-Dokumente für diese Beispiele werden nicht automatisch mit ausgedruckt, bei eingebundenen Beispielen ist eine ausgedruckte Version natürlich auch nicht besonders zur Analyse oder Weiterverwendung der Quelltexte der Beispiele geeignet.

Buchpatenschaft/Ansprechperson:**Doktorchen**[3]

3 http://de.wikibooks.org/wiki/Benutzer%3ADoktorchen

1 Einführung

Dieses Kapitel soll kurz die Motivation des Buches erläutern und helfen, sich im Buch zurechtzufinden.

1.1 Die Frage nach dem Warum

1.1.1 Warum eigentlich Vektorgrafiken?

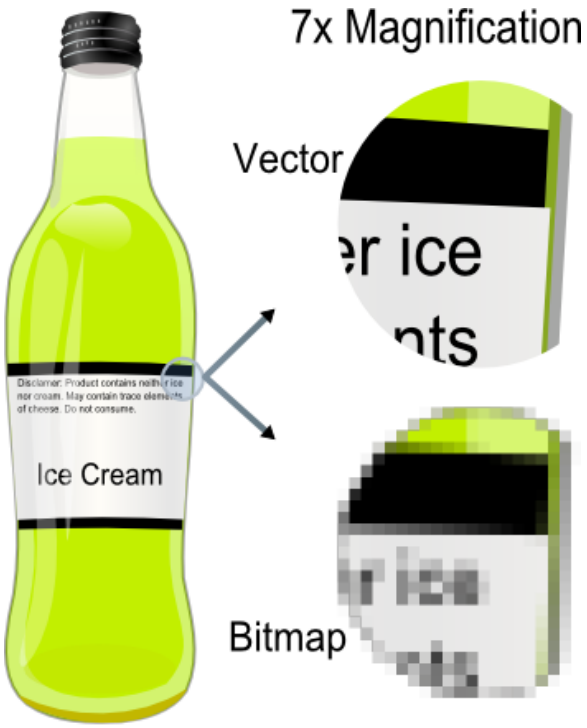


Abb. 2 Vektorgrafiken lassen sich ohne Qualitätsverlust beliebig skalieren.

„Normale“ Rastergrafiken wie zum Beispiel JPEG/JFIF, PNG oder GIF speichern das Bild als reine Farbinformation, also jedes einzelne Pixel im Falle von GIF und PNG oder aufgrund geschickter Transformationen eher als Farbverteilungen bei JPEG/JFIF. Vektorgrafiken speichern hingegen, aus welchen Elementen ein Bild besteht, also was die darstellenden Programme tun müssen, um das Bild zu präsentieren. Mit Vektorgrafiken ist im Dokument notierbar, was eine inhaltlich wichtige Struktur ist, worauf es ankommt.

Wenn es darum geht, Photographien abzuspeichern, so bieten Rastergrafiken klare Vorteile, denn eine Kamera braucht bei einem Rohdatenformat einfach nur Pixel für Pixel abzuspeichern, was der oder die Bildsensoren wahrnehmen. Weil viele Digitalkameras eine Bayer-Matrix im Sensor haben, interpolieren diese sogar die meisten Pixel nur und führen zumeist auch eine etwas aufwendigere Kompressionstransformation durch, um das Bild (auch) als JPEG/JFIF abspeichern zu können. Bei JPEG/JFIF ist allerdings wiederum die Farbtiefe begrenzt. So kommt es also zu Verlusten, bereits bedingt durch die Aufnahmetechnik oder das Format, mit welchem die Aufnahme abgespeichert wird.

Die Digitalkamera braucht indes nicht zu verstehen, was das Motiv ist, wenngleich dies bei besseren Modellen oftmals der Fall ist, um die automatische Fokussierung und Belichtung zu optimieren. Für die Aufnahme selbst braucht die Digitalkamera allerdings nicht zu wissen, welche Elemente auf dem gewählten Bildausschnitt dargestellt und wichtig sind, auch sind die auf Photographien dargestellten Geometrien und Farbverläufe oftmals derart komplex, dass eine vektorielle Beschreibung ein Vielfaches der Speichergröße benötigen würde.

Vektorformate bieten sich jedoch dort an, wo Grafiken digital gezeichnet und verändert werden und einfach veränderbar bleiben sollen. Dies ist zum Beispiel der Fall bei Konstruktionszeichnungen, Symbolen, Logos, technischen oder wissenschaftlichen Daten und deren Visualisierung. Auch können vorhandene Rastergrafiken um solche Elemente ergänzt werden, etwa durch einen Text, der lesbar bleiben soll, oder durch eine Markierung, die gegebenenfalls wieder entfernbar sein soll.

Vektorgrafiken bieten viele Vorteile:

- Sie lassen sich vergrößern, ohne dass es zu „Treppenbildung“ kommt (siehe Beispielbild)
- Sie lassen sich leicht bearbeiten (wenn man z.B. einen Text etwas höher platzieren möchte, muss man nur die Koordinaten des Textelements anpassen, bei Rastergrafiken müsste man die Stelle, an der sich der Text vorher befand, aufwendig nachbearbeiten)
- Eventuell vorhandener Text lässt sich maschinell durchsuchen
- Die Dateigröße ist unabhängig von der Auflösung. Sie hängt nur von der Anzahl der dargestellten Elemente ab und ist deshalb bei den angesprochenen Anwendungen oft geringer als die Speicherung als Rastergrafik.

Auch Animationen oder Interaktionen des Nutzers lassen sich bei Vektorformaten leichter berücksichtigen. Bei Animationen muss nicht wie bei Videos die Bewegung bildweise gespeichert werden. Vielmehr ist es möglich, nur einzelne Eigenschaften einzelner Objekte unabhängig von anderen Bildinhalten zu ändern und dies auch nicht bildweise, sondern durch eine Angabe der Interpolationsmethode zwischen wenigen Stützstellen.

Diese Vorteile beruhen vor allem darauf, dass bei der Speicherung als Rastergrafik ein Informationsverlust stattfinden kann, denn ein Pixel in einer Rastergrafik ist einfach nur ein Pixel mit einer bestimmten Farbe, nicht mehr und nicht weniger, das eigentliche Bild war jedoch aus verschiedenen Elementen (z.B. einem Kreis) aufgebaut, die als Rastergrafik nichts anderes als eine Ansammlung aus Pixeln sind, ihre eigentliche Bedeutung geht verloren. Anders bei der Speicherung als Vektorgrafik, die Elemente behalten hier weiterhin ihre Bedeutung und der Informationsverlust (z.B. die letzten Bearbeitungsschritte) wird minimiert.

Mehr dazu im Abschnitt Vektorgrafik.[1]

1 Kapitel 1.4 auf Seite 9

1.1.2 Warum SVG?

Der SVG-Standard wurde vom W3C (World Wide Web Consortium) ins Leben gerufen, demselben Gremium, das sich auch um die (X)HTML-Spezifikationen und viele andere Standards für das Internet kümmert. SVG ist also ein herstellerunabhängiger Standard auf dem sonst sehr zersplitterten Markt der Vektorformate. Bei dem Entwurf von SVG wurde darauf geachtet, dass es sich möglichst gut in andere Spezifikationen einfügt, so wurde nicht versucht, ein allmächtiges Format zu schaffen, sondern ein Format, das in dem Bereich, für den es konstruiert wurde, sehr mächtig ist (im Falle von SVG Grafikanwendungen), aber nicht versucht, Funktionalität zu beinhalten, die eher in den Aufgabenbereich eines anderen Standards fällt. Um dies zu ermöglichen, wurde SVG auf Basis von XML spezifiziert, was es erlaubt, SVG mit allen anderen XML-basierten Auszeichungssprachen zu kombinieren (vorausgesetzt das Anzeigeprogramm versteht diese) und die etablierten Werkzeuge zum Bearbeiten von XML zu benutzen. Davon wird auch oft Gebrauch gemacht: Metadaten (z.B. Autor und Lizenz, Titel und Beschreibung) werden mittels RDF gespeichert (RDF - englisch: Resource Description Framework), Verweise und Referenzen mittels XLink realisiert, SVG-Dokumente können in andere XML-Dateien (z.B. XHTML oder DocBook) integriert werden, sie können mittels ecmascript dynamisch verändert und mit CSS den aktuellen Bedürfnissen gemäß dekoriert werden (ohne die Grafik selbst zu ändern).

Ein wichtiger Bestandteil von SVG ist auch, dass dieses Format Strukturen beinhaltet, welche die eigentliche Information und Struktur des Bildes zugänglich machen kann. Es gibt ausgereifte Möglichkeiten, im Bild selbst Alternativtexte für alle relevanten Strukturen zu notieren, so dass ein Verständnis der Information des Bildes auch mit einer alternativen Textrepräsentation möglich wird, wenn keine visuelle Präsentation möglich ist – warum auch immer.

Die Verwendung von XML hat den angenehmen Nebeneffekt, dass die Dateien mit einem einfachen Texteditor relativ komfortabel manuell bearbeitet werden können.

1.2 Wie man dieses Buch liest

Nun, grundsätzlich ist das natürlich jedem selber überlassen und hängt ganz von den persönlichen Vorlieben und Lernverhalten ab. Wer etwas Bestimmtes sucht, sollte sich am besten in der Referenz[2] oder im Inhaltsverzeichnis[3] die entsprechenden Stellen heraussuchen. Möchte man eher einen Überblick über das Thema erhalten, so kann man Kapitel anlesen und sich einen Überblick verschaffen, gegebenenfalls zum nächsten Kapitel übergehen. Das Durchlesen des gesamten Buches bietet sich für diejenigen an, die sich ausführlich mit dem Thema beschäftigen wollen.

1.3 Konventionen

1.3.1 Elemente, Eigenschaften und Attribute

Attribut-, Eigenschafts- und Elementnamen werden immer mit der gleichen Dekoration versehen, um sie vom übrigen Text abzuheben: Ein *Attribut* oder eine *Eigenschaft* und ein ***Element***.

2 Kapitel 23 auf Seite 757

3 `http://de.wikibooks.org/wiki/SVG`

1.3.2 Beispiele

Zu jedem Element oder Attribut sollte mindestens ein Beispiel vorhanden sein, welches in nichtproportionaler Schrift mit Syntaxhervorhebung, bei längeren Beispielen auch mit Zeilennumerierung dargestellt wird:

```
<element attribut="wert">
    <text>Ich bin ein Quelltext</text>
</element>
```

Teils wird bei umfangreicheren Beispielen auch direkt auf die SVG-Datei verwiesen, wo dann sowohl das graphische Ergebnis im Darstellungsprogramm betrachtet werden kann als auch der Quelltext der SVG-Datei, die wiederum selbst oft eine entsprechende Beschreibung enthält.

In einigen Fällen wird auch eine automatisch erzeugte Vorschau im Format PNG angeboten, über welche dann zudem auch zur zugehörigen SVG-Datei gelangt werden kann.

1.3.3 Unterstützung

Zu jeder Eigenschaft, zu jedem Attribut oder Element wird angegeben, seit welcher Version des SVG-Standards es vorhanden ist und in welchen Programmen es unterstützt wird. Die Liste der Programme ist nicht vollständig und meist liegt ein kompletter Test einer Eigenschaft oder eines Attributes oder gar eines Elementes gar nicht vor, daher kann dies nur als Anhaltspunkt gesehen werden, der eine kritische Prüfung im Einzelfall leider nicht erspart.

Ein Element, das seit der Version 1.1 zum SVG-Standard gehört, von Firefox und Safari mindestens seit der Version 4, bei Opera 9, teilweise auch schon in Version 8 und von Konqueror gar nicht unterstützt wird, bei batik/Squiggle gar falsch implementiert wurde und für welches für librsvg etc. kein Testergebnis vorliegt, erhält dann folgende Kennzeichnung:

element

Squiggle (Batik)	1.7 (falsch)
Opera (Presto)	9 (8 teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	–
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Hier folgt die Beschreibung des Elements…

Text

Text

Text

Die Versionsangaben bedeuten dabei **nicht**, dass das Element erst seit der angegebenen Version unterstützt wird, sondern dass die Unterstützung mit dieser Version erfolgreich getestet wurde, also das Element *mindestens* seit der angegebenen Version unterstützt wird. Ein Fragezeichen deutet darauf hin, dass die Unterstützung mit dem angegebenen Programm noch nicht getestet wurde und ein Strich, dass bis jetzt kein Test erfolgreich war.

Weitere Informationen zur Software-Unterstützung, was zu tun ist, wenn die von Ihnen genutzte Software nicht in der Liste auftaucht, und vieles mehr erfahren Sie im Kapitel Software[4] und auf der Dokumentationsseite der verwendeten Vorlage[5].

1.4 So können *Sie* mithelfen

Sie haben einen Fehler gefunden, oder wollen die Versionsangabe aktualisieren? Oder Sie haben ein kompaktes Anwendungsbeispiel, welches im Buch nicht fehlen sollte? Gut! Da sich dieses Buch auf einem Wiki befindet, müssen Sie den Autoren keine email schreiben und darauf hoffen/warten, ob das Buch entsprechend angepasst wird, sondern Sie können die Änderungen selbst durchführen. Wenn Sie also einen Fehler (sei es in der Rechtschreibung oder im Inhalt) finden, klicken Sie einfach auf „Bearbeiten“ und beheben Sie ihn – nur Mut, Sie können nicht viel falsch- oder gar kaputtmachen. Sollten Sie sich nicht sicher sein, ob überhaupt ein Fehler vorliegt, zwar wissen, dass ein Fehler vorliegt aber keine Korrektur dafür parat haben, oder einfach nur eine Frage haben, so sind Sie auf der Diskussionsseite zu dem entsprechenden Kapitel genau richtig, zu der Sie gelangen wenn Sie oben auf „Diskussion“ klicken.

Hinweise dazu, wie Sie die Versionsangaben aktualisieren, finden Sie im Kapitel Software[6] und auf der Dokumentationsseite der verwendeten Vorlage[7].

Nun wünschen wir Ihnen noch viel Spaß beim Lesen dieses Buches und hoffen, dass es Ihnen gefällt.

1.5 Kurze Einführung in die Welt der Vektorgrafik

Eine **Vektorgrafik** ist eine Computergrafik, die aus grafischen Primitiven wie Linien, Kreisen, Polygonen oder allgemeinen Kurven (splines) zusammengesetzt ist. Meist sind mit Vektorgrafiken Darstellungen gemeint, deren Primitive sich zweidimensional in der Ebene beschreiben lassen. Eine Bildbeschreibung, die sich auf dreidimensionale Primitive stützt, wird eher 3D-Modell oder Szene genannt.

Um beispielsweise das Bild eines Kreises zu speichern, benötigt eine Vektorgrafik mindestens zwei Werte: die Lage des Kreismittelpunkts und den Kreisdurchmesser. Neben der Form und Position der Primitiven werden eventuell auch die Farbe, Strichstärke, diverse Füllmuster und weitere das Aussehen bestimmende Daten angegeben.

1.6 Eigenschaften

4 Kapitel 1.7 auf Seite 13
5 http://de.wikibooks.org/wiki/SVG%2F_Vorlage%2F_Versionen%2F_Doku
6 Kapitel 1.7 auf Seite 13
7 http://de.wikibooks.org/wiki/SVG%2F_Vorlage%2F_Versionen%2F_Doku

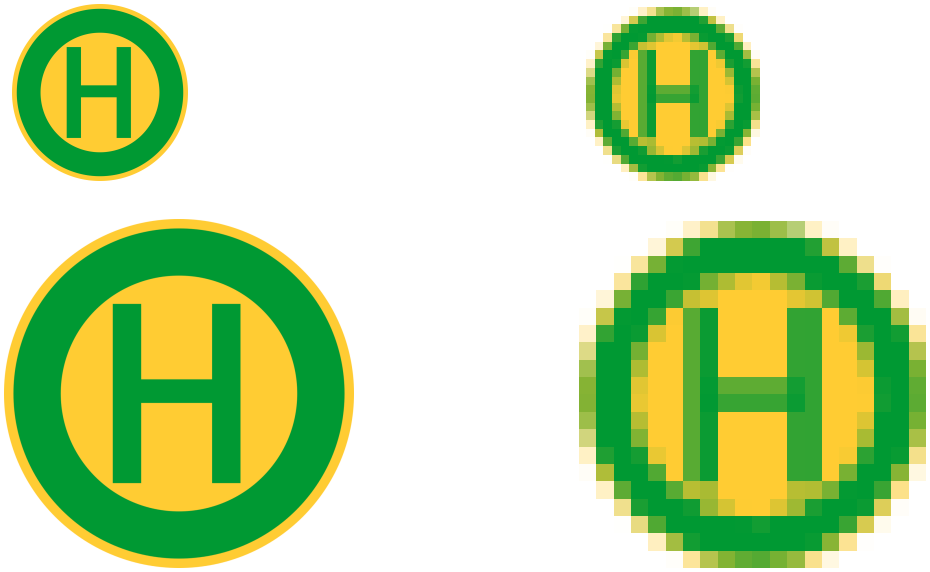


Abb. 7 Vektorgrafik

Abb. 8 Rastergrafik

Vektorgrafik

Rastergrafik

Vektorgrafiken lassen sich ohne Qualitätsverlust beliebig skalieren.

Vektorgrafiken basieren anders als Rastergrafiken nicht auf einem Pixelraster, in dem jedem Bildpunkt ein Farbwert zugeordnet ist, sondern auf einer Bildbeschreibung, die die Objekte, aus denen das Bild aufgebaut ist, exakt definiert. So kann beispielsweise ein Kreis in einer Vektorgrafik über die Lage des Mittelpunktes, den Radius, die Linienstärke und die Farbe vollständig beschrieben werden; nur diese Parameter werden gespeichert. Im Vergleich zu Rastergrafiken lassen sich Vektorgrafiken daher oft mit deutlich geringerem Platzbedarf speichern.

Die Erzeugung von Vektorgrafiken ist Gegenstand der geometrischen Modellierung und geschieht meist mittels eines Vektorgrafikprogramms oder direkt mit einer Auszeichnungssprache (wie hier in unserem Lehrbuch). Rastergrafiken können mit gewissen Einschränkungen in Vektorgrafiken umgewandelt werden (Vektorisierung[8]); manche Texterkennungsprogramme basieren auf einem Vektorisierungsalgorithmus. Mittlerweile bieten gängige Vektorgrafikprogramme Funktionen an, die es erlauben, Vektorgrafiken mit Farbverläufen und Transparenzstufen zu speichern und damit eine größere Zahl von Bildern zufriedenstellend zu beschreiben. Auch solche Vektorgrafiken lassen sich, im Gegensatz zu Rastergrafiken, bequem und verlustfrei verändern und transformieren.

Auf bis in die 1980er Jahre verbreiteten Vektorbildschirmen wurden Vektorgrafiken durch einen Kathodenstrahl direkt auf dem Bildschirm gezeichnet. Zur Anzeige auf heute üblichen Rasterbildschirmen müssen Vektorgrafiken gerastert werden.

8 `http://de.wikipedia.org/wiki/Vektorisierung%20%28Grafik%29`

1.7 Anwendungen

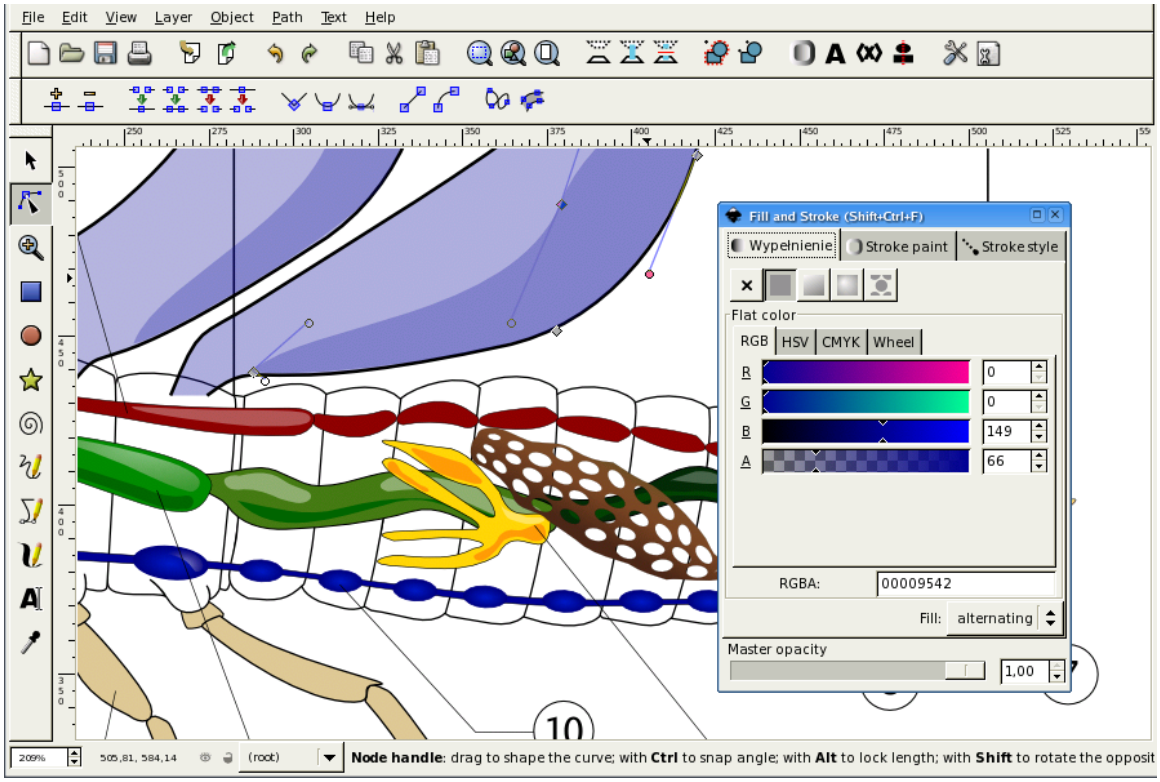


Abb. 9 Bearbeitung einer Vektorgrafik mit Inkscape[a]

a `http://de.wikibooks.org/wiki/Inkscape`

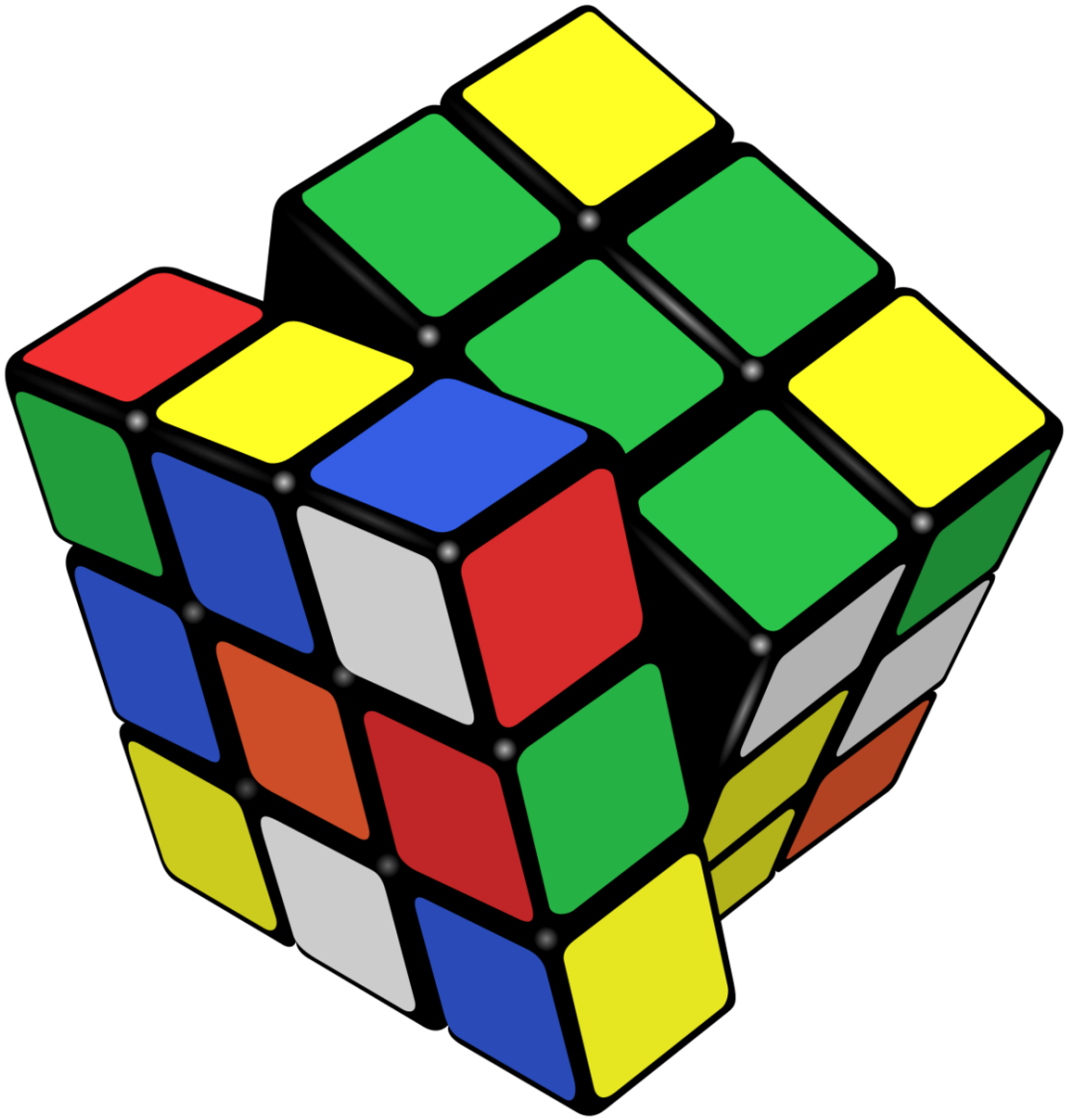


Abb. 10 Zauberwürfel im SVG-Format

Die Stärke von Vektorgrafiken liegt bei Darstellungen, die als Zusammenstellung von grafischen Primitiven befriedigend beschrieben werden können, zum Beispiel Diagramme oder Firmenlogos. Sie sind nicht geeignet für gescannte Bilder und Digitalfotos, die naturgemäß als Rastergrafik erfasst werden und nicht verlustfrei umgewandelt werden können. Ebenfalls an die Grenzen stoßen Vektorformate bei komplexen berechneten Bildern, die ebenfalls direkt als Rastergrafik berechnet werden. Allerdings spezialisieren sich mehr und mehr Firmen auf die Vektorisierung von Rastergrafiken. Dies ist vor allem von Interesse für großflächige Bildwerbung, Fahrzeugbeschriftung oder wenn die Vektorisierung als grafischer Effekt genutzt wird.

Grafikanwendungen

Zur Erstellung von Illustrationen, insbesondere für die Erstellung von Logos, können vektorbasierte Zeichenprogramme verwendet werden. Für technische Zeichnungen finden CAD-Programme Verwendung; hier wird meist das Drawing Interchange Format (DXF) zur Speicherung verwendet, das in Zukunft in weiten Teilen auch von SVG abgelöst werden soll.

Seitenbeschreibungssprachen

Vektorgrafiken erlauben es, Dokumente unabhängig von der Auflösung des Ausgabegeräts zu beschreiben. Mit Hilfe einer vektorgrafikfähigen Seitenbeschreibungssprache wie PostScript oder dem daraus hervorgegangenen Portable Document Format (PDF) können Dokumente, im Gegensatz zu Rastergrafiken, mit der jeweils höchstmöglichen Auflösung auf Bildschirmen verlustfrei dargestellt oder gedruckt werden.

Computerschriften

Auf gängigen Computersystemen finden heute überwiegend so genannte Outline-Schriften Verwendung, die die Umrisse jedes Zeichens als Vektorgrafik beschreiben. Wichtige Formate sind TrueType, PostScript und OpenType. SVG bringt zusätzlich ein eigenes Fontformat mit (SVG-Fonts) dazu später mehr.

Computerspiele

Frühe Arcade-Spiele liefen mit Vektorgrafik. Das erste war Space Wars aus dem Jahr 1977. Bekannt waren auch Lunar Lander (Atari, 1979) und Star-Wars-Spiele[9]. Es gab auch die Spielkonsole Vectrex. Der Vorteil lag in der höheren Geschwindigkeit im Vergleich zur Rastergrafik. Auch 3D-Spiele bestehen in weiten Teilen aus Vektoren, die meist mit Rastertexturen "überzogen" werden. 2D-Vektorgrafiken können sehr gut als Grundlage für 3D-Modelle verwendet werden, z.B. lassen sich SVG-Pfade mit Blender[10] importieren.

Internet

Im World Wide Web liegen Vektorgrafiken meist im offenen Format SVG oder als proprietäre SWF-Dateien (Adobe Flash) vor.

Geoinformationssysteme

Bei Geoinformationssystemen (GIS) kann die Geometrie von Flurstücken und Landkarten in Form von Vektordaten gespeichert werden. Solche Vektorgrafiken lassen sich vergleichsweise einfach mit Sachdaten verknüpfen.

Wer mehr Informationen über die Anwendungsmöglichkeiten von SVG in der Geoinformation wissen möchte, ist bei cartoo.net[11] richtig.

1.8 Darstellungsprogramme

Zur Darstellung von SVG gibt es zahlreiche Programme. Auch technisch aktuelle Darstellungsprogramme, die eine graphische/visuelle Präsentation von Inhalten erlauben, haben heutzutage (2010) auch eine Anzeigemöglichkeit für SVG.

9 http://de.wikibooks.org/wiki/Star%20Wars%20%28Farb-Vektorgrafik%29
10 http://de.wikibooks.org/wiki/Blender
11 http://www.carto.net/papers/svg/

Beispiele für häufig verwendete Programme zur Darstellung und Interpretation, welche unter anderem SVG verstehen, sind:

- Presto (Opera)
- Mozilla, Gecko (Firefox, SeaMonkey, Iceweasel; Iceape, Epiphany, Galeon ...)
- WebKit (Apple Safari, Arora, Konqueror4+WebKit, Google Chrome ...)
- KHTML+KSVG (Konqueror3/4+KSVG)
- Trident (Microsoft Internet Explorer ab Version 9, Vorschauversion verfügbar seit März 2010)
- Amaya

Bei älteren Versionen von Darstellungsprogrammen kann vor allem eine Aktualisierung oder ein Wechsel zu einem aktuelleren Programm sinnvoll sein.

1.9 Darstellungsprogramme für Mobile Geräte

Dann gibt es auch spezielle, zum großen Teil kommerzielle Programme für Mobiltelephone, die eine Anzeige besonders auch von SVG tiny 1.2 ermöglichen, Anbieter sind zum Beispiel:

- Opera mini
- BitFlash
- Ikivo
- Abbra
- eSVG viewer (Intesis)
- Streamezzo

1.10 Erweiterungen

Ferner gibt es auch für ältere Darstellungsprogramme die Möglichkeit, Erweiterungen zu installieren, mit denen SVG interpretiert werden kann. Bekanntere Beispiele sind:

- Adobes SVG-Erweiterung
- KSVG (für den Konqueror, siehe oben)
- Renesis SVG Player
- GPAC

1.11 Konvertierung in andere Formate

Es gibt auch Ideen, SVG in andere Formate zu konvertieren, was aber zumeist durch die begrenzten Möglichkeiten dieser Formate nur eingeschränkt möglich ist.

Verwendet ein Autor solche Skripte, um die Inhalte auch Nutzern zugänglich zu machen, die keine Darstellungsmöglichkeit für SVG haben, besteht leicht die Möglichkeit, dass versehentlich auch die originale Anzeige für jene Nutzer unterbunden wird, bei denen eine Anzeige möglich ist. Die Strategie erfordert also eine Menge Tests mit diversen Programmen und Voreinstellungen derselben.

1.12 Spezielle Programme zur Darstellung und Bearbeitung von SVG

Es gibt auch zahlreiche Programme, die mehr auf die Darstellung von Graphik oder auch nur SVG spezialisiert sind, zum Beispiel

- Apache Batik/Squiggle (Darstellung, Formatkonvertierung und Einsatz auf Web-Servern)
- Inkscape[12] (Editor mit graphischer Oberfläche)
- Sodipodi (Editor mit graphischer Oberfläche, die Entwicklung wurde 2005 eingestellt)
- Amaya (siehe unten, Entwicklungsumgebungen)

1.13 Implementierungsstand

Oft werden in den Programmen nur Teile bestimmter Profile interpretiert, die auch noch von Programm zu Programm unterschiedlich sind. Die Implementierungen sind unterschiedlich weit fortgeschritten. Ob eine sinnvolle Anzeige zustande kommt bei einem Programm, welches die zu erwartende Zielgruppe verwenden wird, sollte also vom Autor geprüft werden. Allerdings ist es aufgrund der großen Zahl von verschiedenen Darstellungsprogrammen wenig sinnvoll, für eines zu optimieren oder gar Fehler einer aktuellen Version eines Programmes auszunutzen, weil dies dann zu Unsinn in neueren Versionen oder anderen Programmen führen wird. Somit sollten die Dokumente also jedenfalls immer technisch korrekt sein und zugleich einfach genug für die Programme der Zielgruppe. Das impliziert zumeist, dass Autoren mit dem Fortschreiten der Implementierungen auch technisch bessere Versionen von Dokumenten anbieten können, statt in alten Kompromissen zu verharren.

1.14 Installation

Die Installation hängt vom konkreten Programm ab und von dem Betriebssystem, auf dem es installiert werden soll. Die Anbieter sollten entsprechende Instruktionen bereit halten. Es sind auch nicht alle Programme für alle Betriebssysteme verfügbar und auch nicht für alle Abspielgeräte. Meist gibt es etwa vorinstallierte oder spezielle Versionen auf Mobiltelephonen.

1.15 OpenOffice.org

Das von OpenOffice verwendete Dateiformat ODF (open document format) ist selbst ein Format der XML-Familien und verwendet von verschiedenen Standards, so auch von SVG und XHTML, einige Fragmente. Eine komplette Übernahme von SVG-Profilen ist im Gespräch, ist aber derzeit (2010) noch nicht umgesetzt.

12 `http://de.wikibooks.org/wiki/Inkscape`

1.16 Graphviz

Graphviz kann als Ausgabe auch diverse Graphikformate verwenden. Unterstützt werden neben SVG unter anderem die Dateiformate Postscript, JPEG/JFIF, PNG und PDF.

1.17 Entwicklungsumgebungen

1.17.1 Texteditor

Als hervorragende Entwicklungsumgebung für SVG erweist sich ein einfacher Texteditor, vorzugsweise einer mit einer (farblichen) Hervorhebung von Strukturen und Elementen ganz allgemein für XML-Dokumente. Von solchen Editoren gibt es viele verschiedene. Die Namen sind meist abhängig vom aktuell verwendeten Betriebssystem.

1.17.2 Amaya

Amaya ist eine Kombination von Darstellungsprogramm und Editor des W3C, welcher verschiedene Formate interpretieren und editieren kann, etwa HTML, XHTML, SVG, MathML, CSS, RDF, XLink, XPointer. Die XML-Formate lassen sich darin auch miteinander kombinieren. Als Editor bietet Amaya auch eine Überprüfung von Dokumenten anhand ihm bekannter Dokumenttypdeklarationen.

1.17.3 Netbeans

Netbeans von SUN unterstützt die SVG-Entwicklung inklusive Testhilfsprogramm und Vorschau für das SVG-Mobile-Profil, weiterhin bietet Netbeans eine hervorragende JavaScript-Unterstützung.

1.17.4 Eclipse

Für Eclipse gibt es ausgezeichnete XML-Entwicklungswerkzeuge, Unterstützung und Integration der Darstellungsprogramm-Vorschau mit Fehlersuche (über Aptana Studio), weiterhin gibt es eine (zurzeit veraltete) Batik-Erweiterung.

1.17.5 emacs

Emacs ist ein freier Editor und eine allgemeine Entwicklungsumgebung. Es gibt ein Modul für diesen Editor mit SVG-Vorschaufunktion. Zwischen Textmodus und Vorschau kann dann mit strg-c strg-c gewechselt werden.

1.17.6 Firebug

Für kleinere Korrekturen, um Fehler zu finden und als praktische Methode, um direkt zum Quelltext eines Elements zu gelangen, eignet sich eine Erweiterung für das Darstellungsprogramm Firefox namens Firebug.

1.18 Kurze Einführung in XML

Soweit es von Belang ist, soll hier kurz auf die prinzipielle Sprachkonstruktion eines XML-Formates eingegangen werden (XML - erweiterbare Auszeichnungssprache, englisch: eXtensible Markup Language).

Bereits vor der Jahrtausendwende hat sich am Beispiel von HTML (Versionen 1 bis 4, Hypertextauszeichnungssprache, englisch: HyperText Markup Language) gezeigt, dass diese auf SGML (englisch: Standard Generalized Markup Language) basierende Sprache derart komplex ist, dass es bis zum heutigen Tage kein Darstellungsprogramm geschafft hat, diese komplett zu interpretieren. Auch die offengelassene Fehlerbehandlung hat letztlich zusammen mit der unvollständigen oder fehlerhaften Interpretation der Darstellungsprogramme dazu geführt, dass dieses Format gewissermaßen korrumpiert ist, gleichzeitig aber durch die häufige Verwendung im Internet zu einem der wichtigsten Formate überhaupt geworden ist. Aufgrund der Komplexität und der zahlreichen Probleme der Programme schien indes eine Erweiterung nur schwierig möglich.

Mit XML wurde daher eine Sprachfamilie konzipiert, die eine einfachere Syntax hat und gleichzeitig die Möglichkeit der Erweiterbarkeit bietet. Zusätzlich hat es zumindest hinsichtlich grober Strukturfehler eine definierte Fehlerbehandlung, die Darstellungsprogramme dazu bringt, Strukturfehler anzuzeigen, am besten noch bevor ein Autor fehlerhafte Dokumente veröffentlicht.

Die Grundidee von XML ist vorzugeben, wie eine Sprache dieser Sprachfamilie zu spezifizieren ist, statt die Sprache selbst zu spezifizieren. Gemeinsam vorgegeben ist allen XML-Formaten die Konstruktion mit Elementen und Attributen.

Ein XML-Dokument besteht aus Klartext und beginnt generell mit einer XML-Verarbeitungsanweisung, die dem Darstellungsprogramm kenntlich macht, dass es sich um XML handelt und gegebenenfalls auch, wie das Dokument kodiert ist und ob es Abhängigkeiten zu anderen Dokumenten gibt.

Als nächstes können eine oder mehrere XML-Stilvorlagenverarbeitungsanweisungen folgen. Diese enthalten Verweise zu Stilvorlagen, mit denen die Darstellung des Dokumentes beeinflusst werden kann. Beispiele für Sprachen für Stilvorlagen sind CSS oder XSL (CSS - kaskadierende Stilvorlagen, englisch: Cascading StyleSheets, XSL - erweiterbare Stilvorlagensprache, englisch: eXtensible Stylesheet Language).

Darauf kann ein sogenannter 'doctype' folgen. Dies ist eine Dokumenttypdeklaration, die angibt, welches das Hauptelement des Dokumentes ist und entweder direkt definiert, welche Elemente auftreten können, oder auch auf eine Quelle verweist, in der dies formal beschrieben ist. Diese Beschreibung ist in einer Schemasprache maschinenlesbar angelegt. Weil die per Dokumenttypdeklaration verwendbare Schemasprache, welche selbst kein XML ist, deutliche Grenzen darin aufweist, was man wie genau über die zu definierende Sprache aussagen kann, gibt es auch einige andere Schemasprachen mit mehr Möglichkeiten. Da diese immer noch sehr beschränkt sind, werden neuere

Formate auch zunehmend nur noch oder vorrangig mit einem Prosa-Dokument beschrieben. Dann kann die Dokumenttypdeklaration entfallen (wie etwa in SVG tiny 1.2). Die Dokumenttypdeklaration kann aber dennoch zusätzlich genutzt werden, um Abkürzungen anzugeben (sogenannte Entitäten, englisch: entities).

Auf den 'doctype' folgt das Hauptelement des Dokumentes, welches alle weiteren Elemente des Dokumentes enthält. Es gibt genau ein Hauptelement pro Dokument mit dieser ausgezeichneten Position im Quelltext (englisch: root). Je nach verwendeter Sprache kann das für das Hauptelement verwendete Element auch mehrmals im Dokument auftauchen. Dies kann zum Beispiel in SVG mit dem Element ***svg*** passieren.

Zumeist wird die Sprache oder das Format einem Namensraum zugeordnet. Der Namensraum kann mit dem Attribut *xmlns* angegeben werden. Dies schließt auch eine Notation mit einem Präfix mit ein, was hilfreich ist, wenn ein Dokument Elemente oder Attribute aus verschiedenen Namensräumen enthält. Das ist dann ein gemischtes Dokument oder ein Dokument, welches mehrere XML-Formate enthält (englisch auch compound-document).

Als Beispiel sei hier ein einfaches, nahezu leeres SVG-Dokument angegeben:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="200px" height="200px" viewBox="0 0 200 200"
  xmlns="http://www.w3.org/2000/svg" version="1.1" baseProfile="tiny"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xml:lang="de">

<title>Beispiel für ein SVG-Dokument</title>
<desc>SVG-Beispiel eines Grundgerüstes baseProfile tiny</desc>
</svg>
```

Es beginnt also mit der XML-Verarbeitungsanweisung, welche angibt, dass die Version 1.0 von XML verwendet wird, ferner die Kodierung iso-8859-1. Dann folgt die Dokumenttypdeklaration. Das Hauptelement ist bei SVG-Dokumenten das Element ***svg***, dies hat Attribute, hier *width*, *height*, *viewBox*, *version*, *baseProfile*. Dazu kommt noch ein Attribut *xml:lang*, welches die im Dokument verwendete Sprache festlegt. Das Präfix (Vorsilbe) 'xml' gibt an, dass dieses Attribut ganz allgemein für alle XML-Formate bereits in der XML-Spezifikation definiert ist (siehe unten), also selbst ohne Kenntnis des Einzelformates eine bekannte Bedeutung hat. xmlns="http://www.w3.org/2000/svg" ist die Angabe zum Namensraum von SVG. Da hier kein Präfix angegeben ist, bedeutet dies, dass das Element ***svg*** selbst, seine Attribute und Kindelemente zu diesem Namensraum gehören, sofern es keine anderen Angaben gibt. Mittels xmlns:xlink="http://www.w3.org/1999/xlink" wird ein anderer Namensraum für die Sprache XLink angegeben (eine Sprache zur Auszeichnung von Verweisen; englisch: XML Linking Language), hier mit dem Präfix 'xlink'. Im Dokument können also Elemente oder Attribute aus diesem Namensraum auftauchen, die werden dann mit diesem Präfix angegeben. In SVG taucht besonders häufig *xlink:href* auf, um ein Beispiel für ein Attribut zu nennen (die Sprache XLink hat nur Attribute, keine Elemente).

Im Hauptelement ***svg*** sind die Kindelemente ***title*** und ***desc*** notiert, die in diesem Falle keine Attribute angegeben haben und nur Text als Inhalt enthalten.

1.19 Verarbeitungsanweisungen

Eine XML-Verarbeitungsanweisung sagt einem Darstellungsprogramm, dass es sich bei dem vorliegenden Dokument um ein XML-Format handelt, es also nach den Regeln von XML zu verarbeiten ist.

Oft kommt darin auch eine Angabe zur Kodierung vor. Ein offensichtliches Problem dabei ist, dass bis zu der Stelle, wo die Angabe erfolgt, das Dokument schon dekodiert sein muss, um die Angabe berücksichtigen zu können. Das typische Vorgehen besteht darin, dass das Programm erstmal eine Kodierung rät und guckt, ob damit der Dokumentinhalt einen Sinn ergibt. Wird so die erste Zeile eines Dokumentes dekodiert, in der die XML-Verarbeitungsanweisung steht, kann gegebenenfalls die Dekodierung noch angepasst werden. Das liegt daran, dass die Angaben in der XML-Verarbeitungsanweisung so ausgelegt sind, dass sie bei gängigen Kodierungen gleich ausfallen, also keine kritischen Sonderzeichen enthalten. Liegt keine Angabe vor, wird eine Kodierung als UTF-8 angenommen. Ein anderes Vorgehen kann sich ergeben, wenn das Dokument von einem Server kommt. Dieser kann verbindlich die Kodierung des Dokumentes angeben, wonach sich ein Darstellungsprogramm zu richten hat. Das vermeidet das Problem, dass die Datei bereits dekodiert sein muss, bevor man die Angabe zur Kodierung erhält. Gibt der Server eine Kodierung an, so wird die Angabe im Dokument ignoriert.

XML-Stilvorlagenverarbeitungsanweisungen können auf die Verarbeitungsanweisung folgen und referenzieren dann externe Dateien, in denen Stilvorlagen notiert sind.

1.19.1 Typische Beispiele

Nur eine Angabe, dass nach XML1.0 zu interpretieren ist. Sofern keine anderen Angaben gemacht wurden, wird UTF-8 als Kodierung unterstellt:

```
<?xml version="1.0" ?>
```

Zusätzliche Angabe der Kodierung als iso-8859-1 (das reicht für Umlaute, ß-Ligatur und einige andere Sonderzeichen aus dem europäischen Sprachraum):

```
<?xml version="1.0" encoding="iso-8859-1" ?>
```

Beziehungsweise wenn man das Euro-Zeichen braucht, kann auch iso-8859-15 verwendet werden:

```
<?xml version="1.0" encoding="iso-8859-15" ?>
```

Eine Kodierung mit UTF-8 eignet sich für die meisten Sprachen und kann insbesondere empfohlen werden, wenn neu angefangen wird und keine Altlasten mit anderer Kodierung bereits vorhanden sind.

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Zu beachten ist, dass einige Editoren bei UTF-8 an den Beginn einer Datei bisweilen ein BOM setzen (Byte-Anordnungsmarkierung, englisch: byte order mark). Das ist für UTF-8 nicht unbedingt notwendig, weil dort die Anordnung der Bytes eindeutig ist. Dies kann aber mit der Forderung kollidieren, dass zu Beginn des Dokumentes die XML-Verarbeitungsanweisung stehen soll. Entwicklern einiger Programme ist das Problem bekannt und sie umgehen es, indem sie das vorangestellte BOM

hinsichtlich der Forderung ignorieren, dass die XML-Anweisung zu Beginn des Dokumentes stehen soll. Bei anderen Programmen oder älteren Versionen kann es hingegen passieren, dass versucht wird, das BOM zur Anzeige zu bringen, statt ein XML-Dokument zu verarbeiten. Deswegen ist es in der Praxis empfehlenswert, dem Editor nicht zu erlauben, ein BOM an den Anfang einer mit UTF-8 kodierten XML-Datei zu setzen.

Folgendes ist ein Beispiel mit XML-Stilvorlagenverarbeitungsanweisungen. Hier werden CSS-Dateien referenziert. Zudem gibt es drei verschiedene Vorlagen. Das Darstellungsprogramm sollte einen Auswahlmechanismus bereitstellen, um zwischen den drei Möglichkeiten zu wechseln (ist zum Beispiel mit Mozilla-Geckos oder Opera auch nachvollziehbar).

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet href="css1.css" type="text/css" title="Stil 1"?>
<?xml-stylesheet href="css2.css" type="text/css" title="Stil 2"
   alternate="yes" ?>
<?xml-stylesheet href="css0.css" type="text/css" title="Kein CSS"
   alternate="yes" ?>
```

Mit dem Attribut *href* wird die externe Stilvorlage referenziert. Die Angabe des Attributes ist erforderlich. Der Attributwert ist eine absolute oder relative URI der Datei mit der Stilvorlage, hier einfach der Dateiname einer CSS-Datei im gleichen Verzeichnis. Es kann auch ein Fragmentidentifizierer sein, der dann eben auf ein Dokumentfragment verweist, welches die Stilvorlage enthält.

Mit dem Attribut *type* wird angegeben, welches Stilvorlagenformat verwendet wird, hier also CSS. Der Wert ist ein Inhaltstyp (ehemals MIME-Typ). Die Angabe des Attributes ist erforderlich.

Mit dem Attribut *title* kann ein Titel oder eine Überschrift angegeben werden. Die Angabe des Attributes ist nicht erforderlich, sie wird jedoch immer benötigt, wenn mehrere Stilvorlagen alternativ angeboten werden. Das Attribut ist also insbesondere relevant, wenn das Darstellungsprogramm mehrere alternative Stilvorlagen zur Auswahl stellen soll, der Attributwert wird dann als Menüeintrag der Auswahl verwendet.

Daß Stilvorlagen alternativ und nicht alle gleichzeitig angewendet werden sollen, wird dann mit dem Attribut *alternate* angegeben. Die erste Angabe im obigen Beispiel ist ohne *alternate*, was dann bedeutet, das dies die Voreinstellung ist. Der Attributwert 'yes' gibt eben an, dass es sich um eine alternative Stilvorlage handelt. 'no' täte angeben, dass es zusätzlich zur Voreinstellung anzuwenden ist. Eine Angabe des Attributes ist nicht erforderlich, Voreinstellung ist 'no'.

Mit dem Attribut *charset* kann ein Hinweis auf die Kodierung des referenzierten Dokumentes angegeben werden. Eine Angabe des Attributes ist nicht erforderlich. Die möglichen Werte entsprechen denen von *encoding*. Der Hinweis ist allerdings nur relevant, wenn es keine anderen bindenden Informationen gibt, zum Beispiel von einem server, der das referenzierte Dokument ausliefert, oder im Dokument selbst.

Mit dem Attribut *media* kann angegeben werden, für welches Ausgabemedium die Stilvorlage gedacht ist. Eine Angabe des Attributes ist nicht erforderlich, Voreinstellung ist in CSS 'screen'. Der Attributwert ist ein Medientyp oder eine Liste von Medientypen, die jeweils mit einem Komma voneinander separiert sind. Welche Medientypen verfügbar sind, hängt von der verwendete Stilvorlagensprache ab. CSS2 gibt zum Beispiel folgende Möglichkeiten an:

all

Für alle Ausgabemedien geeignet.

aural

Für eine akustische Wiedergabe vorgesehen.

braille

Für Braille-Geräte vorgesehen.

embossed

Für Braille-Seitendrucker vorgesehen.

handheld

Für Handheld-Geräte vorgesehen (normalerweise mit kleinem Schwarzweißbildschirm und begrenzter Bandbreite).

print

Für Drucker vorgesehen, ebenso für den Druckvorschaumodus auf dem Bildschirm.

projection

Für projizierte Präsentationen vorgesehen, wie zum Beispiel Projektoren oder den Ausdruck auf Folien.

screen

Hauptsächlich für normale Farbmonitore vorgesehen.

tty

Für Medien vorgesehen, die ein festes Zeichenraster verwenden, wie beispielsweise Fernschreiber oder Terminals.

tv

Für fernsehartige Geräte vorgesehen (geringe Auflösung, begrenzt rollbare Bildschirme, aber in Farbe und mit Ton).

Für spätere Versionen von CSS sind weitere Typen und Schreibweisen vorgesehen, insbesondere auch, weil es inzwischen eine größere Anzahl von verschiedenen Geräten gibt, zum Beispiel weichen bei notebooks oder Mobiltelephonen die typischen Abmessungen von Höhe und Breite erheblich von denen von früheren Monitoren ab (auch im Seitenverhältnis).

1.20 Dokumenttypdeklaration und Dokumenttyp-Definition

Die Dokumenttypdeklaration kann hilfreich sein, wenn mit einem Validator untersucht werden soll, ob die Struktur eines Dokumentes korrekt ist. Ein Validator ist ein spezielles Programm, welches derartige Untersuchungen der Dokumentstruktur durchführt. Kennt der Validator das Format oder kann die entsprechende Datei mit Strukturinformationen herunterladen, kann er untersuchen, ob das Dokument die Verschachtelungsregeln und einige andere relevante Dinge einhält, die in der Dokumenttyp-Definition (DTD) angegeben sind, welche mit der Deklaration im Dokument zugeordnet wird. Weil man in der gängigen Notation als DTD längst nicht alles angeben kann, was hilfreich wäre, wird darauf auch bei neueren oder komplexeren Formaten verzichtet. In SVG etwa

kann mittels der DTD nicht überprüft werden, ob ein Autor Pfade oder Animationswerte korrekt notiert hat, weil die DTD solche Feinheiten nicht hergibt. Dem Validator fallen solche Fehler nicht auf. Andersherum können Beschränkungen in dem, was in einer Dokumenttypdefinition angegeben werden kann, dazu führen, dass der Validator Fehler anzeigt, die keine sind, weil in der Spezifikation etwas anderes angegeben ist, dies aber in solcher Allgemeinheit nicht in der Dokumenttypdeklaration angegeben werden kann.

Relevant ist bei einer konkreten Angabe einer Dokumenttypdeklaration weniger, genau zu verstehen, was da warum angegeben ist, sondern die Deklaration exakt so hinzuschreiben, wie es spezifiziert ist. Nur dann kann der Dokumenttyp von einem Programm erkannt werden. Als Autor ist nicht mehr zu tun. Die Entwickler der Formate haben sich zu überlegen, was wie deklariert wird.

Während ein Programm jedes XML-Dokument daraufhin prüfen kann, ob es korrekt nach den XML-Regeln erstellt ist (Wohlgeformtheit), braucht es ein solches für die Sprache spezifisches Schema wie eine DTD, um zu erkennen, ob die Bedingungen des Formates erfüllt sind. Ob also die Elemente und Attribute richtig verwendet werden und richtig ineinander verschachtelt sind. Letzteres wäre dann also eine Prüfung auf Validität des jeweiligen Formates.

Bei SVG 1.1 dient die Deklaration auch der Angaben der Version und des Profiles der Sprache. Alternativ kann das dort auch mit Attributen getan werden, letzteres ist dann die in SVG tiny 1.2 mögliche Methode, um eindeutig festlegen zu können, welche Version und welches Profil verwendet wird.

In der Praxis sind Angaben zur Version hilfreicher für den Autor oder spätere Leser des Quelltextes als für ein Programm, welches das Dokument interpretiert. Das liegt daran, dass aktuell real existierende Programme einfach nicht so ausgelegt sind, dass sie für verschiedene Versionen verschiedene Interpretationen implementiert haben. Entwickler von Formaten versuchen daher nach Möglichkeit, Inkompatibilitäten zu vermeiden, was aber nicht immer komplett möglich ist. Autoren können anhand der Versionsangaben bei Unstimmigkeiten gucken, wie das Dokument eigentlich gedacht war und es dann im Bedarfsfalle vorsichtig an eine neue Version anpassen. Bei Dokumenten ohne Versionsangabe wird in der Regel die aktuelle Version des Formates als relevant angenommen. Das hat zur Konsequenz, dass sich bei einer inkompatiblen Änderung der Spezifikation die Bedeutung von solchen Dokumenten ändern können. Bei Dokumenten mit relevantem, kritischem Inhalt ist also immer zu empfehlen, eine eindeutige Angabe zur verwendeten Sprachversion anzugeben. Der Autor kann nicht vorhersagen, wie zukünftige Versionen des Formates aussehen werden.

1.20.1 Typische Beispiele

SVG tiny 1.1:

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
```

SVG 1.1:

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
```

In SVG tiny 1.2 ist kein doctype mehr vorgesehen. Ein solcher kann aber für Erweiterungen weiterhin genutzt werden. Hier wird die Erweiterung (ENTITY) 'Pfad' definiert, eine Abkürzung für das, was hinter 'Pfad' in Anführungsstrichen angegeben ist. Man kann da auch ganze Elemente oder Gruppen

von Elementen angeben, die dann einfach wiederverwendet werden können, im Dokument kann die Entität 'Pfad' dann mittels &Pfad; verwendet werden. Diese Zeichenkette wird durch die für 'Pfad' definierte ersetzt, bevor das Dokument interpretiert wird:

```
<!DOCTYPE svg
   [<!ENTITY Pfad "M200,300 L300,100">
   ]
>
```

In der Zeichenkette dürfen natürlich nur Zeichen stehen, die nicht mit der Syntax im Konflikt stehen, hier also keine doppelten Anführungszeichen.

Das Analogon entsprechend für SVG 1.1:

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"
   [<!ENTITY Pfad "M200,300 L300,100">
   ]
>
```

1.21 Namensräume

Namensräume sind eine Konstruktion in XML, mit der Überschneidungen von Bedeutungen von Elementen von verschiedenen Sprachen vermieden werden können. Etwa hat das Element ***font*** in SVG eine andere Bedeutung als in XHTML. Das Element ***title*** kommt in verschiedenen Formaten meist mit ähnlicher Bedeutung vor, wo es aber auftreten kann, kann von Format zu Format sehr unterschiedlich sein. Ein Darstellungsprogramm muss nun bei einem Dokument wissen, dessen Inhalt aus verschiedenen Formaten zusammengemischt wurde, aus welchem Format das jeweilige Element oder Attribut kommt. Dazu dient die eindeutige Zuordnung zu einem Namensraum. Das löst wiederum nicht das mögliche Problem, dass bei verschiedenen Versionen eines Formates die Elemente oder Attribute leicht unterschiedliche Bedeutungen haben können. Um die Version eines Formates zu kennzeichnen, sind besondere Mechanismen im jeweiligen Format zu definieren.

Typisch wird eine URI als Namensraum verwendet, damit die Zuordnung eindeutig ist. Andere eindeutige Zeichenketten sind aber nicht ausgeschlossen. Bei einer URI ist es meist üblich, dass man dort zumindest Angaben darüber findet, um was für ein Format es sich handelt.

Häufig auftauchende Formate und Namensräume:

- SVG: http://www.w3.org/2000/svg
- XLink: http://www.w3.org/1999/xlink
- XHTML: http://www.w3.org/1999/xhtml
- MathML: http://www.w3.org/1998/Math/MathML
- RDF: http://www.w3.org/1999/02/22-rdf-syntax-ns#
- Dublin Core, DCMI: http://purl.org/dc/elements/1.1/

Der Namensraum wird mit dem in XML selbst definierten Attribut *xmlns* angegeben, Beispiel siehe oben. Wird ein Präfix verwendet, ist dies den Elementnamen und Attributnamen jeweils voranzustellen. Der Autor selbst legt das Präfix fest. Hinsichtlich XLink in SVG ist allerdings das Präfix 'xlink' dringend zu empfehlen, weil einige Darstellungsprogramme Mängel in der Interpretation

von Namensräumen aufweisen und dieses Präfix wegen seiner Erwähnung in der DTD von SVG 1.1 durchgehend getestet ist.

Ist bei einem Element kein Namensraum angegeben, so gehört es zum Namensraum des Elternelementes, sofern dies zu einem Namensraum gehört. Typisch wird etwa bei Dokumenten, die nur ein Format verwenden, der Namensraum im Hauptelement angegeben, womit die Angabe dann für alle Kindelemente gilt, die keine eigenen Angaben zum Namensraum machen. Bei Attributen ist die Lage formal etwas komplizierter, weil sie nicht direkt zu einem Namensraum gehören, wenn für sie mittels Präfix keiner angegeben ist. Allerdings werden die Attribute dann in dem Namensraum definiert und interpretiert, der zu dem Element gehört, in dem sie notiert sind. Etwa bei der deklarativen Animation von SVG kann es notwendig sein, dass Attributnamen samt Namensraumzugehörigkeit als Werte anderer Attribute anzugeben sind. In solchen Fällen ist in dem jeweiligen Format, hier also SVG, festgelegt, wie dabei Präfixe zu interpretieren sind und die Attribute zu identifizieren sind.

1.22 Elemente und Attribute

Das prinzipielle Konzept von XML-Formaten liegt darin, Inhalte auszuzeichnen. Dazu werden Konstruktionen verwendet, die Funktion oder Semantik des Inhaltes beschreiben. Zum Beispiel gibt es in SVG ein Element, welches ***rect*** heißt. Dies beschreibt ein Rechteck, gegebenenfalls mit abgerundeten Ecken. Generell sind damit alle Arten von Rechtecke gemeint. Nun gibt es aber verschiedene Arten von Rechtecken, die sich insbesondere in Breite und Höhe unterscheiden und in der Positionierung, auch wie stark die Ecken abgerundet sind. Dies wird mit Attributen genauer festgelegt. Auch die Erscheinung oder Präsentation, etwa die Farbe, kann genauer festgelegt werden. Um nun zu verstehen, was genau gemeint ist, wenn jemand sagt oder schreibt, es soll ein ***rect***-Element verwendet werden - oder irgendein anderes Element - kommt es darauf an, die Bestandteile eines solchen Elementes genau bezeichnen zu können.

Dazu ein willkürliches, ausgedachtes Beispiel, einen Apfel, repräsentiert durch das Element ***apfel***:

```
<apfel sorte="Boskop" zustand='reif'><kern
 /><fruchtfleisch>lecker</fruchtfleisch></apfel>
```

Dabei ist nun dies das gesamte Element (englisch: element). 'apfel' ist der Name des Elementes, Groß- und Kleinschreibung ist signifikant. ***Apfel*** und ***apfel*** können also zwei komplett verschiedene Elemente sein.

Folgendes ist die Anfangsmarkierung (englisch: begin tag). Diese enthält ferner die Attribute (englisch: attribute, Mehrzahl: attributes) *sorte* und *zustand*:

```
<apfel sorte="Boskop" zustand='reif'>
```

Ein Element hat immer eine entsprechende Endmarkierung (englisch: end tag), die keine Attribute enthalten kann, hier:

```
</apfel>
```

Achtung, aufgrund mangelnden Verständnisses, einer gewissen Sprachkonfusion und einer Tendenz besonders in Deutschland, hinter jedem englischen Begriff gleich ein Fachwort zu vermuten, bezeichnen einige Menschen fälschlich ein Element als 'tag' oder noch schlimmer 'Tag', Mehrzahl auch 'tags'. Diese Buchstabenkombination hat allerdings im Deutschen eine komplett andere Bedeutung

(Tagesabschnitt, wo die Sonne über dem Horizont steht, im Gegensatz zu 'Nacht', oder aber auch ein Zeitraum von 24 Stunden, tags auch im Sinne von 'am Tage' oder bei 'tags drauf' eben 'am nächsten Tag'). Das ist so falsch und wird auch im englischen Sprachraum von kundigen Fachleuten nicht so verwendet. 'tag' ist einfach auf deutsch eine Markierung oder Kennzeichnung und hat nicht den Anspruch eines Fachwortes. Die Fachbezeichnung ist auf englisch wie auf deutsch jedenfalls (mal abgesehen von Groß- und Kleinschreibung) die gleiche - Element. Anfang und Ende davon werden markiert. Dazu dienen Markierungen, Anfangs- und Endmarkierung, die jeweils den Elementnamen enthalten. Die Markierungen sind aber nicht das Element selbst, ebenso wenig wie Anfang und Ende von einem Würstchen das Würstchen selbst sind.

Attribute haben immer einen Wert. Es werden also getrennt mit Leerzeichen in der Anfangsmarkierung die Attributnamen angegeben, wobei jeweils der Wert oder Attributwert nach einem Gleichheitszeichen in Anführungszeichen folgt. Das sind entweder einfache oder doppelte Anführungszeichen. Enthält der Attributwert einfache Anführungszeichen, sind doppelte zu verwenden und umgekehrt. Taucht beides im Attributwert auf, so sind Anführungszeichen dort am besten zu maskieren, dafür hat XML spezielle Abkürzungen eingeführt (siehe unten). Hier ist der Wert des Attributes *sorte* eben 'Boskop' und der von *zustand* 'reif'.

In obigem Beispiel ist dann

```
<kern /><fruchtfleisch>lecker</fruchtfleisch>
```

der Inhalt von ***apfel***. Das sind zwei weitere Elemente ***kern*** und ***fruchtfleisch***. Beide haben keine Attribute und ***kern*** hat auch keinen Inhalt. Das ist ein besonderer Fall in XML, ein sogenanntes inhaltsleeres Element (englisch: empty element). Entweder folgt dann die Endmarkierung direkt auf die Anfangsmarkierung

```
<kern></kern>
```

oder es wird wie im Beispiel eine spezielle Kurzschreibweise verwendet:

```
<kern />
```

Das Element ***fruchtfleisch*** enthält als Inhalt nur den Text 'lecker'. Reiner Text wird in XML auch 'CDATA' genannt, der wird nicht mehr interpretiert. Text, der PCDATA genannt wird, wird interpretiert, also darin vorkommende Entitäten oder auch Markierungen werden interpretiert.

In unserem Falle sind also ***kern*** und ***fruchtfleisch*** Kindelemente von ***apfel***. Sie sind in ***apfel*** verschachtelt, stehen also zwischen Anfangs- und Endmarkierung von ***apfel***, welches dann das Elternelement ist.

Was der Inhalt eines Elementes sein darf, ist in der jeweiligen Spezifikation genau angegeben und kann - sofern vorhanden - mittels eines Schemas wie einer DTD auch formal überprüft werden.

Durch das Verschachteln der Elemente und das Zusammenspiel mit den Attributen entsteht so ein strukturiertes Dokument. An dem Beispiel mit dem Apfel ist bereits zu erkennen, dass damit maschinenlesbar und -verstehbar sehr genaue Angaben darüber gemacht werden können, worum es geht. Elemente, Attribute und zumindest festgelegte Attributwerte sind in dem Sinne einem Programm verständlicher Inhalt, der auch hinsichtlich der Zugänglichkeit so vermittelbar ist. Freier Text ist nicht zwangsläufig der Maschine verständlich, als Text aber jedenfalls von der Maschine Menschen vermittelbar, die die Sprache verstehen, in der der Text verfasst ist.

1.23 Attribute und Entitäten von XML

In XML sind bereits einige Attribute vordefiniert. Diese werden mit dem ebenfalls bereits in XML fest vorgegebenem Präfix 'xml' notiert.

Ein Attribut ist *xml:lang* zur Kennzeichnung der Sprache, die im jeweiligen Element verwendet wird. Der Wert ist eine mit Leerzeichen separierte Liste von Sprachkennungen wie 'de' (deutsch) oder 'en' (englisch) oder auch 'en-us' (englisch im amerikanischen Dialekt). Siehe auch BCP47[13].

Ein weiteres ist *xml:space* mit den Werten 'preserve' oder 'default'. Dies hat Einfluss auf die Interpretation von Leerraum (Leerzeichen, Zeilenumbrüche) im Textinhalt. Bei 'preserve' wird der Leerraum in der Präsentation erhalten. Bei 'default' werden bei der Präsentation mehrere aufeinanderfolgende Leerzeichen oder Zeilenumbrüche zu einem Leerzeichen zusammengefasst. Je nach Format (besonders bei einem graphischen wie SVG) kann es weitere Angaben in der Spezifikation geben, wie mit Textinhalt im Detail umzugehen ist, wie es zum Beispiel mit Hilfe von Glyphen zu einer graphischen Repräsentation des Textes kommt.

xml:base gibt als Wert die Basis-URI an, von der aus Pfadangaben zu absoluten URIs aufgelöst werden sollen. In der Praxis problematisch kann dieses Attribut sein, wenn vom Autor nur Fragmentidentifizierer oder Bezeichner in einem Attribut angegeben werden, welches zu einer absoluten URI aufzulösen ist.

xml:id ist ein dokumentweit eindeutiger Bezeichner oder Fragmentidentifizierer für das Element, entsprechend den Attributen *id* in (X)HTML oder SVG. In SVG ist die Verwendung dieses erst spät in XML eingeführten Attributes erst in SVG tiny 1.2 alternativ zu *id* vorgesehen. Der Vorteil gegenüber formatspezifischen Bezeichnern wie *id* liegt darin, dass die Bezeichner formatübergreifend gelten und auch Fragmente in Formaten referenzierbar sind, ohne dass das genaue Format dem Darstellungsprogramm bekannt sein müsste.

Als Entitäten (maskierte Zeichen) sind in XML vordefiniert, was in Konflikt mit der Notation geraten kann:

- & (&)
- < (<)
- > (>)
- ' (')
- " (")

Weitere Details über XML kann man im Wikibuch XML[14] lernen.

13 http://www.rfc-editor.org/rfc/bcp/bcp47.txt
14 http://de.wikibooks.org/wiki/XML

2 Start mit SVG

2.1 Einführung: Beispiel 'Hello World'

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	3
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	1.x

Einer alten Tradition folgend soll mit einem Beispiel begonnen werden, welches die Zeichenfolge 'Hello World' zur Präsentation bringt:

```
<?xml version="1.0"?>

<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1"
     xml:lang="en">
  <title>Hello World</title>
  <text x="50.5" y="80.25" font-size="20">Hello World</text>
</svg>
```


Abb. 11 Beispiel 'Hello World'

Da jedes SVG-Dokument auch ein XML[1]-Dokument ist, beginnt dies mit einer XML-Verarbeitungsanweisung in der ersten Zeile ohne vorhergehende weitere Zeichen.

Die zweite Zeile ist hier leergelassen. Hier können optional mehrere Zeilen mit XML-Stilvorlagenverarbeitungsanweisungen stehen, gefolgt von einer ebenfalls optionalen Dokumenttyp-deklaration (siehe dazu auch das Kapitel Kurze Einführung in XML[2]), im Bedarfsfalle auch mit der Definition eigener Abkürzungen darin.

Dann beginnt das SVG-Dokument mit der Anfangsmarkierung des Elementes ***svg***. Dies ist das Wurzelelement des Dokumentes. Elemente wie ***svg*** bilden die Struktur des Dokumentes, ohne

1 http://de.wikibooks.org/wiki/XML
2 Kapitel 1.17.6 auf Seite 17

selbst unmittelbar zu einer Anzeige eines graphischen Inhaltes zu führen. Solche Elemente der Dokumentstruktur gruppieren Inhalte oder referenzieren sie. Näheres zu solchen Elementen wird im Kapitel Dokumentstruktur[3] diskutiert.

Das Wurzelelement enthält die Angabe des Namensraumes von SVG mit dem Attribut *xmlns*. Dies ist wichtig, weil jeder XML-Formate spezifizieren darf, zum Beispiel auch die **S**traßen**v**erkehrs**g**emeinschaft, die **S**chienen**v**erkehrs**g**esellschaft, der '**S**port**v**erein **G**uckste' oder auch die **S**ylter **V**erkehrs**g**esellschaft. Es gibt viele Möglichkeiten, wofür das Element ***svg*** stehen könnte, nicht nur für **S**kalierbare **V**ektor**g**raphik. Erst die Angabe des Namensraumes legt fest, wie die Elemente und die Struktur zu verstehen sind. Der obige gehört zur **S**kalierbaren **V**ektor**g**raphik und zu keiner anderen Möglichkeit.

Auf Namensräume wird ebenfalls näher im Kapitel Kurze Einführung in XML[4] eingegangen.

Ferner ist mit dem Attribut *version* angegeben, dass die Version 1.1 von SVG verwendet wird.

Ebenfalls notiert ist mit dem allgemeinen XML-Attribut *xml:lang* die verwendete Sprache für Textinhalt. Weil 'Hello World' englisch ist, wird das entsprechend auf englisch (Wert: en) festgelegt.

Als nächstes folgt ein Element ***title*** mit einem für den Inhalt charakteristischen Titel für das Dokument. Ein SVG-Dokument ohne ***title*** ist zwar technisch möglich, überspitzt formuliert aber inhaltlich sinnlos. Mehr zur sinnvollen Verwendung dieses Elementes ist dem Kapitel Barrierefreiheit[5] zu entnehmen.

Der Inhalt des Elementes ***title*** wird nicht direkt als Graphik ausgegeben, sondern erscheint vielleicht als Überschrift des Fensters mit der Graphik. Es handelt sich ebenfalls um ein Element der Kategorie Dokumentstruktur[6].

Damit der Text als Graphik sichtbar wird, wird sodann das Element ***text*** verwendet.

Mit den Attributen *x* und *y* wird der Text im Darstellungsbereich positioniert.

Das Element ***svg*** hat diesen Darstellungsbereich des Dokumentes erzeugt, in dem die graphischen Elemente dargestellt werden können. Ohne weitere Angaben, welche allerdings sowohl einfach möglich als auch in der Praxis sinnvoll sind, deckt sich der Darstellungsbereich des Dokumentes mit dem aktuell verfügbaren. Wird das Dokument nicht in ein anderes eingebettet, so ist dies der Darstellungsbereich der Anzeige, etwa das Fenster des Darstellungsprogrammes auf dem Monitor oder bei einem Drucker die Größe des Papieres. Wird das Dokument in ein anderes eingebettet, so bestimmt das einbettende Dokument in der Regel den verfügbaren Darstellungsbereich. Das kann allerdings auch so erfolgen, dass wiederum auf das einzubettende Dokument verwiesen wird, um die Größe zu bestimmen. Für den obigen Fall ohne weitere Angaben sollte das einbettende Format allerdings dann Regeln angeben, was zu tun ist, wenn das einzubettende Dokument selbst keine Angaben zur Größe enthält. Eine eindeutige Angabe mit dem noch zu diskutierenden Attribut *viewBox* für das Element ***svg*** vermeidet derartige Komplikationen, ohne eine Anpassung an den verfügbaren Anzeigebereich durch Skalierung zu verhindern.

3 Kapitel 3 auf Seite 43
4 Kapitel 1.17.6 auf Seite 17
5 Kapitel 4 auf Seite 77
6 Kapitel 3 auf Seite 43

Elemente, welche direkte graphische Formen repräsentieren, wie eben ***text*** oder auch Basisformen wie Rechtecke, Kreise, Ellipsen, Polygone und allgemeine Pfade, werden in weiteren Kapiteln erläutert. Dies sind Grundformen[7], Pfade[8] und Text als Graphik[9].

Welche spezifischen Attribute es für ein Element gibt, ist jeweils bei der Beschreibung des Elementes angegeben. Oft hat der Wert eines Attributes eine bestimmte Struktur. Diese ist dann auch genau beschrieben. Sofern es sich um einen immer wiederkehrenden Typ handelt, ist er auch im Kapitel Grunddatentypen[10] erläutert. In obigem Beispiel ergibt sich zum Beispiel die Frage, was als Wert jeweils für *x* und *y* und *font-size* notiert werden kann. Es wird etwa die Schreibweise '50.5' verwendet und nicht '50,5'. Letzteres hätte gegebenenfalls entweder eine andere Bedeutung für *x* und *y* als vermutlich beabsichtigt oder wäre sogar ein fehlerhafter Wert für *font-size*. Wie Zahlen zu notieren sind, ist ebenfalls im Kapitel Grunddatentypen[11] erläutert.

Während die Elemente samt ihren Attributen festlegen, was dargestellt werden soll, gehört obiges *font-size* zu den Eigenschaften oder Präsentationsattributen, die festlegen, wie etwas dargestellt werden soll. Mit dem Präsentationsattribut *font-size* wird die Größe der Schrift des Textes angegeben. Präsentationsattribute werden wie gewöhnliche Attribute notiert, haben im Detail aber ein etwas anderes Verhalten, was dem von Eigenschaften in CSS entspricht, die es gleichnamig ebenfalls gibt. Bei obigem Beispiel bleiben da noch zahlreiche Fragen hinsichtlich der Präsentation offen, etwa in welcher Farbe, welche Schriftart etc. Eigenschaften und Präsentationsattribute werden im Kapitel Grafiken formatieren[12] ausführlich vorgestellt. Diese Fragen werden in dem konkreten Beispiel für die Präsentation vom Darstellungsprogramm automatisch beantwortet. Das funktioniert, weil SVG für alle Präsentationsattribute, Eigenschaften und Attribute Rückfallwerte oder ein Rückfallverhalten vorsieht, auf welche vom Darstellungsprogramm zurückgegriffen wird, wenn es im Dokument keine weiteren anwendbaren Angaben gibt. In einigen Fällen, die in obigem Beispiel nicht vorliegen, ist allerdings die Angabe gewisser Attribute notwendig, sonst ist das Dokument fehlerhaft und die Darstellung kann abgebrochen werden.

7 Kapitel 6 auf Seite 119
8 Kapitel 7 auf Seite 143
9 Kapitel 8 auf Seite 193
10 Kapitel 2.2 auf Seite 31
11 Kapitel 2.2 auf Seite 31
12 Kapitel 9 auf Seite 209

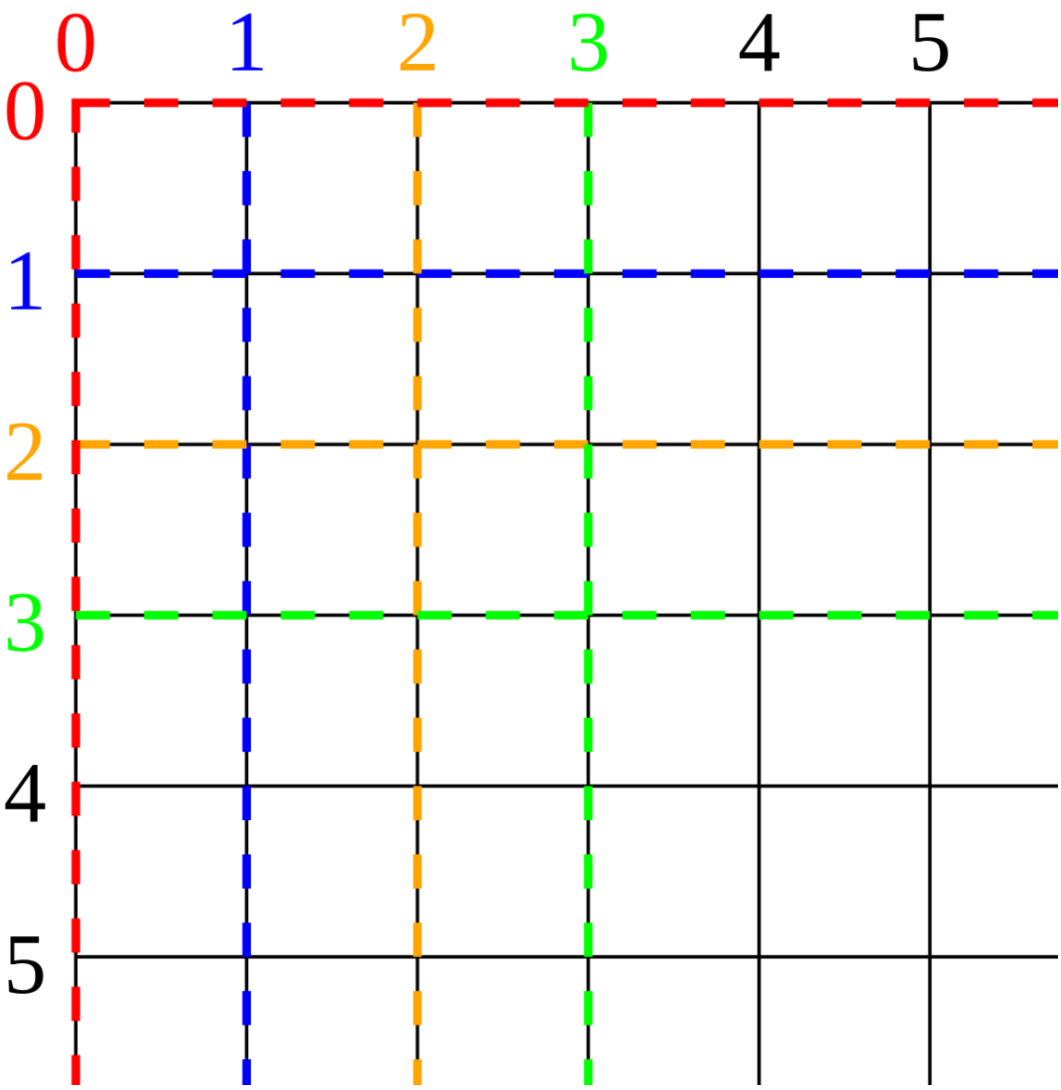


Abb. 12 Das anfängliche Koordinatensystem von SVG

Der Ursprung des anfänglichen Darstellungsbereiches befindet sich links oben, die x-Richtung zeigt nach rechts, die y-Richtung nach unten. In SVG ist es auch einfach möglich, neue Koordinatensysteme einzuführen oder Inhalte zu transformieren. Dies wird im Kapitel Transformationen[13] erklärt.

Alle Elemente werden entsprechend wieder geschlossen. Das ist immer so bei XML und nicht spezifisch für SVG.

Rechts oben neben diesem Text ist zu sehen, wie eine typische Ausgabe aussehen könnte. Besonders bei Textinhalt sind die Rückfallwerte so definiert, dass die Ausgabe nicht eindeutig ist, also je nach Darstellungsprogramm etwas anders ausfallen darf.

13 Kapitel 5 auf Seite 99

2.1.1 Welche Informationen vermittelt das erste Beispiel?

1. Jedes SVG-Bild befindet sich im Element ***svg*** zwischen der Anfangsmarkierung `<svg>` und der Endmarkierung `</svg>`
2. In dem ***svg*** wird ein Namensraum angegeben: `xmlns="http://www.w3.org/2000/svg"`. Dies ist notwendig, um das Format SVG selbst zu identifizieren. Gegebenenfalls gibt es Angaben zu weiteren Namensräumen, um weitere verschiedene verwendete XML-Formate zu identifizieren und miteinander zu mischen. Bei SVG sind es oft XLink, XHTML, RDF oder MathML. Es ist aber auch jedes andere XML-Format möglich. Es wird allerdings nicht jedes Format von jedem Darstellungsprogramm interpretiert. SVG hat aber definierte Regeln, wie nicht interpretierte andere Formate im Zweifelsfalle zu ignorieren sind. Es hat auch Regeln, wo Inhalte in anderen Formaten zu notieren sind, damit sie nicht ignoriert werden, sofern das Darstellungsprogramm sie kennt.
3. Eine Versionsangabe dokumentiert, welche Version von SVG im Dokument verwendet wird.
4. Es gibt Elemente, die der Struktur und der Arbeitserleichterung für den Autor dienen, und andere für die Angabe von graphischen Inhalten. Daneben gibt es weitere Elementtypen, etwa solche, die einen Maldienst wie ein Muster oder einen Farbverlauf definieren.
5. Ein Element ***title*** ist wichtig als Textrepräsentation des gesamten Dokumentes.
6. Die Elemente können Attribute besitzen (hier dargestellt in blau), denen Werte (rot) zugewiesen werden. Elemente samt ihren Attributen beschreiben, was für ein Objekt dargestellt wird. Das Element selbst macht mit seinem Namen eine allgemeine Angabe, die Attribute bestimmen eher im Detail, um was es sich handelt.
7. Daneben gibt es weitere allgemein anwendbare Attribute wie *xml:lang*, die jedes Element und dessen Inhalt näher beschreiben können.
8. Eigenschaften oder Präsentationsattribute bestimmen, wie Inhalte dargestellt werden.
9. Das anfängliche Koordinatensystem von SVG arbeitet mit der Einheit Pixel des jeweiligen Ausgabegerätes. Der Ursprung befindet sich links oben, davon wird die x-Achse nach rechts gezählt und die y-Achse nach unten.

2.2 Grunddatentypen

In SVG teilen einige Attributwerte und Eigenschaftswerte eine gemeinsame Datenstruktur oder -syntax. Wenn ein Attributwert oder Eigenschaftswert eine solche allgemeine Struktur hat oder aus einer Liste solcher Strukturen besteht, ist in der Beschreibung jeweils nur ein Schlüsselwort angegeben.

Solch allgemeine Strukturen sind hier erläutert. Beim ersten Lesen kann ein Überfliegen reichen, im Zweifelsfalle ist das aber später genau nachzusehen, weil davon die Funktion diverser Anwendungen abhängen kann.

2.2.1 Zahlen

Häufig ist in Form von Zahlen anzugeben, wo eine graphische Form positioniert sein soll oder wie groß sie ist, allgemein der Wert eines Attributes oder einer Eigenschaft. Die verwendbaren Zahlen sind in SVG angegeben. Wichtig ist etwa bei nicht ganzzahligen Zahlen, dass zur Abtrennung des nichtganzzahligen Teiles ein Punkt '.' verwendet wird, nicht wie im deutschen Sprachraum ein

Komma. Einmal abgesehen von Farbangaben, die getrennt diskutiert werden, erfolgen die Angaben im Dezimalsystem (Ziffern 0 bis 9).

Ganze Zahlen als einzig erlaubte Werte kommen in SVG eher selten vor, können aber ebensogut wie nichtganzzahlige Zahlen notiert werden.

Folgendes sind Beispiele für *korrekt* notierte Zahlen:

- 73.1234
- -0.002
- .154
- -.897
- +13.2
- 0000.123
- 1.24E+2 (wissenschaftliche Notation, hinter dem E nur ganze Zahlen, Vorzeichen optional)
- 1.24e+2 (wissenschaftliche Notation)
- 1E0 (wissenschaftliche Notation)
- -.0E-1 (wissenschaftliche Notation)

Folgendes sind Beispiele für *falsch* notierte Zahlen:

- 73,1234 (Komma nicht als Trennzeichen für den nicht ganzzahligen Bereich)
- 1.000,7432 (keine weiteren Trennzeichen außer dem Punkt für den nicht ganzzahligen Bereich)
- 1,000.7432 (keine weiteren Trennzeichen außer dem Punkt für den nicht ganzzahligen Bereich)
- - .897 (Leerzeichen zwischen Vorzeichen und Zahl nicht erlaubt)
- + 13.2 (Leerzeichen zwischen Vorzeichen und Zahl nicht erlaubt)
- 1.24D+2 (für wissenschaftliche Notation nur E oder e)
- 7.9e-1.7 (hinter dem E nur ganze Zahlen)
- A8 (keine Hexadezimalnotation)
- E+2 (vor dem E muss eine Zahl stehen)

Ferner ist der Bereich eingeschränkt, aus dem Zahlen verwendet werden können, bei den tiny-Profilen ist der erlaubte Bereich -32767.9999 bis +32767.9999

Zudem dürfen nicht mehr als 4 Stellen hinter dem Punkt angegeben werden.

Diese Einschränkung hängt vor allem damit zusammen, dass tiny-Dokumente von Mobiltelephonen interpretiert werden können sollen. Diese haben nicht so leistungsfähige Prozessoren wie vollwertige Rechner oder notebooks. Aufgrund der technischen Einschränkung hilft es natürlich auch nicht, diese Beschränkung mit Tricks wie Skalierungen und Verschiebungen von Koordinatensystemen zu umgehen (obgleich dies nicht explizit ausgeschlossen wurde).

Bei dem kompletten Profil 1.1 ist der erlaubte Bereich deutlich größer, mindestens -3.4e+38 bis +3.4e+38 muss interpretiert werden. Natürlich, sofern es für das Dokument ausreicht, ist es vorteilhaft, sich auf Werte zu beschränken, die für tiny erlaubt sind, so können diese Dokumente von allen Darstellungsprogrammen interpretiert werden, wenn sie keine weiteren Angaben enthalten, die im tiny-Profil nicht vorkommen.

2.2.2 Längen

Längen beschreiben einen Abstand. Eine Länge ist eine Zahl, optional gefolgt von einer Einheit. In den tiny-Profilen entfällt die Einheit mit Ausnahme der Längenangaben in den Attributen *width* und *height* des Hauptelementes ***svg***. Ohne Angabe einer Einheit ist die Einheit '1' im lokalen Koordinatensystem. Mögliche Einheiten sind die in CSS2.0 (!) genannten, für SVG 1.1 und SVG tiny 1.2: em, ex, px, pt, pc, cm, mm, in und Prozentangaben.

em und ex beziehen sich auf die aktuelle Größe der Schriftart.

px entspricht 1 in lokalen Koordinaten.

Die absoluten Einheiten pt, pc, cm, mm und in sind alle eindeutig ineinander umrechenbar. Zur Umrechnung dieser Einheiten in px oder lokale Koordinaten braucht das Darstellungsprogramm allerdings immer eine weitere Information, zum Beispiel über die Auflösung des Monitors oder Druckers, wo die Graphik darzustellen ist. Auflösung ist hier im üblichen Sinne gemeint als Pixel pro Meter oder in amerikanischen Einheiten dpi (dots per inch). Zum einen ist dieser Umrechnungsfaktor insbesondere bei Monitoren vielen Darstellungsprogrammen nicht bekannt und zum anderen kann der Faktor je nach Monitor und Nutzer variieren. Genaugenommen wäre die Angabe der Auflösung bei den meisten Betriebssystemen und Monitoren seit Jahren problemlos verfügbar, viele Darstellungsprogramme lesen diese verfügbare Information aber nicht aus, sondern arbeiten mit willkürlichen Schätzungen.

So oder so ist dem Autor der Umrechnungsfaktor nicht bekannt, was die Anzeige für ihn beim späteren Nutzer schwer vorhersagbar macht, wenn absolute Einheiten insbesondere mit lokalen Einheiten gemischt werden.

Die Größe von em oder ex kann der Autor hingegen im Dokument entweder mit absoluten Einheiten oder in lokalen Einheiten eindeutig festlegen. Geschieht dies nicht, ergibt sich em aus den Voreinstellungen des Nutzers. ex ergibt sich aus der jeweiligen Schriftart relativ zu em. Soweit der Autor keine eigene Schriftart angelegt hat, wo diese Einheit präzise festgelegt ist, ist auch hier der Umrechnungsfaktor relativ zu em nur schwer vorhersagbar.

Bei einer Mischung von solchen Einheiten mit Angaben im lokalen Koordinatensystem wird es also dazu kommen, dass der Autor das jeweilige Ergebnis nicht vorhersagen kann. Es empfiehlt sich also in der Regel das Vorgehen wie bei den tiny-Profilen, Einheiten nur bei *width* und *height* des Hauptelements ***svg*** anzugeben, dann ist sichergestellt, dass die Größen innerhalb der Graphik relativ zueinander immer gleich sind.

Prozentangaben werden bei verschiedenen Anwendungen unterschiedlich interpretiert. Bezieht sich die Angabe auf einen Anzeigebereich, wie bei *width* und *height* des Hauptelementes ***svg***, so ist der prozentuale Anteil des verfügbaren Anzeigebereiches gemeint. Bei anderen Prozentangaben, die sich auf den Anzeigebereich beziehen, ist das der prozentuale Anteil der Diagonale desselben.

Bei der Interpretation des Dokumentes werden die Einheiten jeweils in lokale Einheiten umgerechnet, bevor der graphische Inhalt dargestellt wird. In der Praxis zeigt sich, dass viele Darstellungsprogramme Probleme mit der korrekten Umrechnung haben (Stand: Ende 2010). Besonders bei der Addition und Interpolation bei Animationen können grobe Fehler auftreten, weswegen dringend zu raten ist, keine verschiedenen Einheiten anzugeben und am besten alles in lokalen Einheiten zu notieren, um Problemen aus dem Wege zu gehen. Explizit ausgenommen sind davon die genannten *width* und *height* des Hauptelements ***svg***, kombiniert mit einer *viewBox* in lokalen Koordinaten kann dort eine

Größenangabe in absoluten Einheiten, in em, ex oder Prozent sehr sinnvoll sein und hat nur zur Folge, dass der Inhalt einmal komplett skaliert wird. Allerdings ist meist bei der Anzeige auf einem Monitor nicht damit zu rechnen, dass bei einer Angabe wie *width*="10cm" wirklich eine Anzeige mit einer Breite von 10cm erfolgt. Aufgrund der genannten Mängel der Darstellungsprogramme wird sich vermutlich meist eine andere Breite der Darstellung auf dem Monitor ergeben. Drucker können solche Angaben eher korrekt umsetzen, sofern diese SVG interpretieren können. Da der Fehler der Darstellungsprogramme allerdings nur auf einem falschen Konversionsfaktor beruht, bleibt immerhin das Aspektverhältnis korrekt.

Umrechnung zwischen den Einheiten und Beschreibung in Tabellenform:

Einheit	SVG-Abkürzung	Beschreibung
Pixel	px	Ein Pixel, eine Einheit in lokalen Koordinaten
Geviertbreite und -höhe	em	Schriftgröße der aktuellen Schriftart
x-Höhe	ex	Die Höhe des Buchstabens x in der aktuellen Schriftart (bei einer Schriftart immer relativ zu em exakt festlegbar, beziehungsweise geschätzt, wenn nicht festgelegt und kein x vorhanden ist)
Punkt, englisch: point	pt	0.3527mm oder 1/72 Zoll
Pica	pc	4.23mm oder ⅙Zoll oder 12pt
Millimeter	mm	ein Millimeter
Zentimeter	cm	zehn Millimeter
Zoll, englisch: inch	in	25.4 Millimeter

2.2.3 Koordinaten

Eine Koordinate ist eine Längenangabe im lokalen Koordinatensystem, also der Abstand zum Ursprung in Richtung der x- beziehungsweise y-Achse.

2.2.4 Listen

In SVG sind Attributwerte häufig Listen, zum Beispiel von Zahlen. Wie die Listen zu notieren sind, ist insbesondere in SVG 1.1 nicht ganz einheitlich geregelt, in SVG tiny 1.2 wurde jedoch versucht, dies weitgehend zu verschiedenen Gruppen zu vereinheitlichen.

Oft geht es um Listen, bei denen die Listenpunkte mit Leerraum (englisch: whitespace) zu separieren sind. Dabei handelt es sich neben dem normalen Leerzeichen auch um jene Zeichen, die verwendet werden, um Zeilenumbrüche im Quelltext zu erreichen. Je nach Betriebssystem wird das mit einem oder mehreren Zeichen erreicht, die alle als Leerraum angesehen werden. Es können dann mehrere davon in beliebiger Reihenfolge zur Separation genommen werden. Auch der Liste vorangestellter oder nachfolgender Leerraum ist zulässig.

Ein weiterer Listentyp ist der mit einem Komma als Separator. Auch da kann vor und nach dem Komma noch Leerraum stehen. Ein Komma darf jedoch weder dem ersten Listenwert vorangehen, noch dem letzten folgen (Im Falle von Animationen wird anstatt des Kommas ein Semikolon verwendet. Damit können dann auch Listen animiert werden).

Zudem gibt es dann die Möglichkeit, dass der Autor sich bei jedem Separator aussuchen kann, ob dies nur Leerraum ist oder auch ein Komma enthält. Dies ist die häufigste Möglichkeit. Die wird auch kurz 'durch Leerraum oder Komma separierte Liste' genannt.

Leerraum in diesem Sinne sind als unicode: #x9, #xD, #xA, #x20

oder anders notiert: "space" (U+0020), "tab" (U+0009), "line feed" (U+000A), "carriage return" (U+000D),

also das normale Leerzeichen, Tabulatorzeichen, Zeilenvorschub und Wagenrücklauf.

2.2.5 IRI, URI, URL

IRI (englisch: internationalised resource identifier), URI (englisch: unified resource identifier) und URL (englisch: unified resource locator) sind Referenzen zu (anderen) Dokumenten oder Dokumentfragmenten. Die Identifizierer sind etwas allgemeiner als die Lokatoren. Letztere bezeichnen vor allem eindeutig, wo das bezeichnete Dokument(fragment) zu finden ist, während Identifizierer auch der eindeutigen Identifizierbarkeit dienen können, nicht immer auch der direkten Lokalisierbarkeit.

Derartige Konstruktionen sind formatunabhängig festgelegt, also nicht spezifisch für SVG.

An einigen Stellen ist auch eine funktionale Notation vorzunehmen, besonders bei Eigenschaften oder den damit äquivalenten Präsentationsattributen trifft dies zu. Die funktionale Notation einer IRI sieht wie folgt aus: url("IRI") Dabei steht IRI für die zu notierenden Referenz. Die Buchstabenkombination 'url' gilt dabei unabhängig davon, ob es sich um eine IRI, URI oder URL handelt.

2.2.6 Sprachidentifizierer

Ein Sprachidentifizierer dient der Angabe einer gesprochenen oder geschriebenen Sprache wie deutsch oder englisch. Insbesondere kann für Fragmente von Dokumenten, die Text enthalten, damit maschinenlesbar und -verstehbar die verwendete Sprache angegeben werden. Dazu dienen eindeutig definierte Sprachmarkierungen. Derartige Konstruktionen sind formatunabhängig festgelegt, also nicht spezifisch für SVG.

2.2.7 Fragmentidentifizierer

Fragmentidentifizierer sind eindeutige Bezeichner für Dokumentfragmente, die also in dem jeweiligen Dokument nicht mehrmals in gleicher Form auftauchen dürfen. Typische Attribute, die einen Fragmentidentifizierer als Wert haben, sind in SVG *id* und in SVG tiny 1.2 und XML allgemein auch *xml:id*.

Die Konstruktion beginnt immer mit einem Buchstaben (a-z, A-Z ohne Sonderzeichen) oder dem Zeichen '_' oder ':', optional gefolgt von weiteren Buchstaben oder Ziffern oder folgenden Zeichen: '.', '-', '_', ':'. Derartige Konstruktionen sind formatunabhängig festgelegt, also nicht spezifisch für SVG. Exakt sind die erlaubten Zeichen in der XML-Spezifikation unter dem Stichwort 'Name' nachzulesen.

2.2.8 Medientypen

Insbesondere wenn auf andere Dokumente verwiesen wird oder diese in das aktuelle Dokument eingebettet werden sollen, ist es relevant, den Medientyp des jeweiligen anderen Dokumentes maschinenlesbar und -verstehbar angeben zu können, dazu dienen speziell definierte Medientypenmarkierer oder internet-Medientypen, ehemals auch MIME-Typen genannt (englisch: Multipart Internet Mail Extensions).

2.3 Genauigkeit

Hinsichtlich der Genauigkeit der Darstellung werden in den Spezifikationen ebenfalls Angaben gemacht. In der finalen Darstellung auf dem Bildschirm (oder beim Ausdruck) soll eine Genauigkeit nicht schlechter als ein Pixel des Ausgabegerätes erreicht werden, wenn möglich besser. Bei einer Entdeckung einer größeren Ungenauigkeit bei einem Darstellungsprogramm ist dies also das Kriterium, um zu beurteilen, ob das Programm fehlerhaft arbeitet oder ob das eine erlaubte Toleranz ist.

2.4 Zeichenreihenfolge

In SVG wird das sogenannte Malermodell verwendet, um die Zeichenreihenfolge festzulegen, wenn mehrere darzustellende Elemente in einem Dokument vorliegen - was meistens der Fall sein wird. Die darzustellenden Elemente werden in der Reihenfolge gemalt, wie sie im Quelltext stehen. Also was weiter vorne steht ist unten und gegebenenfalls teilweise oder ganz verdeckt. Was zuletzt im Quelltext steht, ist oben und verdeckt gegebenenfalls Elemente, die früher im Quelltext stehen.

Ausnahmen ergeben sich dadurch, dass nicht alle Elemente darzustellen sind. Das ist dann im Detail zu diskutieren, wenn die entsprechenden Elemente vorgestellt werden. Als kurzer Ausblick: Elemente, die in einem ***defs***-Element notiert sind, sind nicht direkt darzustellen, haben im Malermodell also auch keine bestimmte Position, solange sie nicht zum Beispiel mit dem Element ***use*** zur Darstellung referenziert werden. In diesem Falle wird das darzustellende Element ***use*** dann durch das referenzierte Elemente ersetzt und die Position von ***use*** im Quelltext bestimmt die Zeichenreihenfolge.

Für einen Autor ist also hauptsächlich relevant, die Elemente im Quelltext in der für ihn richtigen und wichtigen Reihenfolge anzuordnen, wenn sie sich überlappen. Diese Reihenfolge ist allenfalls mit subtilen Tricks zu ändern. Für zukünftige Versionen von SVG wird allerdings über eine Funktionalität ähnlich der Eigenschaft z-index von CSS diskutiert.

Der wesentliche Trick bei einer scheinbaren Änderung der Zeichenreihenfolge besteht darin, den Betrachter über den Sachverhalt hinwegzutäuschen, dass die Zeichenreihenfolge gar nicht geändert wurde. Dazu werden einige verschiedene Möglichkeiten von SVG ausgenutzt und Eigenschaften des

menschlichen Denkens und der Wahrnehmung, welche von der Evolution her darauf optimiert sind, Objekte räumlich einzuordnen und zu interpretieren und veränderliche Strukturen als Bewegungen zu interpretieren.

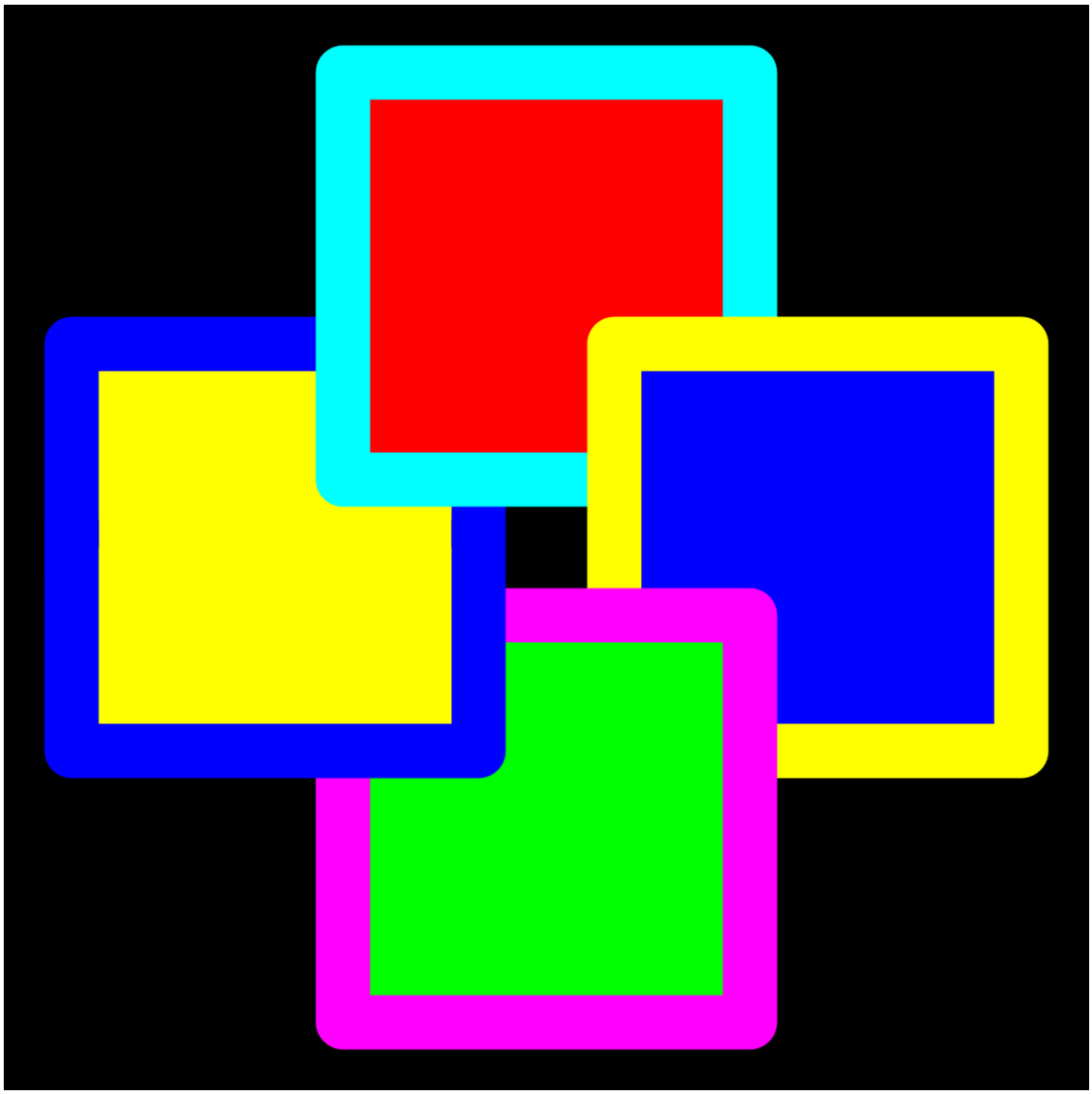


Abb. 13 Verkeilte Vierecke

Das erste Beispiel zeigt sowohl den einfachen Sachverhalt, dass nacheinander im Quelltext stehende Objekte übereinander gemalt werden, als auch eine Methode, mit der suggeriert wird, dass dem nicht so sei:

Verkeilte Vierecke[14].

14 http://de.wikibooks.org/wiki/media%3ASVGzeichenreihenfolge01.svg

Vier teiltransparente symmetrisch angeordnete Rechtecke liegen so, dass eine Ecke eines Rechtecks jeweils eine Ecke eines anderen Rechtecks verdeckt. Das passiert reihum, bis es beim letzten Rechteck zu einem Problem der Zeichenreihenfolge kommt. Daher wird dieses in Teile zerlegt und der problematische Teil wird vor den anderen Rechtecken gemalt.

Ein problematisches Objekt wird also an unkritischen Stellen in mehrere Objekte aufgetrennt, die entsprechend dem gewünschten Effekt im Quelltext in die beabsichtigte Reihenfolge gebracht werden.

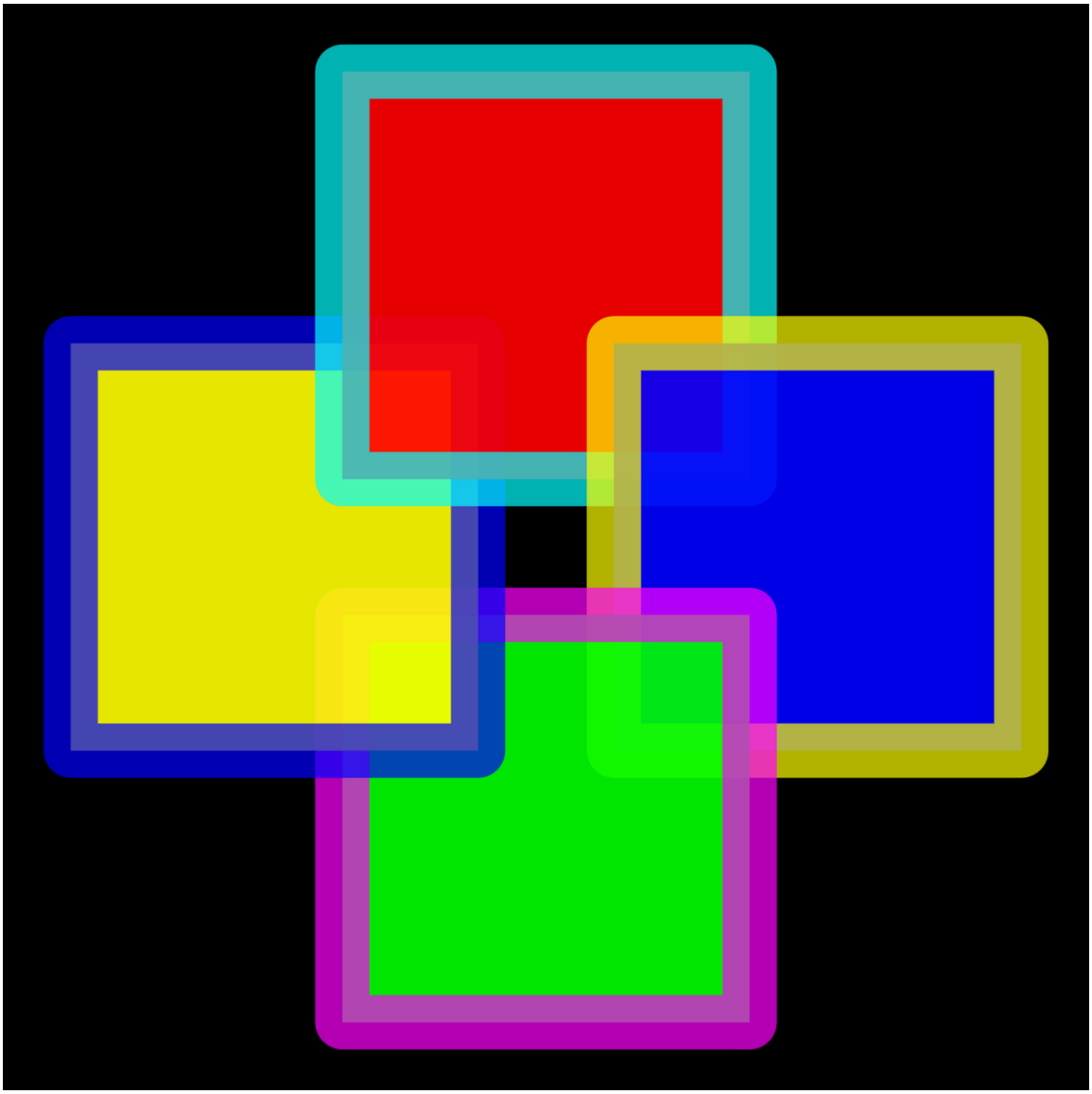


Abb. 14 Teilweise durchsichtige verkeilte Vierecke

Das Vorgehen wird insbesondere bei teildurchsichtigen Objekten problematisch, wenn diese aufgrund von Rundungsfehlern vom Darstellungsprogramm geringfügig falsch positioniert oder dimensioniert werden. Wegen der Teiltransparenz kann hier nicht mit einem Überlapp gearbeitet werden, um solche Ungenauigkeiten der Darstellungsprogramme zu vertuschen. Folgendes ist ein

entsprechendes Beispiel mit teilweise durchsichtigen Füllungen und Strichen:

Teilweise durchsichtige verkeilte Vierecke[15].

Je nach Größe des Darstellungsbereiches ist bei einigen Programmen der Schnitt zu erkennen. Dies kann weitgehend umgangen werden, wenn der Schnitt entlang von ohnehin vorhandenen Kanten erfolgt. Bei so einfachen Objekten wie in diesem Beispiel ergibt sich ferner die Möglichkkeit, die aus der Teiltransparenz resultierenden Farbwerte zu berechnen und die kritischen Stellen mit entsprechenden nicht transparenten Objekten zu verdecken.

Abb. 15 Tribar als Beispiel für ein Penrose-Vieleck

Mit präziser Berechnung und mit ebenfalls in SVG verfügbaren Farbverläufen kann ferner der Eindruck eines dreidimensionalen Objektes erweckt werden, obwohl es in SVG immer nur um Flächen geht. Dies kann sogar so weit ausgenutzt werden, dass dem Betrachter Objekte untergeschoben werden können, die lokal räumlich aussehen, als Gesamtobjekt aber so gar nicht möglich sind:

15 `http://de.wikibooks.org/wiki/media%3ASVGzeichenreihenfolge02.svg`

Tribar als Beispiel für ein Penrose-Vieleck[16].

Der Eindruck von vorne und hinten wird hier einfach durch eine präzise berechnete Anordnung von Flächen erweckt, die zudem mit Farbverläufen gefüllt sind, die einen Beleuchtungseffekt vortäuschen und so das Objekt plastisch erscheinen lassen.

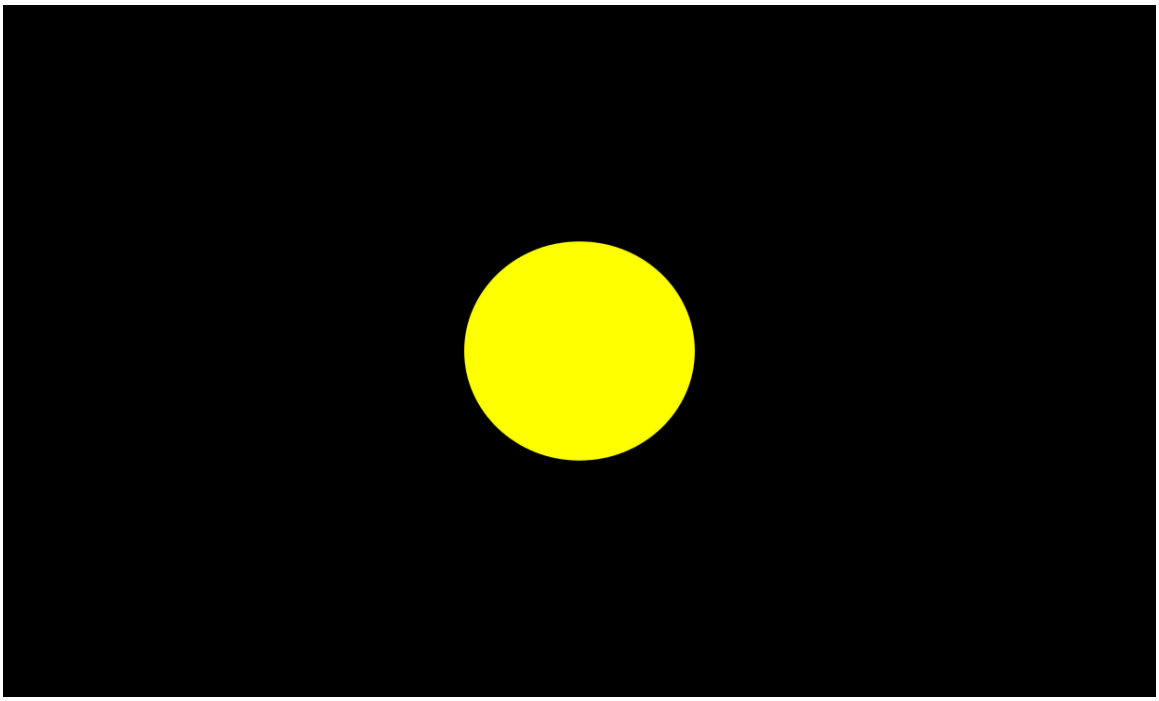


Abb. 16 Umrundung eines Objektes

Noch interessanter wird die Situation, wenn sich die Reihenfolge scheinbar zeitlich ändern soll. Bei folgendem Beispiel soll der Eindruck erweckt werden, dass sich ein Planet um eine Sonne dreht. Bei einer Seitenansicht ist der Planet also einmal vor der Sonne und einmal dahinter:

Umrundung eines Objektes[17].

Der gewünschte Effekt wird bei diesem Beispiel dadurch erreicht, dass einfach ein Objekt vor der Sonne animiert wird und eines dahinter. Mit einem passenden Zeitablauf wird einfach jeweils eines der beiden Objekte komplett transparent gesetzt. Entsprechend könnte auch die Anzeige oder Sichtbarkeit animiert werden. Ein anderer Ansatz wäre ein zeitabhängiger Ausschnitt (clip-Funktion in SVG), oder eine zeitabhängige Maske (mask-Funktion in SVG), wobei das eine Objekt als Ausschnitt beziehungsweise Maske für die Darstellung der anderen dient.

16 `http://de.wikibooks.org/wiki/media%3ASVGzeichenreihenfolge03.svg`
17 `http://de.wikibooks.org/wiki/media%3ASVGzeichenreihenfolge04.svg`

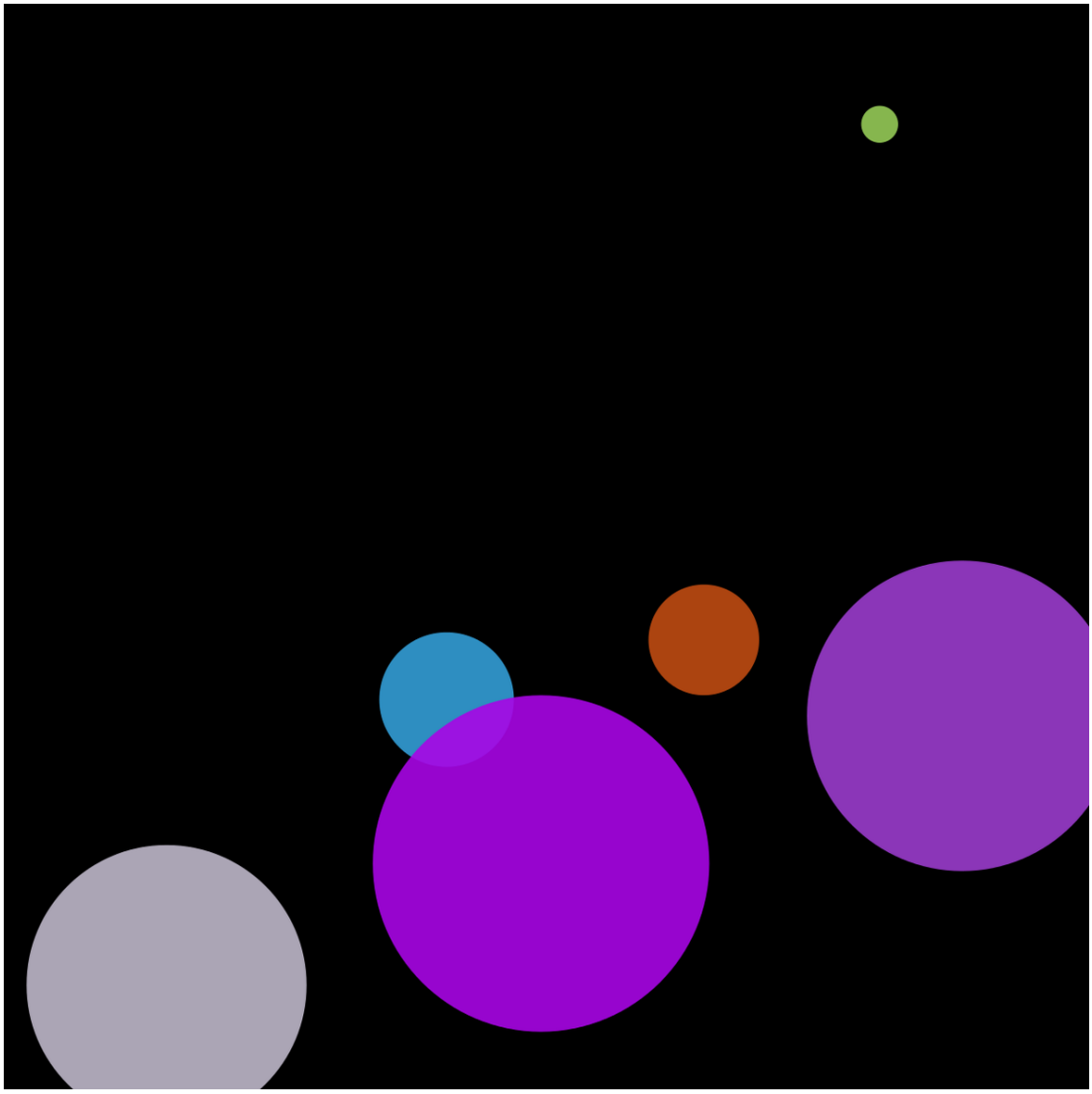


Abb. 17 Zufällig bewegte Kreise

In einer komplizierteren Situation können auch einfach alle Objekte im Element ***defs*** notiert werden. Diese werden von entsprechend vielen ***use***-Elementen referenziert, bei denen mit dem gewünschten Zeitablauf einfach animiert wird, welches Objekt jeweils referenziert wird:

Zufällig bewegte Kreise[18].

18 `http://de.wikibooks.org/wiki/media%3ASVGzeichenreihenfolge05.svg`

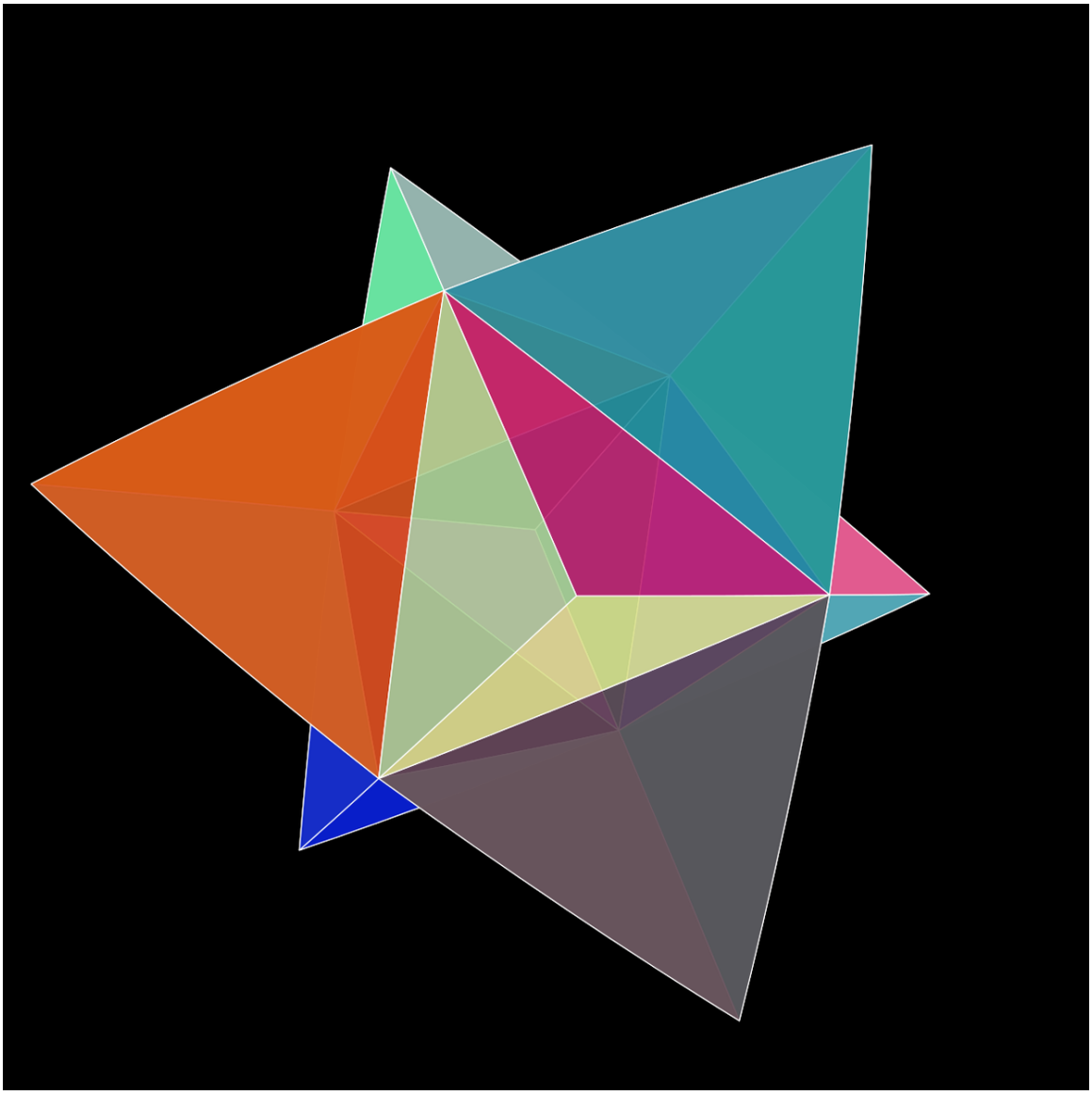


Abb. 18 Animierter Sterntetraeder

Kombiniert mit präziser Berechnung des Zeitablaufes und der perspektivischen Projektion kann so zumindest für einfache Objekte leicht der Eindruck von dreidimensionalen bewegten Objekten erzeugt werden:

Animierter Sterntetraeder[19].

Kompliziertere aperiodische Zeitabläufe sind so allerdings nur über einen endlichen Zeitraum darstellbar. Insofern ist es sehr hilfreich, dass für die nächste Version von SVG eine Möglichkeit vorgesehen ist, die Zeichenreihenfolge zu ändern. Auch perspektivische Projektionen sind in Vorbereitung, wodurch sich in Zukunft für den Autor der rechnerische Zeitaufwand deutlich reduzieren wird, um solche Effekte zu erreichen.

19 `http://de.wikibooks.org/wiki/media%3ASVGzeichenreihenfolge06.svg`

3 Dokumentstruktur

3.1 Dokumentstruktur, Einführung

SVG ist entweder in Form von eigenständigen Dokumenten vorzufinden oder als Dokumentfragment innerhalb von anderen (XML)-Dokumenten, in Zukunft eventuell auch in HTML5-Dokumenten.

SVG braucht einen Zeichenbereich mit klar definierten Koordinaten, daher ist SVG immer in einem Element ***svg*** zu finden, welches das primäre Koordinatensystem aufspannt. Dabei zeigt die x-Richtung von links nach rechts, die y-Richtung von oben nach unten. Der Zeichenbereich ist konzeptionell eine unendliche Ebene, von der im Anzeigebereich ein Teil angezeigt wird, welcher vom Autor präzise festgelegt werden kann.

Weitere Elemente dienen der prinzipiellen Strukturierung und der effektiven Nutzbarkeit von Inhalten, nicht unbedingt der unmittelbaren graphischen Darstellung. Solche Strukturelemente werden in diesem Kapitel beschrieben.

3.2 Element *svg*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Das Element ***svg*** ist ein Gruppierungselement und enthält ganz allgemein den SVG-Inhalt eines SVG-Dokumentes oder -Fragmentes. In den tiny-Profilen gibt es exakt ein ***svg***-Element in einem SVG-Dokument. Dies ist das Haupt- oder Wurzelelement (englisch: root) des Dokumentes im Sinne von XML.

In der Vollversion von SVG 1.1 können darin weitere ***svg***-Elemente ohne diese besondere Funktion eines Wurzelelementes auftreten, welche aber auch einen eigenen Anzeigebereich erhalten.

Für das Element spezifische Attribute:

3.2.1 Attribut *version*

Das Attribut gibt die Sprachversion an, welche für das aktuelle Dokument oder Dokumentfragment verwendet wird. Mögliche Werte sind derzeit '1.0' (veraltet), '1.1' und '1.2'. Ist keine Versionsangabe vorhanden, kann das Darstellungsprogramm oder allgemeiner der Leser des Quelltextes annehmen, dass die Version die 'aktuelle' Version ist. Dabei hängt 'aktuell' zwangsläufig davon ab, wann das

Programm veröffentlicht wurde, beziehungsweise wann jemand sich den Quelltext anguckt. Autoren ist also zu empfehlen, eine konkrete Version anzugeben, um ein definiertes Dokument zu bekommen.

Eine Versionsangabe wird in der Praxis allerdings kaum Einfluss darauf haben, wie ein Darstellungsprogramm den Inhalt darstellt. Nach dem aktuellen Stand der Technik haben Darstellungsprogramme bestenfalls eine Version implementiert. Allerdings kann das Darstellungsprogramm den Nutzer darüber informieren, dass Probleme auftreten können, falls die angegebene Version nicht mit der dem Programm bekannten Version übereinstimmt, was den Nutzer dazu veranlassen könnte, sich gegebenenfalls nach einem neueren Programm umzusehen, wenn es Zweifel an der Darstellung gibt.

Formal kann auch nur mit Angabe der Version entschieden werden, ob das Verhalten eines Darstellungsprogrammes korrekt ist oder nicht. Zudem kann es bei späteren Aktualisierungen oder Modifikationen des Dokumentes helfen zu identifizieren, nach welchen Regeln die bisherige Version erstellt wurde.

Nach Möglichkeit wird versucht, kompatible Versionen einer Sprache in dem Sinne zu spezifizieren, dass Notationen, die in einer früheren Version spezifiziert wurden, in einer neueren nicht plötzlich eine neue Bedeutung bekommen. In einer neueren Version werden daher vor allem neue Funktionalitäten mit neuen Notationen hinzugefügt oder die Notation so erweitert, dass die neue Notation nicht zu Missverständnissen gegenüber der alten Version führt. Indes können in einer alten Version auch Fehler oder Widersprüche vorliegen, die nach Bekanntwerden in einer neuen Version behoben wurden. Für solche problematischen Bereiche der alten Version gibt es kein definiertes Verhalten, welches erst in der neuen Version vorliegt.

Ein vorsichtiger Autor wird also eher eine neue Version einer Sprache wählen, dort aber für wichtige Bestandteile nur Strukturen, die bereits in den älteren Versionen problemlos verfügbar waren. Dies Vorgehen erhöht die Chance, ein Dokument mit gut definiertem Verhalten zu veröffentlichen, was sowohl von alten als auch zukünftigen Programmen hinreichend gut interpretiert werden kann.

Aktuell zeichnet sich zudem für die zweite Ausgabe der Empfehlung von SVG 1.1 ab, dass darin in einigen, zum Glück wenigen Fällen rückwärtsinkompatible Änderungen vorgenommen werden. Meistens passiert das nur für Bereiche, die ohnehing bislang von Darstellungsprogrammen fehlerhaft interpretiert wurden, wo also vermutet wird, dass sie ohnehin kein Autor verwendet hat. Da es allerdings keine Möglichkeit gibt, anzugeben, welcher Ausgabe der SVG 1.1-Empfehlung ein Dokument folgt, empfiehlt es sich, ganz auf solche problematischen Bereiche zu verzichten, da sie nicht eindeutig definiert sind. Bei anderen Bereichen kann auch explizit in der zweiten Ausgabe angegeben sein, dass sie nicht definiert sind, darauf sollte dann natürlich auch im Dokument verzichtet werden - oder zumindest sollte sich ein Autor darauf verlassen, dass verschiedene Darstellungsprogramme derart problematische Bereiche dann auch verscheiden darstellen werden oder können.

3.2.2 Attribut *baseProfile*

Das Attribut gibt das Basisprofil an, welches für das aktuelle Dokument verwendet wird. Das ist dann jeweils eine Auswahl von Elementen oder Attributen der jeweiligen Version.

Mögliche Werte sind derzeit 'none', 'full', 'basic' oder 'tiny'. 'none' liefert keine Information und wird angenommen, wenn das Attribut nicht gesetzt ist.

'tiny' gibt es für die Versionen 1.1 und 1.2 und kennzeichnet einen Teilbereich von SVG, der sich gut für Mobiltelephone und allgemein weniger leistungsstarke Geräte eignet. Das ist bislang das einzige Profil für die Version 1.2 und wird es vermutlich auch bleiben, da die Entwicklung eher dahingeht, dass zum einen neue Module formuliert werden, zum anderen dann eine Version 2.0 versucht wird, in der die Module und weitere neue Möglichkeiten vereint werden.

'basic' gibt es für die Version 1.1. Aufgrund des Leistungszuwachses bei Mobiltelephonen seit der Version 1.1 wurde auf diese Leistungsklasse zwischen der vollen Version und der für Mobiltelephone in 1.2 verzichtet und die 'tiny'-Variante gilt ganz allgemein für weniger leistungsstarke Geräte, grob orientiert am Leistungsstand von 2008.

'full' kennzeichnet den vollen Umfang der jeweiligen Version.

Ein Autor kann also abschätzen, dass tendenziell mehr Darstellungsprogramme seine Dokumente korrekt und vollständig interpretieren können, wenn es gelingt, nur Elemente, Attribute und Eigenschaften aus dem 'tiny'-Profil zu verwenden. Das volle Profil stellt höhere Ansprüche an das Darstellungsprogramm und die Wahrscheinlichkeit ist höher, dass Teile des Dokumentes in einigen Programmen nicht korrekt interpretiert werden.

Im Detail kann sich allerdings die Implementierungsstrategie bei verschiedenen Anbietern von Darstellungsprogrammen von diesen Profilen unterscheiden. Es wäre zu einfach anzunehmen, dass nur Programme veröffentlicht werden, die ein Profil komplett interpretieren können.

3.2.3 Attribute *width* und *height*

width und *height* geben die eigentliche Breite und Höhe des Dokumentes an, der Wert ist jeweils eine Länge größer oder gleich 0. Bei 0 wird das Element samt Inhalt nicht angezeigt. Negative Werte werden nicht unterstützt. Dies bedeutet in SVG tiny 1.2, dass das Attribut mit dem unzulässigen Wert ignoriert wird. In SVG 1.1 ist ein negativer Wert ein Fehler, der zum Abbruch der Darstellung führt.

width und *height* sind animierbar. Sofern nicht angegeben, wird als Wert 100% angenommen. Das entspricht dem gesamten verfügbaren Anzeigebereich in der jeweiligen Richtung.

3.2.4 Attribute *x* und *y*

Die Attribute haben für das Wurzelelement keine Bedeutung, nur wenn ***svg*** als Gruppenelement verwendet wird. Daher treten diese Attribute in den tiny-Profilen nicht auf.

Der Wert ist jeweils eine Koordinate, also wo das Element innerhalb des jeweiligen Koordinatensystems dargestellt werden soll. *x* und *y* geben dann die linke obere Ecke an. Sofern nicht angegeben, wird jeweils 0 angenommen.

Die Attribute sind animierbar.

3.2.5 Attribut *viewBox*

Abb. 19 Beispiel: 'Hello World' jetzt mit *viewBox*

Das Attribut gibt eine rechteckige Region an, in welche der darzustellende Inhalt fällt. Der Wert ist eine Liste von vier Zahlen, mit Leerraum oder Komma separiert. Die erste Zahl ist der minimale x-Wert, die zweite der minimale y-Wert, die dritte die Breite, die vierte die Höhe. Die darzustellenden Inhalte haben also Koordinaten bezogen auf dieses System, bei dem der Ursprung bei 0,0 liegt. Es können auch Elemente außerhalb des Bereiches der *viewBox* notiert werden, diese werden dann vermutlich aber gar nicht oder nur teilweise dargestellt.

Die *viewBox*-Region wird wiederum so skaliert, dass sie in den durch *width* und *height* definierten Anzeigebereich passt. Ist *viewBox* nicht angegeben, so fallen die Koordinaten mit denen des Anzeigebereiches zusammen, der dann den Ursprung links oben mit den Koordinaten 0,0 hat.

Negative Werte für Breite und Höhe sind unzulässig und sind in SVG1.1 ein Fehler, der zum Abbruch der Darstellung des Dokumentes führt. In SVG tiny 1.2 wird in dem Falle das Attribut einfach ignoriert.

In SVG tiny 1.2 kann auch der Wert 'none' angegeben werden, um explizit festzulegen, dass keine *viewBox* anzuwenden ist.

Das Attribut ist animierbar und ist auch für andere Elemente verfügbar, die einen neuen Anzeigebereich aufspannen.

In der Praxis ist die Verwendung von *viewBox* beim Wurzelelement in der Regel zu empfehlen. Dies erleichtert gegebenenfalls die Erweiterung des Inhaltes in der sichtbaren Zeichenebene und auch die Anpassung von der Größe her an verschiedene Anzeigebereiche.

Beispiel:

```
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.2" baseProfile="tiny"
     width="300" height="400"
     viewBox="0,0,100,100"
     xml:lang="en">
   <title>Hello World</title>
         <text x="50" y="80">Hello World</text>
</svg>
```

Ohne Angabe von *width* und *height* wird auf den Darstellungsbereich des Programmes skaliert, das entspricht jeweils einem Wert von 100%; entsprechend größer oder kleiner wird der Bereich, wenn andere Werte als 100% verwendet werden.

```
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.2" baseProfile="tiny"
     viewBox="0,0,100,100"
     xml:lang="en">
   <title>Hello World</title>
         <text x="50" y="80">Hello World</text>
</svg>
```

Beispiel mit *width* und *height* mit Einheiten:

```
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1"
     width="15cm" height="15cm"
     viewBox="0,0,15,15">
   <title>Quadrat</title>
     <rect x="5" y="5" width="5" height="5"/>
</svg>
```

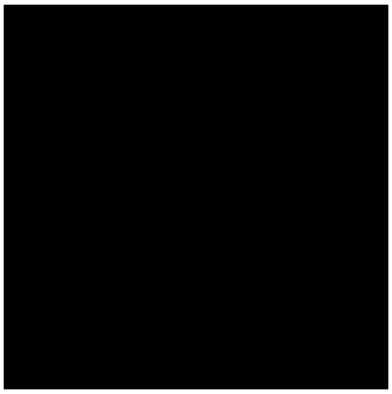


Abb. 20 Beispiel: Quadrat mit Seitenlänge 5cm

Die Größe des Anzeigebereiches ist auf 15cm x 15cm festgelegt und definiert die *viewBox*, so dass der Anzeigebereich innerhalb des Dokumentes nur 15 x 15 Einheiten groß ist. Da wiederum Breite und Höhe als 15cm angegeben sind, hat das Rechteck jetzt einen Abstand 5cm von oben, 5cm von der Seite mit einer Kantenlänge von 5cm.

Beispiel: Quadrat mit Seitenlänge 5cm[1]

Wenn die Auflösung von Drucker oder Bildschirm dem Darstellungsprogramme bekannt ist, sind die Angaben mit einem Lineal nachprüfbar.

1 `http://de.wikibooks.org/wiki/media%3ASVG%20Buch%20Rechteck1.svg`

3.2.6 Attribut *preserveAspectRatio*

Wenn *viewBox* angegeben ist und das Seitenverhältnis nicht zu *width* und *height* passt, kann mit *preserveAspectRatio* angegeben werden, wie die *viewBox* in den Anzeigebereich einzupassen ist.

In SVG tiny 1.1 können die Werte 'none' oder 'XMidYMid' (optional gefolgt von 'meet') angegeben werden. Ersterer Wert bedeutet, dass das Seitenverhältnis nicht erhalten bleiben muss und x- und y-Richtung unabhängig skaliert werden, bei den anderen Möglichkeiten bleibt immer das Seitenverhältnis erhalten. 'XMidYMid' bedeutet einfach, dass so skaliert wird, dass nichts übersteht. In der Richtung, in der dann im Anzeigebereich noch eine freie Fläche verbleibt, wird die skalierte *viewBox* zentriert. 'XMidYMid' ist bei allen Profilen die Voreinstellung.

Die Werte in SVG tiny 1.1 sind also 'none', 'none meet', 'XMidYMid' oder 'XMidYMid meet' - mit oder ohne meet bedeutet dies das gleiche.

In SVG tiny 1.2 gibt es weitere Werte:

'xMinYMin' - die skalierte *viewBox* wird links oben in den Anzeigebereich eingepaßt.

'xMidYMin' - in x-Richtung wird gegebenenfalls zentriert, in y-Richtung nach oben eingepaßt.

'xMaxYMin' - in x-Richtung wird gegebenenfalls rechts eingepaßt, in y-Richtung oben.

'xMinYMid' - in x-Richtung wird gegebenenfalls links eingepaßt, in y-Richtung zentriert.

'xMaxYMid' - in x-Richtung wird gegebenenfalls rechts eingepaßt, in y-Richtung zentriert.

'xMinYMax' - in x-Richtung wird gegebenenfalls links eingepaßt, in y-Richtung unten.

'xMidYMax' - in x-Richtung wird gegebenenfalls zentriert, in y-Richtung nach unten eingepaßt.

'xMaxYMax' in x-Richtung wird gegebenenfalls rechts eingepaßt, in y-Richtung nach unten.

In der Vollversion von SVG 1.1 kann anstatt 'meet' auch 'slice' angegeben werden. damit wird erreicht, dass der Anzeigebereich immer komplett ausgefüllt wird - die zu lange Richtung wird dann gegebenenfalls abgeschnitten.

Dies kann sehr hilfreich sein, wenn zum Beispiel der gesamte Anzeigebereich mit Höhe und Breite gleich hundert Prozent garantiert mit einem Muster oder einer Struktur gefüllt werden soll, die nur endlich groß ist. Damit dies unabhängig vom Aspektverhältnis funktioniert, wird mit der 'slice'-Variante das maximale Rechteck angegeben, welches noch komplett vom Muster oder der Struktur abgedeckt wird. Bei abweichendem Aspektverhältnis wird dieses Rechteck dann entsprechend beschnitten und skaliert.

Ein in SVG tiny 1.2 oder in der Vollversion von SVG 1.1 vorangestelltes 'defer' ist für ein Element ***svg*** ohne Bedeutung. Das Attribut kann jedoch auch bei anderen Elementen, die einen neuen Anzeigebereich öffnen, angewendet werden. Wenn Dateien referenziert werden, die selbst ein bevorzugtes Seitenverhältnis definieren, ist diese Angabe der referenzierten Dateien denen des Attributes vorzuziehen, wenn 'defer' gesetzt ist.

Das Attribut ist animierbar.

Beispiele:

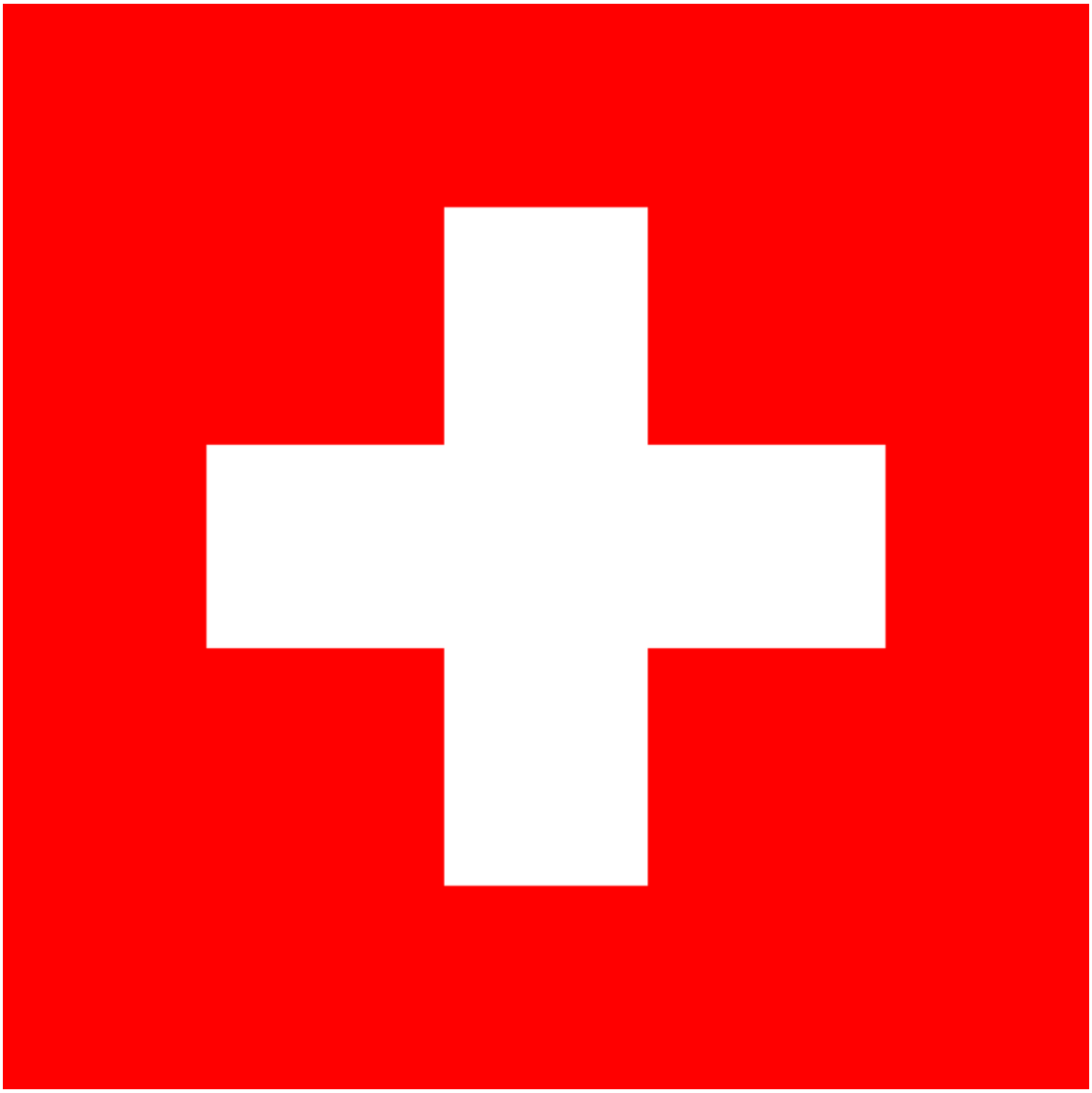


Abb. 21 Die Schweizer Flagge ganz normal

Hier der Quelltext der Schweizer Flagge:

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1"
```

```
     width="300" height="300"
     xml:lang="de">
   <title>Schweizer Flagge</title>
<rect fill="#f00" height="300" width="300"/>
<rect fill="#fff" height="60" width="200" x="50" y="120"/>
<rect fill="#fff" height="200" width="60" x="120" y="50"/>
</svg>
```

Alternativ eine mittige Anordnung:

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1"
     width="600" height="300"
     viewBox="0 0 300 300"
     preserveAspectRatio="xMidYMid meet"
     xml:lang="de">
   <title>Schweizer Flagge</title>
<rect fill="#f00" height="300" width="300"/>
<rect fill="#fff" height="60" width="200" x="50" y="120"/>
<rect fill="#fff" height="200" width="60" x="120" y="50"/>
</svg>
```

Oder nach unten rechts an den Rand gedrängt:

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1"
     width="600" height="300"
     viewBox="0 0 300 300"
     preserveAspectRatio="xMaxYMax meet"
     xml:lang="de">
   <title>Schweizer Flagge</title>
<rect fill="#f00" height="300" width="300"/>
<rect fill="#fff" height="60" width="200" x="50" y="120"/>
<rect fill="#fff" height="200" width="60" x="120" y="50"/>
</svg>
```

Oder links oben, gegebenenfalls abgeschnitten:

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1"
     width="200" height="300"
     viewBox="0 0 300 300"
     preserveAspectRatio="xMinYMin slice"
     xml:lang="de">
   <title>Schweizer Flagge</title>
<rect fill="#f00" height="300" width="300"/>
<rect fill="#fff" height="60" width="200" x="50" y="120"/>
<rect fill="#fff" height="200" width="60" x="120" y="50"/>
</svg>
```

Um einen weiteren Anzeigebereich in einem SVG-Dokument zu etablieren, kann in der Vollversion von SVG 1.1 ein ***svg***-Element in einem ***svg***-Element gesetzt werden, wobei ebenfalls die Attribute *viewBox* und *preserveAspectRatio* darüber entscheiden, wie diese ***svg***-Elemente in den Anzeigebereich eingepaßt werden:

```
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1"
     viewBox="-100 -100 400 400"
     preserveAspectRatio="xMinYMin slice"
     xml:lang="de">
```

```
   <title>SVG-Beispiel zum Element svg</title>
 <svg width="100" height="200" x="-80" y="-90"
      viewBox="0 0 60 200">
 </svg>
 <svg width="100" height="200" x="200" y="100"
      viewBox="0 0 30 30"
      preserveAspectRatio="xMaxYMax meet">
 </svg>
</svg>
```

3.2.7 Attribut *zoomAndPan*

Ein interaktives Darstellungsprogramm sollte eine Funktionalität aufweisen, die es erlaubt, das Dokument zu vergrößern oder zu verkleinern, zu rotieren oder den Anzeigebereich zu verschieben. Dieses kann mit dem Attribut *zoomAndPan* beeinflusst werden.

Mit dem Wert 'disable' wird dies unterbunden, mittels 'magnify' erlaubt. Letzteres ist die Voreinstellung.

3.2.8 Attribut *snapshottime*

Das Attribut wurde mit SVG tiny 1.2 neu eingeführt und ist daher in SVG 1.1 nicht verfügbar.

Mittels *snapshottime* kann ein Zeitpunkt angegeben werden, welcher charakteristisch für ein Standbild des Dokumentes ist - besonders relevant also für animierte Dokumente. Dies ist lediglich ein hilfreicher Hinweis für Darstellungsprogramme, die Vorschaubilder bereitstellen oder gegebenenfalls auch für eine Druckausgabe oder eine Konversion in ein anderes Format ein charakteristisches Standbild erzeugen sollen. Der Wert ist ein Zeitwert, wie im Abschnitt zur Animation genauer erläutert, zum Beispiel '60s' bedeutet 60 Sekunden nach dem Startzeitpunkt des Dokumentes. Alternativ kann auch 'none' angegeben werden, wenn es solch einen charakteristischen Zeitpunkt nicht gibt. Dies ist die Voreinstellung.

3.2.9 Attribut *playbackOrder*

Das Attribut wurde mit SVG tiny 1.2 neu eingeführt und ist daher in SVG 1.1 nicht verfügbar.

In SVG tiny 1.2 gibt es auch eine Möglichkeit, Teile des Dokumentes aus dem Speicher zu löschen, um diesen zu entlasten. Ist für *playbackOrder* der Wert 'forwardOnly' angegeben, soll in dem Dokument nicht zeitlich rückwärts gesucht werden können, weil es solche Löschungen enthalten kann. Beim Wert 'all' sind solche Löschungen nicht zu erwarten und eine entsprechende Suche ist erlaubt, letzteres ist die Voreinstellung.

3.2.10 Attribut *timelineBegin*

Das Attribut wurde mit SVG tiny 1.2 neu eingeführt und ist daher in SVG 1.1 nicht verfügbar.

Für die Synchronisation zeitabhängiger Bestandteile von Dokumenten, wie Animationen, Video und Audio ist ein präziser Zeitnullpunkt vorzugeben. Mit diesem Attribut kann zwischen zwei Zeitpunkten gewählt werden. Mit dem Wert 'onLoad' ist der gleiche Zeitpunkt wie für Dokumente in SVG 1.1

gemeint. Dies ist der Zeitpunkt, wenn das Dokument bis zur Endmarkierung des Wurzelelementes geladen wurde. Mittels 'onStart' liegt der Zeitnullpunkt bereits vor, wenn die Startmarkierung des Wurzelelementes komplett interpretiert wurde.

Der Unterschied kann relevant sein, wenn das Darstellungsprogramm den Inhalt noch während des Ladens progressiv aufbaut. Die Alternative ist, die Graphik komplett unsichtbar für den Betrachter im Speicher zu malen und dann mit einem Male darzustellen. Ein progressiv arbeitendes Programm kann bereits beginnen, die ersten Teile des Inhaltes darzustellen, wenn erst der Anfang der Datei geladen wurde.

Im ersten Falle mit 'onLoad' wird bei einer progressiven Darstellung also erst alles komplett geladen und gegebenenfalls nacheinander angezeigt, dann ergibt sich der Zeitnullpunkt. Im zweiten Falle liegt der Zeitnullpunkt bereits vor, wenn die Startmarkierung des ***svg*** komplett geladen und interpretiert wurde, in der Regel also unmittelbar bevor der erste Inhalt angezeigt wird. Im anderen Falle eines nicht progressiven Dokumentaufbaus wird davon ausgegangen, dass während des Ladens die Darstellung vorbereitet wird und nach Abschluss des Ladens unmittelbar die Darstellung des kompletten Inhaltes beginnt. Bei weniger effektiv arbeitenden Programmen kann also 'onLoad' auch dazu führen, dass der Zeitnullpunkt vor dem Moment liegt, wo das Dokument angezeigt wird, wenn bei solch einem Programm zwischen dem Laden und der Darstellung noch eine nennenswerte Verarbeitungszeit liegt.

Zum Beispiel kann eine mehrfache Verschachtelung von ***use***-Elementen auch bei recht kleinen SVG-Dokumenten zu einer beachtlichen Verarbeitungszeit führen, weil auf diese Weise eine sehr große Anzahl von Kopien erzeugt werden kann.

3.2.11 Attribut *contentScriptType*

Wenn in einem Dokument Skripte vorkommen, kann mit dem Attribut *contentScriptType* die vorausgewählte Skriptsprache angegeben werden. Der Typ ist ein Medientyp (ehemals MIME-Typ). Typisch für Skripte innerhalb von SVG ist zum Beispiel 'application/ecmascript', was in SVG tiny 1.2 auch die Voreinstellung ist, wenn das Attribut nicht angegeben ist. Sofern die einzelnen Skriptelemente selbst angeben, welche Skriptsprache sie verwenden, ist deren Angabe von höherem Gewicht. Der Wert dieses Attributes ist also nur ein Rückfallwert für den Fall, dass die einzelnen Skriptelemente keine Angaben beigefügt haben. In SVG 1.1 ist der voreingestellte Wert noch der veraltete Wert 'text/ecmascript', der nicht mehr verwendet werden sollte, seitdem 'application/ecmascript' verfügbar ist.

Trotz der Voreinstellung ist in SVG keine Skriptsprache vorgesehen, die interpretiert werden müsste. Wenn überhaupt, bietet sich jedoch eine an, die das in SVG definierte Dokument-Objekt-Modell (DOM) ausnutzen kann.

3.2.12 Attribut *contentStyleType*

In der Vollversion von SVG 1.1 kann mit dem Attribut *contentStyleType* der voreingestellte Typ für Stilvorlagen angegeben werden. Der Typ ist ein Medientyp (ehemals MIME-Typ). Typisch ist 'text/css', was auch die Voreinstellung ist. Damit wird die Stilvorlagensprache angegeben, die innerhalb des allgemein verfügbaren Attributes *style* verwendet wird.

3.2.13 Beispiel zu *svg*

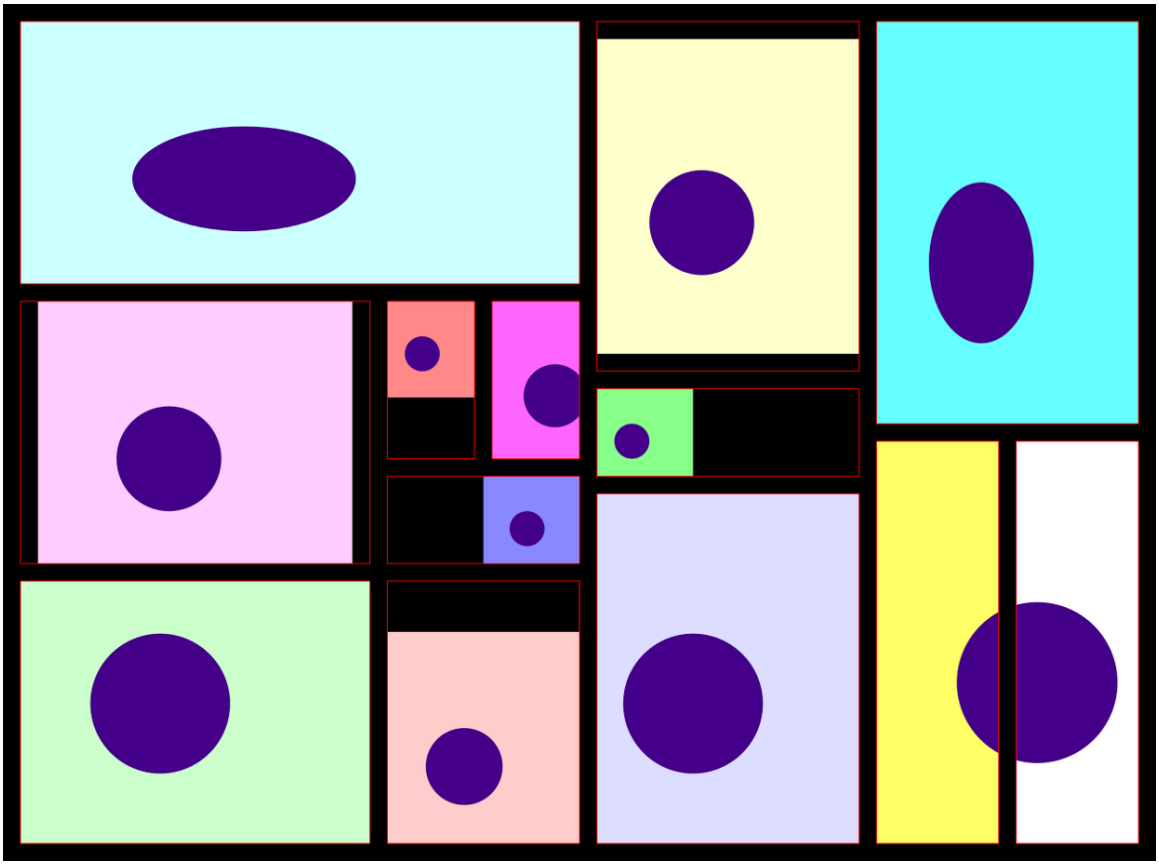


Abb. 22 Beispiel zur Verwendung von ***svg*** als Gruppierungselement

Beispiel zur Verwendung von ***svg*** als Gruppierungselement[2].

In diesem Dokument werden mehrere ***svg***-Elemente verwendet, welche jeweils ein Rechteck und einen Kreis enthalten. *viewBox* ist immer gleich angegeben, Größe oder *preserveAspectRatio* aber unterschiedlich.

Der Anzeigebereich der jeweiligen ***svg***-Elemente ist jeweils mit einem rot gerandeten Rechteck angedeutet.

W3C ***svg***[3]

2 `http://de.wikibooks.org/wiki/media%3ASVGsvg01.svg`
3 `http://www.w3.org/TR/SVG/struct.html#SVGElement`

3.3 Element *g*, allgemeine Gruppierung

Mit dem Element ***g*** können mehrere graphische Elemente zu einer Einheit zusammengefasst werden, die dann zum Beispiel gemeinsam referenziert werden können oder auch transformiert oder mit Eigenschaften ausgezeichnet werden können, die vom ***g*** geerbt werden.

Ferner kann eine solche Gruppe auch mittels der Elemente ***title***, ***desc*** und ***metadata*** eine gemeinsame Überschrift, Beschreibung und allgemein Metainformationen zugeordnet bekommen. Auch wenn bedingte Verarbeitung mit ***switch*** für mehrere Elemente zusammen gelten soll, empfiehlt sich eine Gruppierung mittels ***g***.

Das Element selbst hat keine Attribute oder Eigenschaften, die für dieses Element spezifisch sind. Anders als das Element ***svg*** definiert es keinen neuen Anzeigebereich und hat keine spezifische inhaltliche Bedeutung, ähnlich dem Element ***span*** in (X)HTML.

W3C ***g***[4]

3.4 Element *defs*, Definitionen

Das Element ***defs*** ist ein Behälter für Gruppen oder Elemente, die nicht direkt dargestellt werden, sondern referenziert werden. Die für effektive Verarbeitung empfohlene Platzierung dieses Elementes ist soweit wie möglich vorne im Quelltext, also am besten unmittelbar hinter ***title***, ***desc*** und ***metadata*** des Hauptelementes ***svg***.

Autoren sollten jeglichen Inhalt, der nicht direkt dargestellt wird, in diesem Element unterbringen, damit das Dokument auch hinsichtlich der Zugänglichkeit übersichtlich bleibt und effektiv verarbeitet werden kann. Die Anordnung referenzierter Elemente hat ferner Einfluss auf eine mögliche progressive Darstellung. Liegt der referenzierte Inhalt vor der Referenzierung, begünstigt dies eine progressive Darstellung, liegt der referenzierte Inhalt hinter der Referenzierung, verzögert dies die progressive Darstellung.

Das Element selbst hat keine Attribute oder Eigenschaften, die für dieses Element spezifisch sind.

W3C ***defs***[5]

3.5 Elemente *title* und *desc*, Titel und Beschreibung

Graphische Elemente und Gruppierungselemente können jeweils einen Titel und eine Beschreibung haben. Für den Titel wird das Element ***title*** verwendet, für die Beschreibung das Element ***desc***. Beide sollten nicht mehr als einmal als Kindelement eines Elementes vorkommen. ***title*** sollte gegebenenfalls das erste Kindelement sein, gegebenenfalls direkt gefolgt von ***desc*** (und dann ***metadata***).

4 `http://www.w3.org/TR/SVG/struct.html#Groups`
5 `http://www.w3.org/TR/SVG/struct.html#DefsElement`

Autoren sollten dem Hauptelement ***svg*** immer einen sinnvollen Titel geben. Eine hilfreiche Beschreibung ist ebenfalls zu empfehlen. Nähere Informationen dazu sind dem Kapitel Barrierefreiheit[6] zu entnehmen.

Im einfachsten Falle ist der Inhalt dieser Elemente einfacher Text. Dieser Inhalt wird nicht zusammen mit dem sonstigen graphischen Inhalt dargestellt. Wie dieser Inhalt vom Darstellungsprogramm dargestellt wird, ist nicht eindeutig festgelegt. Jedenfalls soll der Inhalt dem Nutzer zugänglich gemacht werden, wie hängt von den Möglichkeiten des Darstellungsprogrammes und der Art der Präsentation selbst ab.

Die Elemente selbst haben keine Attribute oder Eigenschaften, die für diese Elemente spezifisch sind.

W3C ***title*** und ***desc***[7]

3.6 Element *symbol*, Kombination zu einem Symbol

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5
Konqueror (KSVG)	3.2
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Das Element ***symbol*** dient in SVG dazu, Dokumente semantisch zu strukturieren. Einzelne Elemente werden zu einer Einheit, einem Symbol kombiniert.

Wie das Element ***svg*** öffnet das Element ***symbol*** einen eigenen Anzeigebereich. Der Unterschied liegt vor allem in der semantischen Aussage des Elementnamens. Zudem wird ***symbol*** auch nicht direkt angezeigt, sondern mit dem Element ***use*** referenziert, wo es angezeigt werden soll. Somit bestimmt das ***use*** Positionierung und Größe des Symbols.

Das Element ist in der Vollversion von SVG 1.1 verfügbar, nicht in SVG tiny 1.1 oder tiny 1.2.

3.6.1 Attribut *viewBox*

Das Attribut *viewBox* gibt eine rechteckige Region an, in welche der darzustellende Inhalt fällt. Der korrekte Wert ist bereits für das Element ***svg*** beschrieben worden.

Das Attribut ist animierbar.

6 Kapitel 4 auf Seite 77

7 `http://www.w3.org/TR/SVG/struct.html#DescriptionAndTitleElements`

3.6.2 Attribut *preserveAspectRatio*

Wenn das Seitenverhältnis der *viewBox* des referenzierten Symbols nicht zu *width* und *height* des referenzierenden ***use*** passt, kann mit *preserveAspectRatio* angegeben werden, wie es in den Anzeigebereich einzupassen ist.

Die möglichen Werte sind bereits entsprechend für das Element ***svg*** beschrieben.

Das Attribut ist animierbar.

3.6.3 Beispiel zu *symbol*

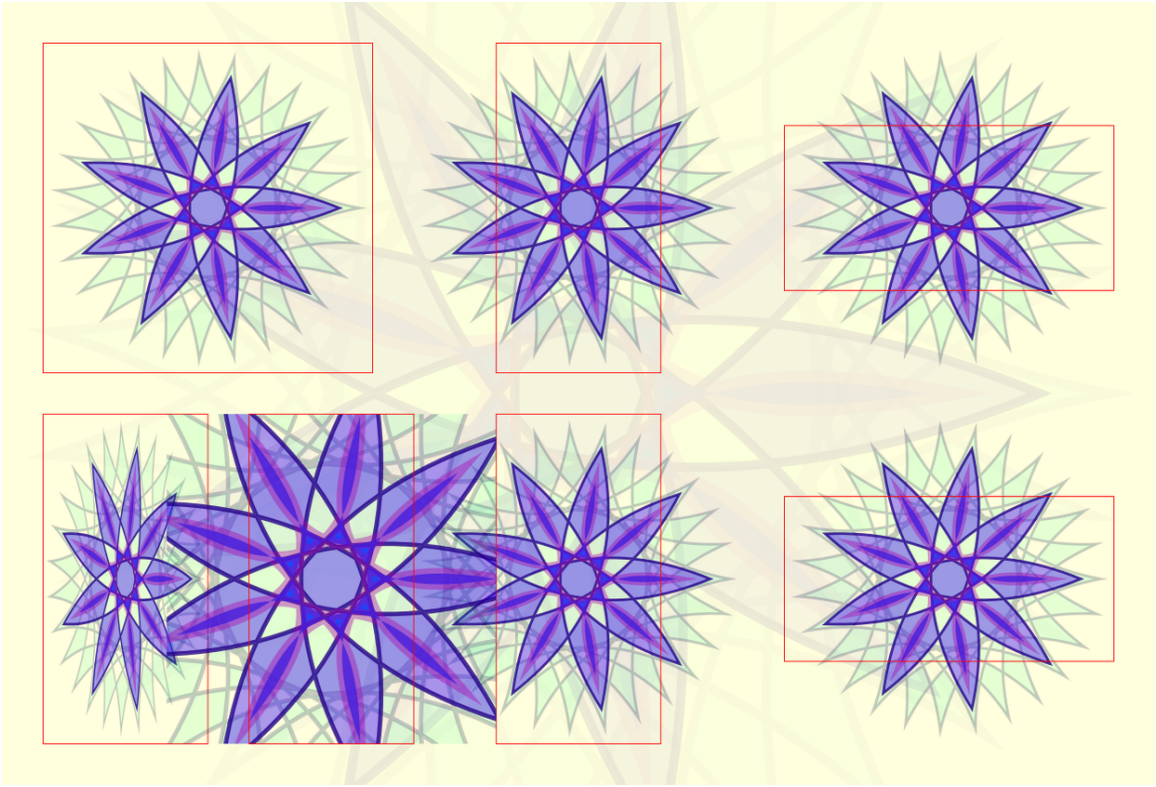


Abb. 23 Beispiel zur Verwendung von ***symbol***

Beispiel zur Verwendung von ***symbol***[8].

Dies Dokument zeigt einige Beispiele für das Zusammenspiel von ***symbol*** mit *viewBox* und *preserveAspectRatio* und der Eigenschaft *overflow*.

Der Anzeigebereich ist jeweils rot gerandet angedeutet.

W3C ***symbol***[9]

8 `http://de.wikibooks.org/wiki/media%3ASVGsymbol01.svg`
9 `http://www.w3.org/TR/SVG/struct.html#SymbolElement`

3.7 Element *use*, Wiederverwendung

Einzelne Elemente oder Gruppen können als Vorlage verwendet und dann wiederverwendet werden. Diesem Zwecke dient das Element ***use***. Es werden keine kompletten Dokumente wiederverwendet, sondern lediglich SVG-Dokumentfragmente. In den tiny-Profilen darf das Element auch kein komplettes ***svg***-Element referenzieren, in SVG tiny 1.1 nur Fragmente aus demselben Dokument.

Die Vererbung von Eigenschaften in die wiederverwendete Vorlage erfolgt vom Element ***use*** aus, beziehungsweise von dessen Elternelementen, nicht von denen der referenzierten Vorlage.

Mit den Attributen *x* und *y* wird der Ursprung für den referenzierten Inhalt festgelegt.

Der effektive visuelle Effekt lässt sich wie folgt beschreiben:

- ***use*** wird durch ***g*** repräsentiert.
- Attribute und Eigenschaften werden auf ***g*** übertragen, mit der Ausnahme von *x*, *y*, *width*, *height* und *href* von XLink.
- Eine Transformation translate(*x*, *y*) wird am Ende der Transformationsliste durchgeführt.
- Der Inhalt der referenzierten Vorlage wird in das g-Element kopiert.
- Im Falle, dass ein ***svg***-Element referenziert wird, überschreiben die Attribute *width* und *height* im Element ***use*** gegebenenfalls die des Elementes ***svg***.
- Wenn ein Element ***symbol*** referenziert wird, wird dies durch ein ***svg*** ersetzt und *width* und *height* werden gegebenenfalls entsprechend überschrieben. Sofern nicht angegeben, werden sie als 100% angenommen.

Referenzierte Elemente ***symbol*** oder ***svg*** können eine wirksame Eigenschaft *overflow* haben, welche darüber entscheidet, ob überstehender Inhalt sichtbar ist oder nicht.

Eine rekursive direkte oder indirekte Selbstreferenzierung ist ein Fehler, der zum Abbruch der Darstellung des Dokumentes führt. Praktisch hängt es allerdings vom Darstellungsprogramm ab, wie schnell bei indirekten Schleifen solch eine unzulässige Konstruktion entdeckt wird.

Allerdings können andere Elemente ***use*** wiederverwendet werden, womit dann eine endliche und erlaubte Rekursion erreicht werden kann, solange dies nicht zu einer indirekten Selbstreferenzierung führt.

Für das Element spezifische Attribute:

3.7.1 Attribute *x* und *y*

Der Wert ist jeweils eine Koordinate, also wo das Element innerhalb des jeweiligen Koordinatensystems dargestellt werden soll, *x* und *y* geben dann den Ursprung an. Sofern nicht angegeben, wird jeweils 0 angenommen.

Die Attribute sind animierbar.

3.7.2 Attribute *width* und *height*

width und *height* geben die Breite und Höhe an, falls ein ***symbol*** oder ***svg*** als referenzierte Vorlage dient. In den tiny-Profilen werden diese Attribute nicht gebraucht und sind daher in SVG tiny 1.2

nicht verfügbar und in SVG tiny 1.1 wirkungslos, sofern die Attributwerte größer als 0 sind, dies gilt auch für das volle Profil, wenn die Vorlage nicht selbst einen eigenen Anzeigebereich öffnet.

Der Wert ist jeweils eine Länge größer oder gleich 0. Bei 0 wird das Element samt Inhalt nicht angezeigt. In SVG 1.1 ist ein negativer Wert ein Fehler, der zum Abbruch der Darstellung führt. Wenn nicht angegeben, hängt das Verhalten vom referenzierten Element ab, siehe oben.

Die Attribute sind animierbar.

3.7.3 Attribute von XLink, insbesondere *href*

Um die Vorlage zu referenzieren, wird das Attribut *href* von XLink verwendet. Typisch wird das Präfix 'xlink' für den Namensraum von XLink verwendet. Sofern so angegeben, wird das Attribut dann als *xlink:href* notiert. Entsprechendes gilt für die anderen in SVG verfügbaren Attribute von XLink, die bei ***use*** somit ebenfalls verwendbar sind.

Das Attribut *href* ist animierbar.

Ein gültiger Wert ist eine URI mit Fragmentidentifizierer oder nur ein Fragmentidentifizierer. In SVG tiny 1.1 ist es immer ein Fragmentidentifizierer im selben Dokument. Beispiel:

```
<g fill="blue">
<circle id="Beispiel" r="20" />
</g>
<use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#Beispiel" x="50" fill="red" />
```

Bei dem Beispiel wird also ein blauer Kreis mit dem Mittelpunkt bei 0,0 dargestellt und ein roter Kreis mit einem Mittelpunkt bei 50,0.

So ergeben sich übrigens zwei blaue Kreise, weil das Präsentationsattribut *fill* hier direkt im Element ***circle*** notiert ist, nicht in einer übergeordneten Gruppe:

```
<circle id="Beispiel" r="20" fill="blue" />
<use xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="#Beispiel" x="50" fill="red" />
```

Anmerkung zu den Beispielen: Sofern der Namensraum für XLink bereits bei einem der Elternelemente mit dem entsprechenden Präfix festgelegt wurde, ist dies natürlich beim jeweiligen Element nicht abermals notwendig, welches ein Attribut aus dem Namensraum verwendet.

3.7.4 Anwendungsbeispiele für *use*

Wiederverwendung von Inhalten ist bei Graphik an vielen Stellen nützlich und hilft, Arbeit zu sparen und die Dateigröße zu reduzieren. Bei Landschaftsarchitektur und Architektur etwa mögen häufig sich wiederholende abstrahierte Formen auftauchen, zum Beispiele Bäume oder immer wiederkehrende Wohnelemente in größeren Wohneinheiten. Bei anderen Anwendungen mögen sich Logos oder Symbole häufiger wiederholen. Eine weitere Anwendung ergibt sich, wenn bei zwei ***use***-Elementen die referenzierten Inhalte per Animation vertauscht werden, damit ergibt sich dann eine Vertauschung der Zeichenreihenfolge.

Abb. 24 Klonschaf Dolly

Klonschaf Dolly[10]

Ein geläufiges Beispiel aus der Biologie, Genetik und Fortpflanzungstechnologie ist das Klonen von Lebewesen, was mit SVG für Abbildungen von Lebewesen ungleich einfacher zu realisieren ist.

Das zunächst als Pfad definierte Schaf wird in einer Gruppe ***g*** zweimal mit ***use*** referenziert. Diese Gruppe wird erneut doppelt in einer Gruppe referenziert und diese abermals.

10 `http://de.wikibooks.org/wiki/media%3ASvgTutorialSchaf.svg`

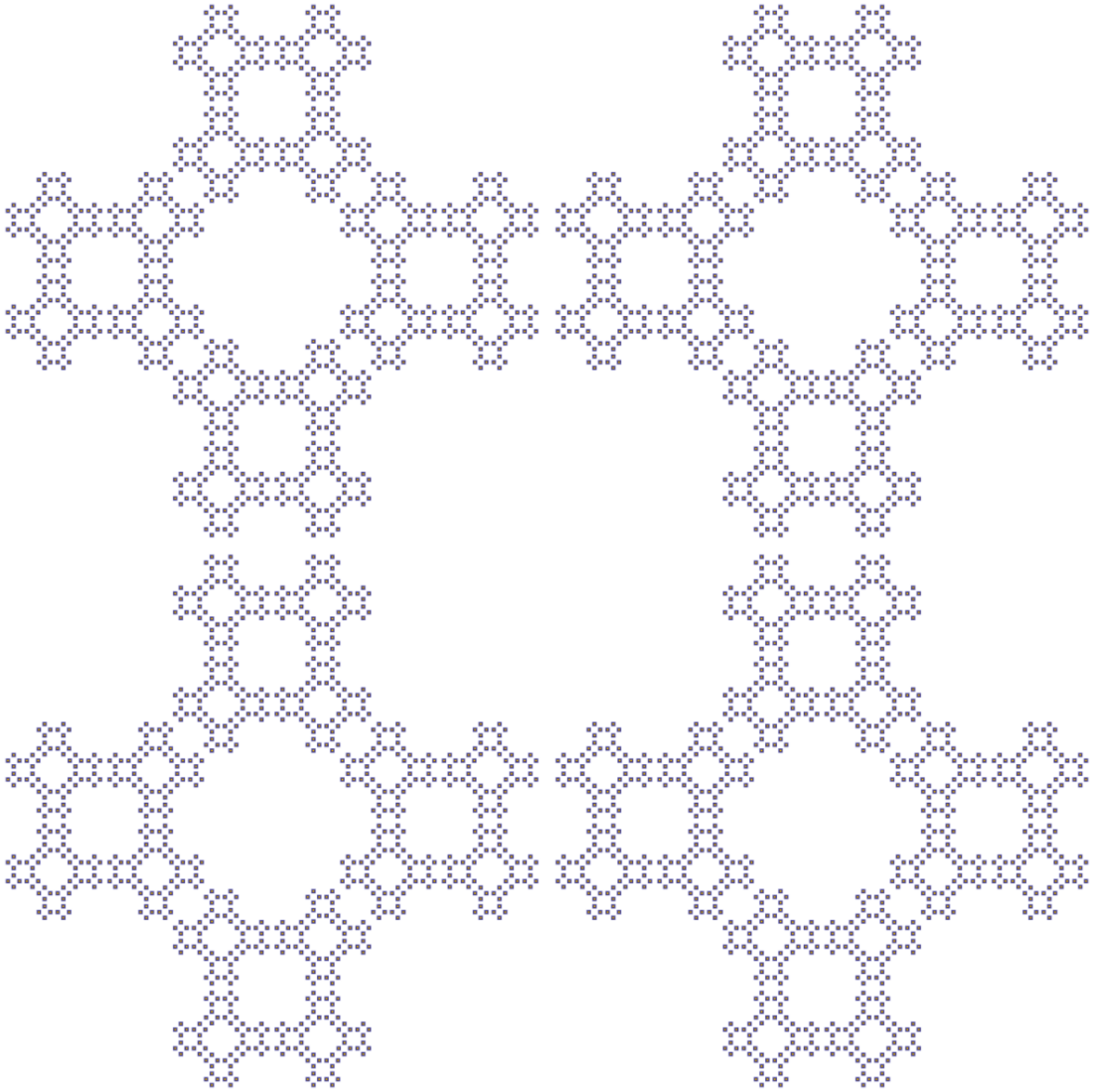


Abb. 25 IFS-Fraktal mit SVG

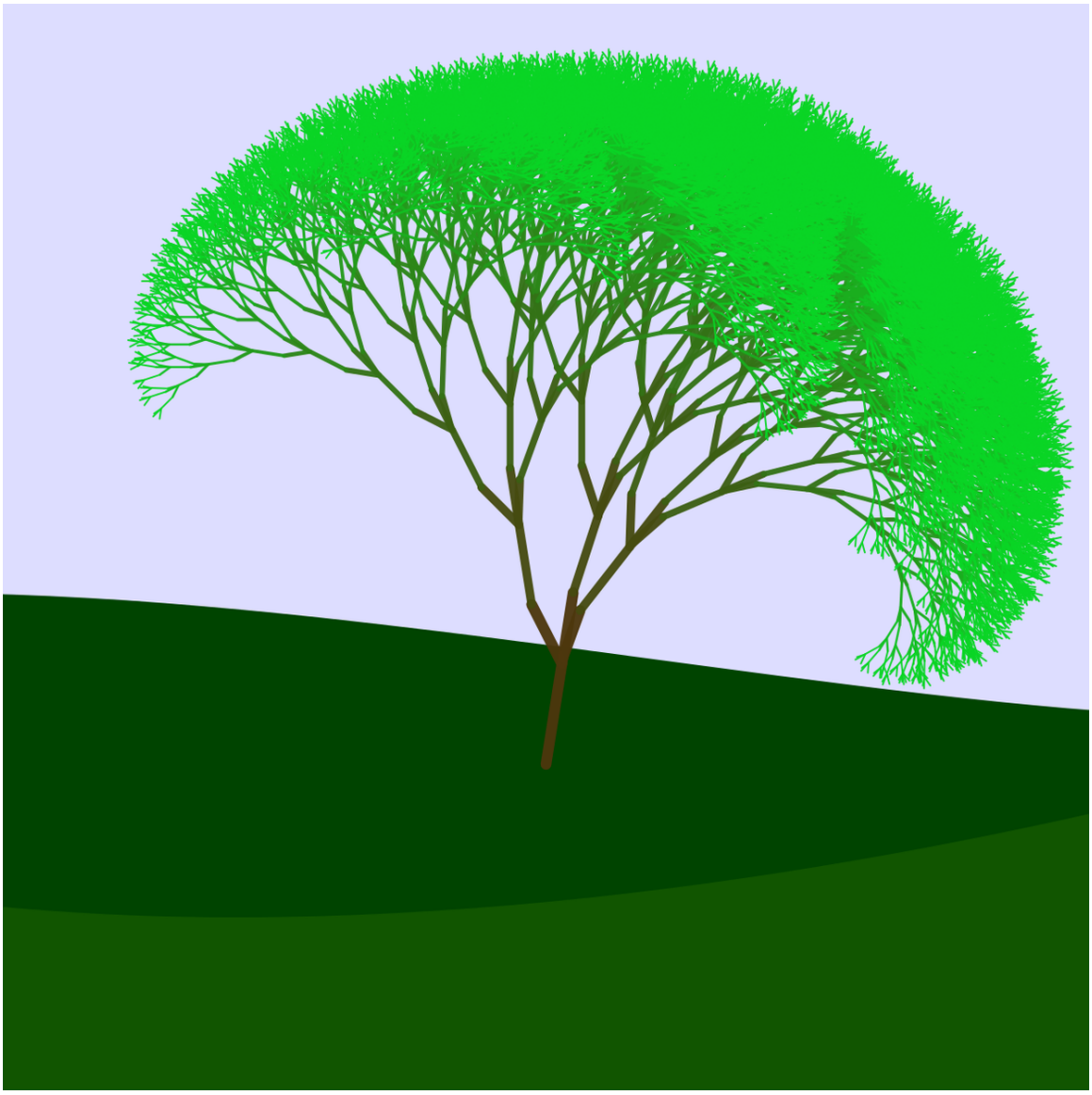


Abb. 26 Fraktaler Baum in kubischer Landschaft

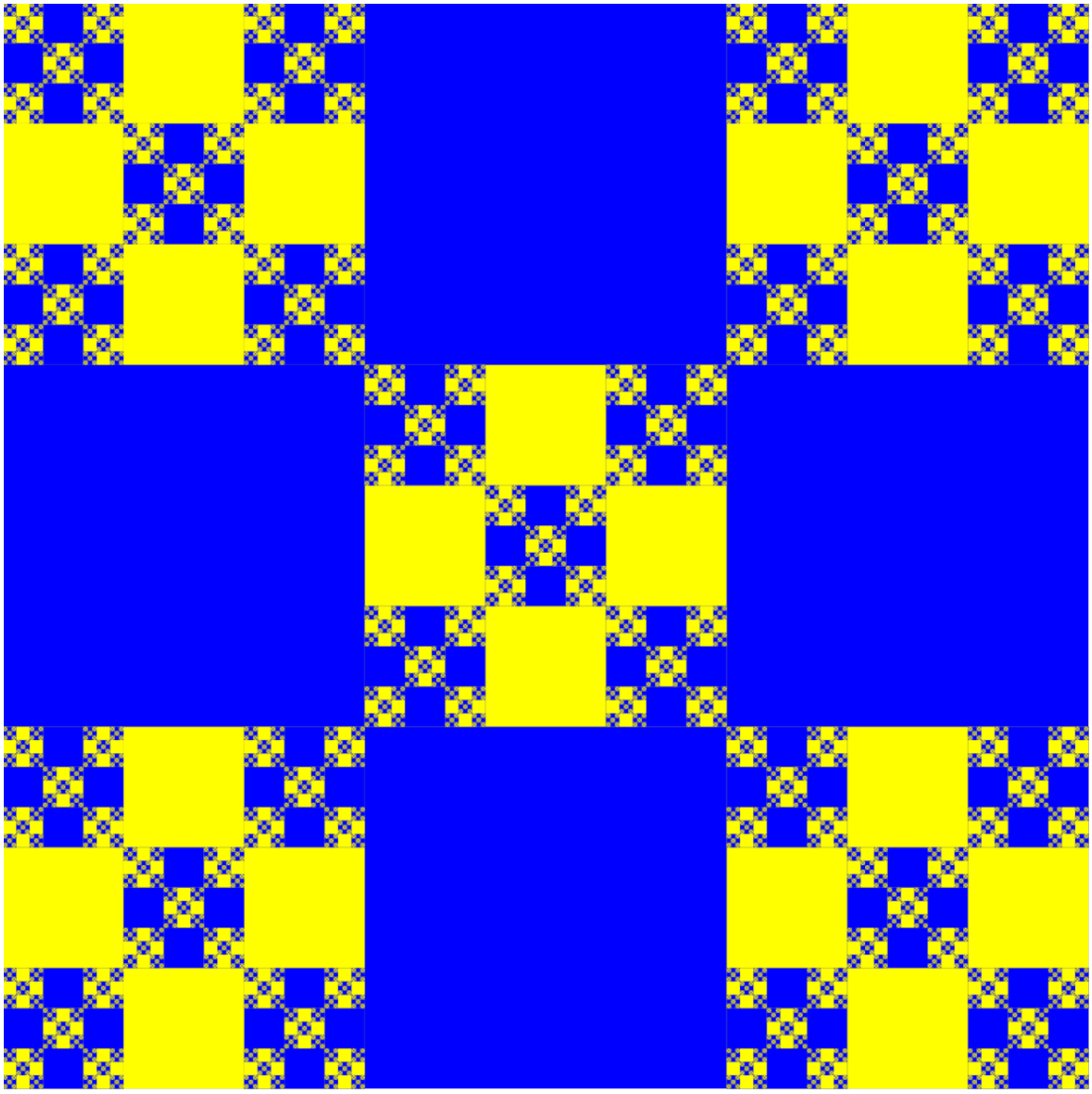


Abb. 27 Darstellung einer Cantor-Menge mit SVG

Die folgenden Anwendungsbeispiele sind eher der Mathematik entlehnt. Weil durch die mehrmalige Referenzierung von ***use***-Elementen sehr viele Elemente dargestellt werden müssen, kann es ein paar Sekunden dauern, bis die Beispiele dargestellt werden. Die Dateien selbst sind eher klein, die Dauer bis zur Darstellung ist also grob ein Maß für die Leistungsfähigkeit des verwendeten Programmes und Computers. Für WebKit (getestet mit Versionen 523 und 528; Safari4, Google-Chrome4) zum Beispiel ist bekannt, dass dieses Darstellungsprogramm zu faul ist, um so viele Element überhaupt darzustellen, bei späteren Versionen kann der Fehler allerdings bereits behoben worden sein - die folgenden Beispiele decken solche Mängel eines Darstellungsprogrammes leicht auf.

IFS-Fraktal mit SVG[11]

11 `http://de.wikibooks.org/wiki/media%3ASVG_IFS01.svg`

Beispiel IFS-Fraktal mit SVG - Iterierte Funktionen-Systeme.

Fraktaler Baum in kubischer Landschaft[12]

Ein baumartiges Fraktal in stark abstrahierter Landschaft, geformt aus zwei kubischen Kurven.

Technik: IFS-Fraktal mit SVG - Iterierte Funktionen-Systeme.

Darstellung einer Cantor-Menge mit SVG[13]

Als Ausgangsmuster wird ein Quadrat gewählt. Die Entstehung der Cantormenge kann man sich so vorstellen, dass das Quadrat in neun gleichgroße Unterquadrate eingeteilt wird, die vier in den Seitenmitten werden entfernt. Auf die verbleibenden fünf Quadrate wird im nächsten Iterationsschritt das gleiche Verfahren angewendet - erneut wird in neun Quadrate unterteilt, von denen wiederum die vier in den Seitenmitten entfernt werden - und so weiter.

Zur konkreten Umsetzung mit SVG wird mit einem Quadrat begonnen, dies wird auf die fünf verbleibenden Positionen kopiert und auf ein Drittel verkleinert. Diese Gruppe wird wiederum kopiert und skaliert und so weiter. Die Gruppen werden abwechselnd in blau und gelb dargestellt, um den Konstruktionsmechanismus zu visualisieren. Die entfernten Bereiche sind jeweils die sichtbaren Quadrate. Es verbleiben gewisse Teile der Quadrate, zum Beispiel die Eckpunkte, die als Punkte an sich nicht darstellbar sind.

Um die Struktur zu verdeutlichen, wird eine Animation verwendet, der Darstellungsbereich wird auf das zentrale Quadrat verkleinert.

12 `http://de.wikibooks.org/wiki/media%3ASVG_IFS02.svg`
13 `http://de.wikibooks.org/wiki/media%3ASVG_Cantor01.svg`

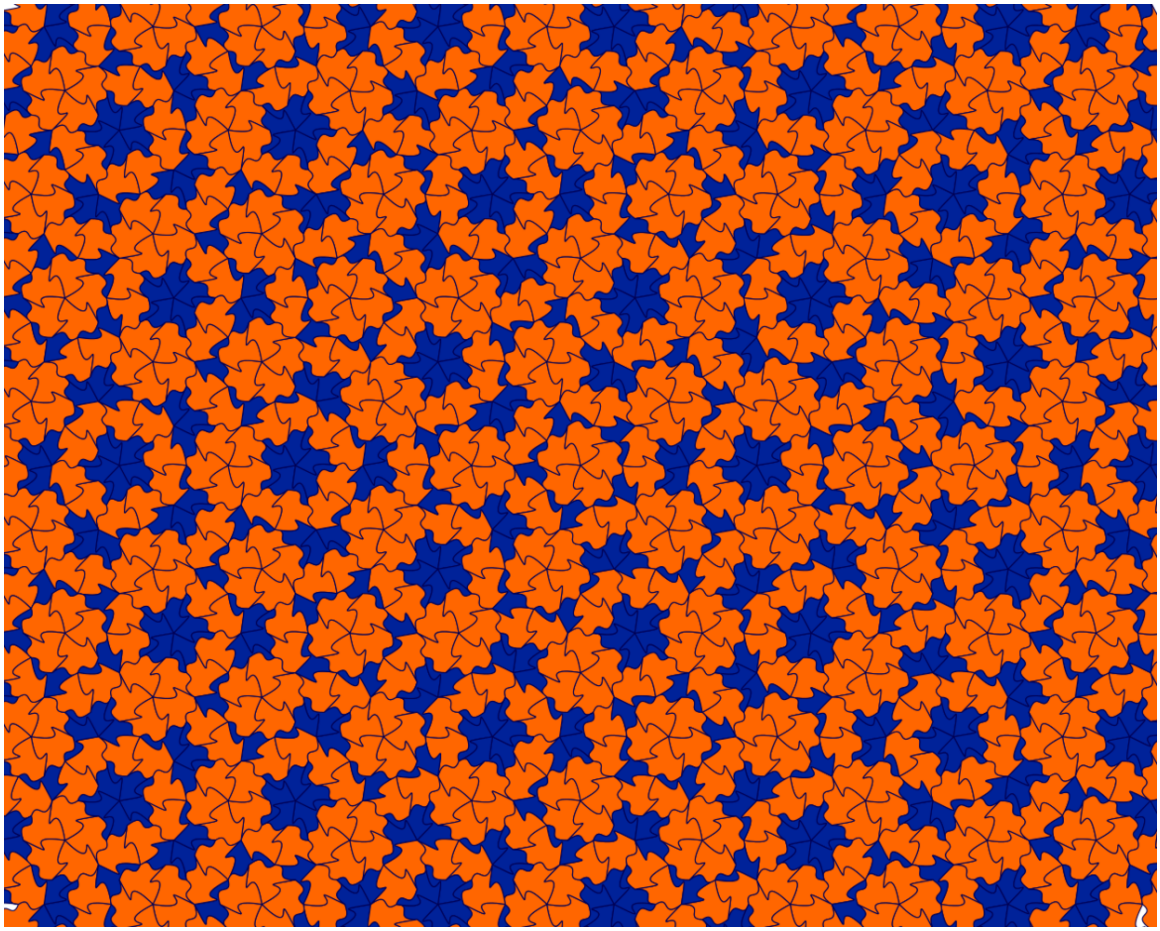


Abb. 28 Drachen und Pfeil

Drachen und Pfeil[14]

Drachen und Pfeil (englisch: kite and dart) sind Formen, mit denen eine aperiodische Parkettierung der Ebene durchgeführt werden kann. Wie das Paar aus dünner und dicker Raute geht diese Parkettierung auf Penrose zurück. Bei Drachen und Pfeil handelt es sich um Vierecke mit exakt einer Spiegelachse. Drachenformen sind beides, denn das jeweilige Viereck hat jeweils zwei Paare von gleichlangen Seiten. Das Verhältnis von langer zu kurzer Seite entspricht dem Goldenen Schnitt mit der Goldenen Zahl $(1+5^{1/2})/2=1.618033988...$. Beim Drachen handelt es sich um die konvexe Form, beim Pfeil sind demgegenüber die kurzen Seiten nach Innen geklappt. Beim Drachen sind drei Innenwinkel 72 Grad und einer 144 Grad (zwischen den beiden kurzen Seiten). Beim Pfeil ist der Winkel zwischen den langen Seiten 72 Grad, der Außenwinkel zwischen den beiden kurzen Seiten 144 Grad und die beiden übrigen Innenwinkel 36 Grad.

Hier dargestellt ist die Deflations-Methode in einer Variation mit komplett asymmetrischen Grundformen, bei denen die jeweiligen vier Seiten von Drachen und Pfeil durch kubische Kurven ersetzt wurden, die zueinander passen, aber so asymmetrisch sind, dass anders als bei den Originalteilen nicht mehr die Möglichkeit gegeben ist, bei der Parkettierung Teile falsch zusammenzulegen.

14 `http://de.wikibooks.org/wiki/media%3ASVG_Penrose01.svg`

W3C ***use***[15]

3.8 Element *image*, Bild einbinden

Mit dem Element ***image*** kann ein (anderes) Bild in das aktuelle Dokument eingebunden werden. In den tiny-Profilen kann dies nicht selbst ein SVG-Dokument sein. Darstellungsprogramme müssen die Formate PNG und JFIF/JPEG interpretieren und einbinden können. Andere Formate sind optional, aber nicht ausgeschlossen. Im Vollprofil von SVG 1.1 ist zudem die Interpretation und Einbettung von SVG-Dokumenten selbst als Bild für ein konformes Darstellungsprogramm notwendig. In SVG tiny 1.2 kann zur Einbindung von SVG-Dokumenten statt ***image*** das neu eingeführte Element ***animation*** verwendet werden, um komplette SVG-Dokumente einzubinden. Dies hat den Vorteil, dass dann auch der Zeitablauf und die Synchronisation der Dokumente eindeutig definiert oder definierbar sind.

Hinsichtlich der Zugänglichkeit und Barrierefreiheit von Information ist es geboten, bei referenzierter Pixelgraphik oder allgemeiner Ressourcen ohne eigene interne Textalternative alternativen Text im Element ***image*** zu notieren. Geeignet sind dafür zum einen die Elemente ***title*** und ***desc*** und ***metadata***, aber auch die Attribute *title* und *role* von XLink.

Für das Element spezifische Attribute:

3.8.1 Attribute *x* und *y*

Der Wert ist jeweils eine Koordinate, also wo das Bild innerhalb des jeweiligen Koordinatensystems dargestellt werden soll, *x* und *y* geben dann die linke obere Ecke an. Sofern nicht angegeben, wird jeweils 0 angenommen.

Die Attribute sind animierbar.

3.8.2 Attribute *width* und *height*

width und *height* geben die eigentliche Breite und Höhe des Darstellungsbereiches für das Bild an. Der Wert ist jeweils eine Länge größer oder gleich 0. Das Bild wird dann entsprechend skaliert. Bei 0 wird das Bild nicht angezeigt. Negative Werte werden nicht unterstützt. Dies bedeutet in SVG tiny 1.2, dass das Attribut mit dem unzulässigen Wert ignoriert wird. In SVG 1.1 ist ein negativer Wert ein Fehler, der zum Abbruch der Darstellung führt.

width und *height* sind animierbar. Sofern nicht angegeben, wird in SVG tiny 1.2 als Wert 0 angenommen. In SVG 1.1 ist das Fehlen eines der beiden Attribute ein Fehler, der zum Abbruch der Darstellung des Dokumentes führt.

3.8.3 Attribut *viewBox*

Das Attribut *viewBox* gibt eine rechteckige Region an, in welche der darzustellende Inhalt fällt. Bei diesem Element wird der Wert allerdings vom referenzierten Dokument übernommen und braucht

15 `http://www.w3.org/TR/SVG/struct.html#UseElement`

nicht angegeben zu werden, beziehungsweise die Angabe ist wirkungslos. Weil aber ein ***image*** formal einen eigenen Anzeigebereich erzeugt, ist das Attribut auch verfügbar. Der formal korrekte Wert ist bereits für das Element ***svg*** beschrieben worden.

Das Attribut ist animierbar.

3.8.4 Attribut *preserveAspectRatio*

Wenn das Seitenverhältnis des referenzierten Bildes nicht zu *width* und *height* passt, kann mit *preserveAspectRatio* angegeben werden, wie es in den Anzeigebereich einzupassen ist.

Die möglichen Werte sind bereits entsprechend für das Element ***svg*** beschrieben. Interessant für Bilder ist die optional vorangestellte Zeichenkette 'defer'. Zum Beispiel der Wert 'defer xMidYMid slice', bedeutet, dass *preserveAspectRatio* vom referenzierten Bild übernommen werden soll, sofern dieses eine solche Angabe hat, was etwa bei einem SVG-Dokument der Fall ist. Ist dem nicht so, wie etwa bei einem PNG, so wird 'xMidYMid slice' verwendet.

Das Attribut ist animierbar.

3.8.5 Attribute von XLink, insbesondere *href*

Um das Bild zu referenzieren, wird das Attribut *href* von XLink verwendet. Typisch wird das Präfix 'xlink' für den Namensraum von XLink verwendet. Sofern so angegeben, wird das Attribut dann als *xlink:href* notiert. Entsprechendes gilt für die anderen in SVG verfügbaren Attribute von XLink, die bei ***image*** somit ebenfalls verwendbar sind.

Ein gültiger Wert ist eine URI oder IRI eines Bildes.

Das Attribut *href* ist animierbar.

3.8.6 Attribut *type*

Das Attribut *type* ist neu in SVG tiny 1.2 eingeführt worden und erlaubt es dem Autor, einen Hinweis zu geben, in welchem Dateiformat das referenzierte Bild vorliegt. Der Wert ist ein internet-Medientyp. Ein Beispiel für einen Medientyp für ein bekanntes Bildformat ist 'image/gif'. Sollte in diesem Falle das Darstellungsprogramm 'image/gif' nicht interpretieren können, kann es den Hinweis dazu benutzen, die Datei erst gar nicht anzufordern.

Das Attribut ist animierbar, weil auch *href* von XLink animierbar ist und sich somit während der Animation das Dateiformat ändern kann. Der Autor ist dann allerdings selbst dafür verantwortlich, dass die Relation zwischen den beiden Animationen ein sinnvoller und zutreffender Hinweis ist.

3.8.7 Beispiel zu *image*

Abb. 29 Beispiel zur Verwendung von ***image***

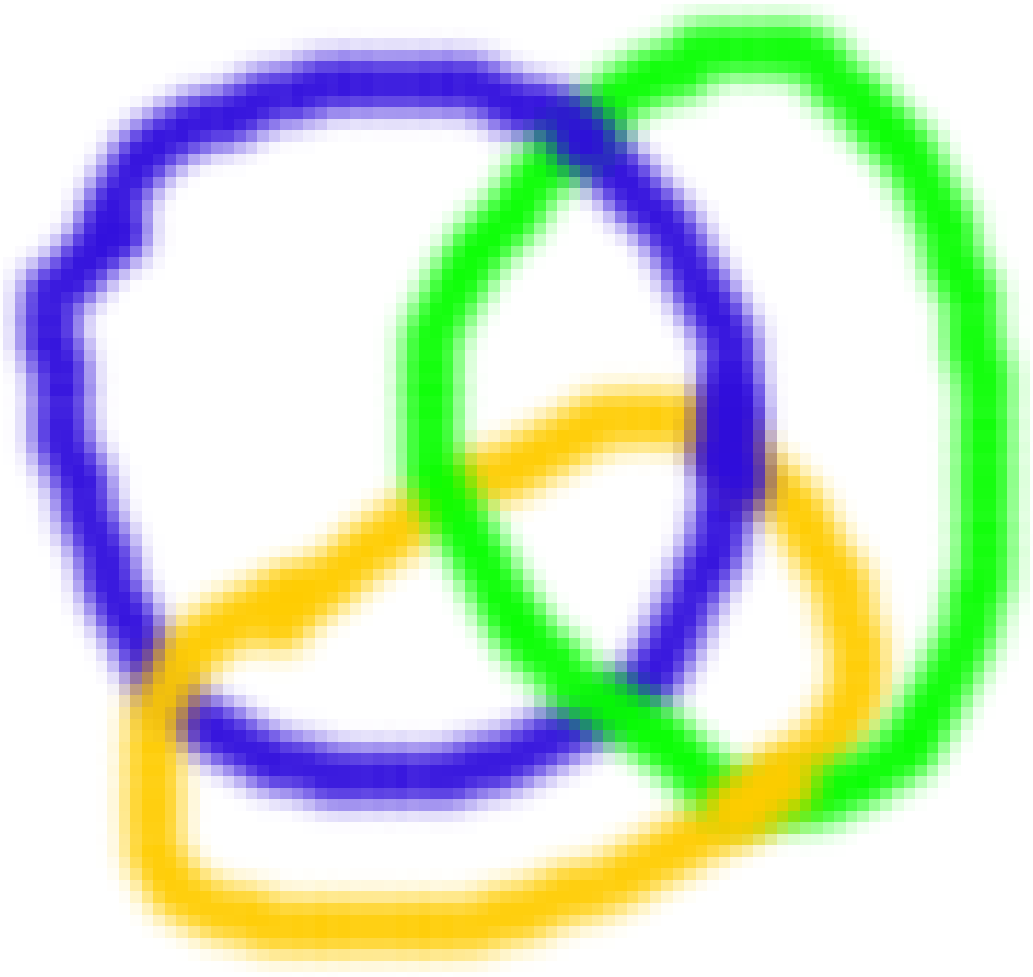


Abb. 30 Beispiel mit referenziertem PNG

Ein einfaches Beispiel zur Verwendung von ***image***[16].

Ein PNG[17] wird mit dem Element image eingebettet und mit Titel und Beschreibung versehen, was ein wichtiger Zugänglichkeitsaspekt von SVG ist.

Zusätzlich wird das Bild transformiert und mit einem Hintergrund hinterlegt.

Es muss natürlich nicht immer, wie in diesem Beispiel, eine komplette URI angegeben werden. Liegt das Bild im gleichen Verzeichnis, so reicht die Angabe des Dateinamens. Liegt das Bild auf der gleichen domain, reichen auch relative Pfade (aufgrund der etwas komplizierteren Ablage bei

16 `http://de.wikibooks.org/wiki/media%3ASVGimage01.svg`
17 `http://de.wikibooks.org/wiki/media%3ASVG_PNGbeispiel01.png`

wikibooks gestaltet sich das Beispiel lediglich etwas komplizierter als für die meisten Anwendungen üblich).

W3C ***image***[18]

3.9 Bedingte Verarbeitung und Element *switch*

Nähere Informationen dazu sind dem Kapitel Barrierefreiheit[19] zu entnehmen.

W3C bedingte Verarbeitung[20]

3.10 Externe Ressourcen und progressive Darstellung

Dokumente referenzieren bisweilen andere Dokumente oder allgemeiner Ressourcen, die inhaltlich wichtig sind. SVG bietet Möglichkeiten anzugeben, wann externe Ressourcen in diesem Sinne (im Rahmen einer progressiven Darstellung) wichtig sind.

3.10.1 Attribut *externalResourcesRequired*

Bei den Gruppierungselementen ***svg***, ***g***, ***a***, ***symbol***, ***clipPath***, ***mask***, ***pattern***, ***marker*** und ***switch*** kann das Attribut *externalResourcesRequired* angegeben werden, ebenfalls bei allen Elementen, die potentiell externe Inhalte referenzieren können. Es hat die Werte 'false' oder 'true' mit der Voreinstellung 'false'. 'false' gibt an, dass externe Ressourcen optional sind, in dem Sinne, dass die Darstellung fortgesetzt werden kann, wenn die Ressource nicht verfügbar ist. Beim Wert 'true' wird die progressive Darstellung des Elementes unterbrochen, bis die externe Ressource verfügbar ist. Insbesondere beginnt die Zeitlinie für Animationen nicht, solange die externe Ressource nicht geladen ist.

Das Attribut ist nicht animierbar. Es ist nicht verfügbar in SVG tiny 1.1.

Es ist zu empfehlen, dies Attribut nur gezielt bei Gruppierungselementen auf 'true' zu setzen, wo dies wirklich notwendig ist, nicht beim ***svg***-Hauptelement. Eine Ausnahme davon ergibt sich, wenn progressives Darstellen unterbunden werden soll.

Insbesondere bei Dokumenten, die Animationen enthalten, kann progressives Darstellen unerwünscht sein. Weil dies weder in SVG 1.1 noch in SVG tiny 1.2 explizit abgestellt werden kann, kann es in solch einem Falle sinnvoll sein, ziemlich zu Beginn mit einem Element ***use*** ein (gegebenenfalls leeres ***g***-) Element am Ende des Dokumentes zu referenzieren und *externalResourcesRequired* 'true' anzugeben. In diesem Falle wird die progressive Darstellung gestoppt, bis das Dokument bis zu diesem letzten Element komplett geladen ist. Entsprechend kann bei Gruppen vorgegangen werden.

In SVG 1.1 gar nicht und in SVG tiny 1.2 ist nur grob beschrieben, was bei einer progressiven Darstellung passieren soll. Dies Attribut samt daraus resultierendem Verhalten ist jedenfalls nur mit

18 `http://www.w3.org/TR/SVG/struct.html#ImageElement`
19 Kapitel 4 auf Seite 77
20 `http://www.w3.org/TR/SVG/struct.html#ConditionalProcessing`

der Ladezeit korreliert. Bei komplizierteren Konstruktionen kann es allerdings auch passieren, dass das Dokument in Sekundenbruchteilen geladen ist, aber mehrere Sekunden braucht, um dargestellt zu werden. Weil die Spezifikation nicht definiert, wie der Prozess der Dokumentinterpretation bis zur Darstellung genau verläuft, kann es neben dem Laden von externen Ressourcen also auch noch andere Aspekte von progressiven Darstellungen geben, die vom Autor nicht direkt kontrollierbar sind.

3.10.2 Element *prefetch*, Ressourcen vorab laden

Neu in SVG tiny 1.2 ist das Element ***prefetch*** spezifiziert, mit dem ein Vorschlag angegeben werden kann, welche externen Ressourcen vorab geladen werden sollten. Wenn das Darstellungsprogramm den Vorschlägen folgt, kann ein flüssigerer, gegebenenfalls auch schnellerer Aufbau der Darstellung des Dokumentes erreicht werden, wenn das Element effektiv eingesetzt wird.

Attribut *mediaSize*

Mit dem Attribut wird die Mediengröße angegeben, also wieviele Bytes (mindestens) vorab geladen werden sollten.

Der Wert ist eine (nicht negative) ganze Zahl nicht größer als 2147483647. Das Attribut ist nicht animierbar. Es ist nicht explizit angegeben, was passiert, wenn das Attribut nicht angegeben ist oder der Attributwert nicht sinnvoll ist, genaugenommen ist der Wert in der Spezifikation nicht einmal auf nicht negative Zahlen eingeschränkt. Es ist zu vermuten, dass unsinnige Angaben dazu führen, dass das Attribut ignoriert wird, wie üblich in SVG tiny 1.2.

Attribut *mediaTime*

Für Medien mit festgelegter endlicher Dauer wie Video oder Audio kann mit dem Attribut *mediaTime* angegeben werden, wie viel vom Dokument bezogen auf die Dauer geladen werden soll. Der Wert ist ein Zeitwert, wie im Kapitel über Animation erläutert, im einfachsten Falle zum Beispiel '120s' für zwei Minuten. Es müsste dann also mindestens soviel von der Datei vorab geladen werden, dass damit zwei Minuten abgespielt werden können.

Medien ohne festgelegte Dauer wie PNGs oder ein Dokument vom Typ SVG tiny 1.2 selbst werden komplett geladen, wenn das Attribut angegeben ist.

Wird ein animiertes Element referenziert, bezieht sich die Angabe auf die aktive Dauer der Animation.

Das Attribut ist nicht animierbar.

Attribut *bandwidth*

Mit *bandwidth* wird die Übertragungsrate in Bits pro Sekunde angegeben, die verwendet werden sollte, um die externe Ressource zu laden. Sofern es nicht angegeben ist, wird die komplette verfügbare Rate verwendet.

Unsinnige Werte oder solche über der verfügbaren Rate werden nicht berücksichtigt werden können. Es ist nicht explizit angegeben, aber es kann vermutet werden, dass unsinnige oder unerfüllbare Angaben dazu führen, dass das Attribut ignoriert wird, wie üblich in SVG tiny 1.2.

Das Attribut ist nicht animierbar.

Attribut *mediaCharacterEncoding*

Das Attribut gibt die Kodierung der zu ladenden Ressource an (UTF-8, ISO-8859-1, etc). Bei der Angabe von *mediaSize* kann diese davon abhängen, von welcher Kodierung bei der Angabe von *mediaSize* ausgegangen wurde. Insofern kann dieser Hinweis nützlich für eine präzisere Schätzung sein. Wird das Attribut nicht angegeben, wird davon ausgegangen, dass die Angabe von *mediaSize* sich auf die aktuelle bezieht, etwa jene, die der server behauptet.

Das Attribut ist nicht animierbar.

Attribut *mediaContentEncodings*

Mit dem Attribut *mediaContentEncodings* kann eine Angabe zur Kompressionsart (Inhaltskodierung) gemacht werden, welche bei der referenzierten Ressource bei der Berechnung der Mediengröße vorausgesetzt wurde. Beim Wert handelt es sich um eine mit Leerzeichen separierte Liste im Sinne von Abschnitt 3.5 von HTTP 1.1 (RFC2616), (gzip, compress, etc) Die Reihenfolge ist jene, die bei der Kompression verwendet wurde. Die Angabe ist erforderlich, wenn eine Kompression vorgenommen wurde. Sofern nicht angegeben, wird keine Kompression angenommen.

Das Attribut ist nicht animierbar.

Attribute von XLink, insbesondere *href*

Um die externe Ressource zu referenzieren, wird das Attribut *href* von XLink verwendet. Typisch wird das Präfix 'xlink' für den Namensraum von XLink verwendet. Sofern so angegeben, wird das Attribut dann als *xlink:href* notiert. Entsprechendes gilt für die anderen in SVG verfügbaren Attribute von XLink, die somit ebenfalls verwendbar sind.

Ein gültiger Wert ist eine URI oder IRI der Ressource. Vorgabe ist ein leerer Wert. Das gilt sinngemäß auch für ungültige Werte, es wird dann also nichts vorab geladen.

Das Attribut *href* ist für dieses Element nicht animierbar.

3.11 Element *discard*, Inhalt streichen

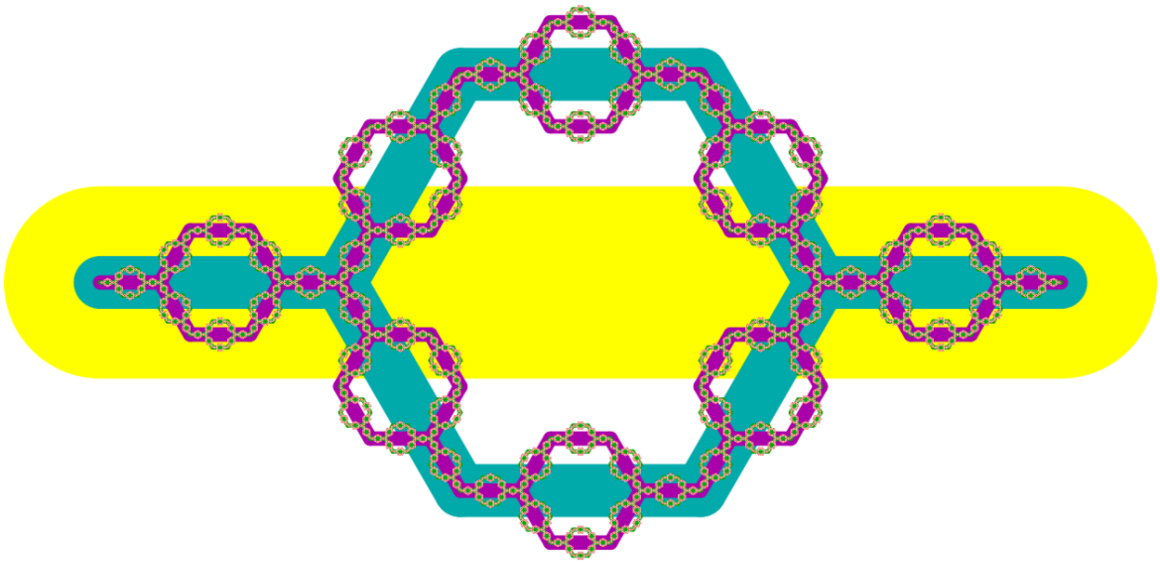


Abb. 31 Beispiel zur Verwendung von ***discard***

Squiggle (Batik)	-
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem in SVG tiny 1.2 neu eingeführten Element ***discard*** kann Inhalt aus dem Dokument gestrichen oder gelöscht werden, um Speicherplatz freizumachen. Nützlich ist dies insbesondere bei Dokumenten mit langen Laufzeiten, zum Beispiel mit einigen Videos oder auch einer Kaskade von ***use***-Elementen. Sofern umfangreicher Inhalt nach einiger Zeit im Dokument nicht mehr gebraucht wird, kann es sich lohnen, den Speicher des Darstellungsprogrammes zu entlasten. Ein gestrichenes Element ist also nach der Streichung nicht mehr verfügbar, als sei es nie im Dokument vorhanden gewesen.

Das Element kann überall auftreten, wo auch Animationselemente wie ***animate*** auftreten können.

Beispiel zur Verwendung von ***discard***[21]

21 `http://de.wikibooks.org/wiki/media%3ASVGdiscard01.svg`

Ein Fraktal in Form eines Iterierte-Funktionen-Systems, ein direkter Verwandter der Koch-Kurve, wird als Beispiel verwendet, um das Element ***discard*** auszuprobieren. Alle zehn Sekunden wird ein dargestellter Iterationsschritt gelöscht, bis nur noch der letzte, weiße übrigbleibt. Dieser kann durch Klicken oder Aktivieren auch noch gelöscht werden.

3.11.1 Attribut *begin*

Mit dem Attribut *begin* wird der Zeitpunkt angegeben, zu dem ein Element aus dem Speicher gestrichen werden soll. Der Wert ist eine Liste entsprechend wie für dies Attribut im Kapitel Animation angegeben. Im einfachsten Falle ist zum Beispiel ein Zeitwert '600s' für zehn Minuten. Ein Anfangswert kann auch ein bestimmtes Ereignis sein wie zum Beispiel 'Zernichter.activate+10s', also zehn Sekunden nach Aktivierung des mit 'Zernichter' identifizierten Elementes. Zum angegebenen Zeitpunkt wird also das referenzierte Element aus dem Speicher gestrichen.

Vorgabe ist der Wert 0, also ein direktes Löschen zum Dokumentbeginn. Das ergibt sich auch bei einem ungültigen Wert.

Das Attribut ist nicht animierbar.

3.11.2 Attribute von XLink, insbesondere *href*

Um das zu streichende Element zu referenzieren, wird das Attribut *href* von XLink verwendet. Typisch wird das Präfix 'xlink' für den Namensraum von XLink verwendet. Sofern so angegeben, wird das Attribut dann als *xlink:href* notiert. Entsprechendes gilt für die anderen in SVG verfügbaren Attribute von XLink, die somit ebenfalls verwendbar sind.

Ein gültiger Wert ist eine URI oder IRI samt Fragmentidentifizierer. Ob bei einer Referenzierung eines externen Dokumentfragmentes ein Effekt resultiert, hängt davon ab, ob das Format des referenzierten Dokumentes eine solche Funktionalität definiert.

Wird das Attribut nicht angegeben, wird das Elternelement gelöscht. Entsprechendes ergibt sich aus einem leeren Attribut oder einem ungültigen Wert.

Das Attribut *href* ist für dieses Element nicht animierbar.

3.12 Allgemein verwendbare Attribute

3.12.1 Attribute *id* und *xml:id*

Zur dokumentweit eindeutigen Identifikation von Elementen dient in SVG das Attribut *id*, in XML ganz allgemein das Attribut *xml:id*. Weil letzteres erst später eingeführt wurde, wird es in SVG 1.1 noch nicht erwähnt und sollte in der Praxis eher nicht dafür verwendet werden, um SVG-spezifisch etwa zur Wiederverwendung mit ***use*** oder zur Animation Elemente zu identifizieren, auch weil ältere Programme die Interpretation von *xml:id* nicht implementiert haben.

xml:id ist andererseits nützlich, wenn es darum geht, über SVG hinausgehende Identifikationen vorzunehmen, wenn es etwa darum geht, Metainformationen eindeutig auf ein Element zu

addressieren. Wenn die Metainformation in einer anderen Sprache abgefaßt ist, erfordert ein per *xml:id* identifizierbares Element nicht einmal die Kenntnis von SVG, um das Element aufzufinden.

In SVG tiny 1.2 ist explizit festgelegt, dass sich der Autor allerdings (bei jedem Element einzeln) entscheiden muss, welche Notation er vornimmt, um Widersprüche zu vermeiden.

Wert ist ein im jeweiligen Dokument einmaliger Fragmentidentifizierer, der also nicht bereits für andere Elemente im selben Dokument verwendet wurde.

Mit ***use*** wiederverwendete Elemente stellen in dem Zusammenhang kein Problem dar, weil konzeptionell die Vorlage das Attribut trägt und somit Aktionen bei der Vorlage gleichermaßen auf alle Kopien wirken.

Die Attribute sind nicht animierbar.

3.12.2 Attribut *class*, Klassenzuordnung

In der Vollversion von SVG 1.1 und in SVG tiny 1.2 kann ein Element ebenfalls selbstdefinierten Klassen zugeordnet werden. Dazu dient das Element 'class'.

Der Wert des Attributes ist eine mit Leerzeichen separierte Liste von Klassennamen.

Ein Klassenname in dem Sinne ist eine Zeichenkombination aus Buchstaben, Zahlen und den Zeichen '.', '-', '_', ':'. Exakt sind die erlaubten Zeichen in der XML-Spezifikation unter dem Stichwort 'Nmtoken' nachzulesen.

Das Attribut ist animierbar.

Vielfach wird das Attribut dazu verwendet, um eine Klasse von Elementen über einen CSS-Selektor mit Eigenschaften zu versehen. Der Wert des Attributes hat allerdings auch eine semantische Bedeutung, denn der Name der Klasse sollte charakteristisch zur Funktion der Elemente einer Klasse gewählt werden.

Zum Beispiel bedeutet *class*="Hilfe Menue", dass das Element zu den Klassen 'Hilfe' und 'Menue' gehört. Eine Bezeichnung wie *class*="klein blau" ist hingegen vermutlich keine semantisch gute Bezeichnung, weil es nichts über die Funktion des Elementes verrät, sondern bestenfalls über die beabsichtigte Präsentation (die bei einer alternativen Stilvorlage anders sein kann).

3.12.3 Attribute für Metainformationen und Beschreibung der Funktion von Elementen

In SVG tiny 1.2 wurden einige allgemein verwendbare Attribute aus anderen Bereichen oder Sprachen (WAI-ARIA und RDFa) übernommen. Diese sollen helfen, die Zugänglichkeit und Verstehbarkeit von Konstruktionen in Dokumenten zu verbessern. Zudem kann damit die Maschinenlesbarkeit und -verstehbarkeit von Inhalt verbessert werden.

Auf diese Attribute wird im Kapitel Barrierefreiheit[22] näher eingegangen.

22 Kapitel 4 auf Seite 77

3.12.4 Allgemeine XML-Attribute

Auf weitere, neben *xml:id* ebenfalls bereits durch XML allgemein verfügbare Attribute wird im Kapitel Kurze Einführung in XML[23] näher eingegangen.

23 Kapitel 1.17.6 auf Seite 17

4 Barrierefreiheit

4.1 Einleitung

Wie alle Auszeichnungssprachen, die vom W3C empfohlen werden, ist auch SVG mit zahlreichen eingebauten Automatismen und Hilfen für den Autor ausgestattet, mit denen die Zugänglichkeit von Dokumenten in diesem Format verbessert wird.

Die eingebauten Automatismen bestehen vor allem darin, dass es sich um ein XML-Format handelt, die Dateien selbst also aus Text bestehen, der mit unspezifischen Werkzeugen lesbar und bearbeitbar ist. SVG ist an sich also Klartext, der ohne spezifische Programme interpretierbar ist. So lässt sich anhand der Namen der verwendeten Elemente und Attribute prinzipiell erkennen, welcher graphische Inhalt repräsentiert wird. So ist es von der Zugänglichkeit her also sinnvoll, statt eines allgemeinen Pfades das Element ***circle*** zu verwenden, wenn ein Kreis gemeint ist. Das ist dann auch besser als das semantisch weniger spezifische ***ellipse*** oder gar ein ***rect*** mit entsprechend abgerundeten Ecken. Entsprechendes gilt für die anderen Basisformen von SVG. Wenn ein gleichseitiges Polygon zu malen ist, ist offenbar auch das Element ***polygon*** eine bessere Beschreibung als das Element ***path***, wo die Eigenschaft, dass es sich um ein Polygon handelt, erst mühsam aus dem Wert des Attributes *d* extrahiert werden muss, was deutlich aufwendiger ist, als dies anhand des Elementnamens zu erkennen. Andersherum ist es nicht so hilfreich, beliebige Kurven mittels ***polyline*** oder ***polygon*** anzunähern, wenn die tatsächliche Kurve viel besser und effektiver mit kubischen Kurvenfragmenten innerhalb eines ***path*** dargestellt werden kann.

In der Praxis fällt es bei nicht trivialen Dokumenten indes schwer, das beabsichtigte Objekt ohne geeignetes graphisches Darstellungsprogramm zu visualisieren. In anderen Fällen wird einigen Betrachtern die Bedeutung der graphischen Repräsentation nicht klar sein, weswegen diese ebenfalls gerne auf eine Textalternative umschalten werden, welche die graphische Repräsentation ergänzt oder einen anderen Zugang zum Verständnis des Bildes ermöglicht.

Für den Fall, dass ein graphisches Darstellungsprogramm nicht verfügbar ist, aber auch um ergänzende Textinformationen angeben zu können, bietet SVG mit den Elementen ***title***, ***desc*** und ***metadata*** dem Autor Hilfen an, um praktisch zu jedem Element eine Textalternative oder -ergänzung bereitstellen zu können. Der Inhalt dieser Elemente kann auch durch Elemente aus anderen Namensräumen strukturiert werden, um die Les- und Verstehbarkeit der Textinformation zu verbessern. Elemente aus anderen Namensräumen stellen andererseits aber auch wieder erhöhte Anforderungen an das Darstellungsprogramm, weil dies diese anderen Formate kennen und interpretieren können muss, um die Struktur sinnvoll in der alternativen Betrachtung präsentieren zu können. Ist dies nicht gegeben, muss der Autor nach wie vor mit einer nicht strukturierten Präsentation rechnen. Zudem ist damit zu rechnen, dass Validatoren, die mit Dokumenttypdeklarationen (DTD) arbeiten, solche Dokumente mit gemischten Formaten nicht korrekt validieren können, wenn die Dokumenttypdeklaration nicht entsprechend angepasst wird. Für Darstellungsprogramme ergibt sich indes in der Praxis kein Problem, weil denen die Angabe des Namensraumes reicht, um Formate zu identifizieren. Aufgrund

der Beschränkungen der Möglichkeiten von Dokumenttypdeklarationen wird auf diese bei neueren Versionen (SVG tiny 1.2) ohnehin verzichtet.

Im Vergleich mit anderen Formaten, besonders Pixelgraphik, wirkt der Begriff Zugänglichkeit auf den ersten Blick vielleicht etwas befremdlich, was hat denn ein Graphikformat mit Barrierefreiheit zu tun? SVG bietet schon alleine dadurch, dass es ein Vektorgraphikformat ist, einen großen Vorteil: Es ist stufenlos skalierbar. Eine weitere Eigenschaft gibt es umsonst, in einem SVG ist der Text nicht in Form von Pixeln und auch nicht als Kurven gespeichert, sondern in Form von Buchstaben, welche in einem ***text***-Element stehen. Daher kann auch zum Beispiel ein Vorlese-browser (screenreader) den Text erkennen und einem Blinden vorlesen. Was hingegen in einer Pixelgraphik dem sehenden Betrachter als Text erscheint, sind eigentlich nur Pixel oder Farbverteilungen, denen ein einfaches Programm keine inhaltliche Bedeutung als Text zuordnen kann, wodurch solche Farbverteilungen in Pixelgraphik Blinden ohne spezielle Programme, die dann versuchen, die Information zu raten, unzugänglich werden.

Auch Verweise sind in SVG problemlos zugänglich und leicht anhand des verwendeten Elementes zu erkennen. Damit diese funktionieren, braucht der screenreader, browser und so weiter noch nicht mal SVG zu interpretieren. Diese werden angegeben mit der Sprache XLink, einer relativ grundlegenden XML-Funktionalität. Wobei inzwischen allerdings die Interpretation von SVG und XLink in SVG weiter verbreitet zu sein scheint als XLink in beliebigen XML-Dokumenten, man betrachte etwa damit eingebetteten Inhalt in diversen browsern - was in SVG problemlos funktioniert, wird in einem beliebigem XML-Format zum Problem für so manchen browser. XLink selbst wiederum bietet Möglichkeiten zu beschreiben, wozu ein Verweis gut ist oder um was es sich beim Verweisziel handelt, das ist zwar bei vielen Darstellungsprogrammen nicht nachvollziehbar, bietet dem Autor aber ganz unabhängig von den Möglichkeiten der Darstellungsprogramme eine weitere Chance, Inhalt und Funktion von Verweisen und referenzierten und eingebetteten Inhalten zu verdeutlichen.

Aber was ist mit dem Inhalt, der Bedeutung einer Graphik oder einer Anwendung, die auf SVG aufbaut? Es ist sehr wohl möglich zum Beispiel für Blinde eine Graphik nutzbar zu machen. Ein screenreader etwa oder auch ein Darstellungsprogramm mit graphischer Ausgabe kann die Inhalte von Text-Elementen, der Elemente ***title***, ***desc*** und ***metadata*** in einer alternativen Textdarstellung zugänglich machen, alternativ oder zusätzlich zur graphischen Darstellung der Datei. So ist ein unabhängiger Zugang zur inhaltlichen Bedeutung des Dokumentes jedem Nutzer, insbesondere auch Blinden, verfügbar.

Wie genau die Textalternative zu präsentieren ist, ist in den SVG-Spezifikationen nicht festgelegt und hängt stark davon ab, für wen und in welcher Relation zum graphischen Inhalt die Darstellung erfolgen soll.

SVG bietet ferner die Möglichkeit der bedingten Verarbeitung, der Autor kann also Alternativen angeben, wenn bestimmte Möglichkeiten beim Darstellungsprogramm oder beim Nutzer nicht verfügbar sind, darunter auch eine Selektion von Text in mehreren Sprachen, je nach Einstellung des Nutzers.

SVG tiny 1.2 bietet dem Autor ferner die Chance, mit neuen Attributen Funktion und Bedeutung von Elementen und Konstruktionen anzugeben, einmal um die semantische Qualität des Inhaltes zu verbessern, zum anderen aber auch, um bei Nutzungseinschränkungen die Möglichkeit zu geben, dass das Darstellungsprogramm die angegebene Funktionalität auf einem alternativen Weg bereitstellt.

4.2 Verwendung von *title*, *desc* und *metadata*

Jedenfalls wird von den Spezifikationen dringend empfohlen, mindestens das Haupt-***svg***-Element (root-Element) mit einem ***title***-Element als erstes Kindelement zu versehen. Der Inhalt dieses Elementes wird als Dokumenttitel verwendet - zum Beispiel im Kopfbereich des Darstellungsfensters und kann auch gut dazu dienen, innerhalb von Textumgebungen auf das Dokument zu verweisen oder es im Textfluß zu bezeichnen.

Titel sind auch bei klassischen Bildern in der Kunst etabliert und finden sich dort oft irgendwo neben dem Bild oder darunter. Bei SVG ist der Titel in das Dokument integriert. Von der historischen Herkunft her ist ein Titel oder eine Überschrift eine Kurzzusammenfassung des Inhaltes, also eine Kurzrepräsentation in Textform oder aber auch ein Bezeichner des gesamten Werkes. Im Bereich der Kunst und Literatur wird dies oft etwas freier gehandhabt und der Titel als inhärenter Bestandteil des Werkes steht manchmal in einer assoziativen Relation zum Inhalt und ist nicht nur eine Kurzzusammenfassung, man denke etwa an Werke von René Magritte oder Salvador Dalí. Der Titel als Bezeichner des Gesamtwerkes ist jedoch durchgehend etabliert, so dass dem Autor zu empfehlen ist, einen ausdrucksstarken und signifikanten Titel zu wählen, der das gesamte Werk gut repräsentiert.

'unbekanntes Dokument' wird zumeist ein schlechter Titel sein, weil er kaum mit dem Inhalt korreliert ist und sich auch kaum als Bezeichner des Werkes eignet. 'Meine Perle' mag für ein Portrait der aktuellen Freundin in einem eindeutig lokalen Umfeld ausreichend sein, ohne weiterem Bezug in beliebigem Zusammenhang aber unzureichend. Besser wäre da etwa: 'Gunilla Schneider; Portrait von Siggi Haase, Hannover 2009'. Dies ist eine recht formale Herangehensweise im Sinne einer Kurzzusammenfassung. Das auf das ***title***-Element folgende ***desc***-Element kann vom Autor genutzt werden, um weitere Zusammenhänge zu erhellen. Etwas künstlerisch kreativer mag Siggi vielleicht auch formulieren: 'Gunillas geheimnisvolles Lächeln in Erwartung ihres Geliebten'. Weitere Angaben schreibt Siggi dann wieder im ***desc***-Element oder maschinenlesbar mit RDF und Dublin-Core im auf das ***desc***-Element folgende Element ***metadata***, indem er ebenfalls notiert, welchen Nutzungsbedingungen das Bild unterliegt.

Bei einem Kunstwerk kann es durchaus erwünscht sein, dem Werk keinen Titel zuzuordnen. Trotzdem ist dringend zu empfehlen, ein ***title***-Element anzugeben, dies dann aber leer zu lassen und auf den Sachverhalt im ***desc***-Element hinzuweisen. Sonst ist es durchaus plausibel, dass sich der Dateiname oder die URI des Dokumentes als Bezeichner etabliert, also vermutlich wird damit genau das Gegenteil von dem erreicht, was eigentlich beabsichtigt war. 'Ohne Titel' als Inhalt des ***title***-Elementes führt wiederum zu einem Widerspruch zwischen Funktion und Inhalt, ist das nicht gerade Bestandteil der Aussage des Kunstwerkes, ist dies dann sicher die falsche Wahl für den Inhalt eines ***title***-Elementes, wenn kein Titel angegeben werden soll. Manche Künstler behelfen sich auch erfolgreich damit, stattdessen eher technische Bezeichner als Titel anzugeben, etwa 'BlaueSerie2009-10-16T12:03:07.3Z'. Das ist als Bezeichner hinreichend signifikant, vermeidet aber weitgehend die gegebenenfalls ungewollte Funktionalität einer Kurzzusammenfassung.

Die Verwendung von ***title***, ***desc*** und ***metadata*** ist bei praktisch jedem Element möglich und repräsentiert dann entsprechend die Textalternative des jeweiligen Elementes oder der Gruppe von Elementen, die das jeweilige Elternelement umschließt.

Der primäre Zweck insbesondere von ***title*** und ***desc*** liegt also darin, eine alternative Textrepräsentation für das jeweilige Dokumentfragment bereitzustellen. Einige graphische Darstellungsprogramme stellen diese Information ähnlich dem *title*-Attribut in (X)HTML als Nutzungshinweis (tooltip) dar,

das kann für Autoren insofern irreführend sein, als dass diese insbesondere im ***title***-Element Nutzungshinweise statt einem Titel oder einer Überschrift, einer Textalternative unterbringen. Schlecht ist in dem Zusammenhang etwa ein Titel wie: 'Anklicken oder Aktivieren, um Animation in 3s zu starten'. Das ist ein Nutzungshinweis, kein Titel und gehört zumindest so nicht in das ***title***-Element, kann aber Bestandteil des Inhaltes von ***desc*** sein. Besser wäre also ein Titel wie 'Knopf, um das Karussel zu starten' mit einer Beschreibung im ***desc***-Element wie: 'Mit dem Anklicken oder Aktivieren dieses Knopfes wird nach drei Sekunden eine Animation eines Karussels gestartet, fröhliche Kinder drehen sich auf einem Kinderkarussel in die Runde.'

Da die graphische Darstellung und die Zeichenreihenfolge im Malermodell von SVG oft bestimmt, in welcher Reihenfolge Elemente im Dokument angeordnet sind, kann es oft schwierig sein, eine passable Textrepräsentation bereitzustellen, indem man allen oder vielen Elementen ***title*** und ***desc*** hinzufügt. Oft ist es einfacher, die Beschreibung nur für das gesamte Dokument vorzunehmen oder für größere Fragmente, die einen in sich abgeschlossenen Sinnzusammenhang ergeben. Es wird dann beschrieben, was das Dokument oder das Fragment repräsentiert, weniger wie dies praktisch realisiert ist (was ja ohnehin anhand des Quelltextes analysiert werden kann).

4.2.1 Elemente

title

Squiggle (Batik)	1.7 (teilweise, falsch im Sinne von tiny 1.2)
Opera (Presto)	8 (nur Dokumenttitel), 9 (teilweise, falsch im Sinne von tiny 1.2)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4 (Gecko 2.0; teilweise, falsch im Sinne von tiny 1.2)
Konqueror (KSVG)	3.2 (Dokumenttitel; komplett mit weiterem plugin)
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Nahezu jedes SVG-Element kann ein Element ***title*** als Kindelement haben, dies sollte dann immer das erste Kindelement sein. Der Inhalt des ***title***-Elementes ist zumeist einfacher Text. In SVG 1.1 sind auch explizit Elemente aus anderen Namensräumen vorgesehen, was jedoch wegen der bereits oben beschriebenen besonderen Verwendung beim ***title***-Element des Haupt-***svg***-Elementes nicht zu empfehlen ist. Das Haupt-***svg***-Element sollte immer ein ***title***-Element haben.

SVG tiny 1.2 empfiehlt explizit, nur einfachen Text als Inhalt zu wählen. Ein Element sollte maximal ein ***title***-Element enthalten. Dieses ist dann der Titel oder eine Kurzzusammenfassung für das Element selbst und seine Kindelemente (insbesondere bei Gruppen). ***title*** zusammen mit ***desc*** dienen primär dazu, eine Textalternative für den graphischen Inhalt bereitzustellen. Andererseits ist es generell als nicht ausreichend anzusehen, den Inhalt statt des graphischen Inhaltes anzuzeigen, wenn eine graphische Repräsentation durch SVG vorgesehen ist.

Wie Darstellungsprogramme im einzelnen die Textalternative darstellen, ist nicht festgelegt. Jedenfalls ist es Aufgabe des Autors, dafür zu sorgen, dass die Abfolge der Texte und Textalternativen in einem Dokument hinsichtlich der Barrierefreiheit und Zugänglichkeit einen Sinn ergibt, der im wesentlichen einen Ersatz für die graphische Repräsentation darstellt, wenn dem Nutzer eine solche nicht zugänglich ist.

Amaya stellt eine alternative Textrepräsentation etwa in einem anderen Darstellungsbereich dar, wenn der Nutzer dies wünscht. Batik/Squiggle zeigt die komplette XML-Struktur als alternative Textrepräsentation, dies kann beim Konqueror ähnlich eingerichtet werden. Einige Programme blenden manche, aber nicht alle ***title***-Elemente je nach Struktur des Dokumentes auch als Nutzungshilfen ein. Generell ist die strukturierte Darstellung von Textalternativen für SVG-Dokumente in Darstellungsprogrammen nicht einheitlich und ist zumeist noch stark verbesserungswürdig.

In SVG tiny 1.2 ist explizit vorgesehen, dass Darstellungsprogramme mit entsprechenden Fähigkeiten ***title*** nur in Form einer Nutzungshilfe (tooltip; wenn mit der Maus, dem Zeigegerät über ein Element mit ***title*** gefahren wird, wird dessen Textinhalt an der aktuellen Position dargestellt) darstellen sollen, wenn *role*="tooltip" angegeben ist. Dies dient vor allem dazu, die Nutzung als Textalternative zu fördern, denn es mag Autoren geben, die eine solche Nutzerhilfe nicht angezeigt haben möchten und ansonsten dann vermutlich eher auf dieses Mittel des barrierefreien Zugangs zum Dokument verzichten, wenn Darstellungsprogramme immer eine solche Anzeige wählen.

desc

Squiggle (Batik)	1.7 (teilweise falsch im Sinne von tiny 1.2)
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	3.2 (mit weiterem plugin)
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Nahezu jedes SVG-Element kann ein Element ***desc*** als Kindelement haben. Dies sollte dann, soweit vorhanden, nach dem Element ***title*** das zweite Kindelement sein. Der Inhalt des ***desc***-Elementes ist zumeist einfacher Text, der eine Beschreibung dessen darstellt, was das Element oder die Gruppe repräsentiert oder welche Funktionalität diese im Gesamtdokument haben oder aufweisen. In SVG 1.1 sind als Inhalt auch explizit Elemente aus anderen Namensräumen vorgesehen. Etwa könnte man das Element ***p*** von XHTML verwenden, um die Beschreibung mit Absätzen zu strukturieren. Auch andere Elemente von XHTML oder anderen Formaten können helfen, die semantische Stuktur des Inhaltes auszuzeichnen. Dies ist insofern hilfreich, als SVG selbst keine Elemente bereitstellt, um den Inhalt zu strukturieren oder Texte semantisch relevant auszuzeichnen.

SVG tiny 1.2 empfiehlt explizit, nur einfachen Text als Inhalt zu wählen. Ein Element sollte maximal ein ***desc***-Element enthalten. In SVG tiny 1.2 ist also an sich keine weitere Strukturierung des Inhaltes von ***desc*** mit Elementen aus anderen Namensräumen wie XHTML vorgesehen. Ist eine strukturierte Beschreibung dennoch notwendig oder gewünscht, bleibt noch die Möglichkeit, innerhalb des Elementes ***metadata*** eine entsprechende strukturierte Beschreibung neben anderen Metainformationen unterzubringen. Die Hinweise bezüglich des Attributes *role*="tooltip" für ***title*** gelten entsprechend.

metadata

Squiggle (Batik)	-
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	3.2 (mit weiterem plugin)
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Nahezu jedes SVG-Element kann ein Element ***metadata*** als Kindelement haben. Dies sollte dann, soweit vorhanden, nach den Elementen ***title*** und ***desc*** das dritte Kindelement sein. Der Inhalt ist Metainformation, ausgezeichnet mit Elementen aus einem anderen Namensraum als SVG, zum Beispiel RDF. Sofern in der verwendeten Sprache vorgesehen, empfiehlt sich explizit anzugeben, ob sich die Metainformation auf das gesamte Dokument bezieht oder nur auf ein bestimmtes Fragment. RDF hat eine entsprechende Funktionalität mit dem Attribut *about*, dessen Wert dann der Fragmentidentifizierer (Wert des Attributes id mit vorangestellten #) des Elementes ist, auf welches sich die Metainformation beziehen soll. Mittels zum Beispiel RDF und Dublin Core lassen sich so (maschinenlesbar) Angaben zu Autor, Erstellungsdatum, Pubklikationsdatum, Nutzungsbedingungen und vielen weitere Metainformationen strukturiert unterbringen.

W3C Referenz[1]

4.2.2 Beispiele zur Verwendung von *title*, *desc* und *metadata*

Einfaches Beispiel für SVG tiny 1.1, ohne Verwendung von Elementen aus anderen Namensräumen, so auch vom W3C-Validator erfolgreich testbar:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG Tiny 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-tiny.dtd">
<svg width="90%" height="100%" viewBox="0 0 1000 1000"
  xmlns="http://www.w3.org/2000/svg"
  version="1.1" baseProfile="tiny"
  xml:lang="de">
<title>Nahezu leeres SVG-Beispiel</title>
<desc>
In einem SVG-Dokument kann alternative oder zusätzliche
Textinformation vom Autor angeboten werden, wobei die
Elemente title und desc verwendet werden.

Das Element metadata kann verwendet werden, um strukturierte
Metadaten oder Metainformation über das Dokument anzugeben,
typisch mit Elementen aus einem anderen Namensraum.

Dies Dokument ist ein Beispiel für die Verwendung von
title, desc und metadata in einem ansonsten nahezu leeren
Dokument.
</desc>

<metadata>
<!-- Einige Metainformationen, zum Beispiel mit RDF und Dublin-Core verwendend,
hier untypisch, aber für den W3C-Validator in Ordnung als Klartext angegeben.
Man kann allerdings nicht erwarten, dass solche Klartextangaben maschinenlesbar
```

1 http://www.w3.org/TR/SVG/metadata.html

```
sind.  -->
Autor: Siggi Haase
Entstehungsdatum: 2009-10-19
Kontext: http://de.wikibooks.org/wiki/SVG/_Barrierefreiheit
etc
</metadata>

<defs>
<!-- nicht direkt dargestellte Elemente, bereitgestellt zur späteren Verwendung,
auch Animationselemente kann man hier unterbringen, um Ordnung und Übersicht zu
schaffen ...  -->
</defs>

<g>
<!--  Ein Element g repräsentiert den restlichen, graphischen Inhalt.  -->
</g>

</svg>
```

Beispiel eines Dokumentes mit Elementen aus anderen Namensräumen:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG Tiny 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-tiny.dtd">
<svg width="100%" height="100%" viewBox="0 0 1000 1600"
  xmlns="http://www.w3.org/2000/svg"
  version="1.1" baseProfile="tiny"
  xml:lang="de">
<title>Gunillas geheimnisvolles Lächeln in Erwartung ihres Geliebten</title>

<desc>
<!-- Beschreibung mit Elementen aus dem Namensraum von XHTML -->
<div xmnls="http://www.w3.org/1999/xhtml">
<p>
<em>Gunilla</em> sitzt vor einem Spiegel und hat ihr gelocktes, rotes Haar
gelöst, welches wallend über ihre nackten Schultern fällt. Verträumt fährt sie
 sich
mit einem Kamm durch die glänzenden Locken, die sie so sehr <em>liebt</em>.
</p>
<p>
Ihre vollen Lippen hat Gunilla leicht lächelnd geöffnet und summt leise ein Lied
von der Liebe zu ihrem <strong><em>Siggi</em></strong>, welchen sie bereits
sehnsüchtig <strong>erwartet</strong>.
</p>
<p>
Herrlich ist das Licht- und Schattenspiel der Sonnenstrahlen auf
der zarten Haut ihres Antlitzes mit den vielen prachtvollen Sommersprossen.<br />
Je nach Lichteinfall erscheinen ihre Augen grau, grün oder blau, <strong>eine
spannende</strong> Mischung.
</p>
<p>
Eine erste kühle Brise fährt durch ihr Haar und kündet von der herannahenden
<strong>Sommernacht</strong> und läßt Gunilla leicht erschauern ...
</p>
</div>
</desc>

<metadata>
<!-- Metainformationen mit RDF und Dublin-Core -->
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/" >
 <rdf:Description rdf:about="">
   <dc:title>Gunillas geheimnisvolles Lächeln in Erwartung ihres
 Geliebten</dc:title>
   <dc:creator>Siggi Haase, Hannover (Linden), Deutschland</dc:creator>
```

```
   <dc:created>2009-10-01/10</dc:created>
   <dc:format>image/svg+xml</dc:format>
   <dc:language>de</dc:language>
   <dc:relation>http://de.wikibooks.org/wiki/SVG/_Barrierefreiheit</dc:relation>
   <dc:description>Gunilla Schneider; Portrait vor einem Spiegel beim
 Haarekämmen.</dc:description>
   <dc:rights>Die Urheber- und Veröffentlichungsrechte liegen beim Autor Siggi
 Haase aus Hannover,
              die abgebildete Person Gunilla Schneider aus Hannover hat einer
 Veröffentlichung
              zugestimmt und überläßt Siggi Haase Entscheidungen über weitere
 Veröffentlichungen.
              Zum persönlichen Gebrauch und zur Verwendung für pädagogische
 Zwecke etwa in Schulen
              und Universitäten darf das Dokument ohne Nachfrage vervielfältigt
 werden.
   </dc:rights>
 </rdf:Description>
</rdf:RDF>
</metadata>

<defs>
<!-- nicht direkt dargestellte Elemente ...  -->
</defs>

<g>
<!--  Ein Element g repräsentiert den restlichen, graphischen Inhalt.  -->
</g>

</svg>
```

4.3 Bedingte Verarbeitung

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	8 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (teilweise)
Konqueror (KSVG)	-
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Abb. 32 Bedingte Ausführung und Erweiterungen, ***switch*** und *requiredExtensions*

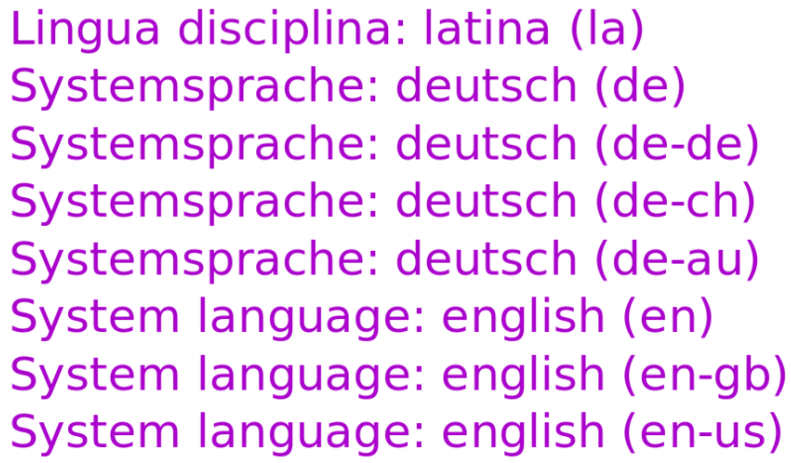


Abb. 33 Bedingte Ausführung und Sprachauswahl, *systemLanguage*, aber in diesem Beispiel einmal ohne ***switch***

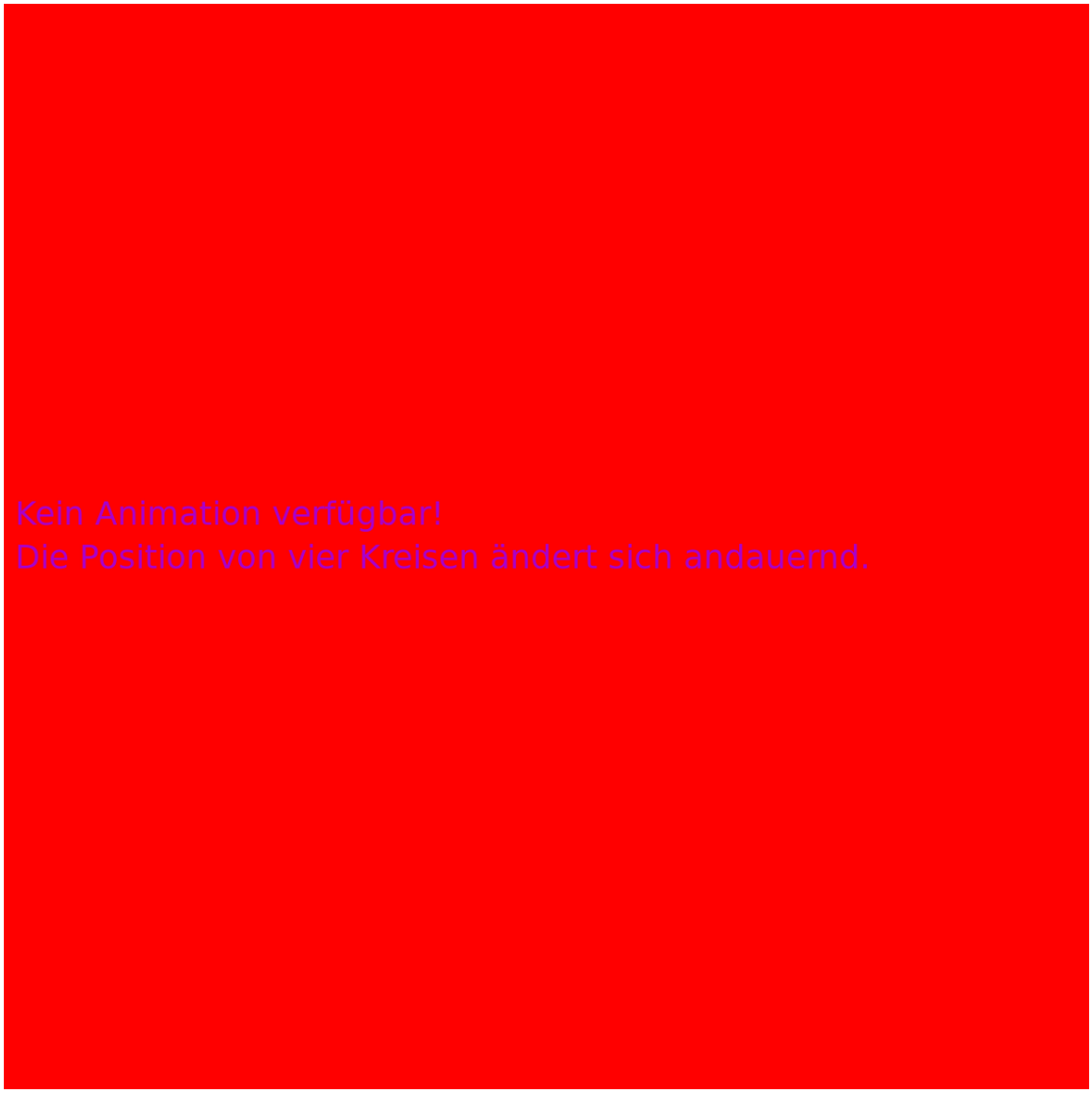


Abb. 34 Bedingte Ausführung und notwendige Merkmale, ***switch*** und *requiredFeatures*

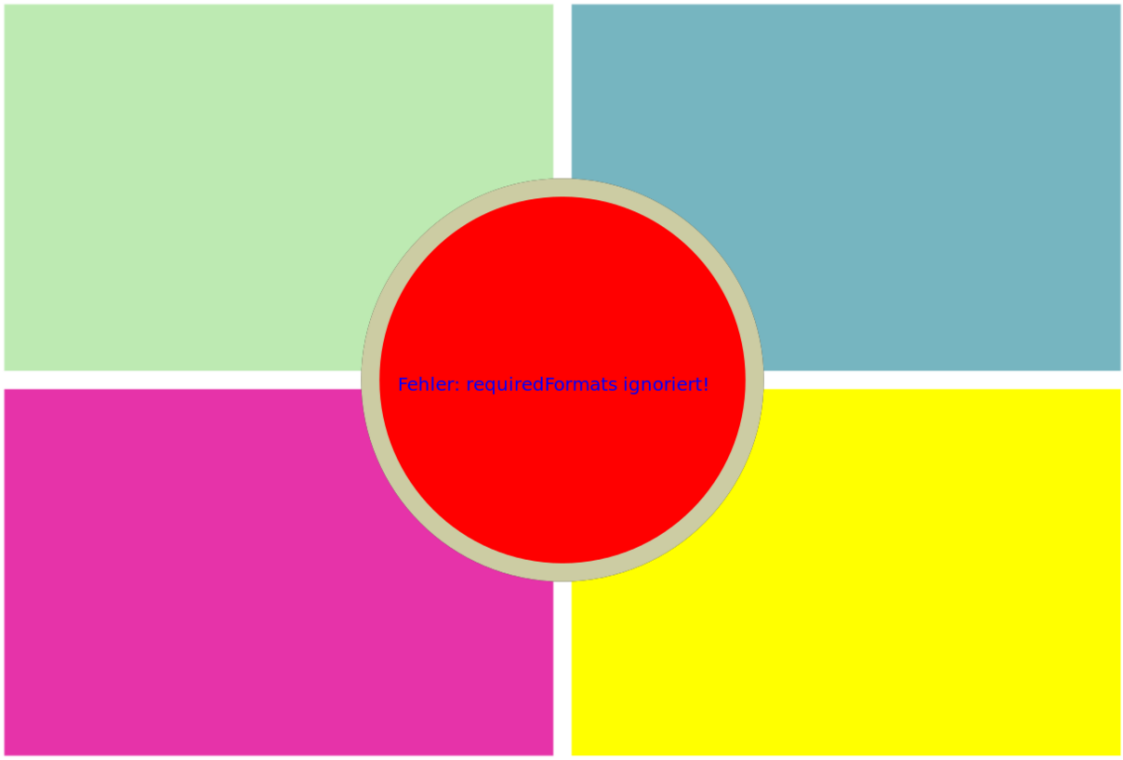


Abb. 35 Bedingte Ausführung und notwendige Formate, ***switch*** und *requiredFormats*; SVG tiny 1.2

Mit dem Element ***switch*** und speziellen Attributen ist es möglich, die Verarbeitung oder Darstellung abhängig von bestimmten Bedingungen zu machen. Die in SVG 1.1 dazu verfügbaren Attribute sind *requiredFeatures*, *requiredExtensions* and *systemLanguage*. SVG tiny 1.2 bietet zwei weitere Attribute zur bedingten Verarbeitung an, *requiredFormats* und *requiredFonts*.

Diese können als Attribut bei einem Element angegeben werden, bei dem eine bedingte Verarbeitung durchgeführt werden soll. Nur wenn die durch das Attribut angegebene Bedingung wahr ist, wird das Element verarbeitet oder angezeigt.

4.3.1 Attribut *systemLanguage*

Mit dem Attribut wird angegeben, für welche Sprachen des Nutzers der Inhalt des Elementes geeignet ist, bei dem das Attribut notiert ist. Der Wert von *systemLanguage* ist eine kommaseparierte Liste von Länderkennungen wie "de" oder "en" oder auch "en-gb", und der Wert ist wahr, wenn der Nutzer im Darstellungsprogramm angegeben hat, dass er Inhalte in wenigstens einer der angegebenen Sprachen verstehen kann.

4.3.2 Attribut *requiredExtensions*

Mit dem Attribut wird angegeben, welche Erweiterungen zur Anzeige des Elementes notwendig sind, bei dem das Attribut notiert ist. Der Wert von *requiredExtensions* ist eine leerraumseparierte Liste

von URIs, die eine notwendige Erweiterung identifizieren. Alle angegebenen Erweiterungen müssen interpretiert werden, damit der Wert wahr ist.

4.3.3 Attribut *requiredFeatures*

Das Attribut dient vor allem als Hilfe für Darstellungsprogramme, welche SVG bislang nur unvollständig implementiert haben. Durch Angabe von notwendigen Funktionalitäten kann der Autor so die Darstellung von den bereits implementierten Fähigkeiten des Darstellungsprogrammes abhängig machen. Der Wert von *requiredFeatures* ist eine leerraumseparierte Liste von Zeichenketten, sogenannten 'feature strings', die in der jeweiligen SVG-Spezifikation genau angegeben sind und Teile oder Module der Spezifikation bezeichnen. Wenn alle angegeben Module vom Darstellungsprogramm interpretiert werden, ist der Wert wahr.

4.3.4 Attribut *requiredFormats*

Mit dem Attribut wird angegeben, welche weiteren Formate zur Anzeige des Inhaltes eines Elementes notwendig sind, bei dem das Attribut notiert ist. Der Wert von *requiredFormats* ist eine leerzeichenseparierte Liste von internet-Medientypen (Inhaltstypen, ehemals MIME-Typen). Der Wert ist wahr, wenn alle angegebene Typen interpretiert werden.

4.3.5 Attribut *requiredFonts*

Mit dem Attribut wird angegeben, welche weiteren Schriftarten oder Zeichensätze zur Anzeige des Inhaltes eines Elementes notwendig sind, bei dem das Attribut notiert ist. Entsprechend ist der Wert von *requiredFonts* eine Liste von Schriftarten im Sinne der Eigenschaft *font-family*. Der Wert ist wahr, wenn alle angegebenen Schrifttypen verfügbar sind.

4.3.6 Element *switch*

Im Element ***switch*** können nun Kindelemente angegeben werden (besonders ***g*** bietet sich da an), von denen nur das erste dargestellt wird, bei dem die bedingte Verarbeitung wahr ist, die anderen werden nicht verarbeitet oder dargestellt. Das ermöglicht also, Alternativen anzubieten, wenn bestimmte Module nicht verfügbar sind oder wenn man mehrere Sprachen in einem Dokument angeben möchte. In der Regel ist es vorteilhaft, das letzte Kindelement von ***switch*** ohne ein Attribut zur bedingten Verarbeitung anzugeben, dann wird dieses in jedem Falle dargestellt, wenn alle vorherigen Kindelemente nicht verarbeitet werden.

Aufgrund der einfachen Syntax sind besonders bei der Sprachauswahl die Möglichkeiten des Autors begrenzt. Hat etwa ein hauptsächlich deutschsprachiger Nutzer die Länderkennungen "de" und "en" angegegeben, möchte aber bevorzugt deutsch lesen, ein anderer mag hauptsächlich englischsprachig sein, täte aber auch notfalls deutsch verstehn, so hat er die gleichen Kennungen angegeben (mit anderen Gewichtungen). Dies ist vom Autor nicht berücksichtigbar, was auch immer zuerst im ***switch*** drinsteht, wird verarbeitet, deutsch oder englisch. Es kann also sinnvoll sein, dem Nutzer eine manuelle Auswahl zusätzlich oder alternativ anzubieten oder die Möglichkeiten der sogenannten

'content-negotiation' von Webservern zu nutzen, um zwischen Darstellungsprogramm und Webserver auszuhandeln, welche Sprachversion bevorzugt ist.

4.3.7 Beispiele für bedingte Verarbeitung

Eine nähere Beschreibung ist jeweils im Dokument als Textalternative vorhanden.

- Bedingte Ausführung und Erweiterungen[2] (***switch*** und *requiredExtensions*)
- Bedingte Ausführung und Sprachauswahl[3] (*systemLanguage*, aber in diesem Beispiel einmal ohne ***switch***)
- Bedingte Ausführung und notwendige Merkmale[4] (***switch*** und *requiredFeatures*)
- Bedingte Ausführung und notwendige Formate[5] (***switch*** und *requiredFormats*; SVG tiny 1.2)

4.4 Elemente mit XLink-Funktionalität und Zugänglichkeit

Squiggle (Batik)	-
Opera (Presto)	9 (nur title)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (nur title)
Konqueror (KSVG)	-
Safari (Webkit)	4 (nur title)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Elemente, für die in SVG eine XLink-Funktionalität angegeben ist, können eine Reihe von Attributen aus dem Namensraum von XLink aufweisen, welche die eingebauten Zugänglichkeitshilfen von XLink bereitstellen. Es soll hier wie üblich für XLink-Attribute das Präfix xlink verwendet werden. Typische solche Elemente sind ***a***, ***image***, ***use*** oder Animationselemente wie ***animate***, in SVG tiny 1.2 auch Elemente wie ***audio***, ***video***, ***animation*** oder ***foreignObject***. Die XLink-Funktionalität ist kurz gesagt bei jenen Elementen gegeben, in denen das Attribut *xlink:href* von XLink zu verwenden ist, um auf andere Dokumente oder Dokumentfragmente zu verweisen oder solche einzubetten. Bei anderen Elementen, wo dies nicht explizit definiert ist, finden auch die anderen Attribute von XLink keine Anwendung. Neben diesem Kernattribut von XLink können weitere angegeben werden, um die Zugänglichkeit des Verweises zu verbessern.

xlink:title mit einer Zeichenkette als Attributwert kann analog zum *title*-Attribut von (X)HTML genutzt werden, um dem Verweis einen Titel zu geben oder auch einen Nutzungshinweis. Nachvollziehbar ist das zum Beispiel mit Gecko von Mozilla oder – sofern das nicht mit der Anzeige von anderen Nutzungshinweisen in Konflikt steht – auch mit Opera.

Mittels *xlink:role* kann auf eine URI verwiesen werden, wo die Rolle oder Funktion des Verweiszieles beschrieben wird. Der Attributwert ist daher also die URI eines Dokumentes oder Dokumentfragmentes, in dem die Rolle oder Funktion des Verweiszieles beschrieben wird.

Mittels *xlink:arcrole* kann auf eine URI verwiesen werden, wo die Rolle oder Funktion des Verweises selbst beschrieben wird. Der Attributwert ist daher also die URI eines Dokumentes oder

2 `http://de.wikibooks.org/wiki/media%3ASVGswitch01.svg`
3 `http://de.wikibooks.org/wiki/media%3ASVGswitch02.svg`
4 `http://de.wikibooks.org/wiki/media%3ASVGswitch03.svg`
5 `http://de.wikibooks.org/wiki/media%3ASVGswitch04.svg`

Dokumentfragmentes, in dem die Rolle oder Funktion des Verweises beschrieben wird. Die Funktion des Verweises ist allerdings in der Regel bereits in SVG durch die Elementdefinition gegeben, die Verwendung ist also allenfalls sinnvoll, wenn darüber hinaus eine besonders bemerkenswerte Funktion vorliegt.

4.4.1 Beispiele zur Verwendung von Attributen aus dem Namensraum XLink

Ein Kreis wird verwendet, um mittels des Elementes ***a*** auf die deutsche wikipedia-Seite zu verweisen. Das Verweisziel wird gegebenenfalls in einem neuen Fenster geöffnet (weil *xlink:show*="new" angegeben ist, alternativ könnte auch *target*="_blank" analog zu (X)HTML transitional angegeben werden):

```
<a
xlink:href="http://de.wikipedia.org"
xlink:title="wikipedia in einem neuen Fenster aufrufen"
xlink:show="new">
<circle r="100" fill="#ccf" stroke="#00f" stroke-width="10" />
</a>
```

Wenn die Darstellung eines Bildes nicht das primäre Einsatzziel eines ***image***-Elementes ist, sondern zum Beispiel in einer Anleitung zur Erstellung von SVG-Dokumenten nur als Beispiel dient, kann es durchaus sinnvoll sein, diese zusätzliche Funktion mittels *xlink:arcrole* anzugeben. Statt das Hauptmotiv des Bildes im Detail zu erläutern, kann es zum Beispiel für einen blinden Nutzer auch hilfreich sein, mittels *xlink:role* einen Verweis auf einen Fachartikel angeboten zu bekommen, wo im Detail nachgelesen werden kann, um was es sich genau handelt. Konkretes Beispiel:

```
<image width="300" height="200"
xlink:href="Zitronenfalter.jpg"
xlink:title="Zitronenfalter auf Sommerflieder"
xlink:role="http://de.wikipedia.org/wiki/Zitronenfalter"
xlink:arcrole="http://de.wikipedia.org/wiki/Beispiel">
<title>Zitronenfalter</title>
<desc>
Ein Zitronenfalter sitzt mit ausgebreiteten Flügeln
auf einem Sommerflieder und saugt Nektar.
Der Hintergrund des Bildes löst sich in diffusem Grün auf.

Beispiel für die Verwendung eines image-Elementes in SVG.
</desc>
</image>
```

Entsprechend bietet sich manchmal eine ähnliche Vorgehensweise beim Element ***use*** an:

```
<g stroke-width="10" stroke="#00f">
<use id="Start" x="100" y="100" fill="#0a0"
xlink:href="#Knopf"
xlink:title="Stoppuhr starten"
xlink:role="http://www.w3.org/TR/wai-aria/#button" />

<use id="Stop" x="100" y="200" fill="#c00"
xlink:href="#Knopf"
xlink:title="Stoppuhr stoppen"
xlink:role="http://www.w3.org/TR/wai-aria/#button" />
</g>
```

Um die Rolle oder Funktion eines jeden Elementes, nicht nur aber auch mit Bezug auf WAI-ARIA angeben zu können, bietet indes SVG-tiny 1.2 auch noch das Attribut *role*.

4.5 Zugänglichkeitsprobleme durch Stilvorlagen und Skripte vermeiden

In SVG tiny 1.1 sind weder Stilvorlagen noch Skripte zu interpretieren, in SVG tiny 1.2 ist die Interpretation von Stilvorlagen und von Skripten optional - und die Spezifikation legt nicht fest, welche Skriptsprache zu verwenden ist, wenn eine verwendet wird. Dies ist auch in SVG 1.1 nicht vorgeschrieben, anders etwa als die Interpretation der Bildformate PNG und JFIF/JPEG, die nicht optional sind, ebenso wenig wie die Interpretation von CSS, so weit es für SVG spezifiziert ist.

Verwendet ein Betrachter nun ein Darstellungsprogramm, welches auf SVG tiny 1.1 ausgelegt ist, kann der Autor eine Interpretation von Stilvorlagen und Skripten nicht erwarten, auch bei SVG tiny 1.2 ist das fraglich, allerdings nicht mehr explizit ausgeschlossen. Bei SVG 1.1 und SVG tiny 1.2 ist immerhin unsicher, ob und welche Skriptsprache vom Darstellungsprogramm interpretiert wird. Zudem kann bei den allermeisten Programmen der Nutzer die Skriptinterpretation deaktivieren - aus Gründen der Sicherheit oder der Weltanschauung, warum auch immer.

Bei SVG tiny 1.2 kann immerhin mittels *requiredFormats* abgefragt werden, ob die Interpretation einer bestimmten Skriptsprache gegeben ist. Allgemeiner kann auch mit *requiredFeatures* abgefragt werden, ob generell eine Interpretation von Skriptsprachen möglich ist (die Antwort zum Beispiel von Opera ist falsch, das Programm behauptet (Stand Mitte 2011, getestet bis Version 11.10) immer, es könne Skripte, insbesondere ecmaScript und JavaScript interpretieren, selbst wenn der Nutzer dies explizit deaktiviert hat und somit auch keine Interpretation in SVG erfolgt).

Zusammenfassend kann also festgehalten werden, dass Stilvorlagen und Skripte ein Zugänglichkeitsproblem für bestimmte Nutzergruppen darstellen können, andererseits für andere aber auch wiederum Hilfen, den Inhalt besser zu verstehen. An der Stelle ist der Autor eines Dokumentes gefragt, diese Methoden so einzusetzen, dass gerade auch ohne deren Interpretation dennoch der Inhalt, die Information des Dokumentes verfügbar und zugänglich ist.

Hinsichtlich der Dekoration mit Stilvorlagen ist die Lösung gleich in SVG eingebaut. Bis auf sehr wenige Ausnahmen (von Eigenschaften, die mehrere verschiedene Eigenschaften zusammenfassen), gibt es für in SVG verfügbare CSS-Eigenschaften Präsentationsattribute, die dann auch verwendet werden sollten, sofern die Eigenschaft zum Informationsgehalt des Bildes beiträgt. Ist es zum Beispiel für das inhaltliche Verständnis relevant, dass ein Rechteck nicht schwarz, sondern rot ist, so wird die Farbe mit einem Präsentationsattribut angegeben. Zudem kann es sinnvoll sein zu prüfen, ob noch eine sinnvolle Anzeige zustande kommt, wenn eine Konversion der Farben in Grauwerte vorgenommen wird, was oft beim Drucken passiert, aber auch bei Grauwert-Monitoren oder anderen Hilfen für Menschen, die spezifische Probleme in der Farbwahrnehmung haben.

Präsentationsattribute sind in (X)HTML weitgehend verpönt, weil sie dort praktisch nur dekorativen Zwecken dienen, nicht inhaltlichen. Das ist bei Graphik ganz anders. Wenn dort alle Elemente schwarz gefüllt sind, ist keine Information mehr erkennbar, also haben Farben und Teiltransparenzen etc meist eine wichtige inhaltliche Funktion, die durch Einsatz von Präsentationsattributen zu gewährleisten ist. Eine andere, alternative Darstellung kann allerdings problemlos mit der Angabe von CSS-Eigenschaften ergänzt werden. Die CSS-Angaben haben eine höhere Spezifität als die Angaben in Präsentationsattributen, überschreiben diese also gegebenenfalls.

Verwendet man bestimmte Hilfsprogramme (wie zum Beispiel inkscape), so ist es nicht unwahrscheinlich, dass inhaltlich wichtige Angaben nicht in Präsentationsattributen abgelegt werden, sondern als Eigenschaften im Attribut *style*. Um den Zugang zum relevanten Inhalt zu optimieren, sind dann

gegebenenfalls manuell Präsentationsattribute zu ergänzen oder die Angaben in style in Präsentationsattribute zu konvertieren, um besser zugängliche Information anzubieten.

Bei Skripten ist die Lösung meist weniger einfach. Generell sollte der Autor immer im Kopf behalten, dass das, was ohne Skriptinterpretation als Inhalt zugänglich ist, der Inhalt, die Aussage seines Dokumentes ist. Die Anwendung von Skripten fügt an sich keine weitere Information hinzu, bietet allenfalls einen alternativen Zugang zum ohnehin vorhandenen Inhalt. Wenn es für den Autor so aussieht, dass dieses Kriterium nicht erfüllt ist, kann er ohne Zweifel davon ausgehen, dass sein Dokument nicht barrierefrei und für alle Nutzer zugänglich ist. Solange nur ein definierter Nutzerkreis mit bekannten Möglichkeiten vorliegt, muss das kein Problem darstellen. Wenn sichergestellt ist, dass jedem, dem das Dokument zugänglich ist, auch der Inhalt wie geplant zugänglich ist, tritt kein Barriereproblem auf. Sobald das Dokument aber allgemein verfügbar veröffentlicht ist, tritt spätestens das Problem der mangelnden Zugänglichkeit auf. Es obliegt dann dem an Zugänglichkeit und Barrierefreiheit interessierten Autor (kann bei einigen gesetzlich vorgeschrieben sein, dass sie interessiert sein müssen), das Dokument so aufzubereiten, dass es zugänglich ist. Oder er muss notfalls Alternativen bereitstellen, die die Information zugänglich werden lassen.

4.6 Funktion und Semantik in SVG tiny 1.2

Squiggle (Batik)	-
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

SVG tiny 1.2 bietet einige Attribute, die von XHTML und RDFa übernommen wurden, um die Zugänglichkeit und semantische Aussagekraft und die Maschinenlesbarkeit und -verstehbarkeit zu verbessern.

Ein Problem nicht nur von SVG ist etwa, dass die Funktion für den Benutzer von komplizierteren Konstruktionen für die darstellenden Programme nicht erkennbar ist. Ist der Nutzer etwa selbst oder durch Mängel des Programmes in der Rezeption eines Dokumentes wie vom Autor gedacht beeinträchtigt, so kann mit WAI-ARIA oder mit anderen Spezifikationen die Funktionalität einheitlich beschrieben werden, so dass Programmierer für die Funktionalitäten im Darstellungsprogramm generische Konstrukte bereitstellen können, welche bei solchen Beeinträchtigungen die Konstrukte des Autors ersetzen können und damit auch die generelle Funktionalität bereitstellen können. Zu diesem Zwecke gibt es das Attribut *role*, mit dem eine solche Funktionalität wie ein Knopf (button) oder Nutzungshinweis (tooltip) angegeben werden können, Die Basiswerte sind in WAI-ARIA definiert, aber Interessierte können darüber hinaus auch eigene Funktionalitäten spezifizieren, wobei der Attributwert dann eine CURIE ist, eine spezielle Methode, um URIs abzukürzen, die auf die Definition der Funktionalität verweisen. Da sowohl WAI-ARIA als auch die Spezifikation einer CURIE noch Arbeitsentwürfe waren, als SVG tiny 1.2 als Empfehlung veröffentlich wurde, ist der Attributwert noch nicht normativ auf die genannte Syntax einschränkt. Für die Funktionalität und Sinnhaftigkeit der Verwendung ist aber davon abzuraten, eigene Wege zu gehen. Sofern die verwendete Sprache selbst über generische Konstrukte verfügt, ist es natürlich besser, diese zu verwenden, statt etwa über teils unzugängliches Skripte ein an sich nicht dafür gedachtes Element mit der Funktionalität zu dekorieren und dann die Funktionalität nur mittels *role* kenntlich zu machen.

Nun ist es auch zwangsläufig so, dass nicht für alles, was mit SVG darstellbar ist, auch ein spezifisches Element vorhanden ist, es ist etwa nicht zu erwarten, dass für eine schlafende Katze oder einen afrikanischen Elefanten in einer Stampede ein passendes Element definiert wird.

RDFa bietet die Möglichkeit, zumindest für häufig auftretende Anwendungen andere Spezifikationen zu referenzieren, um die Bedeutung von eher generischen Elementen zu präzisieren. Insbesondere bei Textinhalten können sogar maschinenlesbare und -verstehbare Notationen von Textstrukturen angegeben werden, wofür es bereits zahlreiche spezielle Vokabularien gibt. SVG hingegen spezifiziert bei Textelementen nur, wie diese graphisch darzustellen sind, nicht was sie bedeuten (Überschrift, Abschnitt, Absatz, Liste etc). Zum Beispiel können dafür Elemente von (X)HTML referenziert werden oder auch Elemente aus anderen Sprachen mit besserer Semantik wie LML.

RDFa ist die Attributversion von RDF, benutzt also das gleiche Prinzip von Tripeln - Subjekt, Prädikat, Objekt. RDFa ist so konstruiert, dass in der Regel das Tripel automatisch aus der Dokumentstruktur und den verwendeten Attributen folgt. Wie bei *role* werden auch hier CURIEs verwendet, um die exakte Bedeutung aus anderen Spezifikationen zu referenzieren. Somit sind auch diese Attributwerte in SVG tiny 1.2 noch nicht normativ eingeschränkt, eine Abweichung von den RDFa-Vorgaben führt aber zu keinem semantisch sinnvollen und eindeutigen Ergebnis. Die Attribute sind im Detail in der Spezifikation von XHTML+RDFa definiert.

4.6.1 Attribut *role*

Mittels *role* wird die Funktion eines Elementes gekennzeichnet. Der Wert ist eine mit Leerzeichen speparierte Liste von Funktionen. Die Funktionen werden entweder durch eine Zeichenkette repräsentiert, die eine von WAI-ARIA vordefinierte Funktion beschreibt, oder durch CURIE, einen (abgekürzten) Verweis auf eine Definition der Funktion.

4.6.2 Attribut *about*

about gibt das Subjekt einer Aussage an, also worum es geht. Der Wert ist im Wesentlichen eine URI zu einem Dokument oder Dokumentfragment, auf welches sich die Aussage bezieht. Sofern nicht angegeben ist dies das aktuelle Element.

4.6.3 Attribut *property*

property gibt das Prädikat einer Aussage an, also in welcher Relation das Subjekt zum Objekt steht. Sofern nicht anders angegeben, ist der Elementinhalt das Objekt, um welches es geht.

4.6.4 Attribut *resource*

resource ist eine Möglichkeit, das Objekt einer Aussage anzugeben, sofern es nicht der Elementinhalt selbst ist. Der Wert ist im Wesentlichen eine URI zu einem Dokument oder Dokumentfragment.

4.6.5 Attribut *datatype*

datatype gibt den Datentyp an, indem eine Information angegeben ist. Der Wert ist eine CURIE zur Definition des Datentyps.

4.6.6 Attribut *typeof*

typeof gibt an, mit welchen Datentypen das Subjekt assoziiert ist. Der Wert ist eine mit Leerzeichen separierte Liste von CURIEs zu den Definitionen der Datentypen.

4.6.7 Attribut *content*

content ist ein Ersatz für den Inhalt des Elementes (oder bei einem inhaltsleeren Element offenbar der Inhalt selbst). Die als Wert notierte Zeichenkette ist ein Textäquivalent für den Inhalt. Sofern *datatype* ebenfalls angegeben ist, entspricht die Zeichenkette der maschinenlesbaren Variante, während der Elementinhalt eine freiere Variante darstellt. Für eine formale Auswertung wird daher der Wert des Attributes verwendet statt der freien, schwerer zu interpretierenden Variante.

4.6.8 Attribute *rel* und *rev*

rel und *rev* sind von (X)HTML übernommen und enthalten eine mit Leerzeichen separierte Liste von CURIEs, um Beziehungen oder Prädikate zwischen zwei Ressourcen anzugeben.

4.6.9 Beispiele zu *role* und RDFa

Ein Knopf mit Nutzungshinweis:

```
<circle id="Start" cx="100" cy="100" r="100"
fill="#0a0"stroke-width="10" stroke="#00f"
role="button">
<title>Knopf zum Beginn der Jonglage</title>
<desc>Ein Clown fährt auf seinem Einrad durch die Manege und jongliert dabei
mit sechs rosafarbenen Pudelmützen mit grünen Bommeln, wenn dieser Knopf
aktiviert oder angeklickert wird.</desc>
<metadata xmnlns:wai="http://www.w3.org/TR/wai-aria/#" role="tooltip"
 property="wai:tooltip"
content="Anklickern oder Aktivieren zum Start der Animation" />
<animateMotion xlink:href="#Clown" dur="600s" begin="Start.click;
 Start.activate">
<mpath xlink:href="#ClownPfad" />
</animateMotion>
</circle>
```

Im folgenden Beispiel werden Überschrift und Absatz aus HTML4 als *property* angegeben, weil SVG tiny 1.2 selbst über keine semantisch relevante Auszeichnung von Text verfügt. Etwas anwenderfreundlicher als HTML4 für diesen Zweck ist wohl zum Beispiel eine Sprache wie LML, die bereits unter anderem zur Verwendung als RDFa-Referenz ausgelegt ist.

Hier wird dann gleichzeitig auch eine neue Funktionalität von SVG tiny 1.2 genutzt, um Fließtext automatisch umzubrechen, was in SVG tiny 1.1 nicht umzusetzen war. Man beachte dabei auch den

Unterschied zwischen der graphischen Formatierung mit dem Element ***tbreak*** und der semantischen Strukturierung durch Anwendung von *property* auf ***tspan*** innerhalb von ***textArea***.

```
<g font-family="sans-serif" xmnls:h="http://www.w3.org/TR/html4/struct/"

<text font-size="20"
x="100" y="100"
property="h:global.html#h-7.5.5">Hallo Welt!</text>

<textArea font-size="10"
x="100" y="150" width="500">
<tspan property="h:text.html#h-9.3.1">So grüßen wir die Stadt,
das Land, die Erde, die Galaxis!<tbreak />
Und ganz besonders die SVG-Arbeitsgruppe des W3C.</tspan><tbreak /><tbreak />

<tspan property="h:text.html#h-9.3.1">Der Gruß impliziert eine gewisse
 Weltoffenheit
und eine positive Grundeinstellung und die philosophische Hypothese, daß es eine
Trennung zwischen dem Ich und der Welt gibt, etwa im Gegensatz zum
 solipsistischen
Ansatz, wo dieser Gruß nur ein Selbstgespräch wäre.</tspan><tbreak /><tbreak />

<tspan property="h:text.html#h-9.3.1">
Die Nutzung des Imperativs zeigt wiederum eine gewissen Entschlossenheit,
der Gruß selbst den Willen, offen auf die Welt zuzugehen, um eine Brücke
zwischen dem Ich und der Welt zu schlagen.</tspan>
</textArea>

</g>
```

Diverse Beispiele, allerdings mit Verwendungen in XHTML, sind in der XHTML+RDFa-Spezifikation angegeben. Spezielle Vokabularien sind derzeit (2011) für SVG noch nicht bekannt, aufgrund der gut durchdachten CURIE-Syntax kann aber eigentlich jede interessierte Gruppe eine Vokabelkollektion spezifizieren, damit diese von allen Autoren einheitlich genutzt werden kann.

4.7 Literatur

Ein Hinweis zur Barrierefreiheit von SVG[6] ist von der SVG-Arbeitsgruppe des W3C veröffentlicht worden und liefert Hilfen für eine bessere Zugänglichkeit von Dokumenten.

Text aus SVG extrahieren[7], nützlich, um ein Gefühl zu bekommen wie sich ein SVG ohne G anfühlt.

XLink[8], XML-Sprache zur Beschreibung von Verweisen, in SVG verwendet.

RDF[9], Rahmenwerk zur Beschreibung von Quellen (Meta-Informationen zu Werken).

Semantic Web Activity[10], Semantik im internet.

DCMI[11] Dublin Core Metadata Initiative - Sprache zur Angabe von Meta-Informationen über Werke.

6 http://www.w3.org/TR/SVG-access/#Accessibil1
7 http://www.w3.org/2002/05/svg2stuff.html
8 http://www.w3.org/TR/xlink/
9 http://www.w3.org/RDF/
10 http://www.w3.org/2001/sw/
11 http://dublincore.org/

Text in SVG[12], Vorschläge zur Verwendung und Interpretation von Text in SVG.

IANA Medientypen[13], Liste registrierter internet-Medientypen (MIME).

RDFa in XHTML[14], Verwendung von RDFa-Attributen

WAI-ARIA[15], Anwendungen und Funktionalitäten angeben und zugänglicher machen

role[16], XHTML: *role*-Attribut

CURIE-Syntax[17], URIs abkürzen

LML[18], Auszeichnungssprache für Literatur, geeignet für RDFa in SVG tiny 1.2, um Text semantisch relevant zu kennzeichnen

12 http://purl.oclc.org/net/hoffmann/svgtext/
13 http://www.iana.org/assignments/media-types/
14 http://www.w3.org/TR/rdfa-syntax/
15 http://www.w3.org/TR/wai-aria/
16 http://www.w3.org/TR/xhtml-role/
17 http://www.w3.org/TR/curie/
18 http://purl.oclc.org/net/hoffmann/lml/

5 Transformationen

5.1 Koordinatensysteme und Transformationen

In SVG können mit diversen Elementen neue Anzeigebereiche eröffnet werden, welche dann mit dem Attribut *viewBox* auch ein neues Koordinatensystem erhalten. Entsprechend wird eine (implizite) Transformation vom lokalen neuen Koordinatensystem vorgenommen, um den neuen Inhalt im alten Koordinatensystem des umgebenden Elementes darzustellen.

Daneben ist das Attribut *transform* verfügbar, und zwar bei graphischen Elementen und Gruppierungselementen. Für Farbverläufe und Muster gibt es zudem die verwandten Attribute *gradientTransform* und *patternTransform*, um diese Maldienste ebenfalls zu transformieren.

Verfügbar sind in SVG 1.1 und SVG tiny 1.2 affine Transformationen, also solche, die sich durch Matrizen darstellen lassen, welche Objekte vom neuen ins alte Koordinatensystem transformieren. Für einen Autor ist es nützlich, sich etwas mit Matrizenrechnung auszukennen, um die Effekte genau zu verstehen oder eine beliebige Transformation direkt angeben zu können. Für typische und gängige Transformationen stellt SVG allerdings auch einige spezielle Typen bereit, so dass solche Kenntnisse und Fertigkeiten nicht unbedingt notwendig sind.

5.1.1 Attribut *transform*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Attribut *transform* können Gruppierungselemente und graphische Elemente transformiert werden.

Der Wert von *transform* ist eine Liste von Transformationen, die dann nacheinander ausgeführt werden. Die jeweiligen Transformationen der Liste sind durch Leerzeichen oder Komma separiert.

Die Transformationen werden durch Transformationstypen angegeben, die aus einem Schlüsselwort bestehen, unmittelbar gefolgt von Zahlen in Klammern. Die Transformationen entsprechen Matrizen, welche vom neuen ins alte Koordinatensystem transformieren. Die Matrix beschreibt also mathematisch, welche Koordinaten der transformierte Inhalt im alten Koordinatensystem hat. Koordinaten und Abmessungen des zu transformierenden Element und gegebenenfalls der Kindelemente sind also im neuen Koordinatensystem angegeben und werden somit durch die Transformation in das alte, umgebende Koordinatensystem übertragen.

Ist V ein Vektor im neuen Koordinatensystem, so ergibt sich zum Beispiel mit anzuwendenden Transformationen A, B, C der Vektor V' im alten Koordinatensystem:

V' = A * B * C * V, wobei im Attribut *transform* die entsprechenden Schlüsselwörter in gleicher Reihenfolge notiert sind, also sinngemäß *transform*="A(...) B(...) C(...)".

Um Verschiebungen mit in die Matrixschreibweise einzubauen, wird ein Trick verwendet. Es wird eine dritte Koordinate eingeführt, die immer 1 ist, ein Vektor sieht dann also wie folgt aus: V = (V_x V_y 1).

Eine Matrix von dieser Art mit den Zahlen a bis f sieht allgemein wie folgt aus:

a	c	e
b	d	f
0	0	1

Zu beachten ist dabei, dass eine Vertauschung der Reihenfolge meistens zu einem anderen Ergebnis führt. Das ist ein Unterschied zur Multiplikation von einfachen Zahlen.

Das Produkt aller Transformationsmatrizen bis hinauf zum nächstgelegenen Elternelement, welches einen neuen Anzeigebereich eröffnet, wird die aktuelle Transformationsmatrix eines Elementes genannt (englisch: Current Transform Matrix; abgekürzt CTM). Die verbindet also ein Objekt im lokalen Koordinatensystem mit der Darstellung im nächstgelegenen Anzeigebereich. Ist also V ein Vektor zu einem Punkt eines Objektes im lokalen Koordinatensystem, so ist V' = CTM * V die Darstellung im nächstgelegenen Anzeigebereich.

Das Attribut *transform* ist animierbar mit dem Element ***animateTransform***. Dazu wird der jeweilige Typ animiert.

Folgendes sind die verfügbaren Transformationstypen (in SVG tiny 1.2 kommen noch eingeschränkte Transformationen hinzu, siehe nächsten Abschnitt):

'translate', Verschiebung

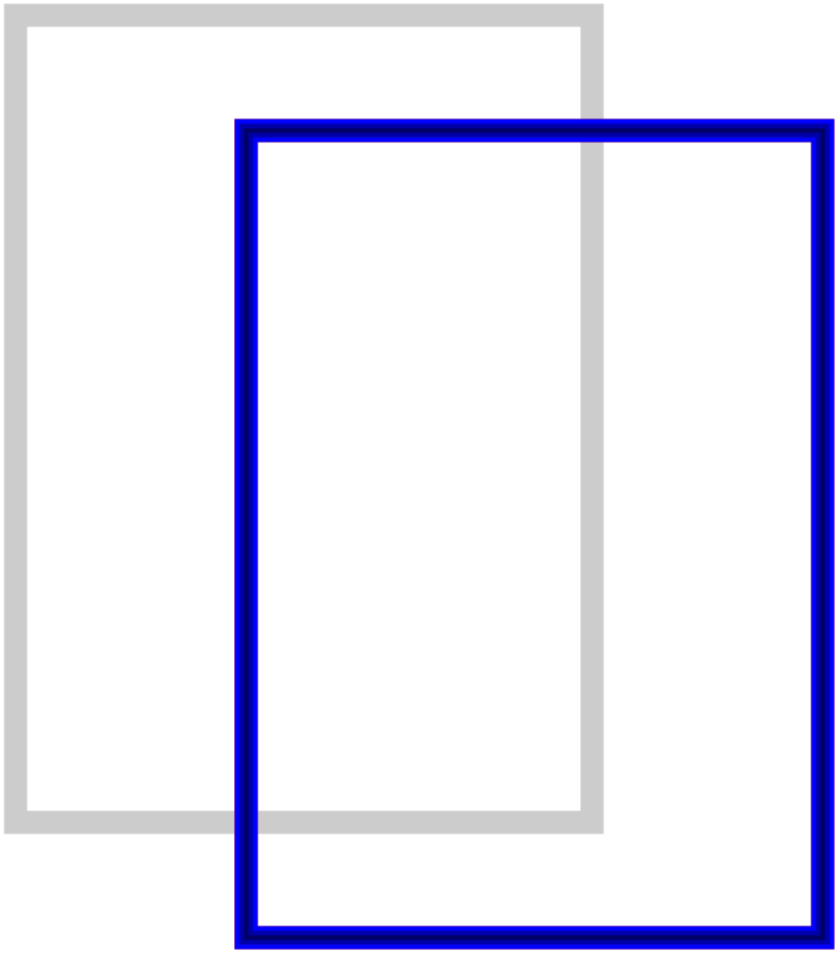


Abb. 36 Beispiel Verschiebung

'translate(e f)' spezifiziert eine Verschiebung oder Translation. e ist eine Verschiebung um e in x-Richtung, f ist eine Verschiebung um f in y-Richtung. f ist optional, sofern nicht angegeben, wird 0 angenommen. Ein Beispiel ist also translate(60 40) für eine Verschiebung um 60 in x-Richtung und um 40 in y-Richtung. translate(50) beschreibt eine Verschiebung um 50 in x-Richtung. translate(0 -50) beschreibt eine Verschiebung um -50 in y-Richtung.

Eine Verschiebungsmatrix um e in x-Richtung und f in y-Richtung sieht wie folgt aus:

1	0	e
0	1	f

0	0	1

Beispiel Verschiebung[1]

Blaue gerandete Rechtecke werden mittels des Attributes transform vom Typ translate verschoben, so dass ein rotes gerandetes Rechteck komplett verdeckt wird. Ein grau gerandetes Rechteck zeigt die unverschobene Position.

Gleiche Transformationstypen vertauschen immer miteinander, dies wird mit den dunkler gerandeten Rechtecken angedeutet.

1 http://de.wikibooks.org/wiki/media%3ASVGtransform_translate01.svg

'scale', Skalierung

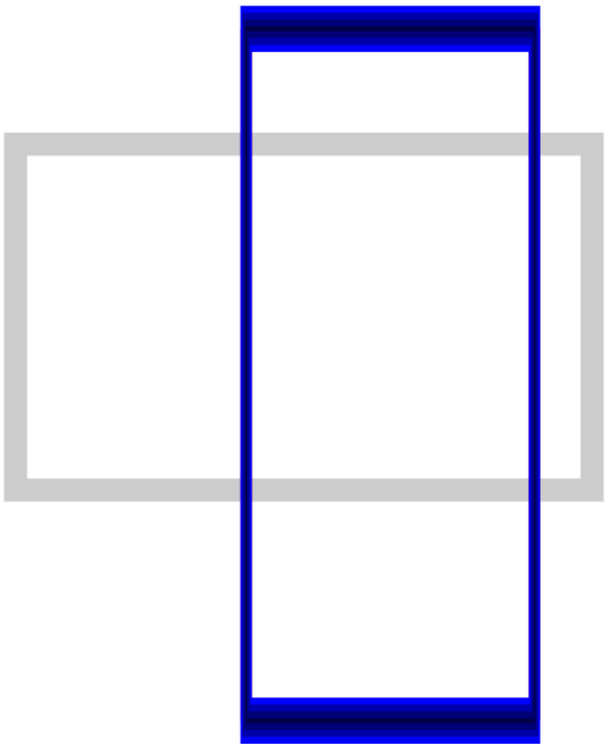


Abb. 37 Beispiel Skalierung

'scale(a d)' ergibt eine Skalierung. a ist eine Skalierung der x-Richtung entsprechend einer Multiplikation mit a. d ist eine Skalierung der y-Richtung entsprechend einer Multiplikation mit d. d ist optional, sofern nicht angegeben, wird der gleiche Wert wie für a angenommen. Ein Beispiel ist also scale(-1 2), x-Werte werden mit -1 multipliziert, also eine Spiegelung an der y-Achse, y-Werte werden mit zwei multipliziert. Mittels scale(-1) erfolgt eine Punktspiegelung am Ursprung, mit scale(1 -1) eine Spiegelung an der x-Achse. Mit scale(0 1) wird alles auf die x-Achse abgebildet. In SVG 1.1 sind so transformierte Inhalte prinzipiell nicht mehr sichtbar. In SVG tiny 1.2 ergibt sich mit *vector-effect*="non-scaling-stroke" gegebenenfalls noch ein sichtbarer Strich.

Eine Skalierungsmatrix um a in x-Richtung und d in y-Richtung sieht wie folgt aus:

a	0	0
0	d	0
0	0	1

Beispiel Skalierung[2]

Blaue gerandete Rechtecke werden mittels des Attributes transform vom Typ scale skaliert, so dass rot gerandete Rechtecke komplett verdeckt werden. Ein grau gerandetes Rechteck zeigt das nicht transformierte Rechteck.

Gleiche Transformationstypen vertauschen immer miteinander, dies wird mit den dunkler gerandeten Rechtecken angedeutet.

2 http://de.wikibooks.org/wiki/media%3ASVGtransform_scale01.svg

'rotate', Drehung

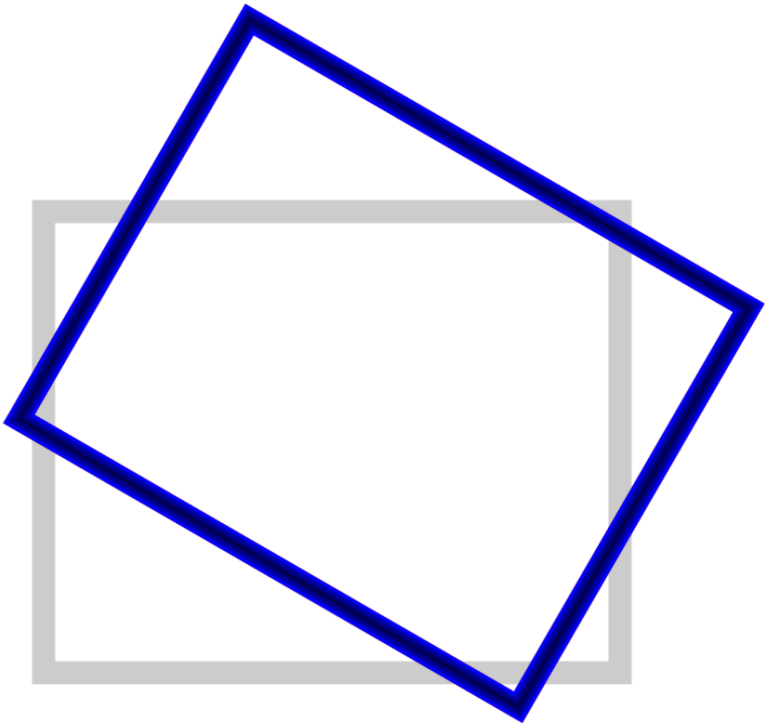


Abb. 38 Beispiel Drehung

Mit rotate(w x y) wird eine Rotation um einen Winkel w (in Grad) um den Punkt (x, y) vorgenommen. Die positive Drehrichtung geht von der Richtung der x-Achse zur positiven Richtung der y-Achse. x und y sind optionale Angaben, sofern nicht vorhanden, wird 0 angenommen. Zum Beispiel dreht rotate(90) die x-Achse in die y-Richtung und die y-Achse in die -x-Richtung, rotate(-90) die x-Achse in die -y-Richtung und die y-Achse in die x-Richtung. rotate(37 11 -13) beschreibt eine Drehung um 37 Grad um den Punkt (11 -13).

rotate(w x y) ist gleich translate(x,y) rotate(w) translate(-x, -y).

Eine Drehmatrix um den Winkel w um den Ursprung sieht wie folgt aus:

cos(w)	-sin(w)	0
sin(w)	cos(w)	0
0	0	1

Beispiel Drehung[3]

Blaue gerandete Rechtecke werden mittels des Attributes transform vom Typ rotate rotiert, so dass ein rot gerandetes Rechteck komplett verdeckt wird. Ein grau gerandetes Rechteck zeigt das nicht transformierte Rechteck.

Verschiedene Kombinationen von translate und rotate oder nur rotate können zum gleichen Ergebnis führen, dies wird mit den dunkler gerandeten Rechtecken angedeutet.

3 http://de.wikibooks.org/wiki/media%3ASVGtransform_rotate01.svg

'skewX' und 'skewY', Scherung

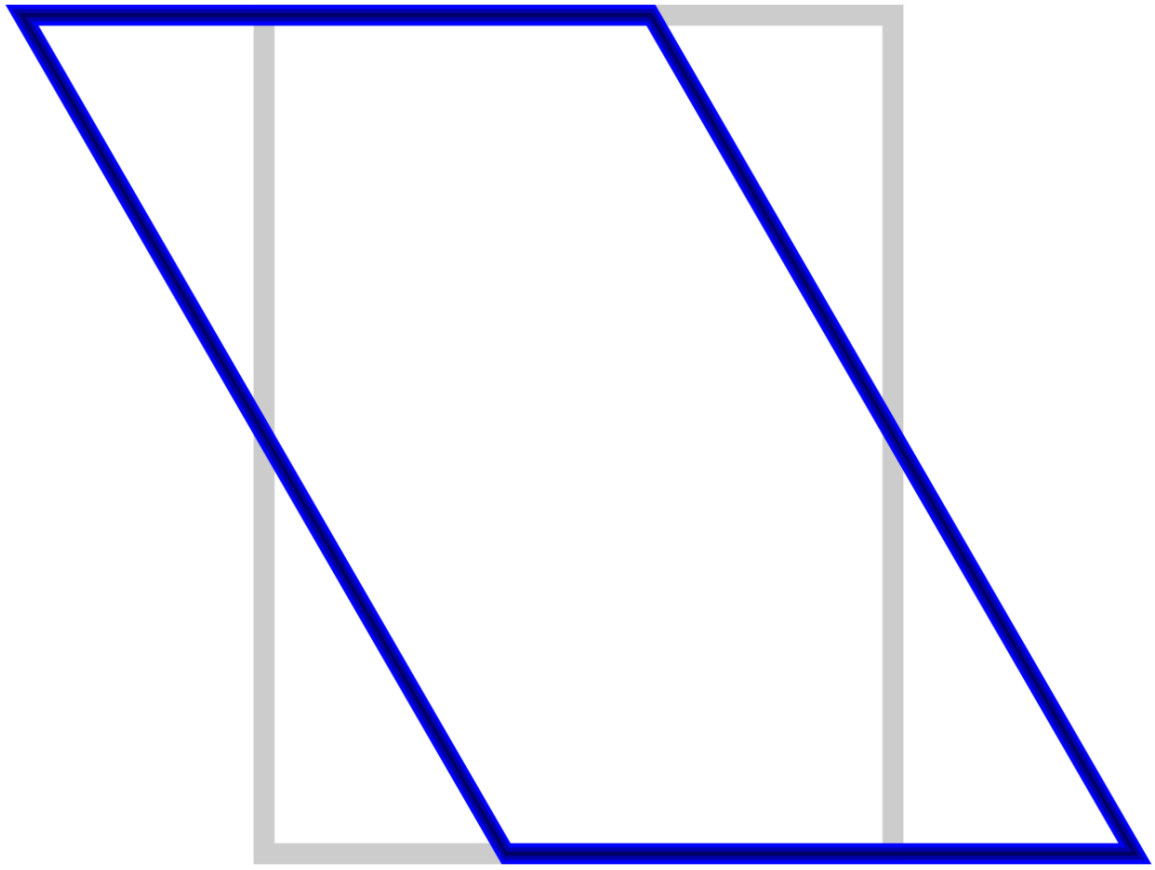


Abb. 39 Beispiel Scherung in x-Richtung

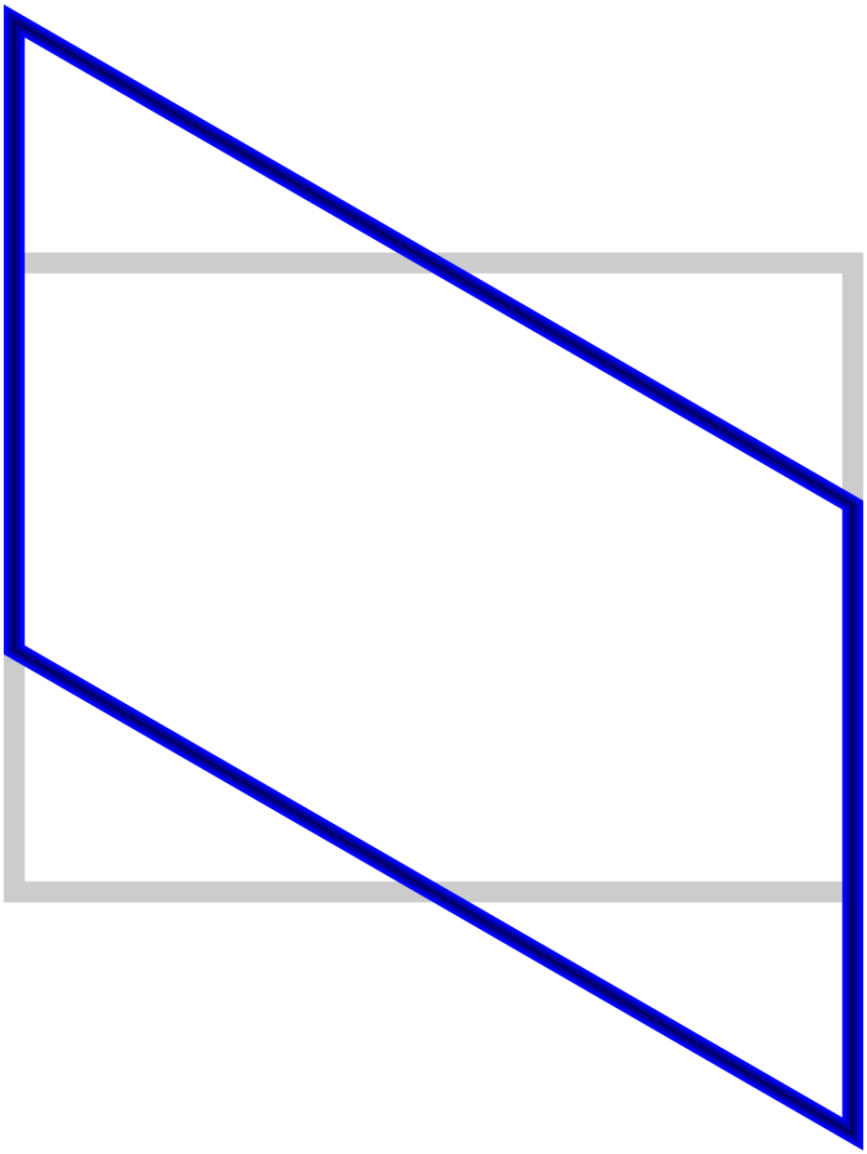


Abb. 40 Beispiel Scherung in y-Richtung

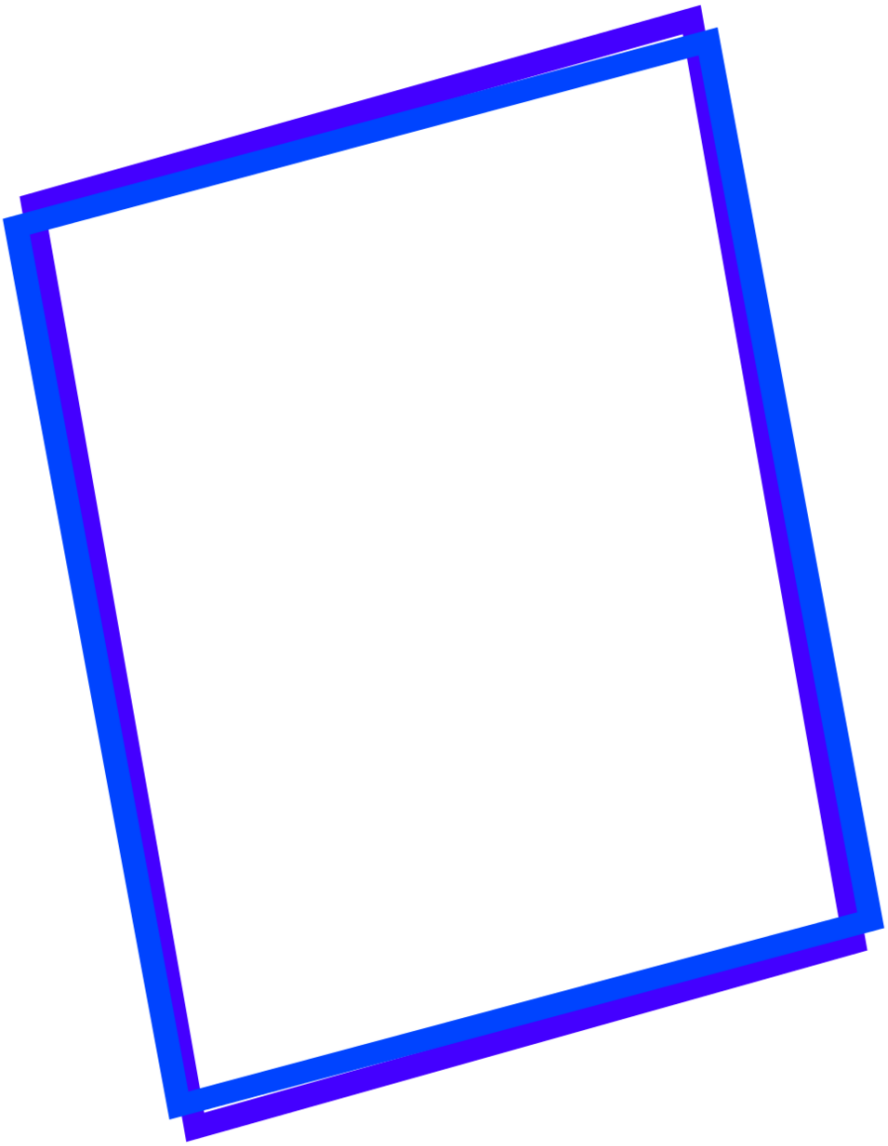


Abb. 41 Beispiel Scherung

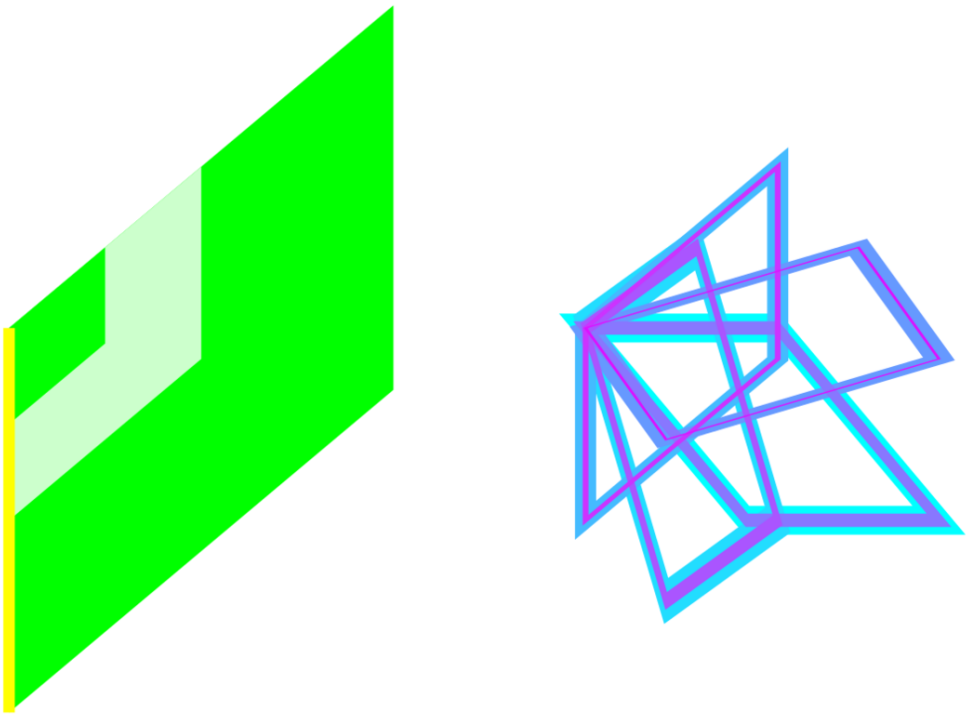


Abb. 42 Beispiel Scherung in beliebiger Richtung

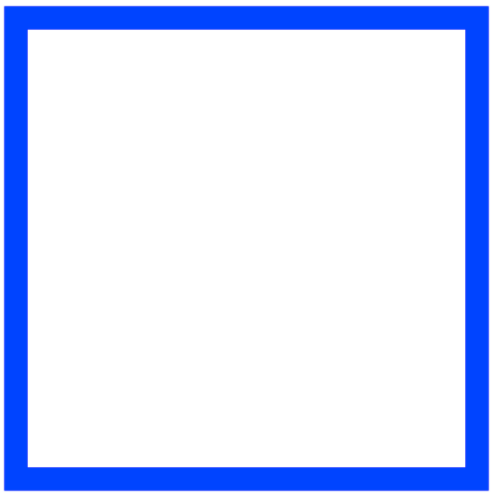


Abb. 43 Beispiel Scherung mit problematischen Winkeln

Eine Scherung in x-Richtung um den Winkel u in Grad wird mit skewX(u) angegeben.

Eine Scherung in y-Richtung um den Winkel v in Grad wird mit skewY(v) angegeben.

Beispiele sind also skewX(30) oder skewY(-45) für eine Scherung in x-Richtung um 30 Grad beziehungsweise -45 Grad in y-Richtung.

Eine Schermatrix um den Winkel u in x-Richtung sieht wie folgt aus:

1	tan(u)	0
0	1	0
0	0	1

Beispiel Scherung in x-Richtung[4]

Blau gerandete Rechtecke werden mittels des Attributes transform vom Typ skewX geschert, so dass das rot gerandete Rechteck komplett verdeckt wird. Ein grau gerandetes Rechteck zeigt das nicht transformierte Rechteck.

Gleiche Transformationstypen vertauschen immer miteinander, dies wird mit den dunkler gerandeten Rechtecken angedeutet.

Eine Schermatrix um den Winkel v in y-Richtung sieht wie folgt aus:

1	0	0
tan(v)	1	0
0	0	1

Beispiel Scherung in y-Richtung[5]

Blau gerandete Rechtecke werden mittels des Attributes transform vom Typ skewY geschert, so dass das rot gerandete Rechteck komplett verdeckt wird. Ein grau gerandetes Rechteck zeigt das nicht transformierte Rechteck.

Gleiche Transformationstypen vertauschen immer miteinander, dies wird mit den dunkler gerandeten Rechtecken angedeutet.

Beispiel Scherung[6]

Verschiedene Transformationen vertauschen in der Regel nicht miteinander, auch skewX und skewY nicht. Das Beispiel zeigt die beiden Möglichkeiten.

Beispiel Scherung in beliebiger Richtung[7]

Um eine Scherung in einer beliebigen Richtung zu erreichen, kann man zum Beispiel die Scherungsrichtung in die x-Richtung drehen, gefolgt von skewX, gefolgt von einer erneuten Drehung, diesmal in die umgekehrte Richtung.

Wenn der Scherungswinkel s ist und der Winkel von der x-Achse zur Scherungsrichtung a, dann entspricht dies der Transformation rotate(a) skewX(s) rotate(-a).

Die Graphik links zeigt Beispiele mit s gleich 40 Grad. Die gefüllten Beispiele links entsprechen einer Drehung von 90 Grad. Dabei ist zu beachten, dass dies für 90 Grad skewY(-90) entspricht. Die gerandeten Beispiele rechts zeigen verschiedene Scherungsrichtungen mit einer Differenz von 45 Grad.

4 http://de.wikibooks.org/wiki/media%3ASVGtransform_skewX01.svg
5 http://de.wikibooks.org/wiki/media%3ASVGtransform_skewY01.svg
6 http://de.wikibooks.org/wiki/media%3ASVGtransform_skew01.svg
7 http://de.wikibooks.org/wiki/media%3ASVGtransform_skew02.svg

Offenbar ergibt sich ein numerisches Problem bei Scherungswinkeln wie 90 Grad oder -90 Grad, allgemein (90 + n*180)Grad mit einer ganzen Zahl n. In solchen Fällen ergibt sich eine unendliche Ausdehnung in der Scherungsrichtung. Die Spezifikationen machen dazu allerdings keine besonderen Angaben, das Darstellungsprogramm muss mit dem Problem selbst fertig werden.

Beispiel Scherung mit problematischen Winkeln[8]

Zwei Rechtecke werden mittels animateTransform vom Typ skewX beziehungsweise skewY geschert, wobei auch problematische Winkel auftreten.

'matrix', Matrix

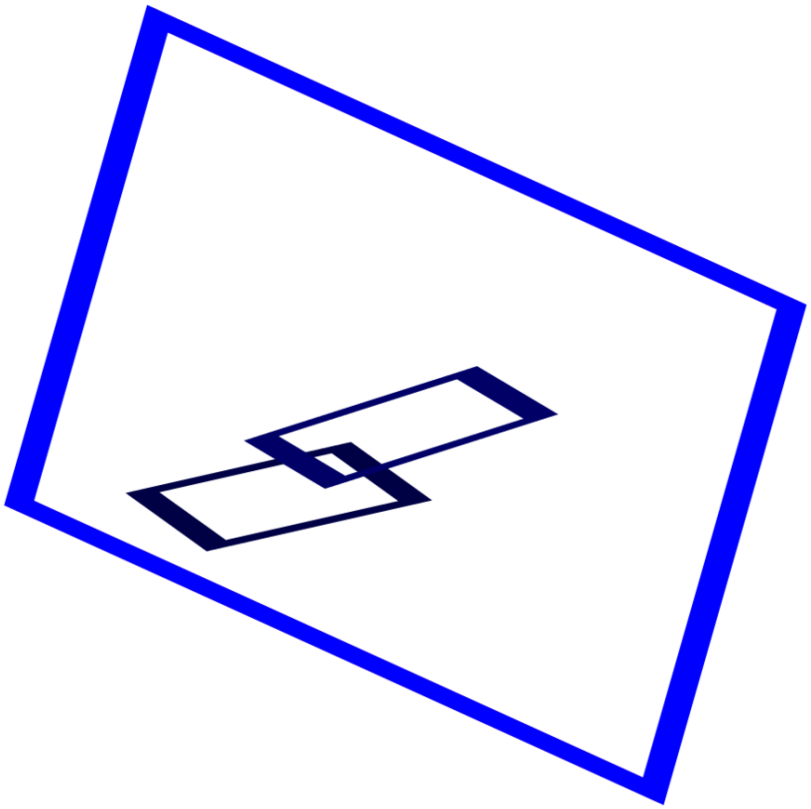


Abb. 44 Beispiel Matrix

8 http://de.wikibooks.org/wiki/media%3ASVGtransform_skew03.svg

Eine allgemeine (affine) Transformation, auch oder gerade wenn sie nicht unter die vorherigen Typen fällt, kann direkt mit matrix(a,b,c,d,e,f) angegeben werden.

Dies entspricht der Matrix:

a	c	e
b	d	f
0	0	1

Der Typ 'matrix' ist nicht animierbar. Gründe dafür sind in den Spezifikationen nicht angegeben. Sind alle Zahlen von 'matrix' 0, so unterbindet dies die Anzeige des transformierten Elementes in SVG tiny 1.2.

Beispiel Matrix[9]

Ein großes blau gerandetes Rechteck wird mittels des Attributes transform vom Typ matrix verschoben, so dass ein rotes gerandetes Rechteck komplett verdeckt wird. Ein grau gerandetes Rechteck zeigt die unverschobene Position.

Transformationen vom Typ matrix vertauschen im Allgemeinen nicht miteinander. Die kleinen Rechtecke zeigen Beispiele dazu.

5.1.2 Eingeschränkte Transformationen

Squiggle (Batik)	-
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit SVG tiny 1.2 wurde eine Möglichkeit eingeführt, Transformationen einzuschränken, so dass die Koordinaten die gleichen sind wie für das Hauptelement ***svg***.

Es gibt verschiedene Transformationen, die Einfluss auf die Darstellung haben können, zum einen sind dies vom Betrachter initiierte Skalierungen, Translationen oder Drehungen. Zum anderen gibt es die implizit durch *viewBox* bewirkte Transformationen und schließlich das Produkt aller Transformationen bedingt durch die Attribute *transform*, welche in Elternelementen notiert sind oder direkt in einem graphischen Element.

Seien die vom Betrachter initiierten Transformationen BS (Skalierung), BT (Translation) und BD (Drehung), insgesamt: B = BT * BS * BD. VS sei die durch *viewBox* beim Element ***svg*** bedingte Transformation. Das Produkt aller Transformationsattribute eines Elementes mit dem Fragmentidentifizierer id bis hinauf zum ***svg*** sei TS(id).

Als aktuelle Transformationmatrix CTM gemäß SVG tiny 1.2 ergibt sich somit: CTM = B * VS * TS(id).

9 `http://de.wikibooks.org/wiki/media%3ASVGtransform_matrix01.svg`

SVG tiny 1.2 führt nun den neuen Transformationstyp 'ref' ein, der die Transformationen ausgenommen B und VS rückgängig macht oder es anders formuliert erlaubt, an beliebiger Stelle im Dokument Elemente anzuordnen, für die das anfängliche Koordinatensystem gilt.

Sofern die inverse Matrix von TS(id) existiert, kann das Ergebnis formal erzielt werden, indem die CTM mit dieser inversen Matrix (von rechts) multipliziert wird. Existiert die inverse Matrix nicht, so muss das Darstellungsprogramm eine andere Methode verwenden, um den gewünschten Effekt zu erreichen, zum Beispiel direkt die Koordinaten als solche im anfänglichen Koordinatensystem interpretieren (was praktisch ein anderes Vorgehen erfordern mag als bei der Darstellung ohne diesen Transformationstyp).

'ref(svg)'

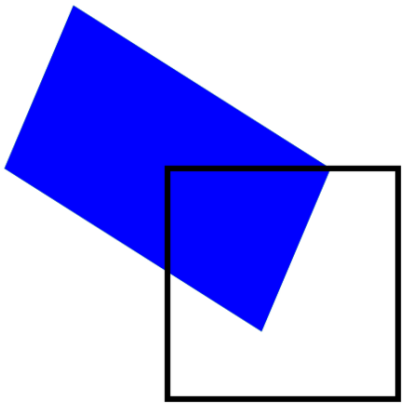


Abb. 45 Beispiel eingeschränkte Transformation

Mittels 'ref(svg)' wird angegeben, dass das Element nicht in lokalen Koordinaten angegeben ist, sondern in denen des anfänglichen Anzeigebereiches (ausschließlich der durch den Betrachter initiierten Transformationen).

Beispiel:

```
<?xml version="1.0" encoding="utf-8"?>
<svg version="1.2" baseProfile="tiny"
   xmlns="http://www.w3.org/2000/svg"
   xml:lang="de"
   viewBox="-100 -100 200 200">
<title>Beispiel eingeschränkte Transformation</title>

<g transform="skewX(30) rotate(45) translate(-20,-20)">
<rect width="40" height="40" fill="green" />
<rect width="40" height="40" fill="blue" transform="ref(svg)" />
</g>

</svg>
```

Während das grüne Viereck geschert, gedreht und verschoben wird und darauf auch die implizite Transformation durch *viewBox* wirkt, wird das blaue Quadrat nicht geschert, gedreht oder verschoben, sondern in den Koordinaten des Anzeigebereiches dargestellt, aber gegebenenfalls aufgrund des Malermodells das grüne teilweise verdeckend, weil es später im Quelltext steht.

Beispiel eingeschränkte Transformation[10]

Eine Gruppe mit einem grünen und einem blauen Quadrat wird verschoben, gedreht und geschert. Für das blaue Quadrat ist die Transformation allerdings eingeschränkt. Position und Größe entsprechen dem Quadrat mit schwarzem Strich.

10 http://de.wikibooks.org/wiki/media%3ASVGtransform_ref01.svg

'ref(svg, x, y)'

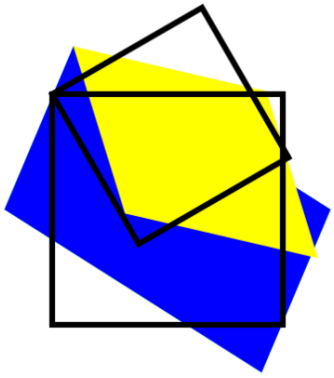


Abb. 46 Beispiel eingeschränkte Transformation

Mittels 'ref(svg, x, y)' wird angegeben, dass das Element nicht in lokalen Koordinaten angegeben ist, sondern in denen des anfänglichen Anzeigebereiches (ausschließlich der durch den Betrachter initiierten Transformationen).

Dabei sind x und y zwei Zahlen, die eine zusätzliche Verschiebung des Elementes relativ zum anfänglichen Anzeigebereich in Koordinaten des anfänglichen Anzeigebereiches bewirken.

Beispiel:

```
<?xml version="1.0" encoding="utf-8"?>
<svg version="1.2" baseProfile="tiny"
```

```
   xmlns="http://www.w3.org/2000/svg"
   xml:lang="de"
   viewBox="-100 -100 200 200">
<title>Beispiel eingeschränkte Transformation</title>

<g transform="skewX(30) rotate(45) translate(-20,-20)">
<rect width="40" height="40" fill="green" />
<rect width="40" height="40" fill="blue" transform="ref(svg, -20,-20)" />
</g>

</svg>
```

Während das grüne Viereck geschert, gedreht und verschoben wird und darauf auch die implizite Transformation durch *viewBox* wirkt, wird das blaue Quadrat nicht geschert, gedreht oder verschoben, sondern in den Koordinaten des Anzeigebereiches dargestellt, aber gegebenenfalls aufgrund des Malermodells das grüne teilweise verdeckend, weil es später im Quelltext steht. Das blaue Quadrat wird zuätzlich im Anzeigebereich verschoben, so dass die linke obere Ecke um -20 horizontal und -20 vertikal verschoben ist (sofern der Betrachter keine zusätzlichen Transformationen veranlaßt hat).

Beispiel eingeschränkte Transformation[11]

Eine Gruppe mit einem grünen, blauen und gelben Quadrat wird verschoben, gedreht und geschert. Für die Gruppe mit dem blauen und dem gelben Quadrat ist die Transformation allerdings eingeschränkt. Das gelbe wird wiederum auch noch direkt gedreht, diese Transformation ist nicht direkt betroffen, weil die Eingeschränkung für die umgebende Gruppe angegeben ist. Position und Größe entsprechen dem Quadrat mit schwarzem Strich.

11 http://de.wikibooks.org/wiki/media%3ASVGtransform_ref02.svg

6 Grundformen

6.1 Einführung Grundformen

Das Konzept von SVG[1] besteht darin, für die Repräsentation von Inhalt zum einen Grundformen anzubieten, die häufig verwendet werden. Es kann auch Text graphisch repräsentiert werden, der aber weiterhin als Text lesbar im Quelltext des Dokumentes steht. Für allgemeine graphische Formen steht zum anderen ein allgemeines Element zur Verfügung, um beliebige Formen anzunähern.

Zumeist wird es sinnvoll sein, den graphischen Formen auch noch weitere Eigenschaften zuweisen zu wollen, etwa Farbe oder Eigenschaften des Randes. Dazu werden bevorzugt spezielle Attribute notiert, die nicht spezifisch für bestimmte Elemente sind, die als sogenannte Präsentationsattribute bei nahezu allen Elementen notiert werden können. Diese werden in einem anderen Abschnitt behandelt und hier nur verwendet, sofern für die Darstellung hilfreich oder notwendig.

Zunächst werden im folgenden Kapitel graphische Grundformen behandelt. Die Erstellung allgemeiner Pfade ist im Kapitel Pfade[2] erklärt. Die Verwendung von Text als eine besondere Form von Pfaden, deren genaues Aussehen von der verwendeten Schriftart abhängt, ist im Kapitel Text als Graphik[3] erläutert.

6.2 Element *line*, Linie

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	2

1 `http://de.wikibooks.org/wiki/SVG`
2 Kapitel 7 auf Seite 143
3 Kapitel 8 auf Seite 193

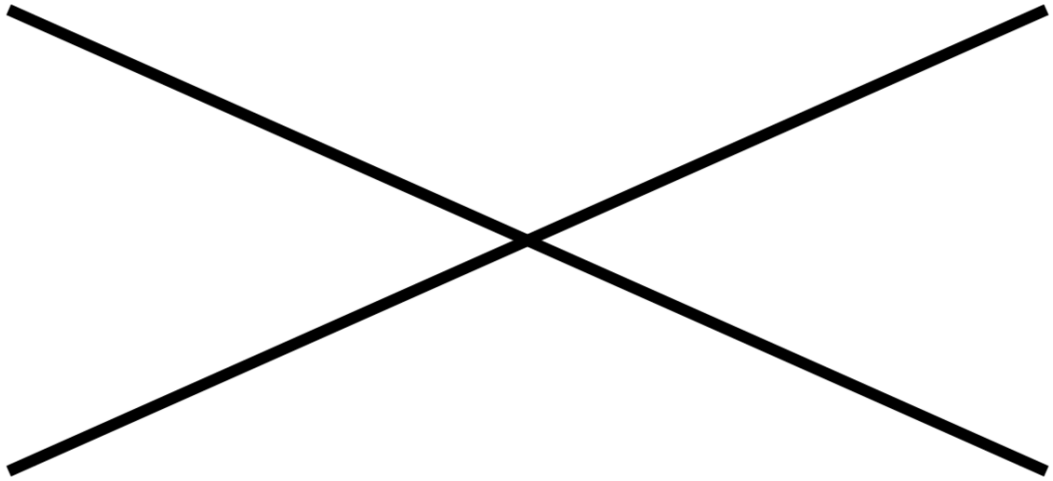


Abb. 47 Linien mit SVG

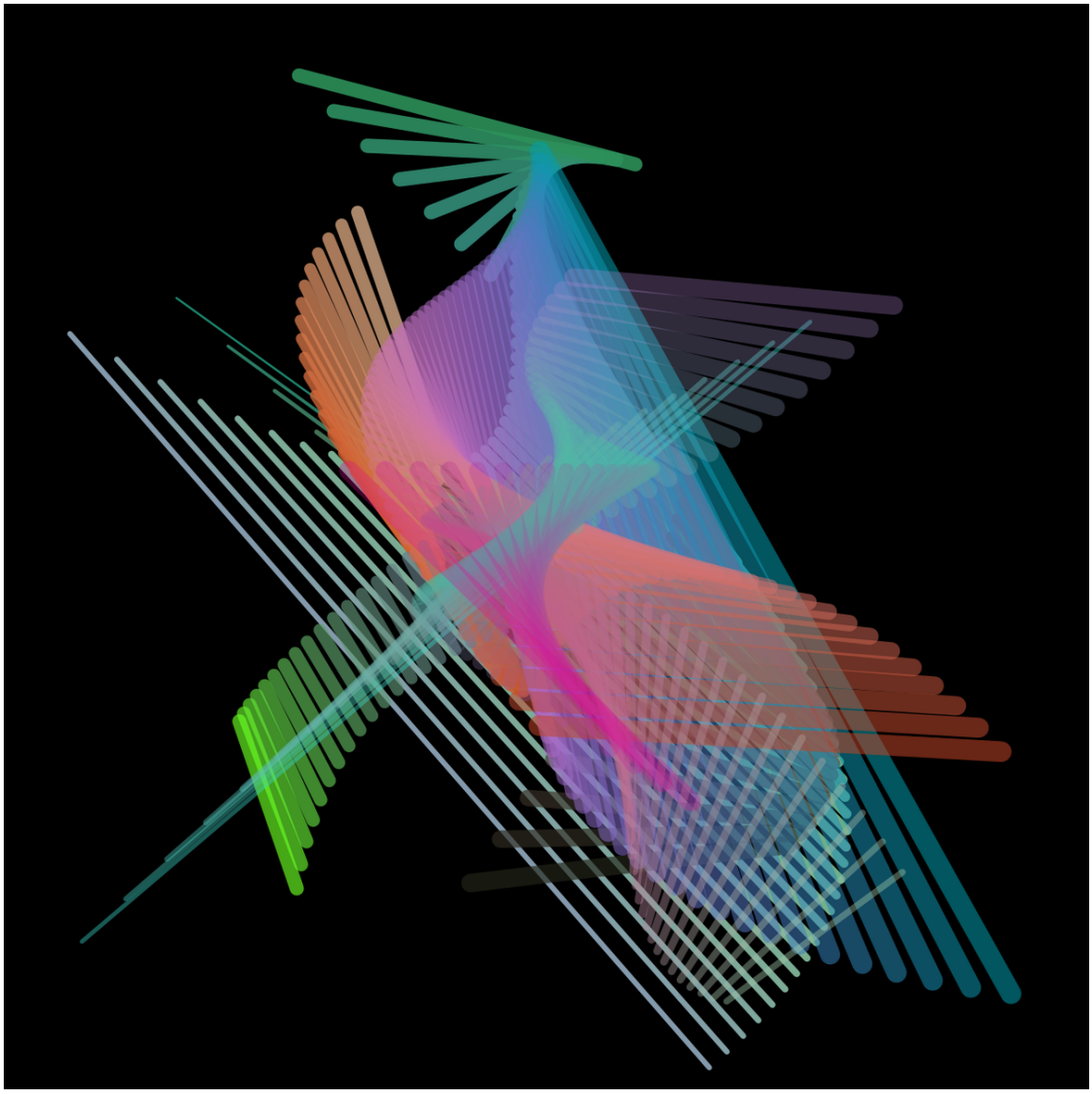


Abb. 48 Linienscharen mit einer kubischen Bézierkurve als Scharfunktion

Linien - genauer gesagt Strecken - werden in SVG durch das Element ***line*** beschrieben. Eine Linie hat keine Füllung, allenfalls nur einen Strich.

6.2.1 Attribute *x1*, *y1*, *x2*, *y2*

Eine Linie wird hierbei durch den Startpunkt mit den Attributen *x1* und *y1* und den Endpunkt mit den Attributen *x2* und *y2* spezifiziert. *x1* und *x2* sind also die x-Koordinaten, *y1* und *y2* die y-Koordinaten. Folglich ist der Wert eines jeden Attributes eine Koordinate.

Nicht angegebene Attribute *x1*, *y1*, *x2*, *y2* werden als 0 angenommen. Die Attribute sind animierbar.

6.2.2 Beispiele Linie

```
<?xml version="1.0" ?>
<svg width="100" height="50" xmlns="http://www.w3.org/2000/svg">
  <line x1="5" y1="5" x2="95" y2="45" stroke="black" />
  <line x1="5" y1="45" x2="95" y2="5" stroke="black" />
</svg>
```

Das Beispiel zeigt zwei sich überkreuzende Linien. Da Linien selbst keine flächige Ausdehnung haben, sind sie nur als Strich sichtbar und das auch nur, wenn wie hier das zugehörige Präsentationsattribut *stroke* explizit angegeben wird, hier mit dem Farbwert 'black', also schwarz.

Nun könnte vermutet werden, dass Linien alleine nicht besonders viel hergeben, schön dekoriert können aber bereits alleine mit ihnen reizvolle abstrakte Bilder erstellt werden: Linienscharen mit einer kubischen Bézierkurve als Scharfunktion[4].

LineElement W3C Referenz[5]

6.3 Element *rect*, Rechteck

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	2

Abb. 49 Einfaches Rechteck

4 http://de.wikibooks.org/wiki/media%3ASVGline01.svg
5 http://www.w3.org/TR/SVG/shapes.html#LineElement

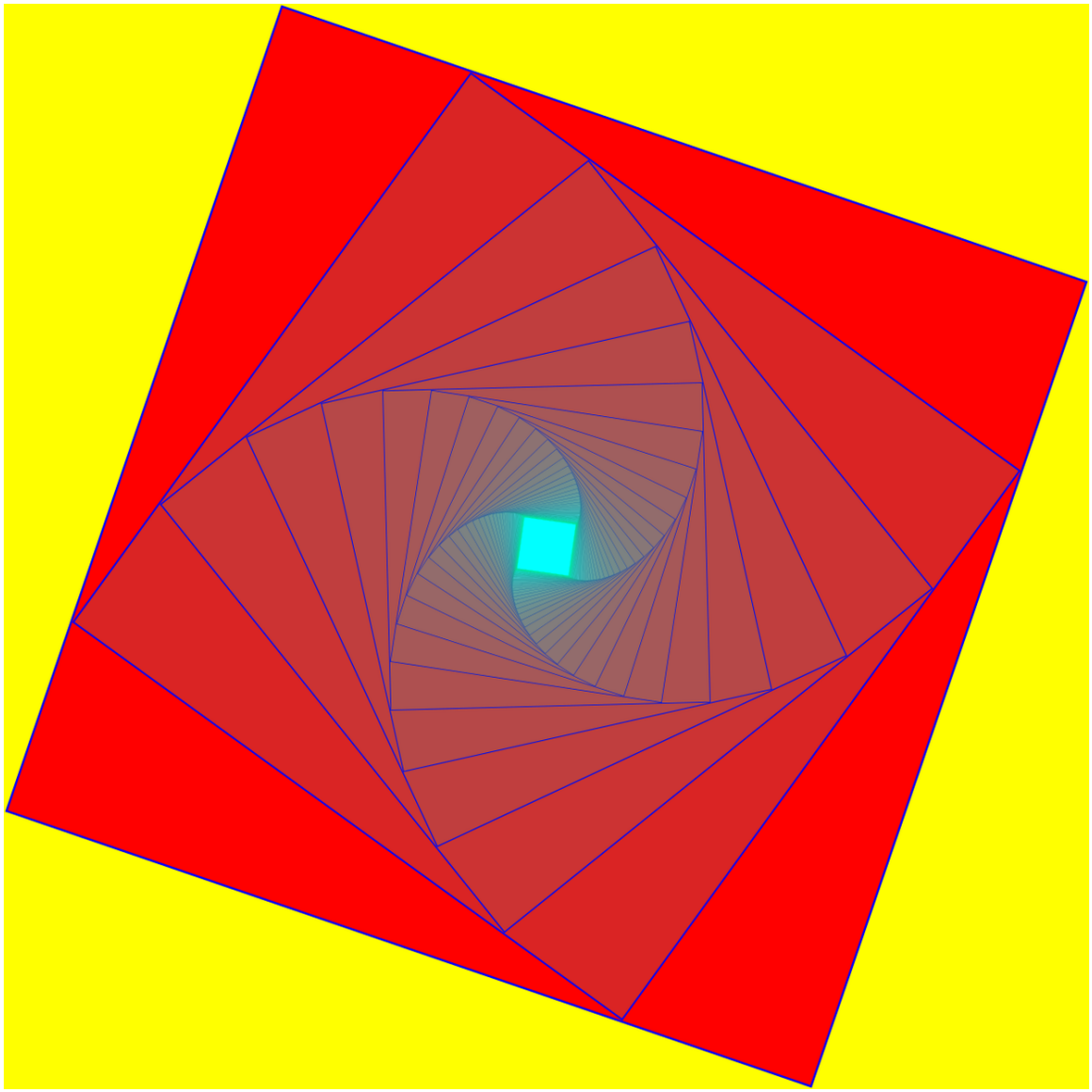


Abb. 50 Eine Schar von Rechtecken mit Farbverlauf, skaliert und rotiert

Ein einfaches Rechteck wird mit dem Element ***rect*** angegeben. Die Position der linken oberen Ecke in Form einer x- und einer y-Koordinate wird mit den Attributen *x* und *y* festgelegt. Breite und Höhe werden mit den Attributen *width* und *height* angegeben. Diese wenigen Angaben reichen bereits aus, um ein Rechteck darzustellen. Für eine willkürliche Ausrichtung ist dann allerdings das allgemein verfügbare Attribut *transform* zu verwenden, damit das Rechteck dann in die gewünschte Ausrichtung gedreht wird.

Beispiel: Eine Schar von Rechtecken mit Farbverlauf, skaliert und rotiert[6].

6 `http://de.wikibooks.org/wiki/media%3ASVGrect02.svg`

6.3.1 Attribute *x* und *y*

Die Attribute *x* und *y* geben die linke obere Ecke des Rechtecks (ohne Berücksichtigung möglicherweiser abgerundeter Ecken) im lokalen Koordinatensystem an. Der Wert ist jeweils eine Koordinate. *x* für die Position in x-Richtung, *y* für die Position in y-Richtung. Nicht angegebene *x* oder *y* entsprechen 0.

Die Attribute *x*, *y* sind animierbar.

6.3.2 Attribute *width* und *height*

width gibt die Breite und *height* die Höhe des Rechtecks im lokalen Koordinatensystem an. Der Wert ist jeweils eine Länge.

Nicht angegebene oder negative *width* oder *height* sind in SVG 1.1 ein Fehler, in SVG tiny 1.2 unterbindet dies die Darstellung, ebenso wie der Wert 0 in beiden Profilen.

Die Attribute *width* und *height* sind animierbar.

```
<?xml version="1.0"?>
<svg width="100" height="50" xmlns="http://www.w3.org/2000/svg">
  <rect x="20" y="5" width="60" height="10" />
</svg>
```

Ohne weitere Angaben ist solch eine Grundform schwarz gefüllt. Dies kann durch die Angabe des Präsentationsattributes *fill* geändert werden, etwas *fill*="red".

6.3.3 Attribute *rx* und *ry*

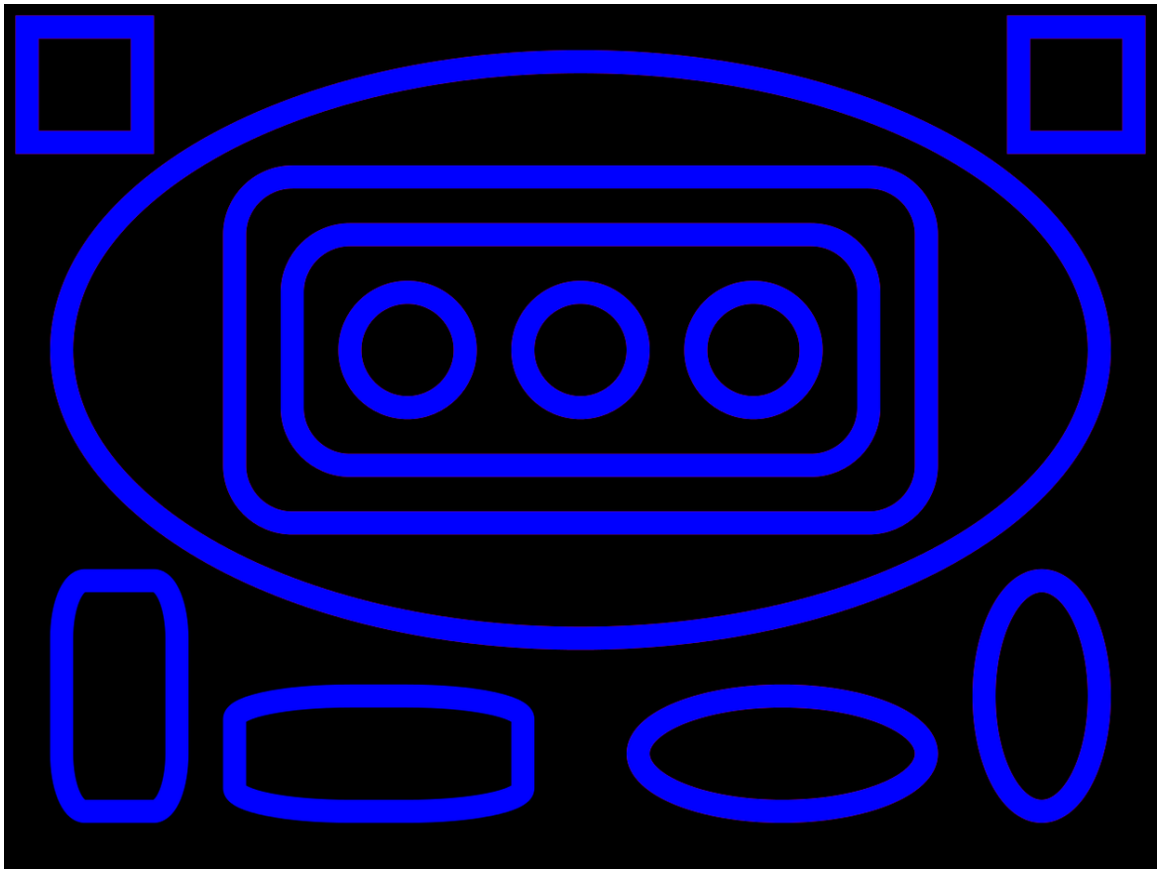


Abb. 51 Beispiel-SVG zur Verwendung des Elementes ***rect***

Ein Rechteck kann auch abgerundete Ecken haben. Abgerundete Ecken werden durch die Angabe der Attribute *rx* und *ry* erreicht. Der Wert ist jeweils eine Länge. Dabei legt *rx* den Radius der Rundung in x-Richtung fest und *ry* in Richtung der y-Achse (genauer handelt es sich da um die Angaben zur Hälfte der Hauptachsen der Ellipsenbögen).

```
<rect x="10" y="10" width="20" height="50" rx="2" ry="3" />
<rect x="50" y="10" width="20" height="50" rx="2" />
<rect x="90" y="10" width="20" height="50" ry="2" />
```

Wird nur eines der Attribute *rx* und *ry* angegeben, wird das andere mit dem gleichen Wert angenommen. Überschreitet der Wert für den Radius den jeweiligen Wert für die halbe Höhe beziehungsweise Breite, so wird der Wert auf die halbe Höhe, beziehungsweise Breite reduziert.

In welcher Reihenfolge Korrektur und Übernahme des Wertes von *rx* und *ry* erfolgen, wird in folgendem Beispiel getestet. Wird etwas Rotes sichtbar, liegt beim verwendeten Darstellungsprogramm jedenfalls ein Implementierungsfehler vor: Beispiel-SVG zur Verwendung des Elementes ***rect***[7]. Genauere Angaben sind den Beschreibungen im Dokument zu entnehmen.

Die Attribute *rx*, *ry* sind animierbar.

7 http://de.wikibooks.org/wiki/media%3ASVGrect01.svg

W3C: ***rect***[8]

6.4 Element *circle*, Kreis

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	2

8 `http://www.w3.org/TR/SVG/shapes.html#RectElement`

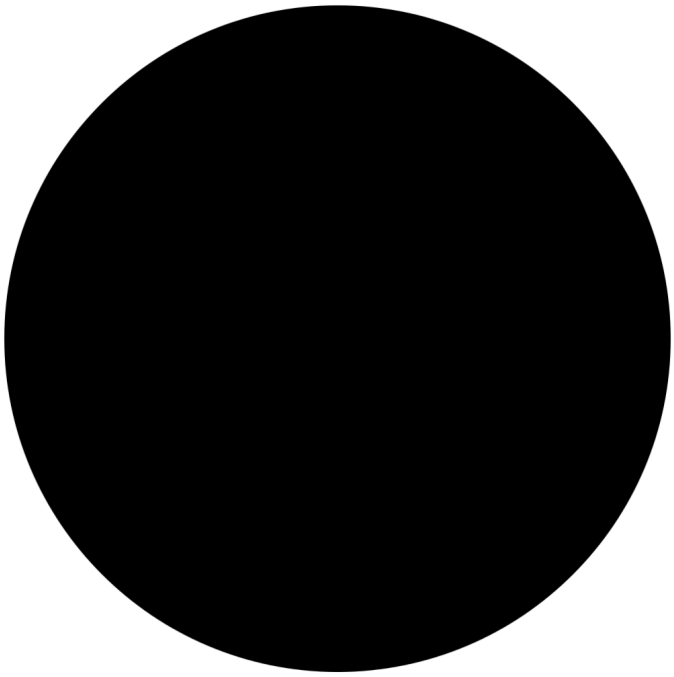


Abb. 52 Einfacher Kreis

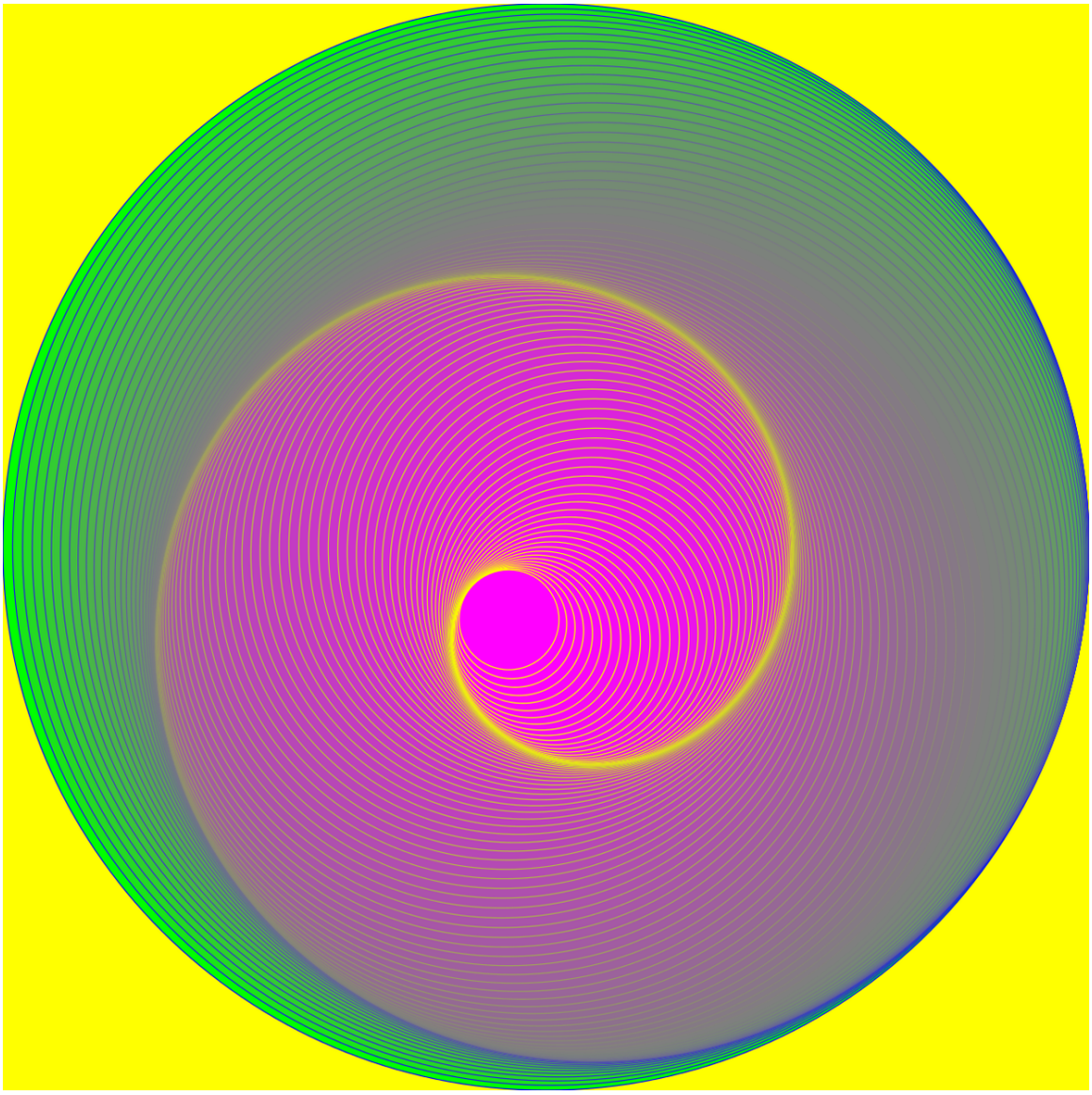


Abb. 53 Schar von Kreisen mit Farbverlauf

Ein Kreis wird in SVG durch das Element ***circle*** repräsentiert. Auch wenn die Kreise lediglich eine Spezialform der Ellipsen sind, besitzt SVG hierfür ein eigenes Element.

6.4.1 Attribute *cx* und *cy*

Der Kreismittelpunkt wird mit den Attributen *cx* und *cy* festgelegt. *cx* steht für die horizontale Koordinate im lokalen Koordinatensystem und *cy* für die vertikale Koordinate. Der Wert ist jeweils eine Koordinate.

Nicht angegebene *cx* oder *cy* werden als 0 angenommen. Die Attribute sind animierbar.

6.4.2 Attribut *r*

Der Radius des Kreises wird mit dem Attribut *r* angegeben. Der Wert ist eine Länge.

Wird in SVG 1.1 der Radius nicht angegeben, so ist dies ein Fehler. In SVG tiny 1.2 unterbindet dies die Anzeige des Kreises.

Das Attribut ist animierbar.

6.4.3 Beispiele Kreis

```
<?xml version="1.0"?>
<svg width="100" height="150" xmlns="http://www.w3.org/2000/svg">
  <circle cx="50" cy="80" r="40" />
</svg>
```

Schar von Kreisen mit Farbverlauf[9].

W3C: ***circle***[10]

6.5 Element *ellipse*, Ellipse

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	2

9 http://de.wikibooks.org/wiki/media%3ASVGcircle01.svg
10 http://www.w3.org/TR/SVG/shapes.html#CircleElement

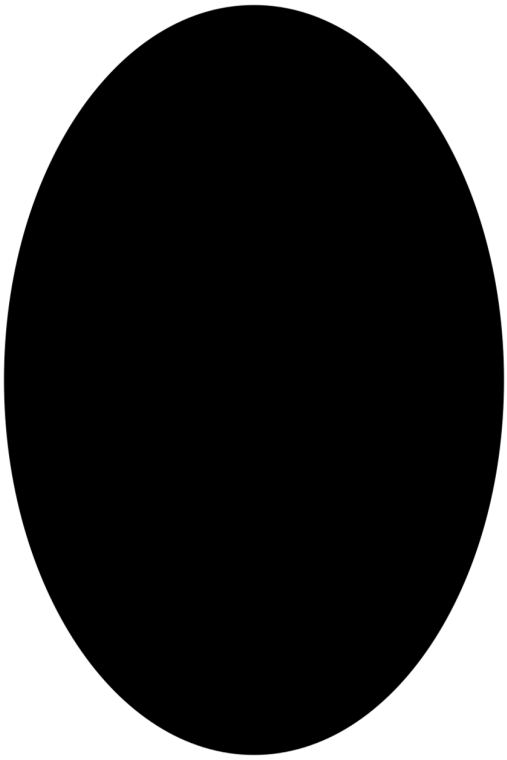


Abb. 54 Einfache Ellipse

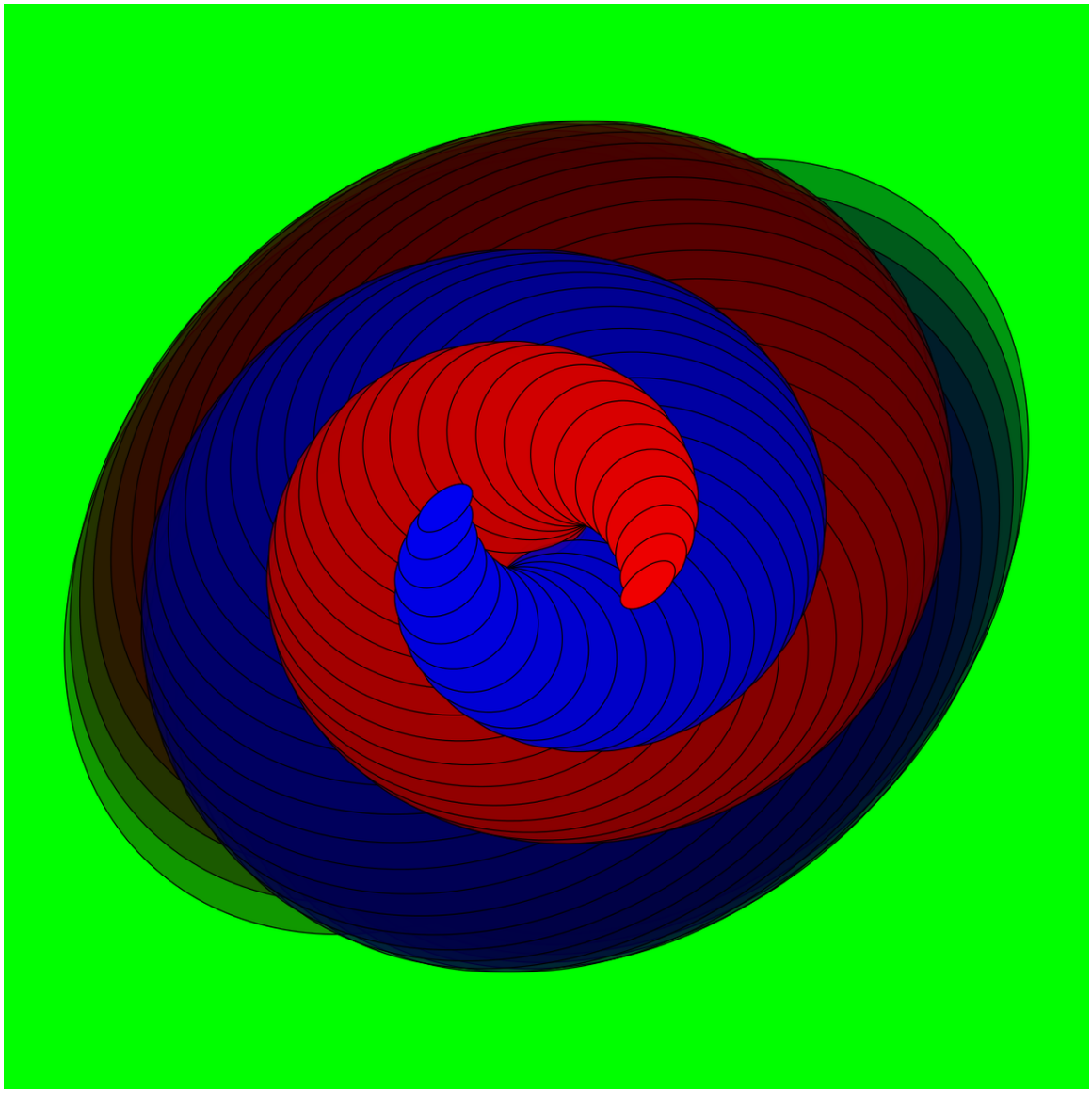


Abb. 55 Ellipsenscharen

Eine Ellipse wird mit dem Element ***ellipse*** beschrieben.

6.5.1 Attribute *cx* und *cy*

Wie beim Kreis legen die Attribute *cx* und *cy* den Mittelpunkt fest. *cx* steht für die horizontale Koordinate im lokalen Koordinatensystem und *cy* für die vertikale Koordinate. Der Wert ist jeweils eine Koordinate.

Nicht angegebene *cx* oder *cy* werden als 0 angenommen. Die Attribute sind animierbar.

6.5.2 Attribute *rx* und *ry*

Eine Ellipse enthält zwei Attribute zur Darstellung der halben Hauptachsen (auch Radien genannt), wobei die Hauptachsen immer in x- und y-Richtung des lokalen Koordinatensystems ausgerichtet sind. Eine andere Ausrichtung ist mit dem allgemeinen Attribut *transform* möglich. Zum einem gibt es für die Horizontale das Attribut *rx* und zum anderen für die Vertikale *ry*. Der Wert ist jeweils eine Länge.

Nicht angegebene oder negative *rx* und *ry* sind in SVG 1.1 ein Fehler. In SVG tiny 1.2 unterbinden sie die Darstellung des Elementes, ebenso wie in beiden Profilen der Wert 0. Die Attribute sind animierbar.

6.5.3 Beispiele Ellipse

```
<?xml version="1.0"?>
<svg width="60" height="100" xmlns="http://www.w3.org/2000/svg">
  <ellipse cx="30" cy="50" rx="20" ry="30" />
</svg>
```

Ellipsenscharen[11]

W3C ***ellipse***[12]

6.6 Element *polyline*, Linienzug

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	2

11 http://de.wikibooks.org/wiki/media%3ASVGellipse01.svg
12 http://www.w3.org/TR/SVG/shapes.html#EllipseElement

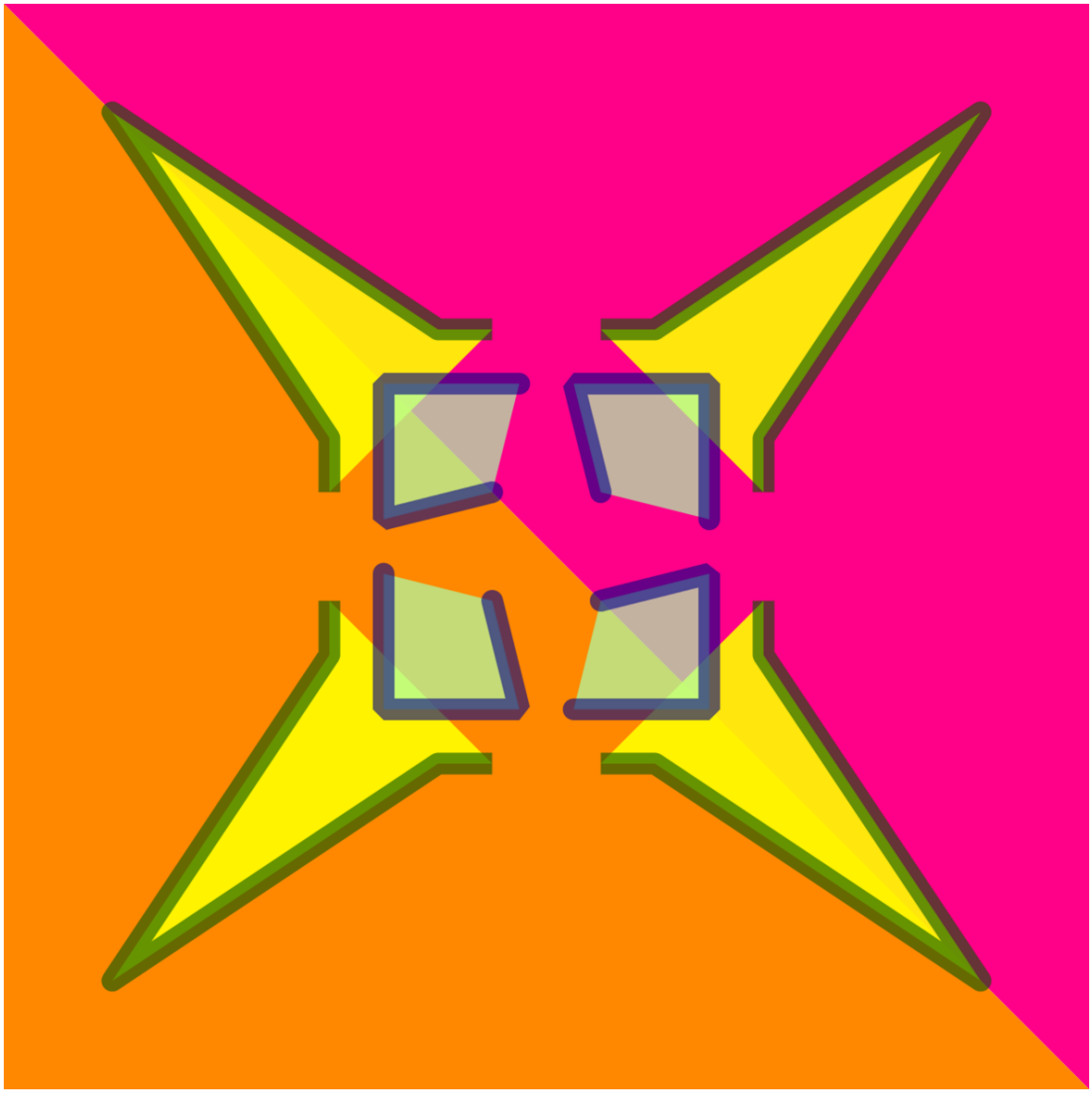


Abb. 56 Beispiel-SVG zur Verwendung des Elementes ***polyline***

Um einen Linienzug anzugeben, wird das Element ***polyline*** verwendet. Ein Linienzug besteht aus mehreren geraden Kurvenstücken, die aneinandergehängt werden. Die Form ist in der Regel offen. Für eine geschlossene Form wäre stattdessen das Element ***polygon*** zu verwenden.

6.6.1 Attribut *points*

Dies Element hat ein spezifisches Attribut *points*. Der Wert dieses Attributes ist eine Liste von Koordinaten, die mit Leerzeichen separiert sind oder jeweils mit einem Komma mit optionalem Leerzeichen davor oder dahinter. Die Koordinaten sind reelle Zahlen, wissenschaftliche Notation ist zulässig. Ferner sind immer Koordinatenpaare anzugeben, erst ein Wert für die x-Richtung, dann einer für die y-Richtung. In SVG 1.1 ist die Angabe einer ungeraden Anzahl von Koordinaten ein Fehler. In SVG tiny 1.2 resultiert daraus, dass das Element nicht dargestellt wird, ebenso, wenn

das Attribut gar keine Koordinaten enthält. Ansonsten ist die Darstellung dieselbe wie bei einem korrespondierenden Pfad.

Das Attribut *points* ist animierbar, jedoch nicht additiv und die Struktur des Attributwertes lässt keine sinnvolle Animation mit dem *calcMode* paced zu. In SVG 1.1 ist es bedingt durch die Äquivalenz zu einem Pfad notwendig, für jeden Animationswert die gleiche Anzahl von Koordinatenpaaren anzugeben. In SVG tiny 1.2 kann aufgrund der gleichen Äquivalenz bei einer diskreten Animation auch die Anzahl der Koordinatenpaare variiert werden.

6.6.2 Beispiel Linienzug

```
<polyline points="200,50 200,100 400,400 100,200 50,200" />
```

Beispiel-SVG zur Verwendung des Elementes ***polyline***[13]

13 http://de.wikibooks.org/wiki/media%3ASVGpolyline01.svg

6.7 Element *polygon*, Polygon

Abb. 57 Beispiel-SVG zur Verwendung des Elementes ***polygon***

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	2

Um ein Polygon oder Vieleck, also einen geschlossenen Linienzug anzugeben, wird das Element ***polygon*** verwendet. Ein Polygon besteht aus mehreren geraden Kurvenstücken, die aneinandergehängt werden. Der letzte Punkt wird mit dem ersten verbunden.

6.7.1 Attribut *points*

Dies Element hat ein spezifisches Attribut *points*. Der Wert dieses Attributes ist eine Liste von Koordinaten, die mit Leerzeichen separiert sind oder jeweils mit einem Komma mit optionalem Leerzeichen davor oder dahinter. Die Koordinaten sind reelle Zahlen, wissenschaftliche Notation ist zulässig. Ferner sind immer Koordinatenpaare anzugeben, erst ein Wert für die x-Richtung, dann einer für die y-Richtung. In SVG 1.1 ist die Angabe einer ungeraden Anzahl von Koordinaten ein Fehler. In SVG tiny 1.2 resultiert daraus, dass das Element nicht dargestellt wird, ebenso, wenn das Attribut gar keine Koordinaten enthält. Ansonsten ist die Darstellung dieselbe wie bei einem korrespondierenden Pfad.

Das Attribut *points* ist animierbar, jedoch nicht additiv und die Struktur des Attributwertes lässt keine sinnvolle Animation mit dem *calcMode* paced zu. In SVG 1.1 ist es bedingt durch die Äquivalenz zu einem Pfad notwendig, für jeden Animationswert die gleiche Anzahl von Koordinatenpaaren anzugeben. In SVG tiny 1.2 kann aufgrund der gleichen Äquivalenz bei einer diskreten Animation auch die Anzahl der Koordinatenpaare variiert werden.

6.7.2 Beispiel Polygon

Als Beispiel ein Pentagramm (mit einigen Präsentationsattributen):

```
<polygon fill="#005" fill-opacity="0.7" fill-rule="evenodd"
stroke="#ccf" stroke-width="32" stroke-opacity="0.7"
stroke-miterlimit="30" stroke-linejoin="miter"
points="500,800 323.6644,257.2949 785.317,592.7051
214.683,592.7051 676.3356,257.2949">
```

(die Präsentationsattribute *fill-opacity* und *stroke-opacity* sind in SVG tiny 1.1 nicht verfügbar, in SVG tiny 1.2 schon.)

Beispiel-SVG zur Verwendung des Elementes ***polygon***[14]

6.7.3 Exkurs Polygone malen

SVG bietet einmal abgesehen vom Element ***rect*** keine weiteren Elemente zum Malen spezieller, benannter Polygone. Der Autor muss sich also selbst darum kümmern, an die Koordinaten der Ecken seines Polygons zu gelangen.

Programme wie Inkscape bieten eine graphische Oberfläche, wo insbesondere symmetrische Polygone wie die regulären (konvexen) und sogenannte Sterne mit vorgefertigten Funktionen gemalt werden können. Bei Inkscape wie bei zahlreichen anderen Programmen ist das Resultat dann allerdings ein allgemeiner Pfad, kein ***polygon***-Element (was hinsichtlich der Qualität solcher Programme Fragen aufwirft, die hier aber nicht diskutiert werden müssen).

So oder so sind die Möglichkeiten solcher Programme, einem Autor zu helfen, spezielle Polygone zu malen, recht beschränkt, insofern ist es naheliegend, ein kleines eigenes Programm oder Skript zu schreiben, welches die Koordinaten berechnet und als Zeichenkette entsprechend dem Attributwert von *points* ausgibt. Dazu sollen im Folgenden ein paar einfache Beispiele betrachtet werden.

14 http://de.wikibooks.org/wiki/media%3ASVGpolygon01.svg

Reguläre (konvexe) Polygone

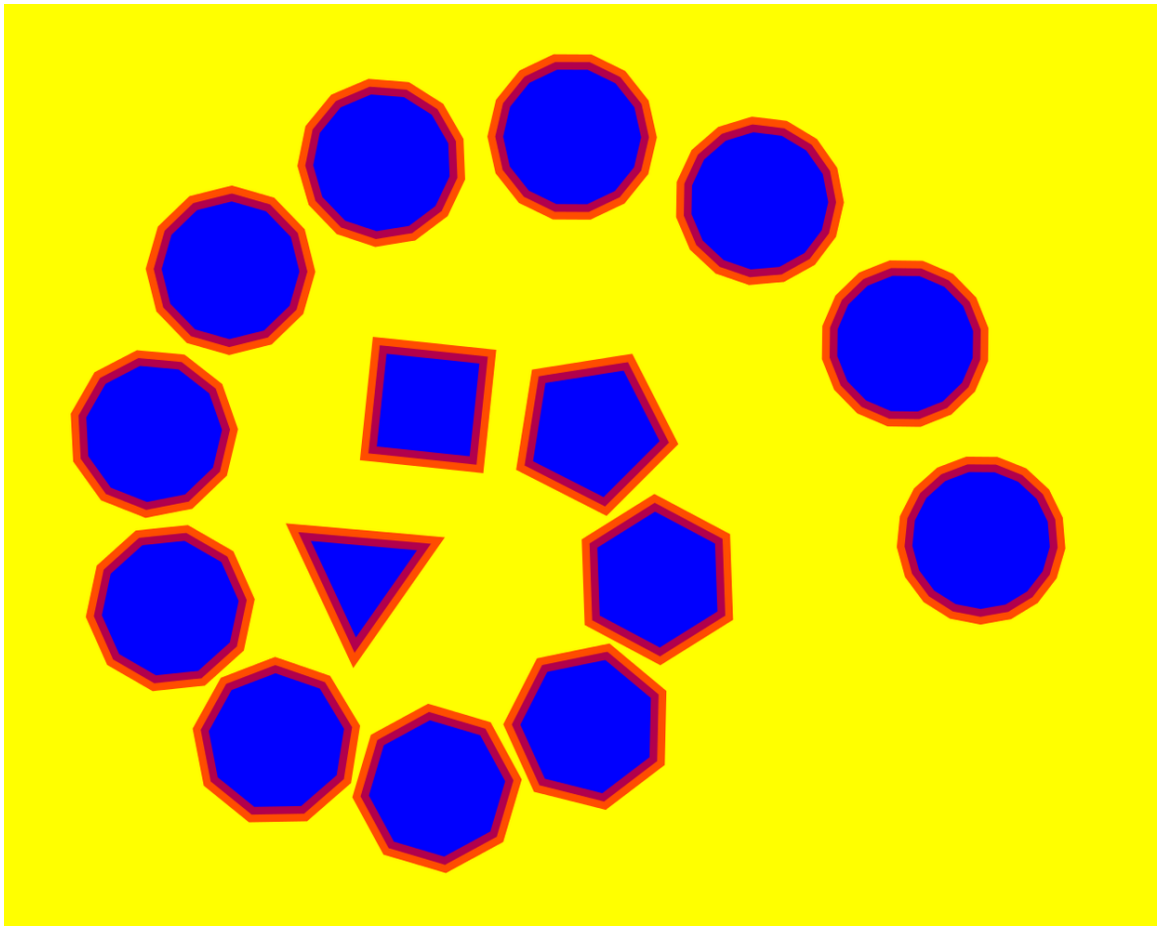


Abb. 58 Reguläre (konvexe) Polygone

Reguläre (konvexe) Polygone zeichnen sich dadurch aus, dass sie gleichseitig sind und zwischen allen benachbarten Seiten der gleiche Winkel auftaucht. Zusätzlich handelt es sich um einen Linienzug, der durch die Verbindung einer Liste von Koordinatenpaaren mit den Ecken erstellt werden kann.

Die Koordinaten (x,y) der Ecken eines solchen regulären (konvexen) Polygons lassen sich wie folgt bestimmen, wobei r als der Radius vom Mittelpunkt (cx,cy) des Polygons zu einer Ecke definiert sei und $n > 2$ die Anzahl der Ecken angibt. Die i-te Ecke, i von 0 bis n-1 mit einem beliebigen Winkel φ der nullten Ecke zur x-Achse, hat dann folgende Koordinaten:

$x(i) = cx + r \cos(2\Pi\ i/n + \varphi)$

$y(i) = cy + r \sin(2\Pi\ i/n + \varphi)$

Ganz allgemein ist eine Fläche konvex, wenn die Verbindungslinie zwischen zwei beliebigen Punkten der Fläche komplett in der Fläche liegt, jeweils einschließlich dem Rand.

Reguläre (konvexe) Polygone[15]

Allgemeine reguläre Polygone

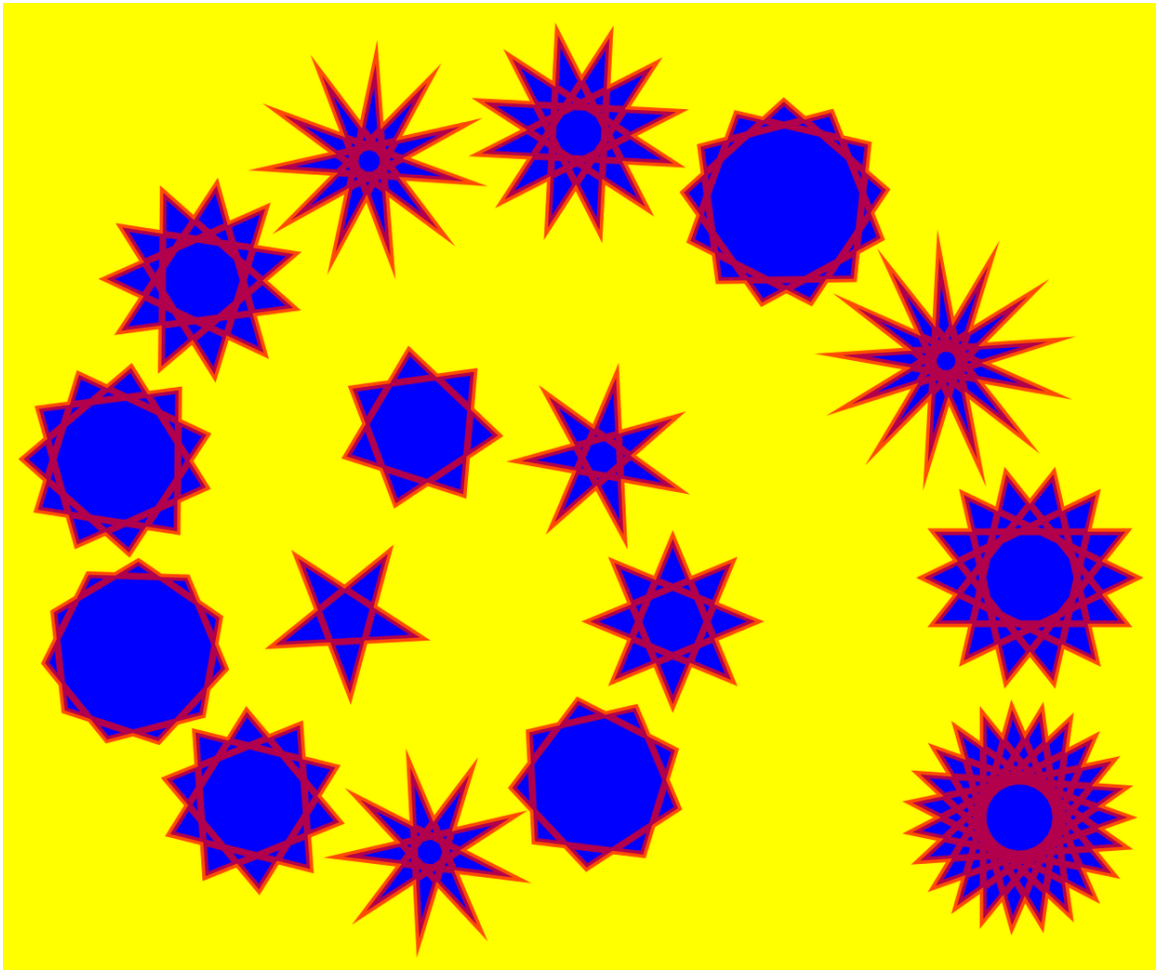


Abb. 59 Sternpolygone

Wird die Forderung zur Konvexität weggelassen, ergibt sich mit einem weiteren ganzzahligen Parameter $0 < m < n/2$ eine allgemeinere Formel für reguläre Polygone, die echte Sternpolygone einschließt, bei diesen überschneiden sich verschiedene Ränder. Sinnvollerweise werden zudem m und n teilfremd gewählt (abgesehen vom Fall m=1, welcher die konvexen Polygone beschreibt), sonst ist das Ergebnis erneut ein reguläres Polygon, bei dem jedoch die Ecken mehrfach notiert sind.

$x(i) = cx + r \cos(2\Pi\, i\, m/n + \varphi)$

$y(i) = cy + r \sin(2\Pi\, i\, m/n + \varphi)$

Sternpolygone[16]

15 http://de.wikibooks.org/wiki/media%3ASVGpolygon02.svg
16 http://de.wikibooks.org/wiki/media%3ASVGpolygon03.svg

Spiegelstern

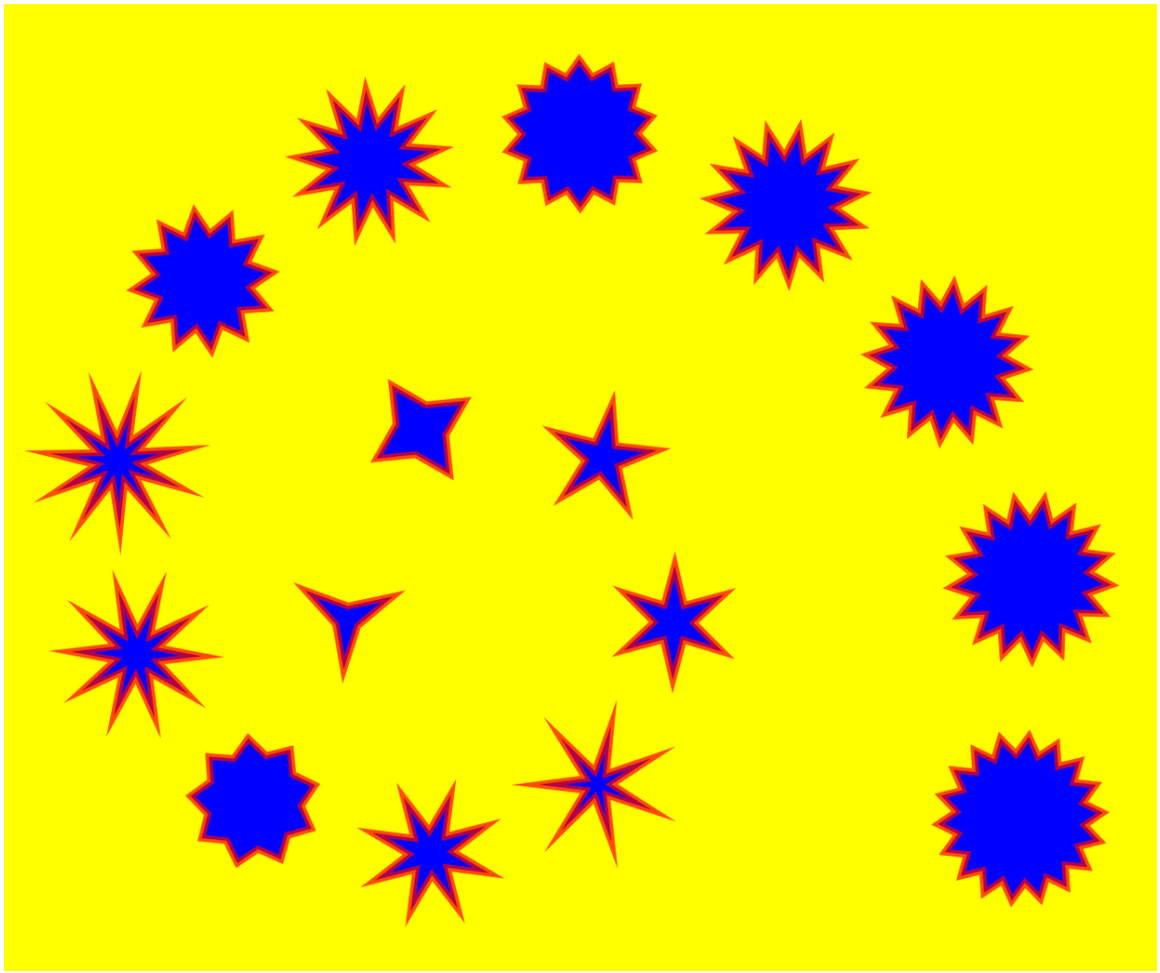


Abb. 60 Spiegelsterne

Was gemeinhin unter dem Begriff 'Stern' bei einer graphischen Form verstanden wird, ist nicht scharf definiert. Zumeist hat dies eine niedrigere Symmetrie, etwa indem zwei Radien oder Winkel im Wechsel vorkommen. Eine Variante ist zum Beispiel, was in diesem Zusammenhang einmal ein Spiegelstern getauft werden soll, weil er Spiegelachsen aufweist):

$x(i,1) = cx + r(1) \cos(2\Pi\ i\ m/n + \varphi)$

$y(i,1) = cy + r(1) \sin(2\Pi\ i\ m/n + \varphi)$

$x(i,2) = cx + r(2) \cos(2\Pi\ (i+0.5)\ m/n + \varphi)$

$y(i,2) = cy + r(2) \sin(2\Pi\ (i+0.5)\ m/n + \varphi)$

Dabei stehen 1 und 2 jeweils für die verschiedenen Radien. Oft wird nur m = 1 verwendet, die anderen sind aber auch erwähnenswert. Die Koordinatenpaare werden dann abwechselnd notiert, sinngemäß so:

points=" x(0,1), y(0,1) x(0,2), y(0,2) x(1,1), y(1,1), x(1,2), y(1,2) x(2,1), y(2,1), x(2,2), y(2,2) x(3,1), y(3,1), x(3,2), y(3,2) x(4,1), y(4,1), x(4,2), y(4,2) ..."

Spiegelsterne[17]

Drehstern

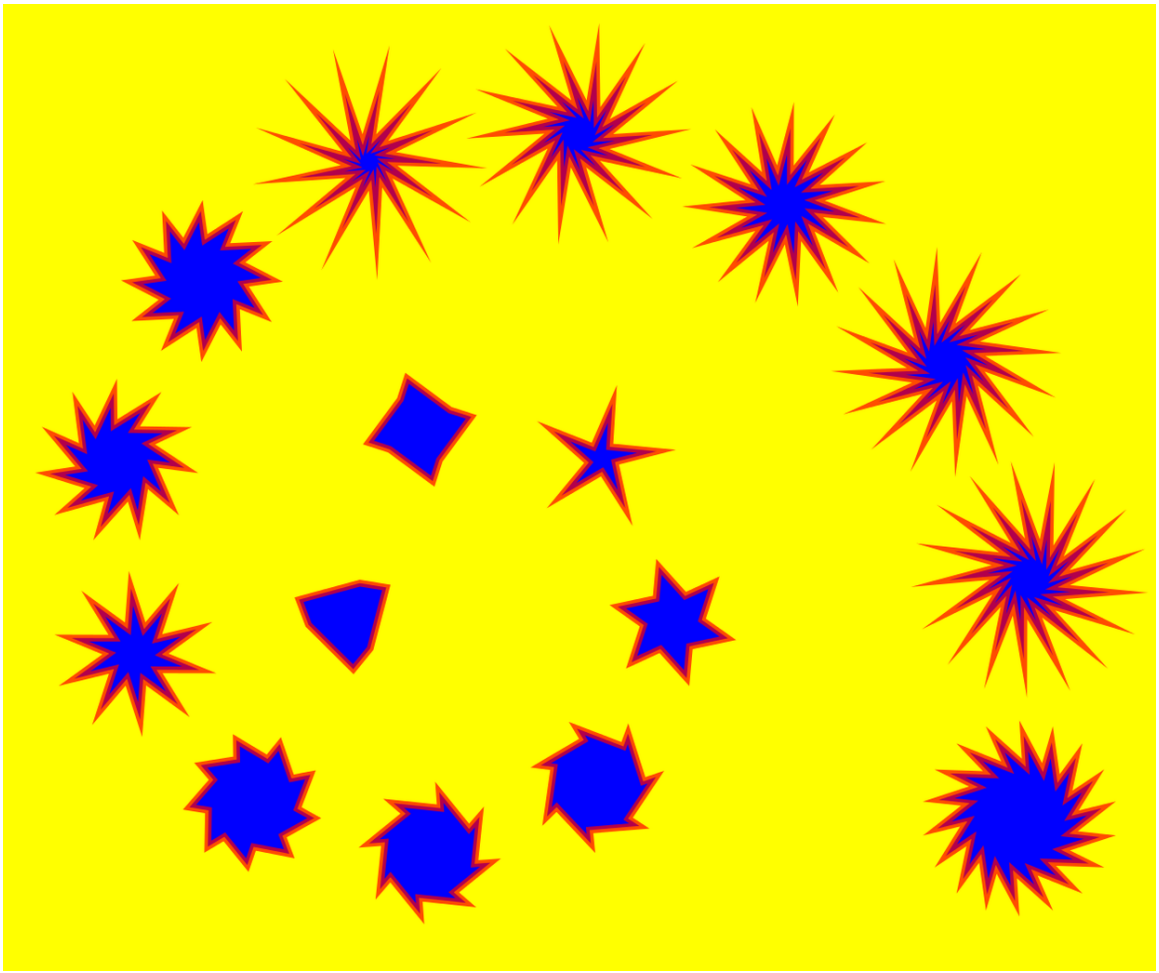


Abb. 61 Drehsterne

Die Spiegelachsen dieser Sterne können vermieden werden, wenn zwei verschiedene Winkel φ angegeben werden. Es verbleibt nur noch eine diskrete Drehsymmetrie, dieser Typ sei hier kurzerhand Drehstern getauft:

$x(i,1) = cx + r(1) \cos(2\Pi\ i\ m/n + \varphi(1))$

17 http://de.wikibooks.org/wiki/media%3ASVGpolygon04.svg

$y(i,1) = cy + r(1) \sin(2\Pi\ i\ m/n + \varphi(1))$

$x(i,2) = cx + r(2) \cos(2\Pi\ i\ m/n + \varphi(2))$

$y(i,2) = cy + r(2) \sin(2\Pi\ i\ m/n + \varphi(2))$

Drehsterne[18]

Allgemeiner Stern

Oftmals sind auch Sterne zu sehen, die weder eine strenge Drehsymmetrie, noch Spiegelachsen aufweisen. Damit die Formel auch diese beschreiben kann, wird zum Radius einer jeden Ecke eine Zufallszahl addiert, deren Betrag sinnvollerweise kleiner ist als der durchschnittliche Radius. Ähnlich kann auch zum Winkel φ einer jeden Ecke ein kleiner Zufallswinkel hinzuaddiert werden, dessen Betrag sinnvollerweise kleiner ist als die durchschnittliche Differenz der Winkel zwischen zwei benachbarten Ecken.

$x(i,1) = cx + r(i,1) \cos(2\Pi\ i\ m/n + \varphi(i,1))$

$y(i,1) = cy + r(i,1) \sin(2\Pi\ i\ m/n + \varphi(i,1))$

$x(i,2) = cx + r(i,2) \cos(2\Pi\ i\ m/n + \varphi(i,2))$

$y(i,2) = cy + r(i,2) \sin(2\Pi\ i\ m/n + \varphi(i,2))$

Leicht lassen sich andere Modifikationen und Variationen zu den Formeln finden, um weitere interessante Polygone berechnen lassen zu können.

18 `http://de.wikibooks.org/wiki/media%3ASVGpolygon05.svg`

7 Pfade

7.1 Pfade - beliebige Formen spezifizieren

Ein Pfad ist wahrscheinlich das meist verwendete Zeichenwerkzeug, welches in SVG Verwendung findet. Um einen Pfad anzugeben, wird das Element ***path*** verwendet. Es hat die spezifischen Attribute *d* und *pathLength*.

7.1.1 Attribut *d*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	2

Das Attribut *d* enthält die eigentlichen Pfaddaten, das sind Kommandos, meist gefolgt von lokalen Koordinaten oder Parametern. Der Wert ist also eine kompliziertere Zeichenkette, die im weiteren Text genauer erläutert wird.

Das Attribut ist animierbar. Allerdings gibt es spezielle Einschränkungen an die Struktur der Attributwerte für Animationen, die im weiteren Text noch erläutert werden.

Kommandos sind einzelne Buchstaben, Koordinaten oder Parameter oder eine bestimmte Anzahl von Zahlen oder einer vielfachen Menge von Zahlen hinter dem Kommando. Sind für ein Kommando etwa mindestens n weitere Koordinaten oder Parameter notwendig, so kann jeweils auch ein ganzzahliges Vielfaches von n an Koordinaten oder Parametern notiert werden. Zu jedem dieser n Koordinaten oder Parameter gehört dann ein Kurvenstück oder -segment, wobei das Kommando bestimmt, wie dieses dargestellt wird. Eines oder mehrere dieser Kurvensegmente können auch Pfadfragment(e) genannt werden, wenn sie im *d*-Attribut aufeinanderfolgen. Ein Pfad kann auch mehrere Unterpfade enthalten, die nicht miteinander verbunden sind. Trotzdem handelt es sich bei allen Unterpfaden, die in einem Attribut *d* notiert werden, um einen Pfad, was Folgen haben kann, wenn sich ein Pfad selbst überkreuzt. Je nachdem, was der Autor angibt, wie der Pfad gemalt werden soll, kann das Resultat für einen oder mehrere sich kreuzende Pfade deutlich unterschiedlich aussehen.

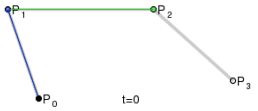
Ein einfaches Beispiel für eine Pfadangabe:

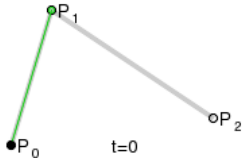
```
<path d="M 20 20 L 18 22 C 24 28 14 25 10 40 Q 20 45 15.33 60" />
```

Sieht kryptisch aus? Richtig, das ist es auch. Folgende Liste bietet eine Kurzübersicht über verfügbare Kommandos. Kommandos werden in großen oder kleinen Buchstaben angegeben. Große Buchstaben

entsprechen absoluten Koordinatenangaben. Kleine Buchstaben entsprechen relativen Angaben, relativ zum letzten Punkt des Pfades (also wirklich Punkt und nicht Kontrollpunkt).

Kommando	Parameter	Funktion	Beschreibung
M, m	(x y)+	Legt die Position des Zeichenstiftes fest und beginnt einen neuen Pfad oder Unterpfad.	Legt die Position des Zeichenstiftes fest, x für waagerecht und y für senkrecht. Es folgt mindestens ein Koordinatenpaar, bei mehreren wird ein Linienzug mit geraden Verbindungslinien angenommen.
L, l	(x y)+	Zeichnet eine gerade Linie	Zeichnet eine gerade Linie von der aktuellen Position zu den angegebenen (x,y) Koordinaten, welche die neue Position festlegen. Weitere Koordinatenpaare können angegeben werden, das ergibt dann einen Linienzug.
H, h	x+	Zeichnet eine horizontale Linie	Zeichnet eine horizontale Linie von der aktuellen Position, die vertikale Position bleibt gleich. Weitere Koordinaten führen wieder zu einem Linienzug.
V, v	y+	Zeichnet eine vertikale Linie	Zeichnet eine vertikale Linie von der aktuellen Position, die horizontale Position bleibt gleich. Weitere Koordinaten führen wieder zu einem Linienzug.

C, c	(x1 y1 x2 y2 x y)+	Zeichnet eine kubische Bézierkurve	Zeichnet eine kubische Bézierkurve von der aktuellen Position nach (x,y) unter der Verwendung von (x1,y1) als dem Kontrollpunkt am Anfang der Kurve und (x2,y2) als dem Kontrollpunkt am Ende der Kurve. Angabe von weiteren Kontrollpunkten und Punkten führt zu einem Kurvenzug. Die Anzahl der (Kontroll-)Punkte muss immer exakt stimmen.  **Abb. 62** Funktionsprinzip einer kubischen Bézierkurve

S, s	(x2 y2 x y)+	Kurzversion zur Fortsetzung einer kubischen Bézierkurve.	Aus dem letzten Kontrollpunkt und Punkt des vorherigen kubischen Bézierkurvensegmentes wird eine stetig differenzierbare Fortsetzung des Pfades berechnet. Ist das vorherige Segment keine kubische Bézierkurve, so wird der fehlende Kontrollpunkt als identisch mit dem Ausgangspunkt angenommen. Angabe von weiteren Kontrollpunkten und Punkten führt zu einem Kurvenzug. Die Anzahl der (Kontroll-)Punkte muss immer exakt stimmen.
Q, q	(x1 y1 x y)+	Zeichnet eine quadratische Bézierkurve	Zeichnet eine quadratische Bézierkurve von der aktuellen Position nach (x,y) unter Verwendung von (x1,y1) als Kontrollpunkt. Angabe von weiteren Kontrollpunkten und Punkten führt zu einem Kurvenzug. Die Anzahl der (Kontroll-)Punkte muss immer exakt stimmen.  **Abb. 63** Funktionsprinzip einer quadratischen Bézierkurve

T, t	(x y)+	Kurzversion zur Fortsetzung einer quadratischen Bézierkurve	Aus dem letzten Kontrollpunkt und Punkt des vorherigen quadratischen Bézierkurvensegmentes wird eine stetig differenzierbare Fortsetzung des Pfades berechnet. Ist das vorherige Segment keine quadratische Bézierkurve, so wird der fehlende Kontrollpunkt als identisch mit dem Ausgangspunkt angenommen. Angabe von weiteren Kontrollpunkten und Punkten führt zu einem Kurvenzug.

A, a	(rx ry x-axis-rotation large-arc-flag sweep-flag x y)+	Zeichnet einen Ellipsenbogen	Zeichnet einen elliptischen Kreisbogen von der aktuellen Position nach (x, y). Die Größe und die Ausrichtung der Ellipse werden bestimmt durch zwei Radien (rx, ry) und einer X-Achsen-Rotation, die bestimmt, wie die gesamte Ellipse relativ zum Koordinatensystem gedreht wird. Da es dann noch die Möglichkeit gibt, wie herum der Bogen gemalt wird und ob ein kleiner oder großer Bogen gemalt wird, gibt es auch noch Boolesche Parameter large-arc-flag sweep-flag. Die Werte sind 0 oder 1. large-arc-flag 1 bewirkt, dass der große Bogen gemalt wird, der Wert 0 entspricht dem kleinen Bogen. sweep-flag 1 entspricht einer positiven Malrichtung, 0 einer negativen. Im Bedarfsfalle, wenn es keinen Ellipsenbogen gibt, der nach den Angaben Anfangs- und Endpunkt verbindet, werden die Angaben vom Darstellungsprogramm umskaliert. Das Kommando ist in den tiny-Profilen nicht verfügbar. Hinsichtlich der (kontinuierlichen) Animation ist bei diesem Kommando bei diversen Darstellungsprogrammen mit Problemen zu rechnen, wobei das korrekte Verhalten auch nicht unbedingt dem entsprechen mag, was viele Autoren intuitiv

Z, z	(keine)	Schließt den (Unter-)Pfad ab	Zeichnet eine Linie von der aktuellen Position zur ersten Koordinate des aktuellen (Unter-)Pfades.

W3C path Referenz[1]

Die Trennung der Zahlen im Attribut voneinander kann auf verschiedene Weise stattfinden. Einige sehr kompakte Notationen, die in der Spezifikation erlaubt sind, können noch ein paar Bytes sparen, werden aber am ehesten kleine Implementierungsprobleme in Darstellungsprogrammen aufdecken. Sofern der Autor nicht daran interessiert ist, Implementierungsprobleme zu finden, empfiehlt es sich, Zahlen mit Leerzeichen zu separieren. Alternativ kann auch ein Komma mit optionalen Leerzeichen davor und danach verwendet werden. Es kann sich also anbieten, die beiden Zahlen eines Koordinatenpaares mit Komma zu trennen und die Koordinatenpaare voneinander mit Leerzeichen oder umgedreht. Das verbessert die Lesbarkeit des Quelltextes und gegebenenfalls das Lokalisieren von Stellen, wo der Autor etwas ändern möchte. Zwischen Kommandos und Zahlen brauchen keine Leerzeichen gesetzt werden und ein Komma wäre dort falsch. Zeilenumbrüche statt Leerzeichen sind auch in Ordnung und können ebenfalls die Lesbarkeit des Quelltextes verbessern.

Insbesondere bei umfangreicheren Pfadangaben kann es für die gesamte Dateigröße entscheidend sein, wie geschickt oder ungeschickt die Pfaddaten notiert werden. Sind in einem Pfad aufeinanderfolgende Punkte auch räumlich benachbart, so sind meist relative Angabe mit kleinen Kommandos effektiver als absolute Angaben. Auch die Anzahl der angegebenen Stellen bei einer Zahl ist relevant sowohl für die Genauigkeit bei Skalierung als auch für die Dateigröße. Wenn drei statt sechs gültige Ziffern ausreichen, um die graphische Information zu vermitteln, kann das bei großen Pfaden eine Reduktion der Dateigröße um die Hälfte ausmachen. Entsprechend kann es sich lohnen, das Koordinatensystem so zu wählen, dass die meisten Koordinaten zum Beispiel zwischen 0 und 999 liegen. Mittels *viewBox* oder Transformationen von Gruppen kann dann dies Koordinatensystem an den gewünschten Anzeigebereich angepasst werden.

Als Sonderfall ist ein (Unter-)Pfad anzusehen, der keine Ausdehnung hat, etwa *d*="M0 0l0 0". Ein solcher ist darzustellen, wenn das Präsentationsattribut *stroke-linecap* nicht 'butt' ist. Die Darstellung besteht dann nur aus den Linienenden.

Angaben wie *d*="M100,100m 100,-100m-100,-100l400,400" sind ebenfalls möglich, der Startpunkt wird dann jeweils verschoben, bis etwas gemalt werden soll, also hier das Kommando l folgt.

Zur Animation des Attributes *d*:

Aufgrund der Struktur des Attributwertes ist *d* nicht additiv und eine Animation mit *calcMode* paced ist nicht sinnvoll. Ferner müssen die Animationswerte für *d* immer exakt dieselbe Abfolge von Kommandos haben und pro Kommando muss die Anzahl der angegebenen Koordinaten und Parameter gleich bleiben. Bei SVG tiny 1.2 können ferner bei verschiedenen Animationswerten große und kleine Kommandos variiert werden, weil die kleinen Kommandos vor der Animation vom Darstellungsprogramm in große umzurechnen sind. Ferner kann bei SVG tiny 1.2 bei einer

1 `http://www.w3.org/TR/SVG/paths.html`

diskreten Animation der Pfad in jedem Animationswert anders aufgebaut sein und unterliegt nicht den Beschränkungen von SVG 1.1.

7.1.2 Attribut *pathLength*

Squiggle (Batik)	-
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

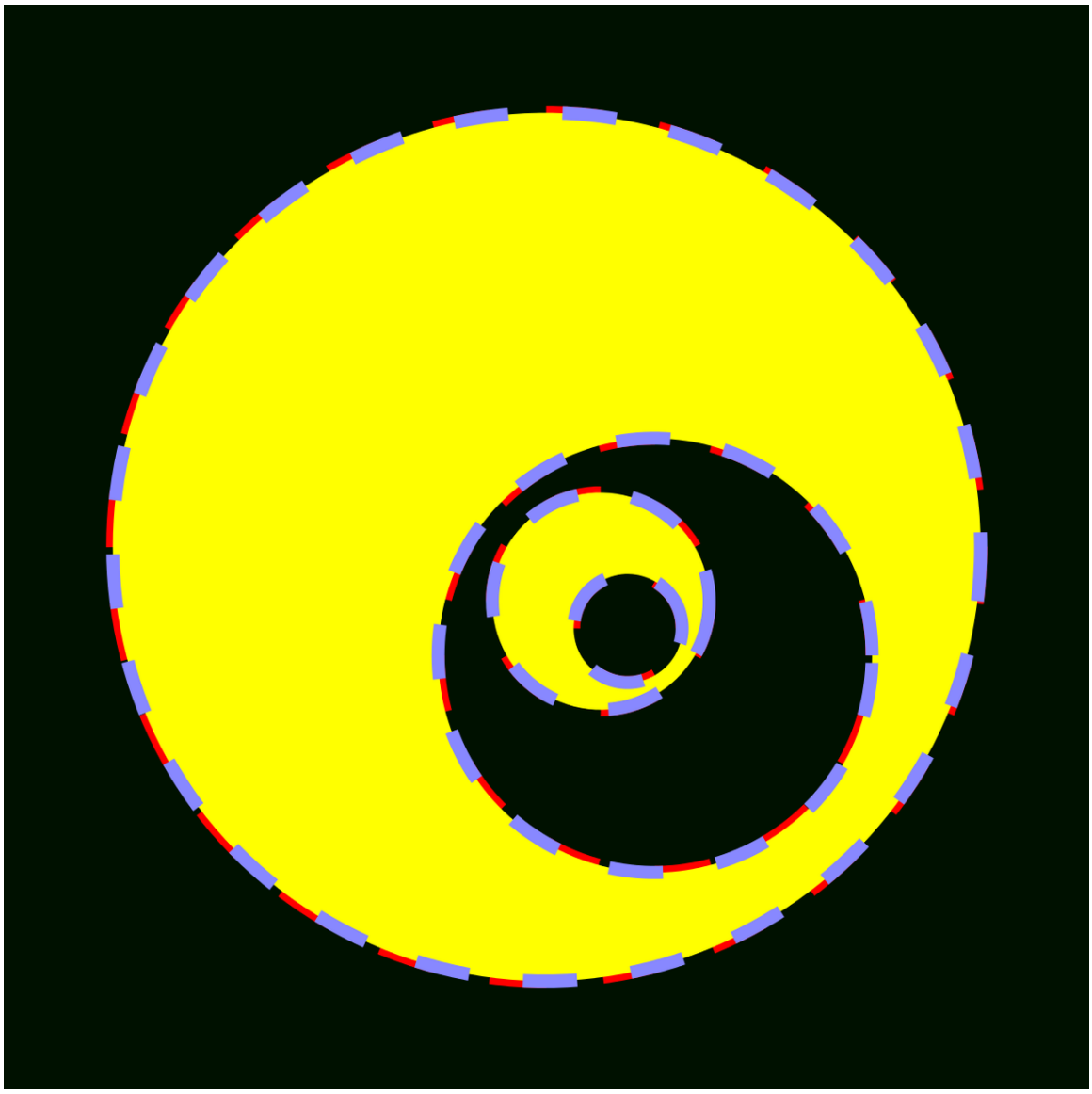


Abb. 64 Beispiel*pathLength*: kubische Pfade als Kreisnäherung mit gestricheltem Rand

pathLength gibt an, welche Pfadlänge der Autor für den angegebenen Pfad berechnet hat. Der Wert ist eine (positive) Zahl für die Pfadlänge in lokalen Einheiten. Ein negativer Wert ist in SVG 1.1 ein Fehler, der zum Abbruch der Darstellung führt, in SVG tiny 1.2 wird in dem Falle nur das Attribut ignoriert.

Das Attribut ist animierbar.

Die explizite Berechnung der Pfadlänge ist teilweise für einige Pfadsegmente nur numerisch möglich, weswegen es da Abweichungen zwischen den Ergebnissen des Autors und dem jeweils verwendeten Darstellungsprogramm geben kann. Das Darstellungsprogramm sollte dann die Darstellung zum Beispiel von gestrichelten Linien entlang des Pfades entsprechend anpassen. Nachvollziehbar ist dies zum Beispiel mit Opera.

Beispiel:

kubische Pfade als Kreisnäherung mit gestricheltem Rand[2]

Mit kubischen Pfaden werden ineinanderliegende Kreise angenähert. Zumindest im tiny-Profil lässt sich die Figur mit einem Pfad nicht besser realisieren, weil es da keine elliptischen Pfadsegmente gibt. Für exakte Kreise wäre die Pfadlänge exakt bekannt, hier 1500α.

Aufgrund der kubischen Beispiele stimmt dies sowieso nicht ganz genau, für das Beispiel wird das großzügig auf 4500 abgerundet, das weicht nur um weniger als 5 Prozent vom richtigen Wert ab.

Besser wäre offenbar 4712.4 als Schätzung. Jenseits von Demonstrationsbeispielen ist es für Autoren immer ratsam, möglichst präzise zu schätzen.

Aufgrund der Verhältnisse der Durchmesser 8:4:2:1 der genäherten Kreise ergibt sich für das Muster von *stroke-dasharray* ein schönes Muster mit dem Wert 50 (oder eben genauer $50\alpha/3$), bei dem wird der Strich des kleinsten Kreises gerade in drei gleichgroße gemalte und 3 weitere ebenso große nicht gemalte Teile zerlegt, die Striche der größeren Kreise werden entsprechend der Verhältnisse der Durchmesser in mehr gleichgroße Teile zerlegt.

Zum Vergleich wird darunterliegend der gleiche Pfad ohne das Attribut verwendet und ein entsprechend optimierter Wert für *stroke-dasharray* verwendet. Dort ist das Strichmuster rot. Wird davon etwas sichtbar, hat das Darstellungsprogramm offenbar einen Fehler.

Opera hat einen für viele Versionen bekannten Fehler für *stroke-dasharray* und geschlossene Unterpfade. Der wird hier kompensiert durch das Anhängen des Anfangspunktes des Pfades mit einem M-Kommando, dies hat keinen Einfluss auf die korrekte Darstellung, kompensiert aber den Opera-Fehler.

2 `http://de.wikibooks.org/wiki/media%3ASVGpathLength01.svg`

7.2 Kurven mit speziellen Programmen erzeugen

Nun sind vermutlich nur wenige Autoren bereit und motiviert, kompliziertere Kurven per Hand mit einem Texteditor einzutippen.

Stattdessen bieten sich Programme wie Inkscape an. Diese haben eine graphische Nutzeroberfläche und weitere Hilfen, um mit einem Zeigergerät wie einer Maus oder auch einer Kombination aus elektronischem Stift und einem Tablett Pfade zu malen. Obwohl solche Programme recht nützlich sind, bleibt es dem Autor meist nicht erspart, nach der Erstellung eines Dokumentes den Quelltext gründlich aufzuräumen. Das reduziert nicht nur meist dramatisch die Dateigröße, sondern verbessert auch die Zugänglichkeit. Offen bleibt dabei die Frage, warum solche Programme dem Autor nicht gleich dabei helfen, zugängliche Dokumente zu erstellen und zu optimieren.

Abb. 65 'Hallo Welt!' mit Inkscape als Kalligraphie

Beispiel: 'Hallo Welt!' mit Inkscape als Kalligraphie[3].

Nachdem das Dokument mit Inkscape erstellt wurde, wurde dies nicht als 'Inkscape-Dokument', sondern als schlichtes SVG-Dokument abgespeichert. Dann wurden die Pfade um die unnötigen Nachkommastellen bereinigt, *style*-Attribute entsorgt, ***title*** und ***desc*** ergänzt, die Version von 1.0 auf 1.1 hochgesetzt, das Profil tiny angegeben, ein Rechteck als Hintergrund gesetzt.

3 `http://de.wikibooks.org/wiki/media%3ASVGHalloWelt01.svg`

Eine weitere Methode besteht darin, eine geeignete Pixelgraphik, etwa das Bild einer Digitalkamera in Pfade zu konvertieren. Dazu dienen zum Beispiel Programme wie Potrace. Potrace läuft als eigenständiges Programm, ist aber auch in Inkscape verfügbar. Eine solche Konversion wird auch Vektorisierung genannt.

Ein erfolgversprechendes Vorgehen bei der Vektorisierung besteht im ersten Schritt darin, ein geeignetes Bild auszuwählen. Dieses sollte aus einem eindeutigen Hauptmotiv bestehen und möglichst einheitlichem Hintergrund. Hoher Kontrast ist ebenfalls von Vorteil. In einem zweiten Schritt kann dieses Bild mit einem Programm wie Gimp manipuliert werden, also störende Details und unregelmäßigen Hintergrund entfernen, gegebenenfalls Kontrast erhöhen. Im dritten Schritt wird mit Potrace die Vektorisierung vorgenommen, wobei zwischen mehreren Strategien gewählt werden kann: Vektorisierung nach Farbe, Kontrast, Anzahl der verschiedenen Farbschichten, aufeinander gestapelt oder unabhängig voneinander etc. Was davon sinnvoll ist, hängt vom Bild ab, aber natürlich auch davon, was der Autor erreichen will.

Zum Beispiel mit Inkscape kann nun das Resultat weiter bearbeitet werden, Pfade können automatisch vereinfacht werden. Die Vereinfachung besteht darin, die Anzahl der Punkte und Kontrollpunkte zu reduzieren und ihre Werte so zu manipulieren, dass ein einfacherer, glatterer Umriß entsteht. Es gibt auch Hilfen, mit denen manuell Punkte und Kontrollpunkte verändert und manipuliert werden können.

Ist das Ergebnis dann zufriedenstellend, kann dies als einfache SVG-Datei abgespeichert werden. Mit einem kurzen Blick in den Quelltext und auf die Dateigröße wird offenbar, dass ein solches Ergebnis aufgrund der Dateigröße nicht besonders brauchbar ist und die Genauigkeit der Pfadangaben deutlich zu hoch für die gewünschte Anwendung ist. Zudem hat Inkscape die unangenehme Angewohnheit, Eigenschaften im Attribut *style* abzulegen, statt in Präsentationsattributen, was für einfachere Darstellungsprogramme, die auf SVG tiny ausgelegt sind, eine Zugänglichkeitsbarriere darstellt.

Der nächste Schritt besteht also darin aufzuräumen. Dies kann rein manuell mit einem Texteditor geschehen. Effektiver ist es natürlich, sich dafür ein entsprechendes Programm oder Skript zu schreiben, welches die Genauigkeit der Angaben reduziert und Eigenschaften in Präsentationsattribute konvertiert, gegebenenfalls auch nur die Pfade zur Weiterverwendung extrahiert. Je nach Motiv, Inhalt, Geschick und Zielrichtung lässt sich die Dateigröße so meist auf einige Prozent bis schlimmstenfalls fünfzig Prozent reduzieren. Jeff Schiller bietet etwa das Skript 'scour' an, welches zum Beispiel Dateien von Inkscape ordentlich aufräumt, allerdings die Pfade nicht rundet oder vereinfacht.

Abb. 66 Vogel

Beispiel: Vogel[4].

Die Abstrahierung des Motivs oder die Optimierung ist bei diesem Motiv nur so weit fortgeschritten, dass noch deutlich die Vorgehensweise von Potrace bei der Vekorisierung an der Art der Kurven zu erkennen ist. Potrace erkennt nicht das Motiv selbst, sondern wählt nach Grauwerten oder hier Farben aus, vereinfacht und umrandet gleichartige Gebiete mit jeweils einem Kurvenzug. Das Programm versteht also nicht, was aus menschlicher Sicht wichtig oder relevant ist, daher sind die erzeugten Pfade oft nicht direkt mit dem zu vergleichen, was ein Mensch als Struktur erkennen und malen würde. Bei einer weiteren Optimierung oder Abstrahierung müsste so vorgegangen werden, dass die Pfade wirklich Bestandteilen des Motivs zugeordnet werden, die von Menschen als relevant angesehen werden. Dies entspricht also einem weiteren Aufspalten von Unterpfaden in selbständige Pfade, eventuell dann weitere Vereinfachungen, an einigen Stellen auch Präzisierungen und Anpassungen der Kurvenform.

Eine weitere Methode der Vektorisierung besteht darin, das Bild der Digitalkamera in ein Programm wie Inkscape zu laden und dann mit diesem das Motiv nachzumalen, also als Hilfe und Vorlage zu verwenden. Das hat den Vorteil, dass der Mensch die relevanten Strukturen besser erkennen und nach Relevanz betonen kann, um die mit der Vektorisierung einhergehende Abstrahierung des Motivs gezielt nach eigenem Verständnis zu gestalten.

4 `http://de.wikibooks.org/wiki/media%3ASVGvogel01.svg`

Für diverse Probleme ist es natürlich effektiver, gleich selbst ein Programm oder Skript zu schreiben, um die gewünschten Pfade in einer sinnvollen Genauigkeit zu erzeugen. Dazu ist es dann notwendig, einiges Hintergrundwissen über die in SVG verwendeten Pfade zu haben. Darum geht es im nächsten Abschnitt.

7.3 Pfadkommandos im Detail

In obiger Kurzbeschreibung ist ja bereits zu lesen, wozu die jeweiligen Pfadkommandos dienen. Um dies wirklich effektiv in eigenen Skripten und Programmen nutzen zu können, lohnt sich ein genauerer Blick auf die Syntax und die Mathematik, welche die Kurvenstücke beschreiben.

Eine Parametrisierung eines Kurvensegmentes ist hilfreich, um das Kurvensegment zu analysieren, Fortsetzungen ohne Knick zu bestimmen, Markierungen anzubringen oder gegebenenfalls andere Kurven zu berechnen, die diese unter einem definierten Winkel schneiden. Es gibt viele weitere Anwendungen, die eine mathematische Herangehensweise erfordern, um zügig ein korrektes Ergebnis zu erhalten. Generell gilt, dass es für dasselbe Kurvensegment nicht nur eine Parametrisierung gibt, sondern viele. Für gerade, quadratische und kubische Kurvensegmente werden die üblichen Parametrisierungen nach Bézier gewählt. Die Darstellung ist unabhängig von der Parametrisierung, solange diese nicht zum Beispiel innerhalb derselben Animation gewechselt wird. Eine Richtungsinformation ist hilfreich, wenn ein Kurvensegment durch ein weiteres ohne Knick fortgesetzt werden soll (stetig differenzierbare Fortsetzung). Interessant ist dies auch, wenn an beliebiger Stelle eine Richtungsmarkierung wie ein Pfeil angebracht werden soll, ebenso, wenn nach einem Pfadsegment gesucht werden, das ein vorgegebenes unter einem bestimmten Winkel schneiden soll. Die Richtung ergibt sich im Regelfall aus der Ableitung der Parametrisierung, wie sich die Krümmung des Kurvensegmentes aus der zweiten Ableitung ergibt.

Eine Linie in Richtung der Linie beim Punkt P ergibt sich aus der Ableitung A als Linie von P nach P + c A mit c größer als 0. Eine senkrechte Linie ergibt sich durch Vertauschung der Komponenten von A und einem Vorzeichenwechsel einer der Komponenten.

Bei Parametrisierungen kann es auch an der Parametrisierung selbst liegen, wenn die Ableitung 0 wird, in solchen Fällen kann die zweite Ableitung verwendet werden, um die Richtung zu bestimmen (Regel von L'Hospital). Dieses Problem tritt bei Bézierkurven auf, wenn ein Kontrollpunkt mit dem zugehörigen Anfangs- beziehungsweise Endpunkt zusammenfällt. Statt die Regel von L'Hospital nun jedes Mal explizit anzuwenden, gilt für Bézierkurven: Sofern nicht alle Punkte zusammenfallen, ergibt sich dann die Richtung aus der Richtung zum nächsten (Kontroll-)punkt des gleichen Segmentes, welches vom aktuellen verschieden ist.

7.3.1 Pfad oder Unterpfad beginnen: M, m

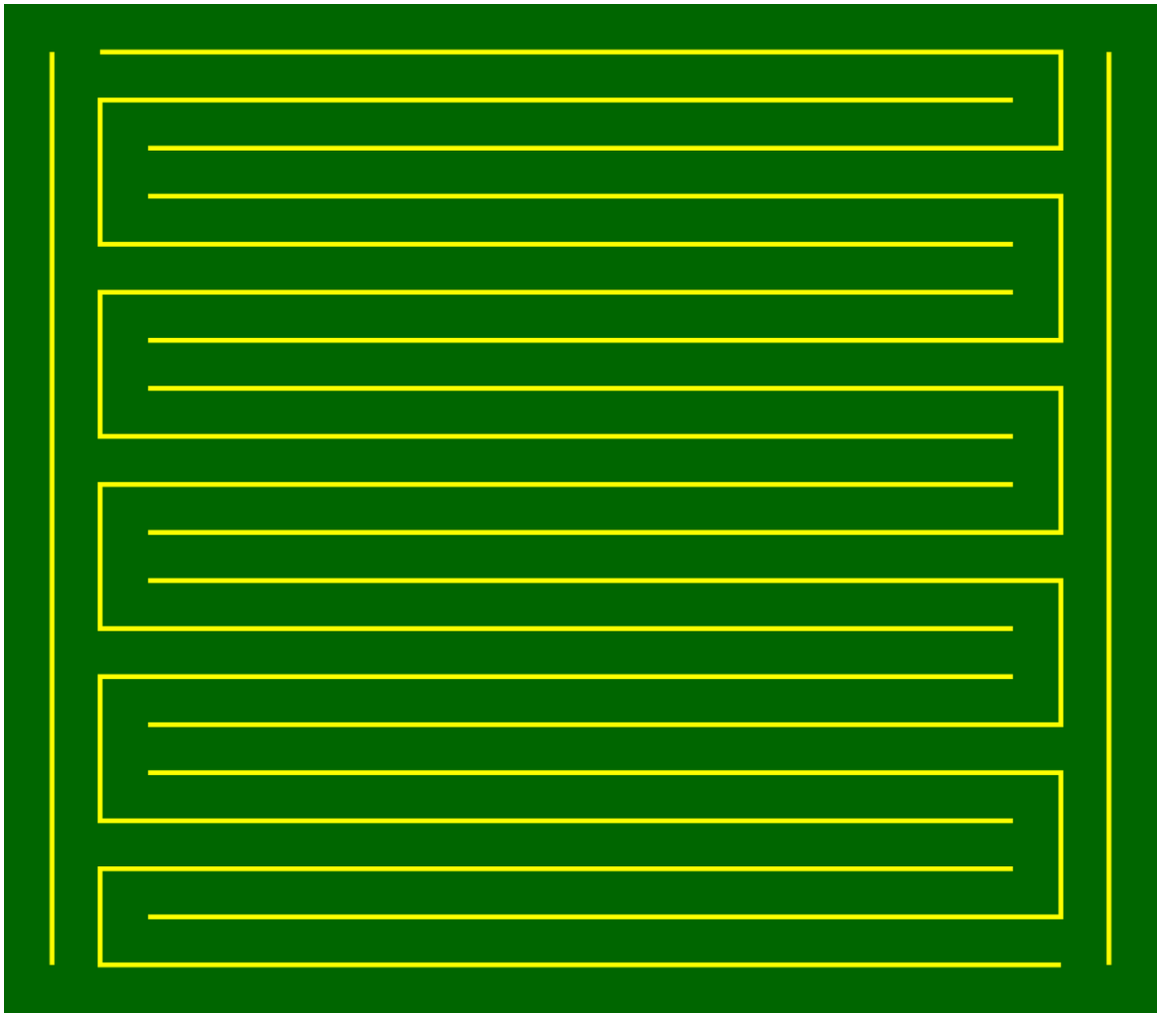


Abb. 67 Pfad mit Unterpfaden, nur mit M, m konstruiert

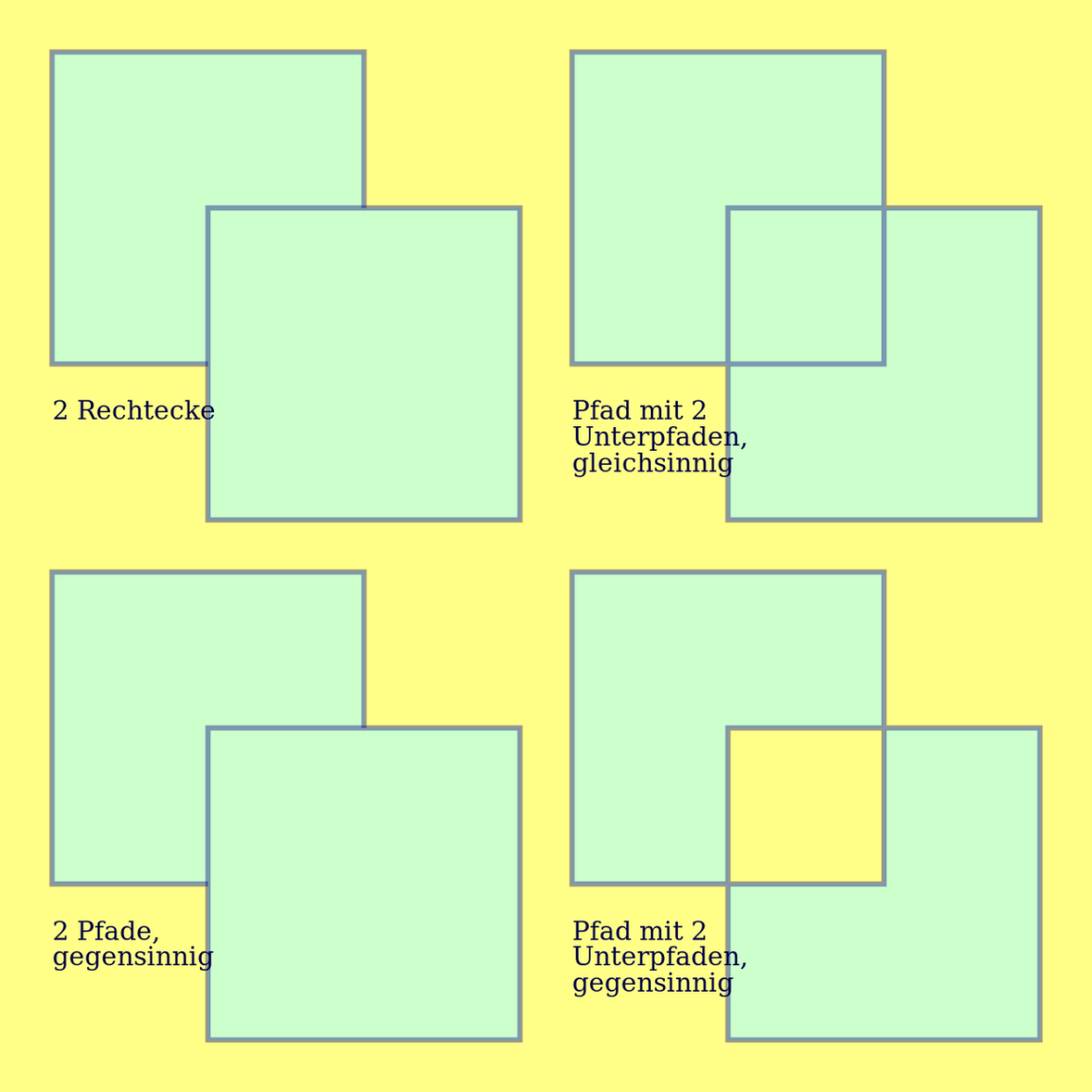


Abb. 68 Pfade mit und ohne Unterpfade, gleichsinnige und gegensinnige Umläufe

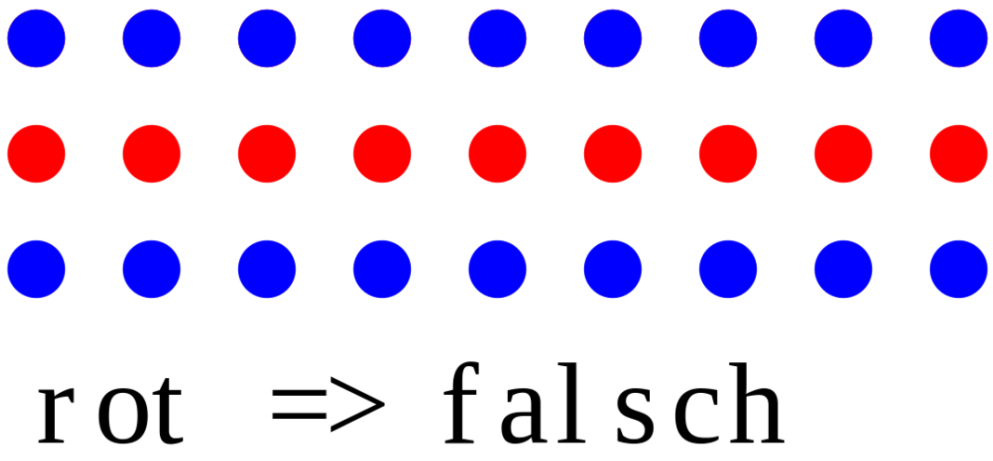


Abb. 69 Pfade der Länge 0

. Mit den Kommandos 'M' oder 'm' beginnt jedes Pfadkommando. Da es keine früheren Pfaddaten gibt, bezieht sich das relative Kommando 'm' einfach auf den aktuellen Ursprung. Darauf folgt mindestens ein Koordinatenpaar, welches angibt, wo der Pfad beginnt. Im weiteren Verlauf können neben anderen Kommandos auch weitere Kommandos 'M' oder 'm' in einem Pfad auftauchen. Die bedeutet dann, dass der vorherige Unterpfad beendet ist und ein neuer Unterpfad an der Stelle begonnen wird, die durch das folgende Koordinatenpaar angegeben ist.

Folgen auf 'M' oder 'm' mehrere Koordinatenpaare, so werden die in der angegebenen Reihenfolge zu einem Linienzug verbunden. Gibt es nur einen Pfad ohne weitere Unterpfade, so entspricht das dem Element ***polyline***, welches dann zu bevorzugen ist.

Beispiel:

```
<path fill="none" stroke="#ff0"
d="M0,0 200,0 200,20 10,20
M190,10 0,10 0,40 190,40
m-180,-10 190,0 0,30 -190,0
m180,-10 -190,0 0,30 190,0
m-180,-10 190,0 0,30 -190,0
m180,-10 -190,0 0,30 190,0
m-180,-10 190,0 0,30 -190,0
m180,-10 -190,0 0,30 190,0
m-180,-10 190,0 0,30 -190,0
m180,-10 -190,0 0,20 200,0
M-10,0 -10,190 M210,190 210,0" />
```

Pfad mit Unterpfaden, nur mit M, m konstruiert[5].

Schon bereits ein weiterer Unterpfad macht einen Unterschied. Bei Überschneidungen kann das Resulat ein anderes sein als das von zwei Pfaden oder Formen.

Beispiel: Pfade mit und ohne Unterpfade, gleichsinnige und gegensinnige Umläufe[6].

5 http://de.wikibooks.org/wiki/media%3ASVGpathM01.svg
6 http://de.wikibooks.org/wiki/media%3ASVGpathM02.svg

Pfade der Länge null werden genau dann gemalt, wenn das Präsentationsattribut *stroke-linecap* auf 'round' oder 'square' steht (dies kann in den Implementierungsanmerkungen der SVG-Spezifikationen nachgelesen werden). Für den Wert 'butt' wird nichts dargestellt. Für 'round' wird ein Kreis mit einem Durchmesser der *stroke-width* dargestellt. Für 'square' wird ein Quadrat mit der Kantenlänge der *stroke-width* dargestellt. Die Kanten des Quadrates sind dann zu den (lokalen) x- und y-Richtungen ausgerichtet.

Dies ermöglicht es, sehr kompakt eine Reihe von 'Punkten' als Kreise oder Quadrate mittels eines Pfades mit Unterpfaden der Länge null zu notieren.

In folgendem Beispiel werden einige mögliche Pfade der Länge 0 ausprobiert. Ist jeweils ein unterliegender roter Kreis zu sehen, ist das verwendete Darstellungsprogramm fehlerhaft.

Beispiel: Pfade der Länge 0[7].

7 `http://de.wikibooks.org/wiki/media%3ASVGpathM03.svg`

7.3.2 Pfad oder Unterpfad beenden: Z, z

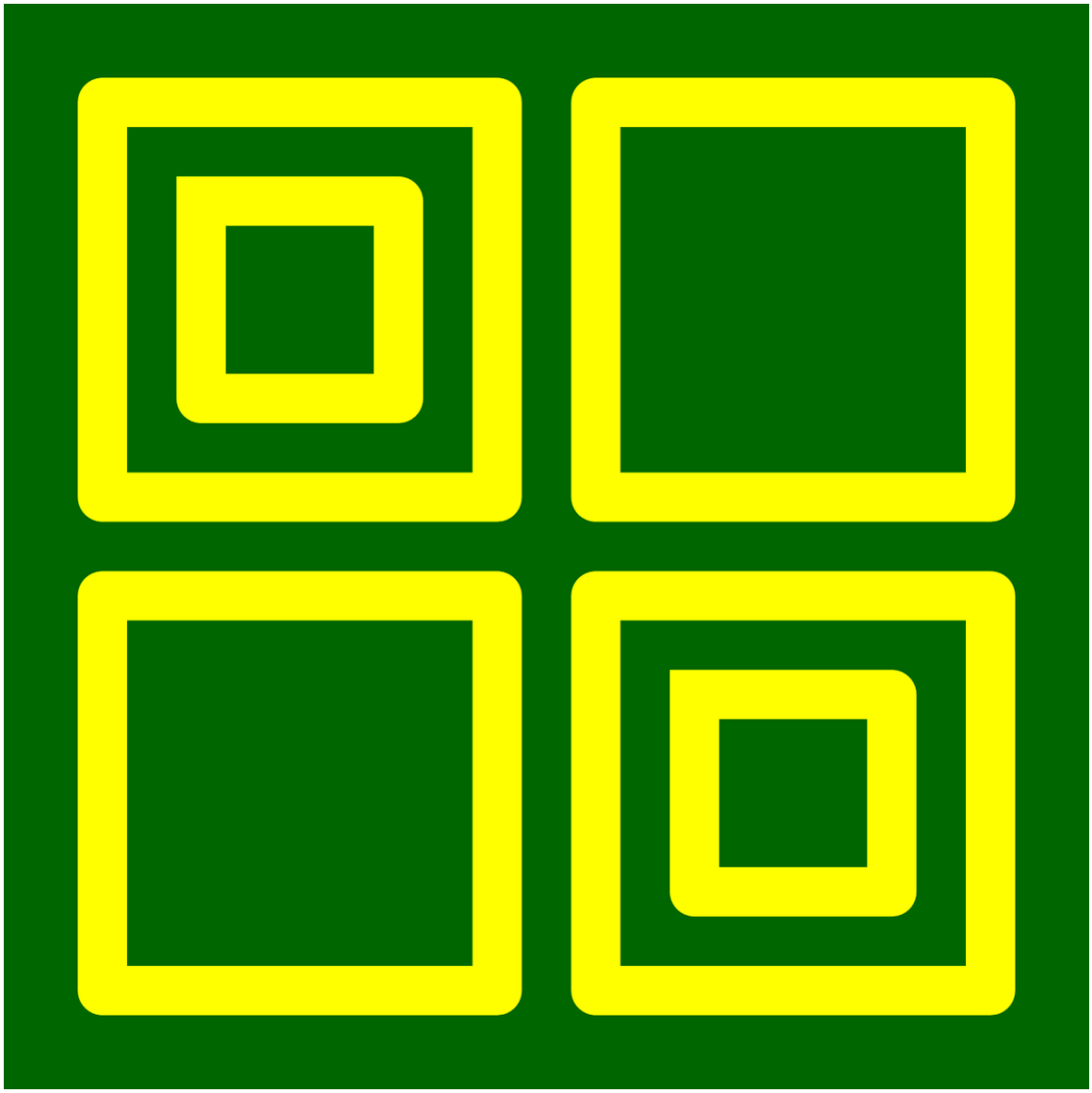


Abb. 70 Pfade mit und ohne Z - Einfluss auf die Darstellung

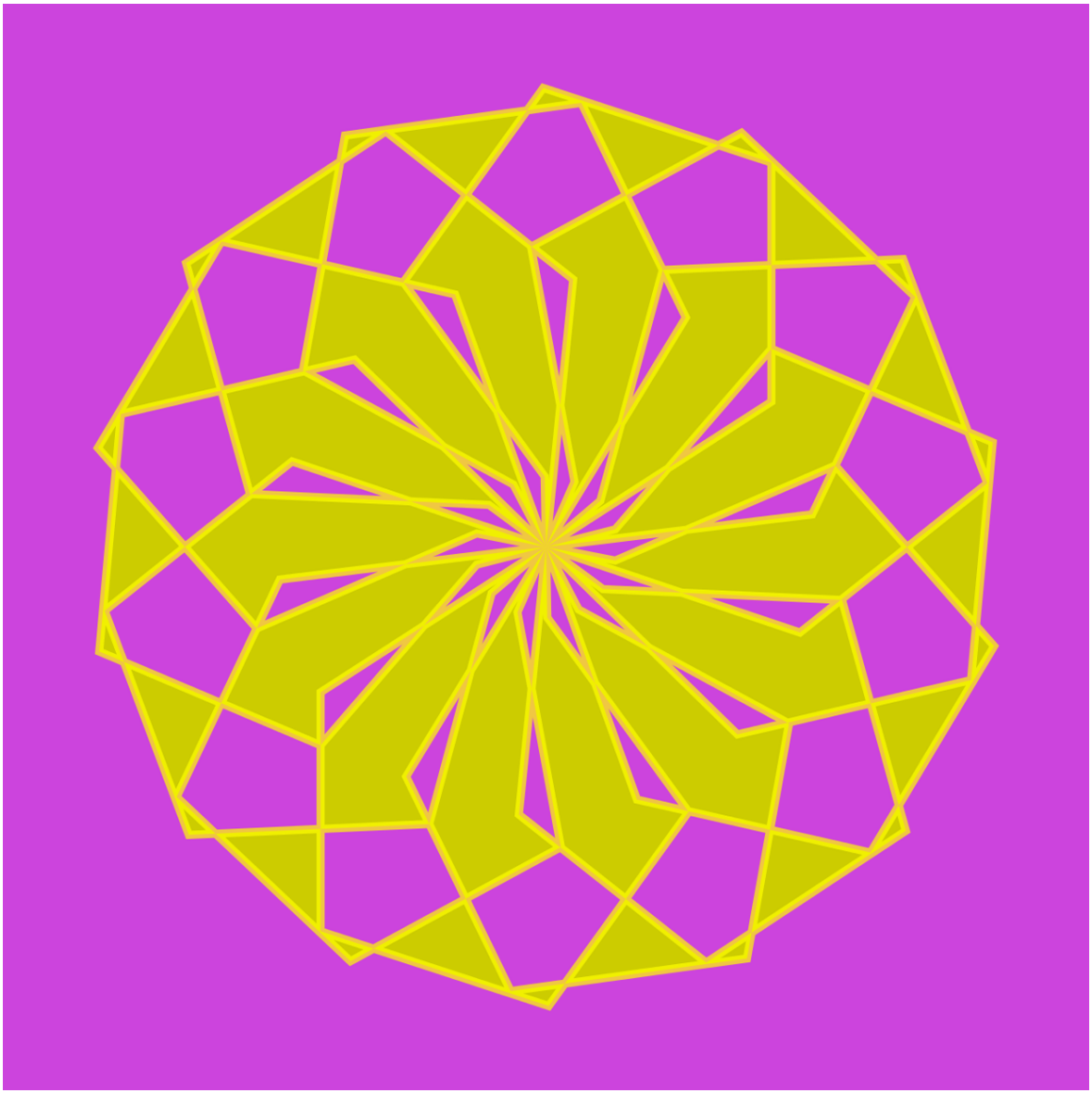


Abb. 71 Fortsetzung von Pfaden hinter dem Kommando Z

Mit dem Kommando 'Z' oder 'z' (was das gleiche bewirkt und keine weiteren folgenden Koordinaten oder Parameter erwartet) wird eine gerade Linie zurück zum Anfangspunkt eines Unterpfades gezogen und somit der Unterpfad geschlossen. Der Anfangspunkt ist gegeben durch das Koordinatenpaar, welches auf das vorherige Kommando 'M' oder 'm' folgt. Sofern auf ein 'Z' oder 'z' kein weiteres 'M' oder 'm' folgt, sondern anderes Kommando, so ist folglich die aktuelle Position die des Anfangspunktes gegeben durch das Koordinatenpaar, welches auf das vorherige Kommando 'M' oder 'm' folgt. Die graphische Repräsentation eines geschlossenen Pfades kann sich leicht von der unterscheiden, die sich ergibt, wenn einfach der Endpunkt gleich dem Anfangspunkt explizit angegeben wird. Das hängt mit den Präsentationsattributen *stroke-linecap* und *stroke-linejoin* zusammen. Ein mit 'Z' oder 'z' geschlossener Unterpfad hat keine Linienenden, die Kurvensegmente sind zusammengefügt. Ohne dieses Kommando hat es Linienenden. Es wird also jeweils das andere Attribut verwendet.

Pfade mit und ohne Z - Einfluss auf die Darstellung[8].

Nach einem Kommando z kann auch ein anderes Kommando als M oder m folgen. In folgendem Beispiel folgt ein L. Folglich geht der Pfad am Anfangspunkt weiter. Hier ist eine Spielerei mit diskreter Drehsymmetrie dargestellt, der wiederholt genutzte Anfangspunkt liegt im Drehzentrum bei 0.

Fortsetzung von Pfaden hinter dem Kommando Z[9].

7.3.3 Gerade Linienstücke: L, l, H, h, V, v

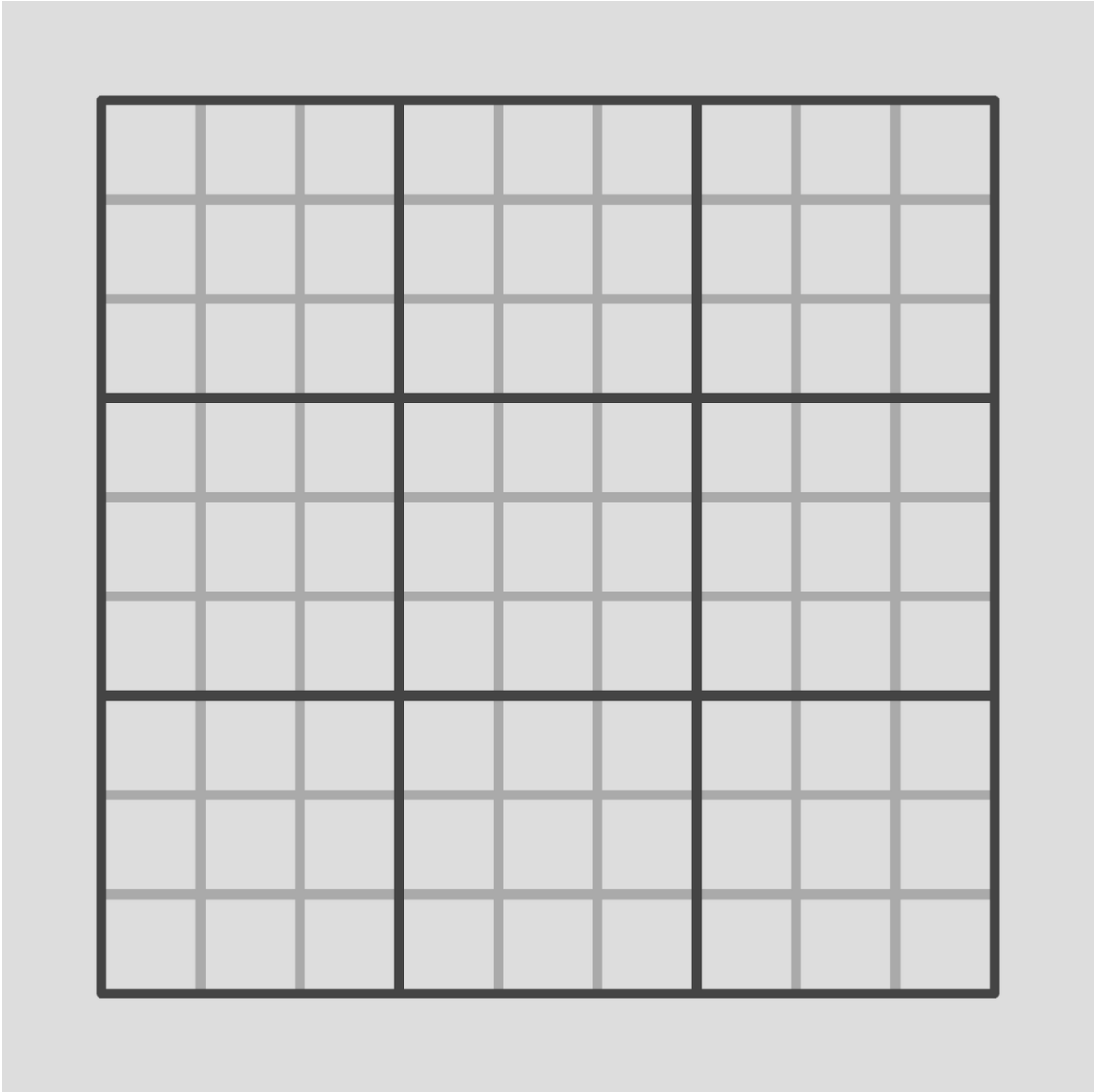


Abb. 72 Sudoku-Spielfeld - kompakte Darstellung mit 'H', 'h', 'V', 'v'

8 http://de.wikibooks.org/wiki/media%3ASVGpathZ01.svg
9 http://de.wikibooks.org/wiki/media%3ASVGpathZ02.svg

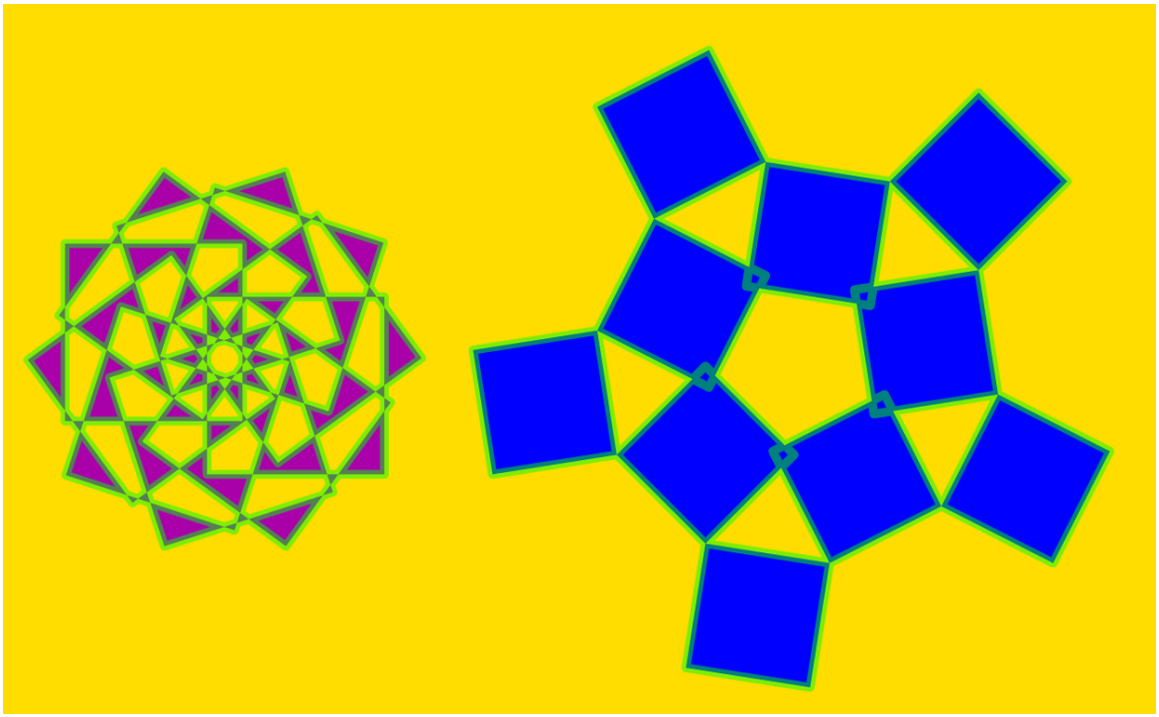


Abb. 73 Spielerei mit Unterpfaden, bestehend aus Quadraten - Beispiel für Kommandos 'L' und 'l'

Mit den Kommandos 'L', 'l', 'H', 'h', 'V', 'v' können gerade Pfadsegmente angegeben werden, mit 'H', 'h' horizontale Linien, mit 'V', 'v' vertikale Linien, indem jeweils die horizontale beziehungsweise vertikale Koordinate des Endpunkts angegeben wird. Mit 'L', 'l' werden Anfangs- und Endpunkt mit einer Linie verbunden.

Eine Parametrisierung ist für ein gerades Pfadsegment recht einfach. Seien Anfangs- und Endpunkt P(0) und P(1), sei der Laufparameter s aus [0,1], so gilt für eine lineare Bézierkurve

$k(s) = (1 - s)P(0) + sP(1)$

mit der Ableitung

$dk(s)/ds = P(1) - P(0)$

und der zweiten Ableitung

$d^2k(s)/ds^2 = 0$

Sudoku-Spielfeld - kompakte Darstellung mit 'H', 'h', 'V', 'v'[10].

Spielerei mit Unterpfaden, bestehend aus Quadraten - Beispiel für Kommandos 'L' und 'l'[11].

10 http://de.wikibooks.org/wiki/media%3ASVGpathHV01.svg
11 http://de.wikibooks.org/wiki/media%3ASVGpathL01.svg

Als kleine Übung sei nun noch die Frage betrachtet, wie zwei gerade Pfadsegmente so zu verlängern sind, dass sie als ein Kurvenzug dargestellt werden können. Das läuft darauf hinaus, erstmal den Schnittpunkt von zwei Geraden zu finden.

Sei die erste Gerade K(t) gegeben durch die nicht gleichen Punkte P und Q mit dem Parameter t in der Form:

K(t) = (1-t) Q + t P = Q + (P - Q) t = Q + R t mit R = P - Q

Sei die zweite Gerade L(s) gegeben durch die nicht gleichen Punkte U und S mit dem Parameter s in der Form:

L(s) = (1-s) S + s U = S + (U - S) s = S + T s mit T = U - S

Um den Schnittpunkt zweier Geraden (in der Ebene) zu finden, wird zunächst geprüft, ob es einen solchen Schnittpunkt überhaupt gibt. Das ist der Fall, wenn sie nicht parallel sind, wenn also T und R nicht parallel sind. Mit einer Bezeichnung der Komponenten in x- und y-Richtung entsprechend sind R und T parallel, wenn Tx Ry - Ty Rx = 0 ist. Dann gibt es also keinen Schnittpunkt.

Sonst ist der Schnittpunkt V der beiden Geraden gegeben durch:

(Q - S) #R /(T #R) = s

beziehungsweise

(S - Q) #T /(R #T) = t

V = Q + R t = S + T s

Dabei sind #R = (Ry, -Rx) und #T = (Ty, -Tx) senkrecht zu R beziehungsweise T.

Sind s und t größer oder gleich 1, so ergibt Q P V S U den Kurvenzug oder auch einfacher Q V U. Sind s und t kleiner oder gleich 0, so sind die Punkte in anderer Reihenfolge zu notieren: P Q V U S oder auch einfacher P V S. Ist nur einer der Werte kleiner oder gleich 0, der andere größer oder gleich 1, so sind nur für den mit einem Wert kleiner oder gleich 0 die beiden Punkte in der Reihenfolge zu vertauschen. In den anderen Fällen liegt der Schnittpunkt auf mindestens einem der beiden Kurvensegmente und der Linienzug ist anders zu malen, zum Beispiel mit zwei Unterpfaden, entsprechend der parallele Fall.

Diese Betrachtung kann im Folgenden noch nützlich sein, wenn für zwei Pfade ein quadratisches Kurvensegment gesucht wird, mit dem die beiden Pfade ohne Knick verbunden werden sollen. Der Kontrollpunkt entspricht dann dem Punkt V, wenn s und t größer oder gleich 1 sind. Da bei vorgegebener Richtung der Pfade eine Vertauschen nicht möglich ist, stellt sich damit bereits heraus, dass es viele Fälle gibt, wo mehr als ein quadratisches Kurvensegment gebraucht wird, um zwei Pfade ohne Knick miteinander zu verbinden.

7.3.4 Quadratische Kurvenstücke: Q, q, T, t

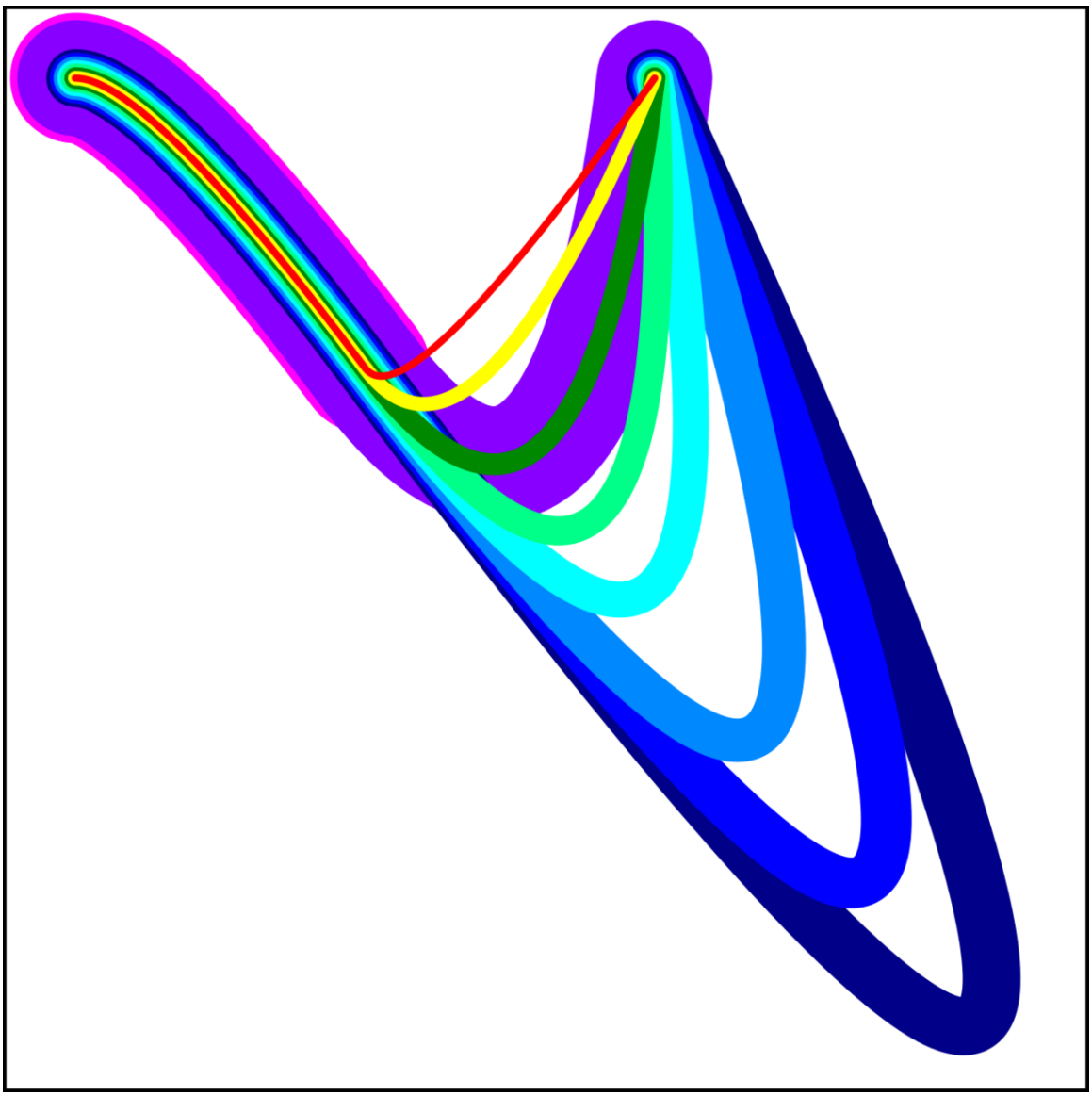


Abb. 74 Fortsetzungen eines quadratischen Pfades ohne Knicke, Kommando T

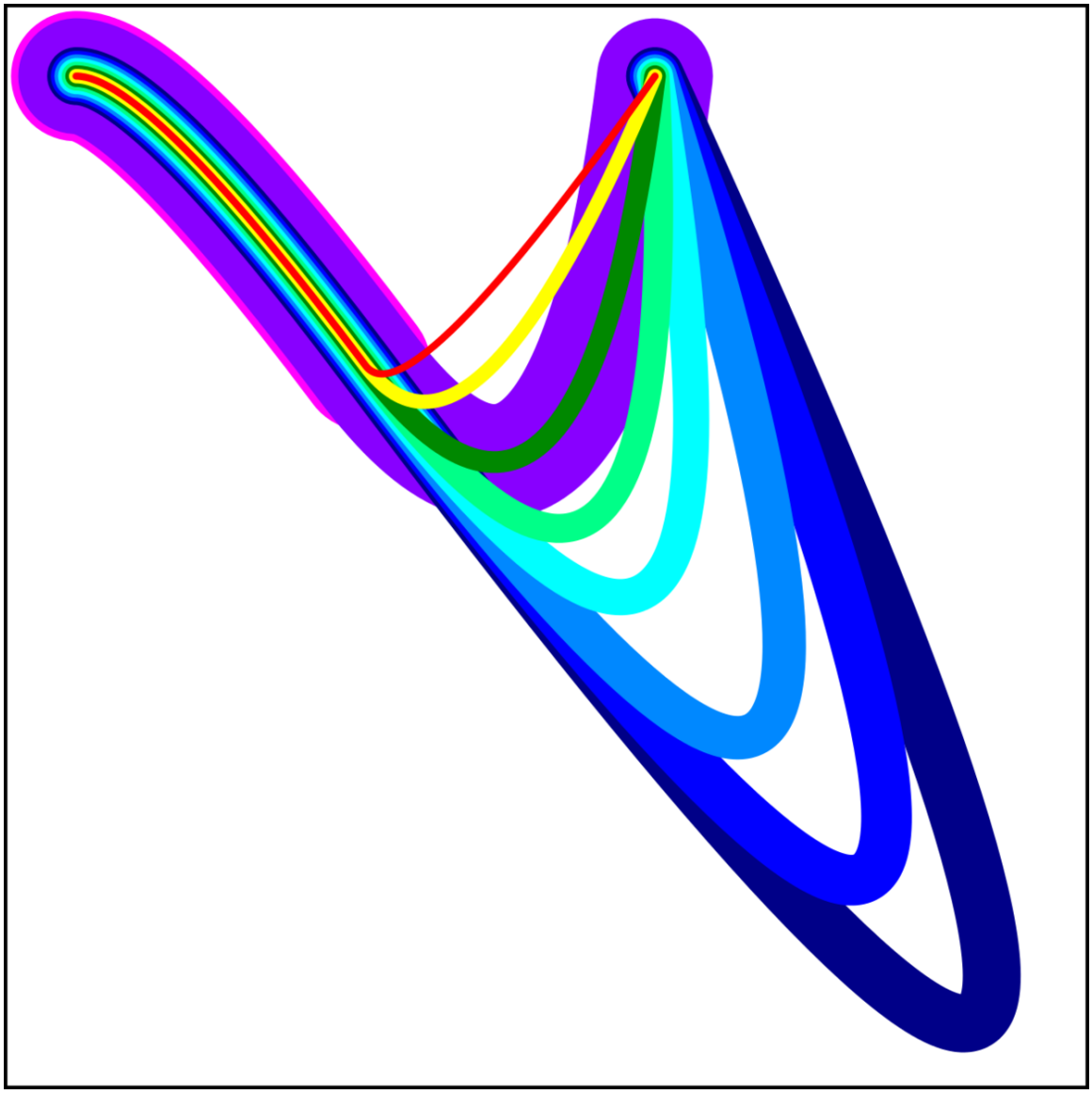


Abb. 75 Fortsetzungen eines quadratischen Pfades ohne Knicke, Kommando t

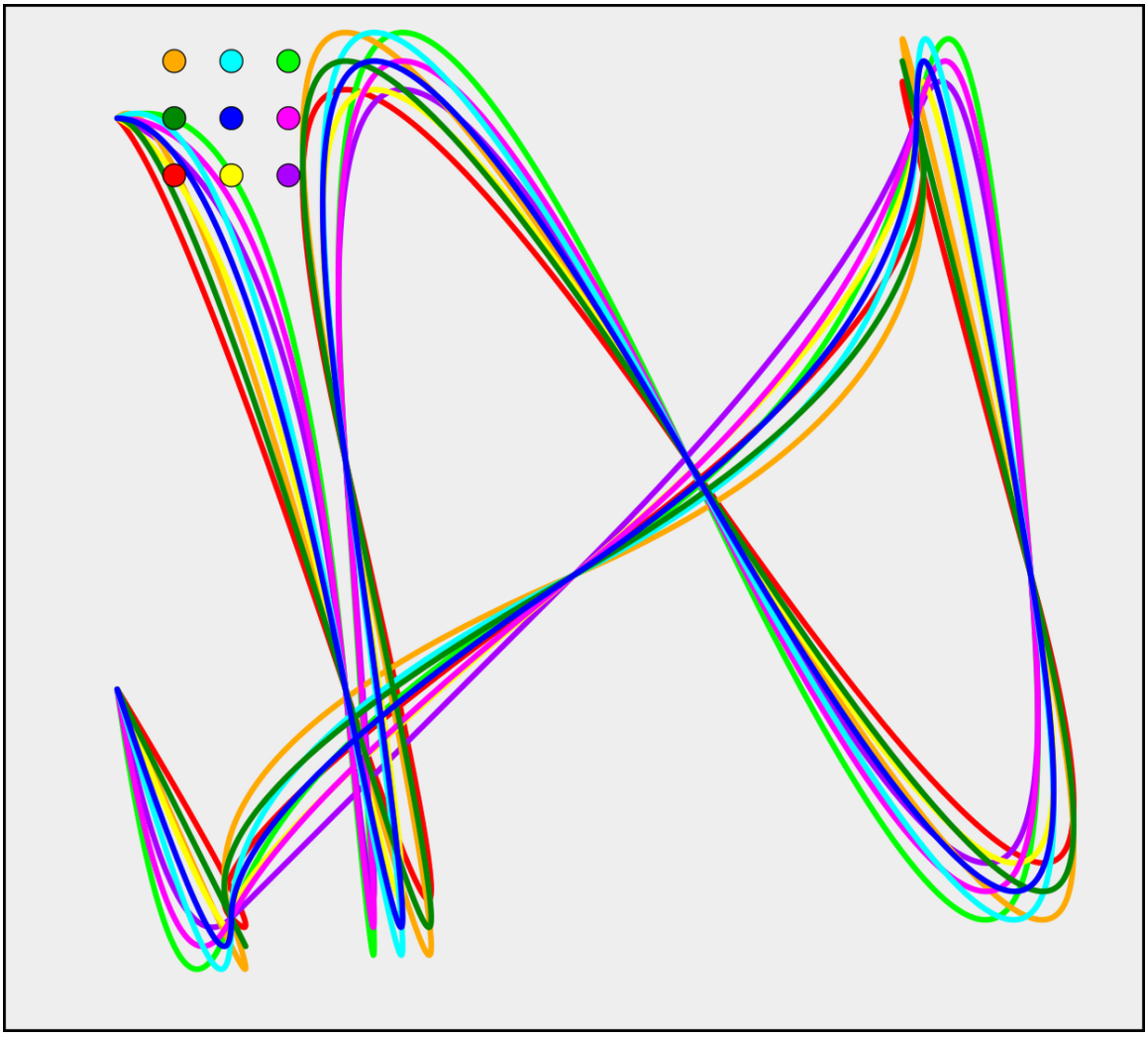


Abb. 76 Nichtlokalität von Kontrollpunkten in QT-Pfaden

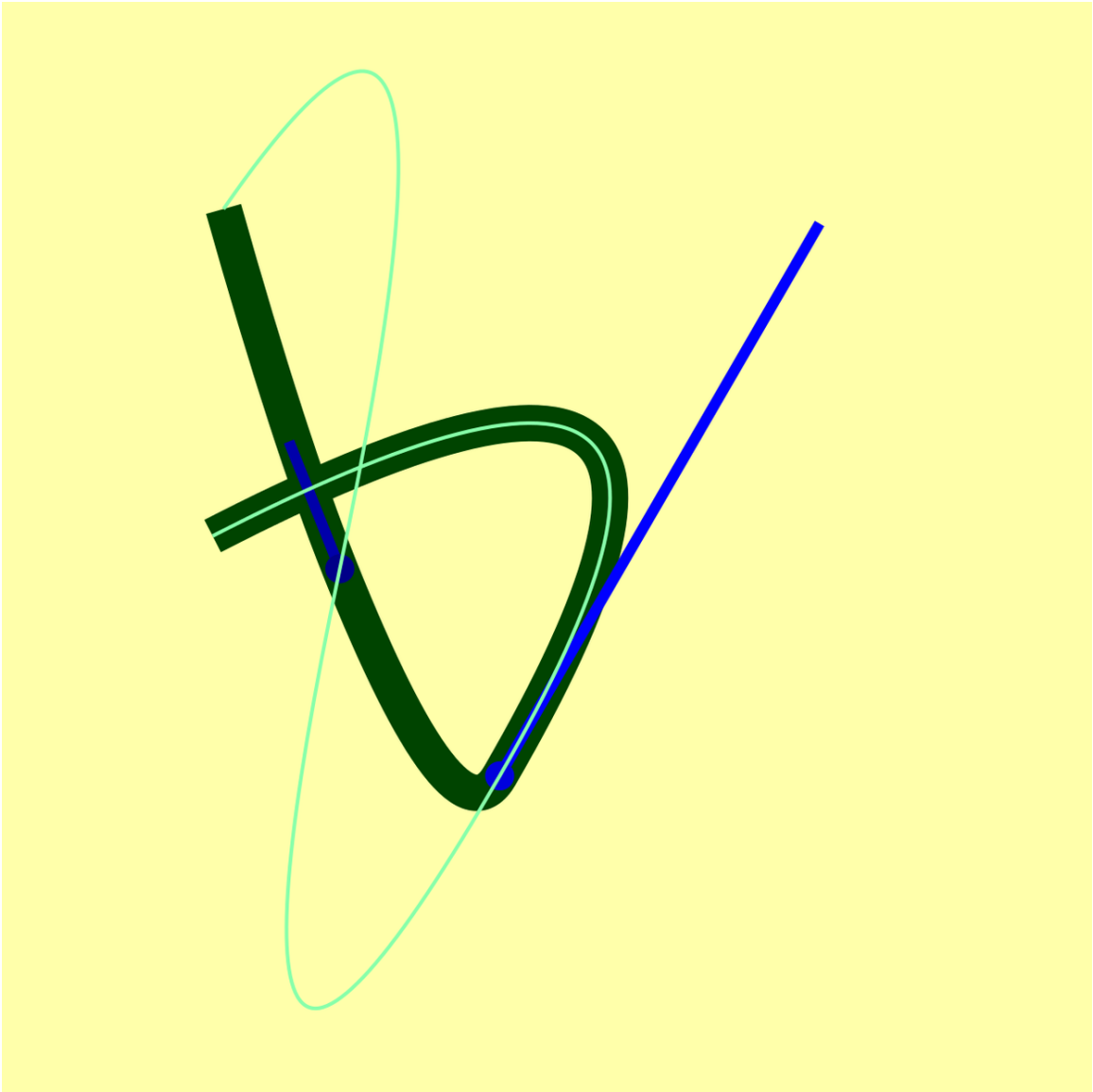


Abb. 77 Kurvensegmente mit quadratischem Pfad verbinden

Mittels 'Q', 'q', 'T', 't' werden quadratische Kurvensegmente angegeben. Der Kontrollpunkt gibt jeweils an, in welche Richtungen der Pfad am Anfangs- und Endpunkt geht. Die Verhältnisse der Entfernungen zwischen den Punkten haben Einfluss auf den sonstigen Kurvenverlauf.

Seien Anfangs- und Endpunkt P(0) und P(2) und der Kontrollpunkt P(1), sei der Laufparameter s aus [0,1], so gilt für eine quadratische Bézierkurve:

$k(s) = (1 - s)^2P(0) + 2s(1-s)P(1) + s^2P(2)$

Und für die Ableitungen nach s gilt folglich:

$dk(s)/ds = 2(s-1)\ P(0) + 2(1 -2s)\ P(1) + 2s\ P(2)$

oder in der Parametrisierung einer linearen Bézierkurve

dk(s)/ds = (1-s) (2P(1) - 2P(0)) + s (2P(2) - 2P(1))

mit der zweiten Ableitung

$d^2k(s)/ds^2$ = 2P(0) + 2P(2) - 4P(1).

Weil P(0) = k(s=0) der Anfangspunkt ist und P(2) = k(s=1) der Endpunkt, ist somit

dk(0)/ds = -2 P(0) + 2P(1)

dk(1)/ds = -2 P(1) + 2P(2)

also:

P(1) = P(2) - dk(1)/ds/2 = P(0) + dk(0)/ds/2

Mit den Kommandos 'T' und 't' erfolgt eine automatische Fortsetzung eines vorherigen quadratischen Pfadsegmentes, welche stetig differenzierbar ist, also keinen Knick an der Verbindungsstelle aufweist. Es gibt allerdings beliebig viele solcher Fortsetzungen, in SVG ist nur eine davon automatisch verfügbar, andere sind manuell zu berechnen und mit weiteren Kommandos 'Q' oder 'q' zu notieren.

Dazu wird wie angegeben die Ableitung am Endpunkt verwendet:

A = dk(1)/ds = -2 P(1) + 2P(2)

Die stetig differenzierbare Fortsetzung ist mit einer positiven Zahl a:

P(1, neu) = P(2, alt) + a A/2, beziehungsweise mit relativen Koordinaten p(1, neu) = a A/2

Das T-Kommando entspricht a = 1.

'alt' und 'neu' stehen hier für das alte Kurvenstück und das neue, also der gewünschten Fortsetzung.

- Beispiel: Fortsetzungen eines quadratischen Pfades ohne Knicke, Kommando T[12].
- Beispiel: Fortsetzungen eines quadratischen Pfades ohne Knicke, Kommando t[13].

Bei einem quadratischen Pfad, der mit dem Kommando T oder t fortgesetzt wird, ist die Wirkung eines Kontrollpunktes in dem Sinne nicht lokal, als die Änderung des Kontrollpunktes des anfänglichen Q oder q den gesamten Verlauf des Pfades ändert.

12 http://de.wikibooks.org/wiki/media%3ASVGpathQT01.svg
13 http://de.wikibooks.org/wiki/media%3ASVGpathQT02.svg

In folgendem Beispiel ist zu sehen, wie mit leicht variiertem Kontrollpunkt zu Beginn des Pfades der gesamte Pfad geändert wird. Der jeweilige Kontrollpunkt ist als Kreis dargestellt, mit der gleichen Farbe gefüllt wie der zugehörige Pfad gemalt ist.

Beispiel: Nichtlokalität von Kontrollpunkten in QT-Pfaden[14].

Daher eignen sich quadratische Bézierkurven eher schlecht, um andere Kurven zu approximieren oder zwischen einzelnen Datenpunkten zu interpolieren oder auch zu extrapolieren. Für solche Anwendungen sind kubische Bézierkurven viel besser geeignet, weil sie wegen des zusätzlichen Kontrollpunktes viel flexibler zu handhaben sind und Änderungen an Kontrollpunkten nur in der Nähe des Kontrollpunktes Auswirkungen haben.

Wie im Abschnitt über gerade Kurvensegmente schon angesprochen, kann in einigen Fällen ein quadratisches Pfadsegment verwendet werden, um zwei Kurvenfragmente ohne Knick miteinander zu verbinden, wenn die Richtungen der Ableitungen an den zu verbindenden Enden geeignet ausgerichtet sind.

Beispiel: Kurvensegmente mit quadratischem Pfad verbinden[15].

Verwendet wird obige Lösung des Problems, den Schnittpunkt zweier Geraden zu finden. Die Richtung der Geraden wird jeweils bestimmt durch die Richtung der Ableitungen an den zu verbindenden Enden, beziehungsweise dort die Richtungen vom Endpunkt zum zugehörigen Kontrollpunkt (oder Anfangspunkt, je nach Typ des Segmentes). Der quadratische Kurvenzug ist grün dargestellt. Die Richtungen der zufällig vorgegebenen Kurvensegmente sind blau dargestellt. Mit Kreisen wird jeweils dargestellt, wo der Kontrollpunkt ist (rosa) oder wo die jeweiligen Pfade beginnen. Hellgrün ist zum Vergleich der Kurvenzug mit automatisch bestimmten Kontrollpunkten bei gleichen Punkten angegeben.

14 http://de.wikibooks.org/wiki/media%3ASVGpathQT03.svg
15 http://de.wikibooks.org/wiki/media%3ASVGpathQ01.svg

7.3.5 Kubische Kurvenstücke: C, c, S, s

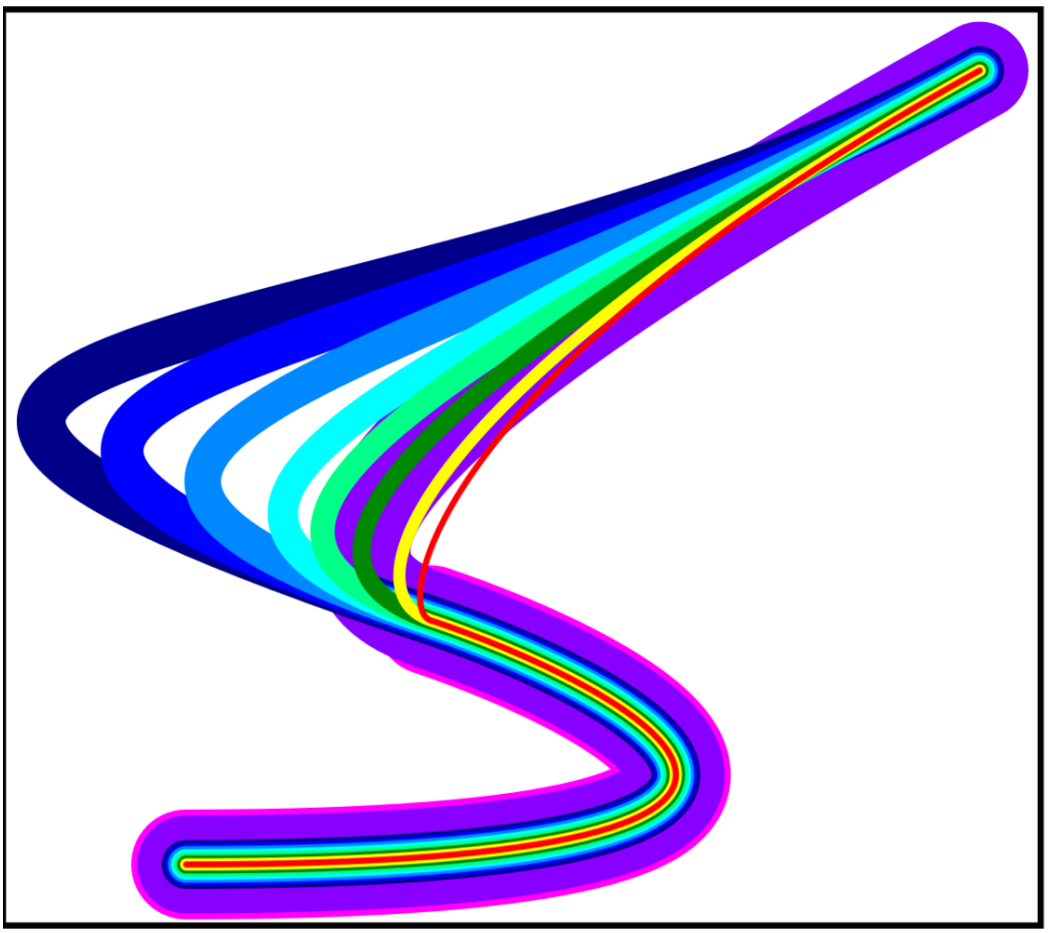


Abb. 78 Fortsetzungen eines kubischen Pfades ohne Knicke, Kommando S

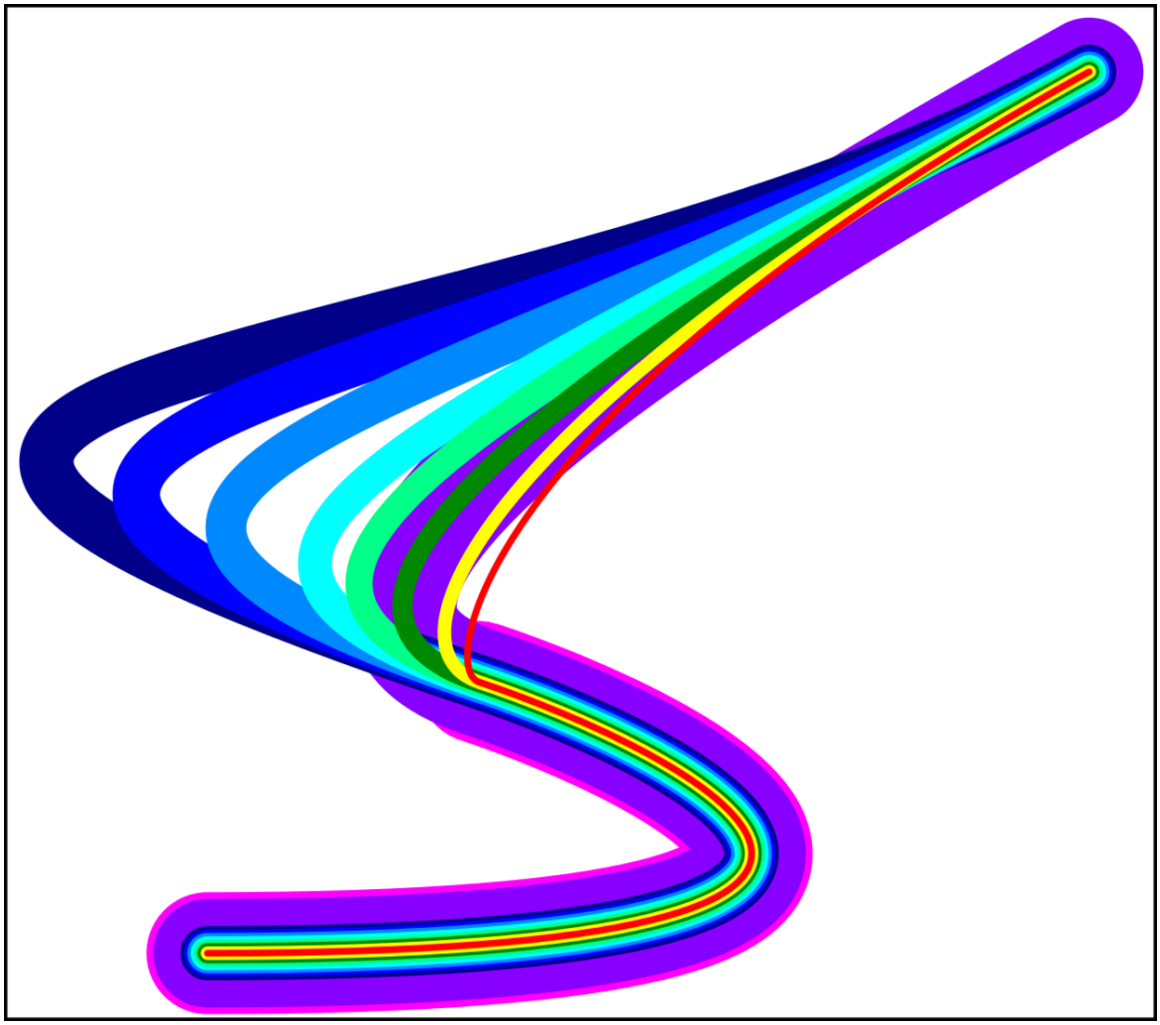


Abb. 79 Fortsetzungen eines kubischen Pfades ohne Knicke, Kommando s

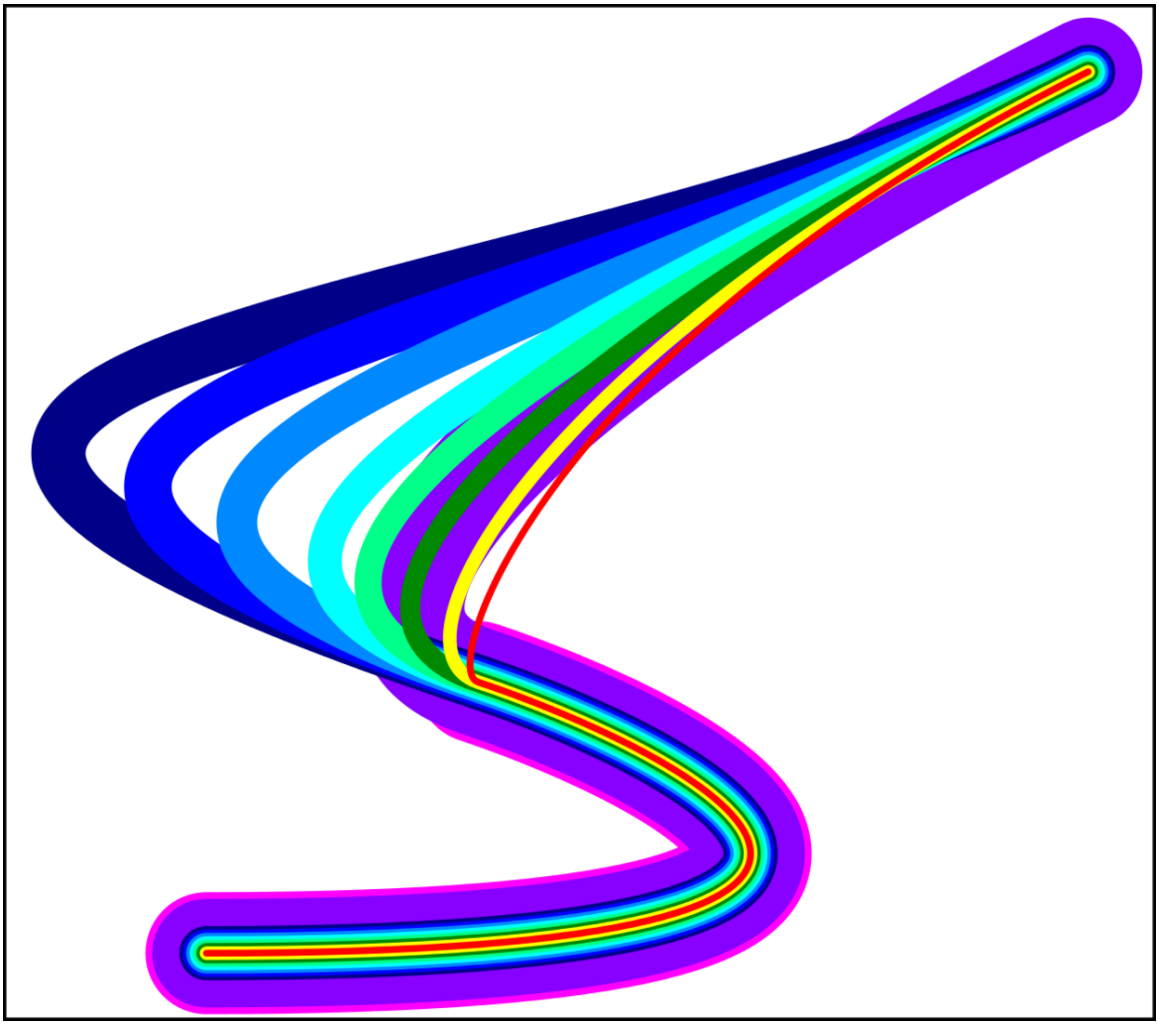


Abb. 80 Fortsetzungen eines kubischen Pfades ohne Knicke, vorne angehängt

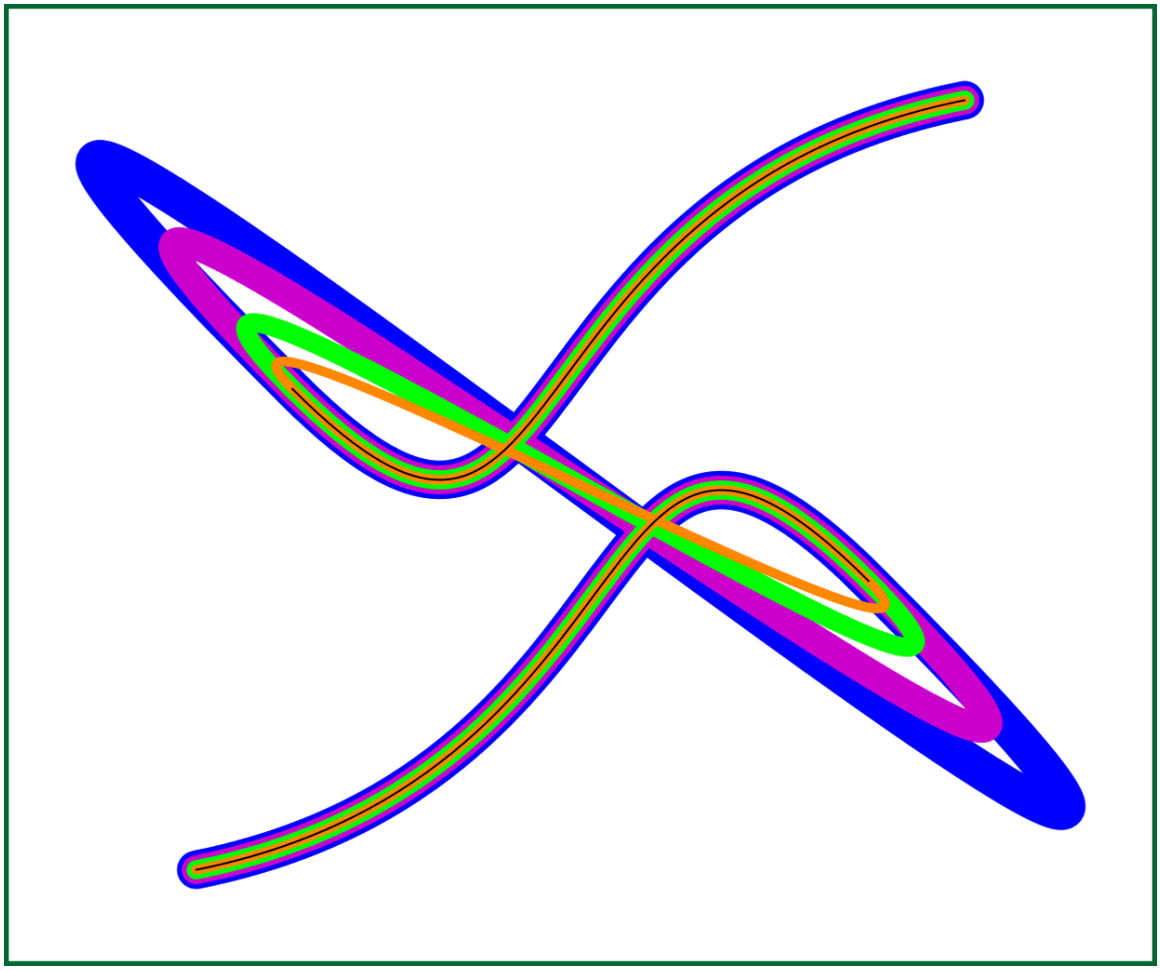


Abb. 81 Verbindung zweier kubischer Pfade ohne Knicke

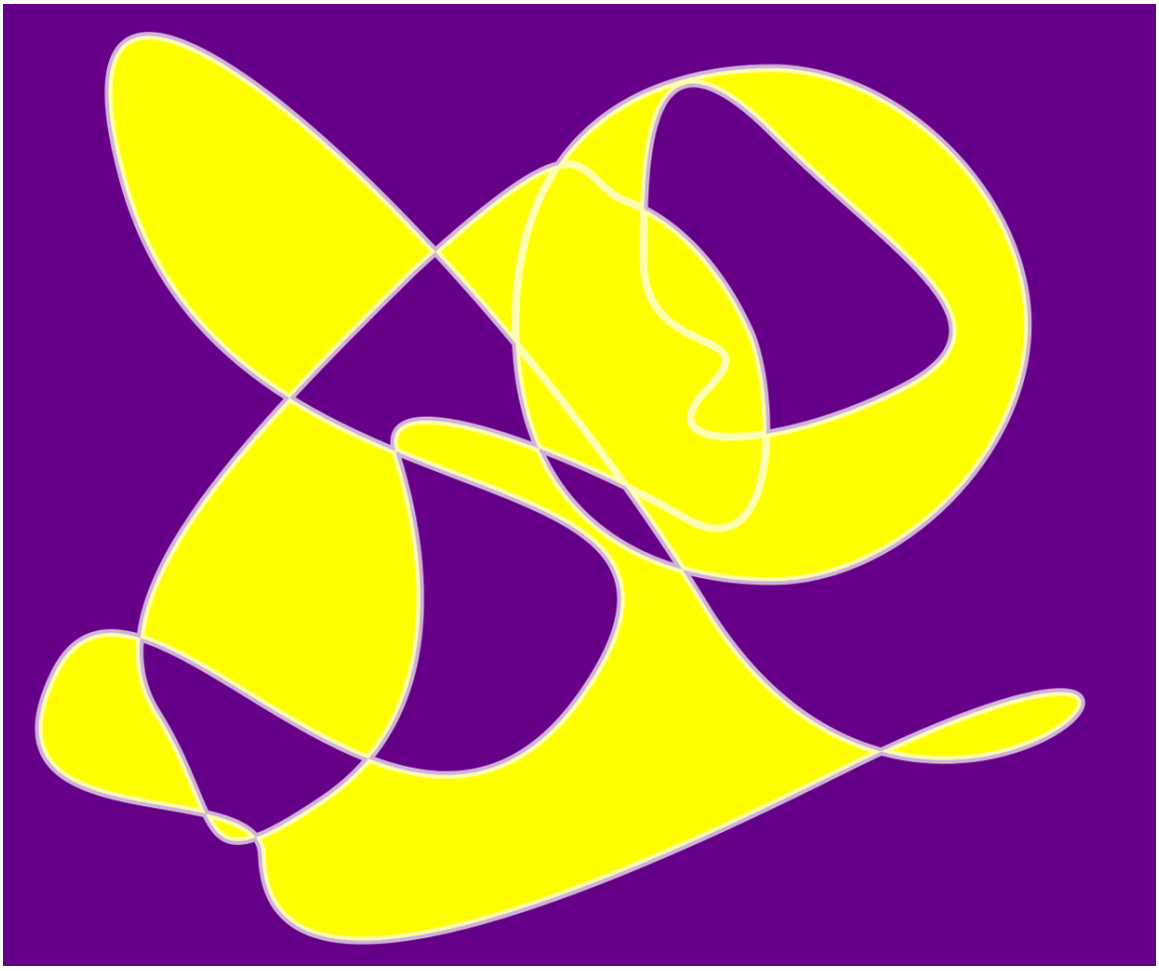


Abb. 82 'Sonne und Seuche'

Mittels 'C', 'c', 'S', 's' werden kubische Kurvensegmente angegeben. Der erste Kontrollpunkt gibt an, in welche Richtung der Pfad am Anfangspunkt geht. Der zweite Kontrollpunkt gibt an, aus welcher Richtung der Pfad am Endpunkt kommt. Die Verhältnisse der Entfernungen zwischen den Punkten haben Einfluss auf den sonstigen Kurvenverlauf.

Seien Anfangs- und Endpunkt P(0) und P(3) und der erste Kontrollpunkt P(1), der zweite P(2), sei der Laufparameter s aus [0,1], so gilt für eine kubische Bézierkurve:

$k(s) = (1 - s)^3P(0) + 3s(1-s)^2P(1) + 3s^2(1-s)P(2) + s^3P(3)$

Die Ableitungen ergeben sich daraus als:

$dk(s)/ds = -3(1 - s)^2P(0) + 3(1-4s +3s^2)P(1) + 3(2s-3s^2)P(2) + 3s^2P(3)$

oder in der Parametrisierung einer quadratischen Bézierkurve

$dk(s)/ds = (1 - s)^2 (3P(1) - 3P(0)) + 2s(1-s) (3P(2) - 3P(1)) + s^2(3P(3) - 3P(2))$

Die zweite Ableitung ist somit

$d^2k(s)/ds^2 = (1-s) (6P(0) - 12P(1) + 6P(2)) + s (6P(3) -12P(2)) + 6P(1))$

Speziell gilt an den Endpunkten:

$dk(0)/ds = -3P(0) + 3P(1)$

$dk(1)/ds = -3P(2) + 3P(3)$

oder entsprechend umgestellt:

$P(1) = P(0) + dk(0)/ds/3$

$P(2) = P(3) - dk(1)/ds/3$

Mit den Kommandos 'S' und 's' erfolgt eine automatische Fortsetzung eines vorherigen kubischen Pfadsegmentes, welche stetig differenzierbar ist, also keinen Knick an der Verbindungsstelle aufweist. Es gibt allerdings beliebig viele solcher Fortsetzungen, in SVG ist nur eine davon automatisch verfügbar, andere sind manuell zu berechnen und mit weiteren Kommandos 'C' oder 'c' zu notieren.

Dazu wird wie angegeben die Ableitung am Endpunkt verwendet:

$A = dk(1)/ds = -3P(2) + 3P(3)$

Die stetig differenzierbare Fortsetzung ist mit einer positiven Zahl a:

$P(1, neu) = P(3, alt) + a\ A/3$, beziehungsweise mit relativen Koordinaten $p(1, neu) = a\ A/3$

Das S-Kommando entspricht a = 1.

'alt' und 'neu' stehen hier für das alte Kurvenstück und das neue, also der gewünschten Fortsetzung. P(2, neu) und P(3, neu) können dann natürlich beliebig gewählt werden, um die Kurve fortzusetzen.

- Beispiel: Fortsetzungen eines kubischen Pfades ohne Knicke, Kommando S[16].
- Beispiel: Fortsetzungen eines kubischen Pfades ohne Knicke, Kommando s[17].

Umgekehrt kann es sinnvoll sein, den Pfad nicht hinten, sondern vorne um ein Segment zu verlängern. Dazu wird die Ableitungsformel für den Anfangspunkt verwendet, also:

$A = dk(0)/ds = -3P(0) + 3P(1)$

16 http://de.wikibooks.org/wiki/media%3ASVGpathCS01.svg
17 http://de.wikibooks.org/wiki/media%3ASVGpathCS02.svg

Die stetig differenzierbare Fortsetzung ist mit einer positiven Zahl b:

P(2, neu) = P(0, alt) -b A/3, beziehungsweise mit relativen Koordinaten p(2, neu) = -b A/3

P(0, neu) und P(1, neu) können beliebig gewählt werden, die Daten des neuen Kurvensegmentes sind den alten voranzustellen.

- Beispiel: Fortsetzungen eines kubischen Pfades ohne Knicke, vorne angehängt[18].

Mit einem kubischen Pfadsegment können anders als mit einem quadratischen Pfadsegment immer zwei Pfade ohne Knick miteinander verbunden werden, sofern aus den zwei Pfaden jeweils die Richtungen ermittelt werden können. Meistens es reicht dazu, dass wie angegeben die Ableitung bestimmbar ist und diese nicht 0 ist, sonst ergibt sich die Richtung entsprechend aus dem anderen Kontrollpunkt, beziehungsweise aus dem anderen Punkt des vorherigen oder des folgenden Pfadsegmentes. Da eine kubische Kurve zwei Kontrollpunkte hat, können diese also unabhängig voneinander in der Richtung des vorherigen beziehungsweise des folgenden Pfades angegeben werden, so dass dadurch immer zwei Pfade mit einem kubischen Pfadsegment ohne Knick miteinander verbunden werden können. Dabei wird vorgegangen, wie bereits beschrieben für die Fortsetzung eines Pfades ohne Knick in den vorherigen Absätzen. Das legt dann für das einzupassende Element bis auf die Parameter a und b die beiden Kontrollpunkte fest. Anfangs- und Endpunkt sind ja ohnehin bereits durch den vorherigen und den folgenden Pfad festgelegt.

- Beispiel: Verbindung zweier kubischer Pfade ohne Knicke[19].

Gilt es, einen längeren Pfad oder Unterpfad so zu schließen, dass kein Knick entsteht, sind die Formeln für die Ableitungen ebenfalls anwendbar.

Mit den Punkten, beziehungsweise Kontrollpunkten P(i, 0) bis Pn(i, n) für Kurvensegmente 0 bis n ist die Kurve dann offenbar geschlossen, wenn gilt:

P(0,0) = P(3,n)

Und wenn an der Verbindungsstelle von Anfang und Ende kein Knick auftreten soll, ergibt sich aus den Formeln für die Ableitungen

P(2, n) = (1+v) P(0, 0) - v P(1, 0) mit einer positiven Zahl v.

Hier ist v = 1 auch wieder der Fall, der einer automatischen Fortsetzung analog zum Kommando S entspricht.

Alternativ kann auch der Kontrollpunkt zu Beginn des Pfades passend gewählt werden, durch Umstellung ergibt sich:

P(1, 0) = (1+w) P(3, n) - w P(2, n) mit einer positiven Zahl w.

18 http://de.wikibooks.org/wiki/media%3ASVGpathCS03.svg
19 http://de.wikibooks.org/wiki/media%3ASVGpathC01.svg

- Beispiel: 'Sonne und Seuche'[20].

Abstrakte Form und Wort/Dialekt-Spielerei. Geschlossene kubische Kurvenzüge ohne Knicke in mehreren, sich überlagernden Unterpfaden (einschließlich einfacher Animationen; Animationen werden dann in einem späteren Kapitel behandelt und müssen an dieser Stelle nicht nachvollzogen werden).

Der Titel ist in mehrfacher Hinsicht doppeldeutig (sonne und seuche - verschiedene Aussprachen oder Dialektformen des Wortes 'solche') und bezieht sich auf die Umlaufrichtungen der Pfade wie auch die daraus resultierenden Typen von Löchern, wenn sich die Pfade überschneiden, was mit einer Animation der Füllregel verdeutlicht wird. Es bezieht sich auch auf die verschiedenen Möglichkeiten, Pfade ohne Knick zu schließen.

Auf Füllregeln wird in einem späteren Kapitel noch eingegangen.

Der Titel kann natürlich auch auf die abstrakten Formen selbst bezogen werden oder eben auf verschiedene Kunstauffassungen, die nebeneinander koexistieren können wie auch Aussprachen von Wörtern oder allgemeiner Dialekte oder die Interpretationen von Kunstwerken. Die im Element ***desc*** vorhandene Beschreibung mag weitere Möglichkeiten zur Deutung eröffnen...

20 `http://de.wikibooks.org/wiki/media%3ASVGsonneUndSeuche.svg`

7.3.6 Elliptische Bögen: A, a

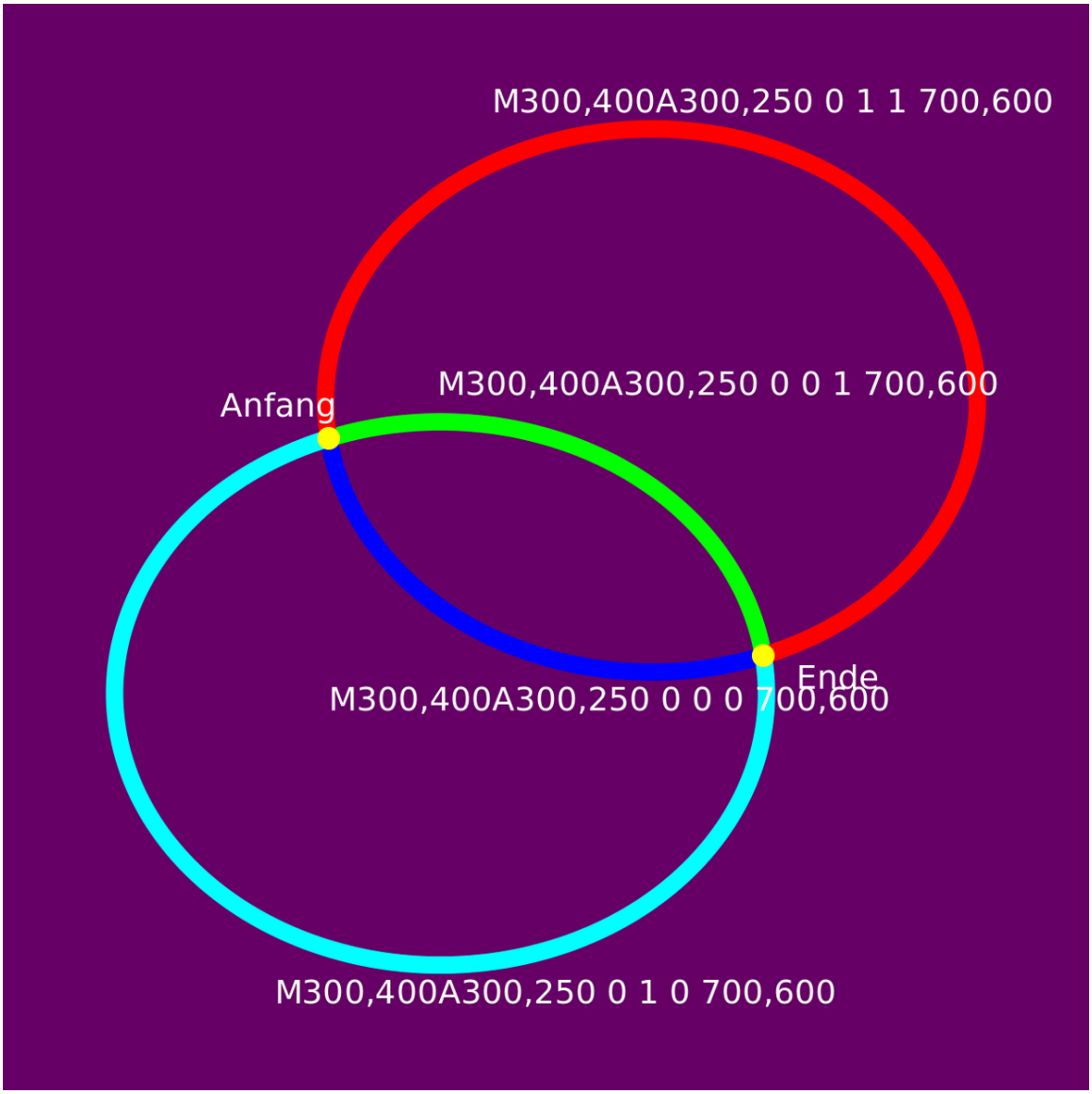


Abb. 83 Pfadkommando A

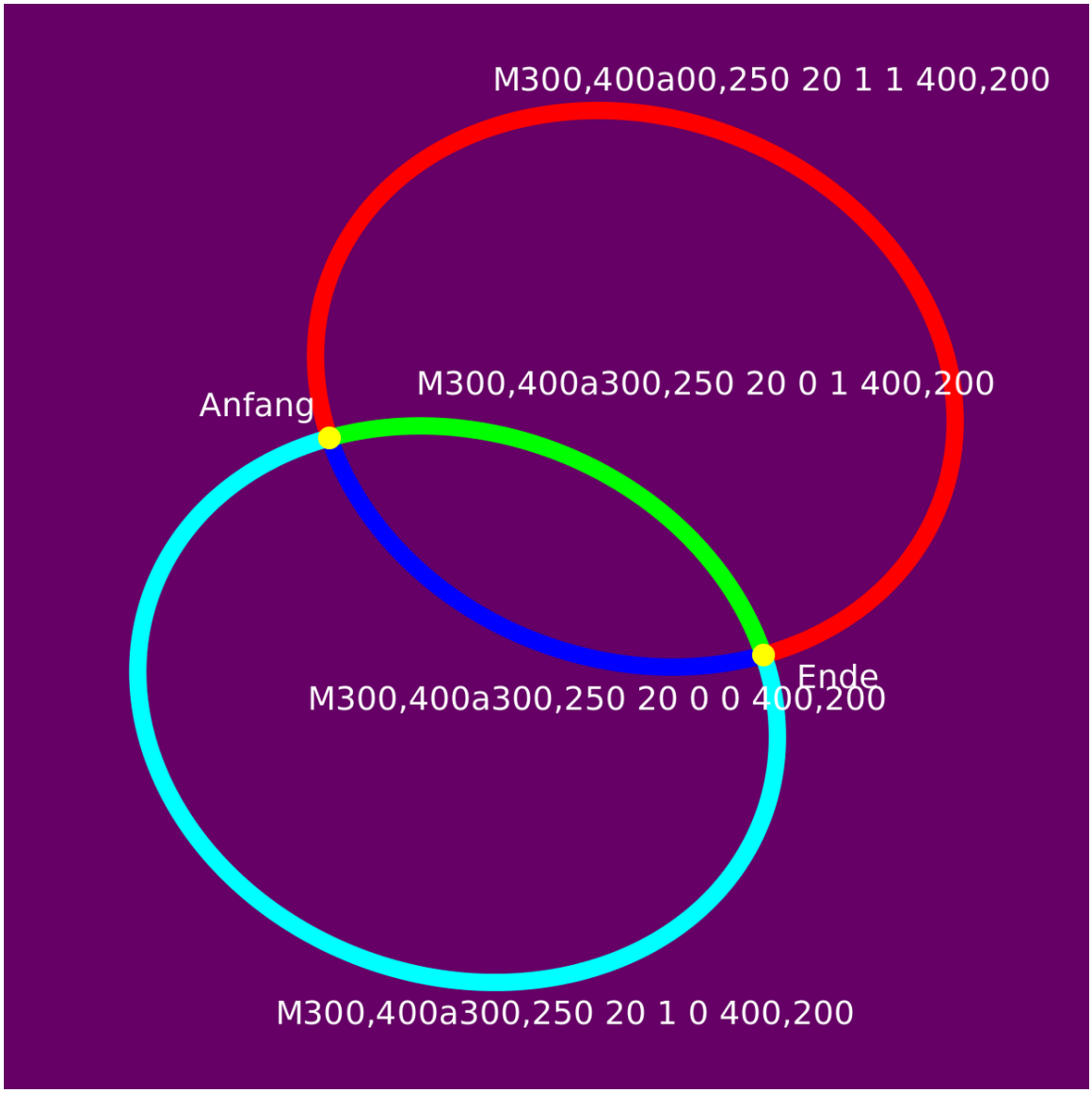


Abb. 84 Pfadkommando a

Mittels 'A' oder 'a' werden elliptische Bögen angegeben. Die Kommandos A und a dürfen nicht in den tiny-Versionen verwendet werden, für diese wären die dann zum Beispiel durch einen Kurvenzug mit kubischen Pfadsegmenten anzunähern.

Beispiele zur Verwendung und zur Bedeutung der Parameter von 'A', beziehungsweise 'a':

- Pfadkommando A[21].
- Pfadkommando a[22].

21 http://de.wikibooks.org/wiki/media%3ASVGpathA01.svg
22 http://de.wikibooks.org/wiki/media%3ASVGpathA02.svg

Bedeutung der Parameter der Pfadkommandos 'A' und 'a':

A rx ry x-axis-rotation large-arc-flag sweep-flag x y

Der elliptische Bogen startet am aktuellen Punkt und endet bei (x,y), bei a entsprechend relativ zum Anfangspunkt verschoben um(x,y).

rx und ry sind die halben Hauptachsen der Ellipse.

x-axis-rotation bezeichnet den Winkel in Grad, um den die Halbachse rx aus der x-Richtung gedreht wird.

In der Regel gibt es dann noch vier Möglichkeiten, Anfangs- und Endpunkt mit einem elliptischen Bogen zu verbinden. Mit large-arc-flag '1' wird von zwei Bögen jener mit der größeren Bogenlänge gewählt, für '0' der mit der kleineren.

Mit sweep-flag wird die Zeichenrichtung des Bogens gewählt.

Mit '1' wird mit positivem Winkel gedreht (In SVG ist der Winkel zwischen der x-Richtung und der y-Richtung +90 Grad, entsprechend der zwischen der x-Richtung und der negativen y-Richtung -90Grad).

Mit '0' wird mit negativem Winkel gedreht.

Für elliptische Bögen gibt es einige häufig verwendete Parametrisierungen. Die in SVG verwendete Endpunktnotation ist nur praktisch für Pfade, wie sie im Attribut *d* auftreten. Eine Umrechnung von den üblichen Parametrisierungen in die Endpunktnotation und umgedreht ist im Anhang der Spezifikation verfügbar.

Eine häufige Parametrisierung ist die Zentraldarstellung. Die sieht wie folgt aus:

$e(s) = D(\varphi)\ (r_X \cos(s), r_Y \sin(s)) + c$

Dabei ist $D(\varphi)$ eine Drehung um den Winkel φ, r_X und r_X sind die Hauptachsen der Ellipse und $c = (c_X, c_Y)$ der Mittelpunkt der Ellipse. Der Parameter s beginnt beim Startwinkel θ und läuft um den Winkel δ weiter.

Als Ableitung ergibt sich offenbar:

$de(s)/ds = D(\varphi)\ (-r_X \sin(s), r_Y \cos(s))$

Mittels einer Transformation kann aus e(s) das jeweilige Pfadkommando bestimmt werden.

Für elliptische Bögen gibt es auch Darstellungen in Polarkoordinaten (r, θ=s), etwa

$r(s) = p/(1 + e \cos(s))$

und die Ableitung:

$dr(s)/ds = - p\ e \sin(s) / (1 + e \cos(s))^2$

Alternativ kann auch einfach durch Verschieben aus der Zentraldarstellung eine Darstellung gewonnen werden:

$r(s) = r_X(1 - e \cos(s))$

mit der Ableitung:

$dr(s)/ds = r_X\ e \sin(s)$

wobei für SVG letztlich ohnehin wieder auf kartesische Koordinaten umgerechnet werden muss:

$e(s) = r\ D(\varphi)\ (\cos(s), \sin(s)) + f$

und die Ableitung

$de(s)/ds = r\ D(\varphi)\ (-\sin(s), \cos(s)) + dr/ds * D(\varphi)\ (\cos(s), \sin(s))$

Dabei ist $r_X = p/(1-e^2)$

$r_Y = p/(1-e^2)^{1/2}$

Oder

$p = r_Y^2/r_X$

$e = (1 - (r_Y/r_X)^2\ )^{1/2}$

In SVG wird die Endpunktnotation verwendet. Um von der Zentraldarstellung in die Endpunktnotation und umgekehrt umzurechnen, dient folgende Betrachtung:

Seien x_1,y_1 die absoluten Koordinaten der aktuellen Pfadposition und x_2,y_2 die der Endposition. Seien r_X und r_Y die halbe große und kleine Hauptachse der Ellipse ('Radien'). Sei φ der Winkel von der x-Achse des aktuellen Koordinatensystems zu r_X. Sei f_A der Parameter für den großen Bogen, 0 falls der Bogen nicht mehr als 180 Grad aufspannt oder 1, wenn der Bogen mehr als 180 Grad

aufspannt. Sei f_S der Parameter, welcher die Drehrichtung festlegt, 1 für größer werdende Winkel im Drehsinn von SVG und 0 sonst.

Folgende Regeln gelten für kritische Werte:

- Sind die Endpunkte identisch, wird kein elliptischer Bogen dargestellt.
- Wenn mindestens eine der Hauptachsen 0 ist, wird der Ellipsenbogen als gerade Linie vom Anfangs- zum Endpunkt angenommen.
- Es werden die Beträge der angegebenen Hauptachsen verwendet.
- Wenn es keine Lösung zu den angegebenen Hauptachsen und dem Winkel gibt, so werden die Hauptachsen gleichmäßig hochskaliert, bis es eine Lösung gibt.
- φ wird Modulo 360 Grad genommen.
- Haben f_S oder f_A einen anderen Wert als 0, so wird 1 angenommen.

In der Zentraldarstellung sei s_1 der Anfangswinkel und s_2 der Endwinkel und ds die Differenz dazwischen.

Umrechnung von der Zentraldarstellung in die Endpunktdarstellung

$(x_1,y_1) = D(\varphi) (r_X \cos(s_1), r_Y \sin(s_1)) + c$

$(x_2,y_2) = D(\varphi) (r_X \cos(s_2), r_Y \sin(s_2)) + c$

f_A ist 1 falls der Betrag von ds größer als 180 Grad ist, sonst 0.

f_S ist 1 falls ds größer als 0 ist, sonst 0.

Umrechnung von der Endpunktdarstellung in die Zentraldarstellung

Zunächst werden Hilfsgrößen berechnet.

$(x_1',y_1') = D(-\varphi) ((x_1 - x_2)/2,(y_1 - y_2)/2)$

$L = x_1'^2/r_X^2 + y_1'^2/r_y^2$

Falls L kleiner oder gleich 1 ist, bleiben r_X und r_Y unkorrigiert, ansonsten werden sie mit der Wurzel aus L multipliziert, damit ein Ellipsenbogen möglich ist.

$R = r_X^2 y_1'^2 - r_Y^2 x_1'^2$

$(c_X',c_Y') = \pm ((r_X^2 r_Y^2 - R) / R) * (r_X y_1'/r_Y , -r_Y x_1'/r_X)$

Das Vorzeichen ist +, falls f_S ungleich f_A ist und - sonst.

$c = D(-\varphi) (c_X',c_Y') + ((x_1 + x_2)/2, (y_1 + y_2)/2)$

Nun fehlt noch der Anfangswinkel und der Differenzwinkel.

Zur Berechnung sollte zunächst bekannt sein, wie ein Winkel zwischen zwei Vektoren berechnet wird. Allgemein ist der Winkel zwischen zwei Vektoren u und v (keiner von beiden 0) gleich $\pm$ arccos u*v/(|u|v|), wobei das Vorzeichen das gleiche ist wie das von $u_X v_Y - u_Y v_X$

s_1 ist der Winkel zwischen (1,0) und $((x_1' - c_X)'/r_X, (y_1' - c_Y)'/r_Y)$

ds ist der Winkel zwischen $((x_1' - c_X)'/r_X, (y_1' - c_Y)'/r_Y)$ und $((-x_1' - c_X)'/r_X, (-y_1' - c_Y)'/r_Y)$ modulo 360 Grad.

Dabei wird s_1 auf den Bereich zwischen -360 Grad und 360 Grad begrenzt, so dass falls f_S=0, dann ist ds kleiner als 0, sonst größer oder gleich.

7.3.7 Konversion zwischen Bézierkurven

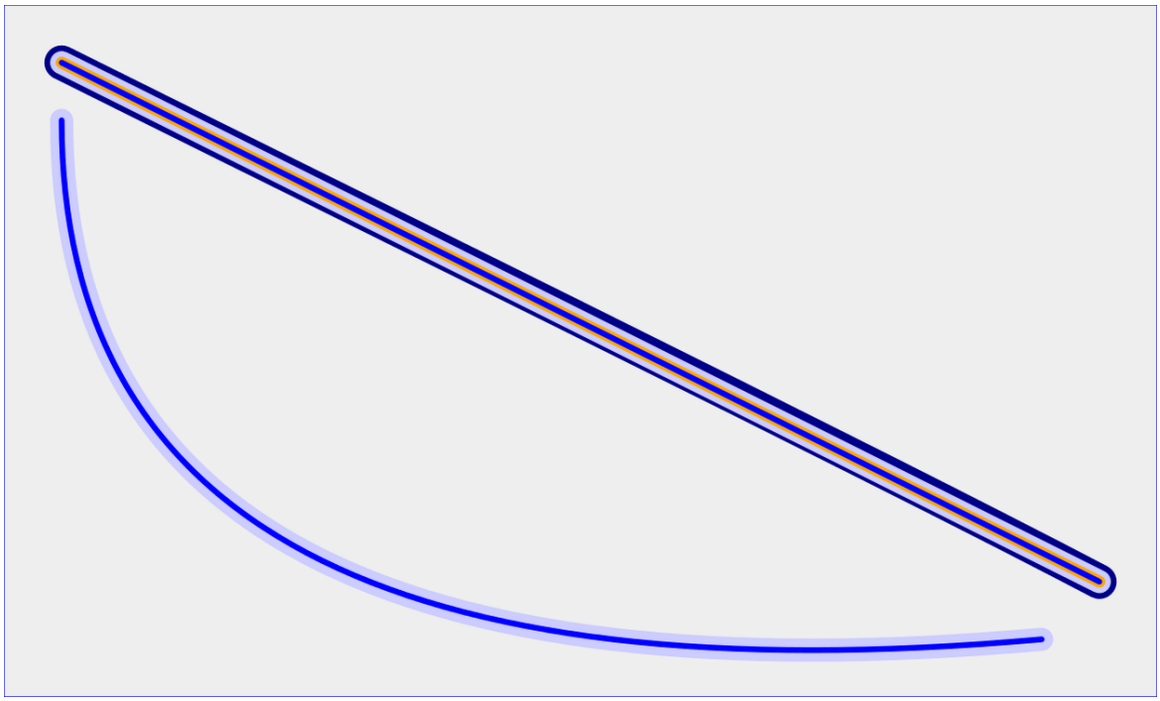


Abb. 85 Pfadkonversionen

Eine Konversion von linearen und quadratischen Kurven in kubische Kurven kann ganz praktisch sein. Dies ist zum Beispiel für Animationen der Fall, weil es nicht zulässig ist, Kommandos zwischen zwei Animationsschritten zu wechseln. Auch sonst kann es zu Vereinfachungen in der Handhabung führen, wenn nur kubische Bézierkurven verwendet werden. Erneut sind oben genannte Parametrisierungen und die Formel für die Ableitungen der Schüssel für die Konversion. Anfangs- und Endpunkt gleich bleiben und ebenfalls die Ableitung dort. Dann ergibt sich:

Konversion L nach Q oder C mit i, f Anfangs- und Endpunkt der Kurve:

M i L f = M i Q (i+f)/2, f

M i C (2i+f)/3, (i+2f)/3, f

Offenbar ist sogar jede quadratische Kurve mit einer Zahl a zwischen 0 und 1 und folgender Form eine Lösung des Problems:

M i L f = M i Q ((1-a)i + a f), f

Entsprechend mit zwei Zahlen a, b und 0 <= a <= b <= 1

M i C ((1-a)i + a f), ((1-b)i + b f), f

Konversion Q nach C (p Kontrollpunkt):

M i Q p, f = M i C (2p+i)/3, (2p+f)/3, f

Auch hier gibt es weitere Möglichkeiten, die Berechnung bleibt dem Leser zur Übung überlassen.

Beispiel: Pfadkonversionen[23].

Konversion von einem geraden Pfadsegement zu quadratischen und kubischen Pfadsegmenten und von einem quadratischen zu einem kubischen Pfadsegment.

23 http://de.wikibooks.org/wiki/media%3ASVGPfadkonversion01.svg

7.4 Weglänge

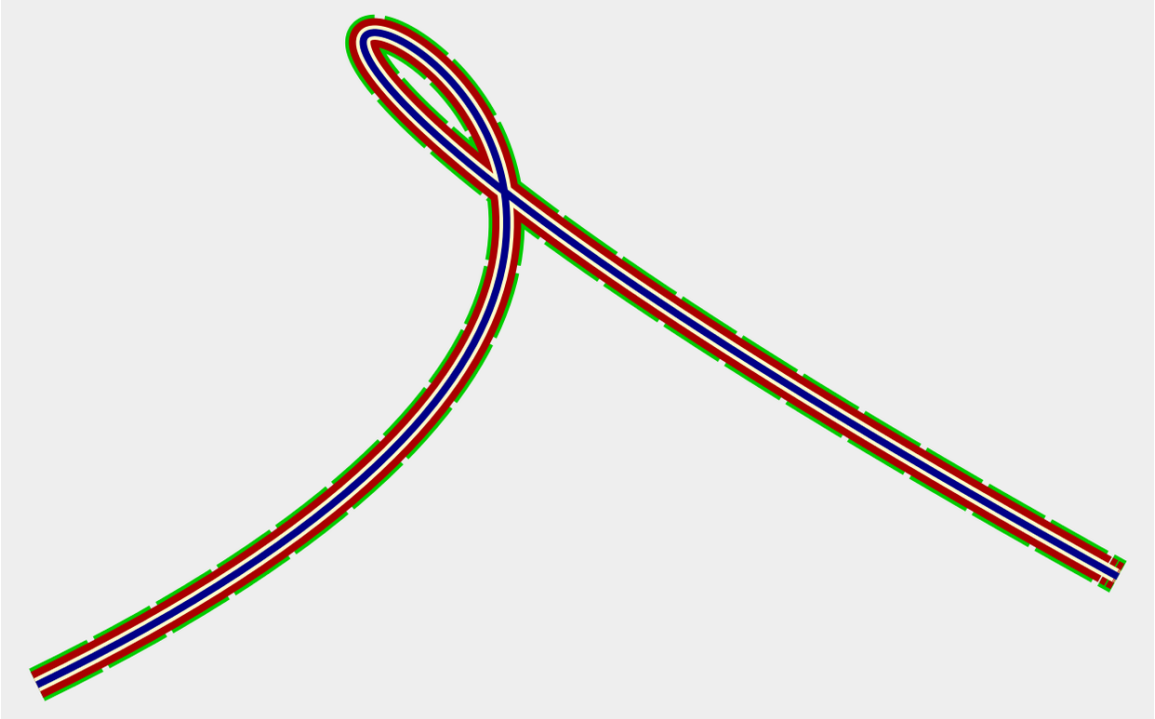


Abb. 86 Weglänge einer Kurve abschätzen durch Verwendung von *stroke-dasharray* als einfachem Maßstab

Nützlich für verschiedene Anwendungen ist die Kenntnis der Weglänge einer Kurve. Dies kann zum Beispiel hilfreich sein, um die Eigenschaft *stroke-dasharray* gezielt einzusetzen, einschließlich des Zeichnens eines Pfades. Auch kann damit die Bewegung eines Objektes entlang des Pfades mit ***animateMotion*** entlang eines Pfades detailliert geplant werden. Ebenfalls interessant kann die Kenntnis der Weglänge werden, wenn Text entlang eines Pfades angeordnet werden soll.

Offenbar ist die Länge eines Pfades die Summe der Längen der Fragmente. Wird mittels 'M' oder 'm' zu einem neuen Unterpfad gesprungen, so trägt dieser Sprung nicht zur Pfadlänge bei.

Ist k(s) wie oben angegeben eine Parametrisierung eines Pfadsegmentes mit s von 0 bis 1, so ist die allgemeine Definition der Weglänge eines zweidimensionalen Pfades mit s von 0 bis f (von 0 bis 1), kx und ky jeweils die x- und y-Komponente des Pfades:

$$w(f) := {}_0\int^f ds\ ((dkx/ds)^2 + (dky/ds)^2)^{1/2}$$

Bei einer geraden Kurvensegment ist das Ergebnis offenbar trivial, die gesamte Weglänge ist der Betrag des Differenzvektors, kurz der Euklidische Abstand oder als Formel für den Anteil f:

$$w(f) = f * ((kx(0)-kx(1))^2 + (ky(0)- ky(1))^2)^{1/2}$$

Bei einer quadratischen Kurvensegment gibt es eine nicht triviale analytische Lösung des Integrals, eine Formel, welche eine Wurzelfunktion und den Logarithmus enthält.

Bei kubischen Kurven und Ellipsenbögen hingegen ergibt sich eine Lösung mit elliptischen Integralen, die normalerweise dann eher numerisch zu lösen sind.

Mittels eines Darstellungsprogrammes des eigenen Vertrauens und der Eigenschaft *stroke-dasharray* kann die von diesem Darstellungsprogramm berechnete Länge für eine Kurve ebenfalls geschätzt werden, dazu wird *stroke-dasharray* als Maßstab verwendet. Liegt die Kurvenlänge zum Beispiel in der Größenordnung 1000, so kann angegeben werden: *stroke-dasharray*="90,10". Daraus ergibt sich dann eine Unterteilung in Abschnitten von 100. Ist die resultierende Abschätzung zum Beispiel 'zwischen 1250 und 1300', so kann der Maßstab verfeinert werden: *stroke-dasharray*="1249,1,9,1,9,1,9,1,9,1,9,1". Und so weiter, bis ein ausreichend genaues Ergebnis erreicht ist.

Praktische Probleme bei der Methode können auftreten, wenn der Pfad aus mehreren Unterpfaden besteht, weil es da je nach Darstellungsprogramm Implementierungsmängel geben kann (statt dass *stroke-dasharray* durchläuft, wird es mit jedem Unterpfad neu begonnen). Deswegen kann es notwendig sein, die Länge eines jeden Unterpfades einzeln zu bestimmen.

Beispiel: Weglänge einer Kurve abschätzen durch Verwendung von *stroke-dasharray* als einfachem Maßstab[24].

Es sind drei Maßstäbe übereinandergelegt. Um den letzten, hellgelben abzulesen, muss gegebenenfalls das Pfadende rechts unten im Darstellungsprogramm vergrößert werden.

24 `http://de.wikibooks.org/wiki/media%3ASVGWeglaenge01.svg`

7.5 Kurven teilen - Algorithmus nach De Casteljau

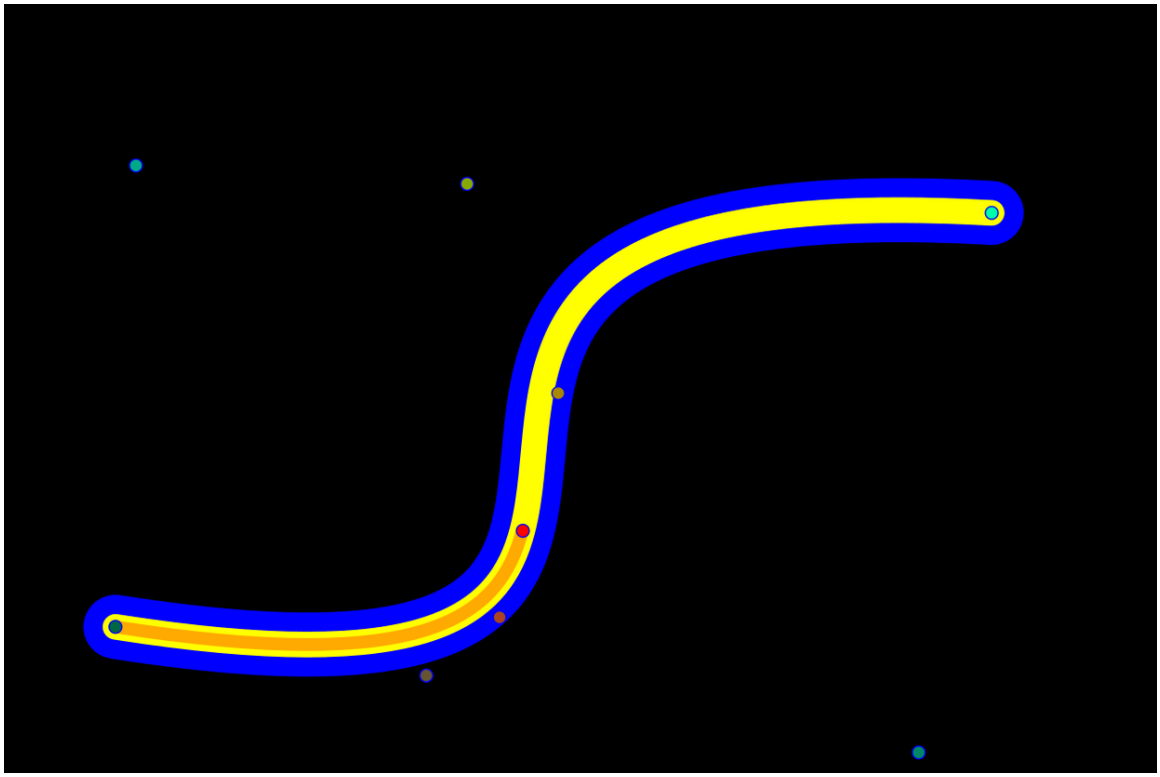


Abb. 87 Algorithmus nach De Casteljau

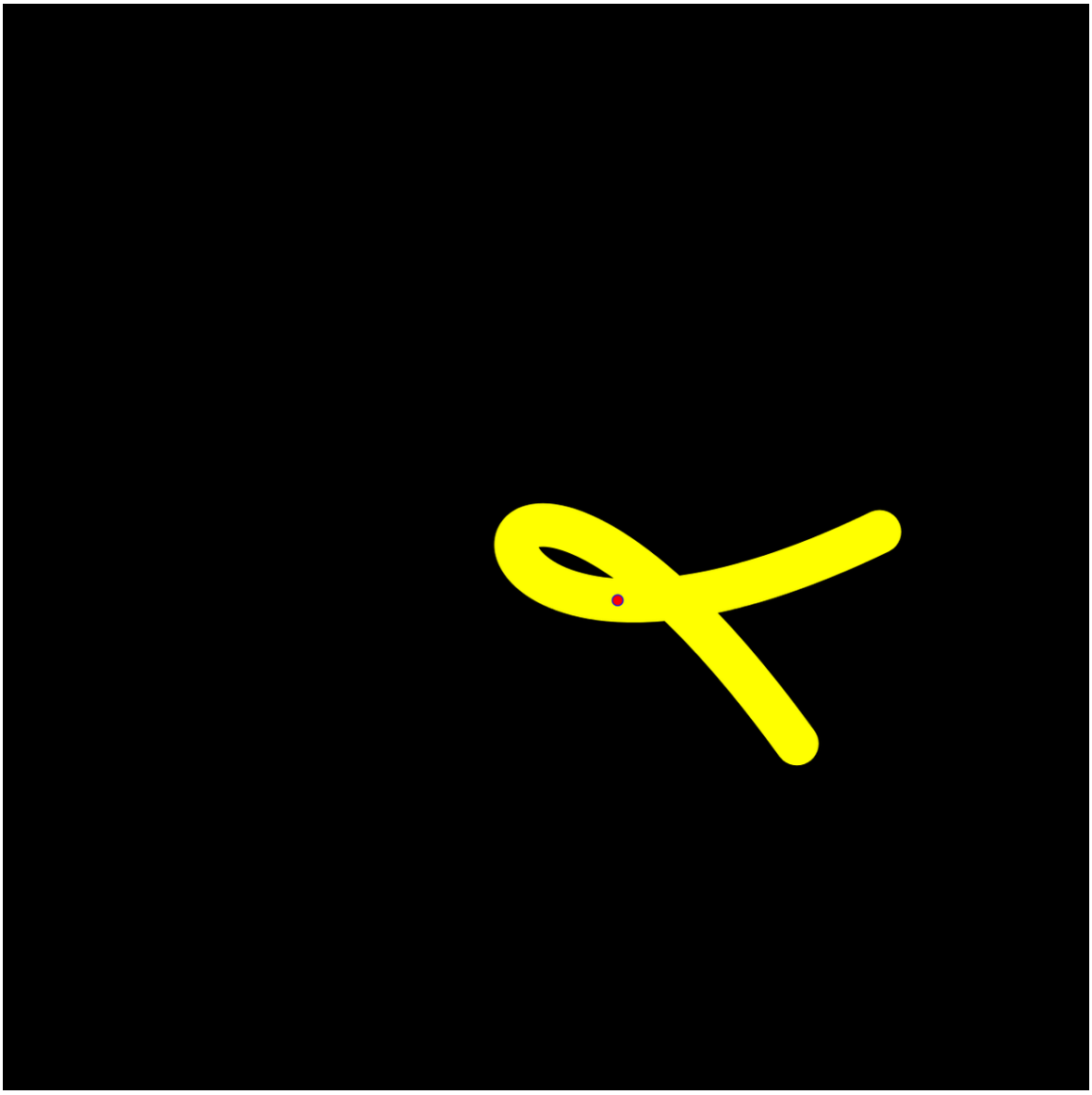


Abb. 88 Algorithmus nach De Casteljau mit animierter Kurve

Manchmal kann es hilfreich sein, ein Kurvensegment in zwei Teile zu teilen, etwa aufgrund der Einschränkungen, die es bei den Angaben zur Pfadanimation gibt. Dort ist die Anzahl und Folge der Kommandos für jeden Animationsschritt gleich zu wählen, so dass die Anzahl und Art der Pfadsegmente in den einzelnen Animationsschritten aneinander anzupassen sind. Dies kann durchgeführt werden, indem die Anzahl der kubischen Pfadfragmente auf das kleinste gemeinsame Vielfache erweitert wird.

Zudem kann so festgelegt werden, welche Fragmente bei einer Animation ineinander übergehen sollen. Eine solche Aufteilung eines Fragmentes einer Bézierkurve in zwei Fragmente jedenfalls geht recht effektiv und einfach mit dem Algorithmus von De Casteljau.

Dieser Algorithmus wird auch oft verwendet, um Bézierkurven mit einem Programm näherungsweise durch Geraden darzustellen, wobei dann so oft aufgeteilt wird, dass die Länge eines Fragmentes so klein wird, dass der Unterschied zu einer Geraden bei der Darstellung nicht mehr auffällt.

Das Verfahren sei hier für ein kubisches Fragment erläutert.

An der Stelle $0 < s < 1$ soll das Fragment aufgeteilt werden.

Anfangspunkt sei **P**(0) und Endpunkt **P**(3).

Die Kontrollpunkte seien **P**(1) und **P**(2).

Die Notation wird nun um einen Iterationsschritt i erweitert:

P(i,0), **P**(i,1), **P**(i,2), **P**(i,3).

Wobei **P**(0,j) = **P**(j) sein soll.

Anschaulich werden die Geraden zwischen zwei aufeinanderfolgenden P im Verhältnis gemäß s aufgeteilt. So entsteht ein neuer Punkt und neue Kontrollpunkte, welche insgesamt und zusammen mit den alten Punkten die neuen Kurvensegmente ergeben.

Nach dem Algorithmus ist, soweit es zu den Indizes Werte gibt:

P(i+1,j) = (1 - s) **P**(i,j) + s **P**(i,j+1),

oder ausgeschrieben:

P(1,0) = (1 - s) **P**(0) + s **P**(1)

P(1,1) = (1 - s) **P**(1) + s **P**(2)

P(1,2) = (1 - s) **P**(2) + s **P**(3)

P(2,0) = (1 - s) **P**(1,0) + s **P**(1,1)

P(2,1) = (1 - s) **P**(1,1) + s **P**(1,2)

P(3,0) = (1 - s) **P**(2,0) + s **P**(2,1)

Die beiden neuen kubischen Fragmente sind dann gegeben durch die Punkte:

P(0), **P**(1,0), **P**(2,0), **P**(3,0) für den ersten Teil und

P(3,0), **P**(2,1), **P**(1,2), **P**(3) für den zweiten Teil.

Folgendes zeigt eine Beispielgraphik mit Pfad und asymmetrischer Teilung:

Algorithmus nach De Casteljau[25]

Das Verfahren allein garantiert allerdings nicht, dass die Interpolation im Rahmen einer Animation besonders sinnvoll oder schön ist. Weil die aufgeteilten Fragmente nicht einfach mit dem Kommando S zusammengefügt werden können, werden in der Regel am Trennpunkt innerhalb der Animation Knicke entstehen.

Das sieht anders aus, wenn bei allen Kurven das s gleich gewählt wird, was recht einsichtig ist, weil jede einzelne Kurven ohnehin schon stetig differenzierbar am Trennpunkt ist, bei gleichem s sind dies dann auch die interpolierten Kurven.

Algorithmus nach De Casteljau mit animierter Kurve[26]

Zwar könnte auch eine stetig differenzierbare Fortsetzung am Trennpunkt mit dem Kommando S/s erzwungen werden, was aber den Kurvenverlauf verändert und meistens nicht erwünscht sein wird. Wenn die Aufgabe also darin besteht, mit einer Animation eine bestimmte Kurve in eine bestimmte andere zu verwandeln, wobei während der Animation keine zuvor nicht vorhandenen Knicke entstehen sollen, so sind derartige Feinheiten bei der Zerlegung zu beachten und die Kurvenfragmente jeweils passend zu wählen. Sozusagen ist es durchaus nicht trivial, aus derAbbildung einer Mücke die eines Elefanten zu machen, ohne dabei während der Verwandlung eigenartige Artefakte zu erhalten.

25 http://de.wikibooks.org/wiki/media%3ASVG_KurvenTeilen01.svg
26 http://de.wikibooks.org/wiki/media%3ASVG_KurvenTeilen02.svg

8 Text als Graphik

8.1 Text als Graphik

Text in SVG 1.1 wird wie anderer graphischer Inhalt wiedergegeben. Die Wiedergabe hängt allerdings davon ab, welche Schriftart dafür verwendet wird. In der verwendeten Schriftart werden die Buchstaben aus dem Quelltext als Glyphen repräsentiert. Die Schriftart kann auch Ligaturen enthalten, typisch wären etwa welche für ft, ff, fl etc. Die noch aus der Frakturzeit stammende ß-Ligatur gilt ohnehin als ein Buchstabe, nicht mehr als Ligatur. In einigen Fällen kann es auch sein, dass ein Buchstabe durch mehrere Glyphen repräsentiert wird. Daher muss die Anzahl der Buchstaben nicht mit der der darzustellenden Glyphen übereinstimmen. Die Zahl der Glyphen ist allerdings relevant dafür, wo welche Glyphe genau dargestellt wird. Die Zählung kann ferner davon abhängen, ob Leerzeichen erhalten bleiben sollen oder komprimiert werden. Bei den meisten Schriftarten ist zudem der von einem Buchstaben benötigte Platz nicht konstant, sondern vom Glyphen abhängig. Daher ist es für den Autor nicht trivial, genau vorherzusagen, wo welche Glyphe dargestellt werden wird. Auch sind nicht in jedem Darstellungsprogramm die vom Autor gewünschten Schriftarten verfügbar, so dass eine Schriftart verwendet werden kann, die lediglich ähnlich zur gewünschten ist, wenn der Autor nicht selbst mit SVG eine eigene Schriftart bereitstellt.

Jedenfalls kann der Autor bei Bedarf detailliert darauf Einfluss nehmen, wo und wie jede Glyphe dargestellt wird. Dazu haben die Textelemente einige Attribute. Für verschiedene Anwendungen stehen mehrere Textelemente zur Verfügung, die als Kindelemente des Basiselementes ***text*** auftreten können. In SVG tiny 1.1 tritt nur das Element ***text*** auf. In SVG tiny 1.2 gibt es darüber hinaus neue Elemente für Text.

8.2 Element *text*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	2

Abb. 89 Hello World

Bereits das Kapitel Start mit SVG[1] hat sich mit der Anzeige von Texten mit SVG beschäftigt. Die einfachste Möglichkeit der Anzeige erfolgt mit dem Element ***text***.

Leider ermöglicht SVG in den Versionen 1.0 und 1.1 keinen Zeilenumbruch. Dieser kleine "Makel" wurde inzwischen erkannt und kann im Bedarfsfalle seit SVG tiny 1.2 mit dem Element ***textArea*** umgangen werden. In der Version 1.1 hat der Autor den Text manuell umzubrechen und die jeweiligen Zeilen zum Beispiel in einzelnen ***text***-Elementen unterzubringen. Es gibt allerdings auch andere, etwas schönere Möglichkeiten.

8.2.1 Attribute *x* und *y*

Mit den Attributen *x* und *y* wird die Textposition angegeben. Der Wert ist eine Liste von Koordinaten. Im einfachsten Falle wird jeweils exakt eine Koordinaten für den gesamten Text angegeben. Die Attribute sind animierbar. Für ein fehlendes Attribut wird jeweils die Koordinate 0 angenommen.

Wie die Glyphen relativ zur angegebenen Position angeordnet werden, hängt von weiteren Präsentationsattributen ab. Die generelle Idee ist aber, dass bei Glyphen anders als bei den anderen Koordinatensystemen die y-Achse von unten nach oben zeigt. Ohne weitere Angaben ist also die *y*-Position unten beim Glyphen zu finden.

1 Kapitel 2 auf Seite 27

Zuerst nochmal das einfache Eingangsbeispiel:

```
<text x="50" y="80">Hello World</text>
```

Die Attribute *x* und *y* können nicht nur eine Koordinate enthalten, sondern jeweils eine mit Komma oder Leerzeichen separierte Liste von Koordinaten für jeden einzelnen Glyphen. Die Koordinaten stehen für die aktuelle Textposition.

Bei der im deutschsprachigen Raum üblichen Schreibrichtung von links nach rechts ergibt sich bei mehr Glyphen als *x*-Koordinaten eine automatische Fortsetzung der aktuellen Textposition nach rechts.

Sind mehr Koordinaten als Glyphen angegeben, werden die überschüssigen Koordinaten ignoriert.

```
<text
  x="50 60 80 70 90 80 70 60 50 30 20"
  y="20 30 50 40 60 70 80 90 110 100 130">Hello World</text>
```

8.2.2 Attribut *rotate*

Mit dem Attribut *rotate* können einzelne Glyphen um die aktuelle Textposition gedreht werden. Der Attributwert ist wiederum eine mit Komma oder Leerzeichen separierte Liste von Winkeln in Grad. Das Attribut ist animierbar (nicht additiv) Sofern nicht angegeben, wird 0 angenommen.

Unstrittig ist die Situation, wenn genau ein Winkel angegeben ist, dann werden alle Glyphen um diesem Winkel gedreht. Gibt es mehr Winkel als Glyphen, werden die überschüssigen Winkel ignoriert.

In der anderen Situation mit mehr Glyphen als Winkeln kann es praktische Probleme mit Implementierungen geben und einer als Fehler ausgegebenen Änderung in der Spezifikation. Die neue Variante ist zwar sinnvoller als die alte, bei älteren Darstellungsprogrammen ist natürlich ungewiß, wie diese die Situation interpretieren. Die aktuelle Interpretation ist, dass der letzte Winkel auf alle überschüssigen Glyphen angewendet wird. Die alte, alternative Interpretation war, dass die überschüssigen Glyphen ungedreht bleiben. Praktisch sollte ein Autor diesen Problemfall also vermeiden.

```
<text x="50" y="20" rotate="-10">Hello World</text>
<text x="50" y="60"
 rotate="-10 -15 -20 -15 -10 -5 0 5 10 15 20 25">Hello World</text>
```

8.2.3 Attribute *dx* und *dy*

Mit den Attributen *dx* und *dy* kann die aktuelle Textposition relativ um den angegebenen Wert verschoben werden. Als Wert kann jenweils eine mit Komma oder Leerzeichen separierte Liste von Längen angegeben werden. Die Liste wird analog zu der von *x* und *y* interpretiert.

Die Attribute sind in SVG tiny nicht verfügbar.

```
<text x="50" y="80"
 dx="0 -2 2 3 3 7 5"
 dy="0 -5 0 0 5 0 5 0 0 0 -5">Hello World</text>
```

8.2.4 Attribut *textLength*

Mit dem Attribut *textLength* kann der Autor die von ihm berechnete Länge des Textes angeben. Weicht die Berechnung des Darstellungsprogrammes von der Angabe ab, hat das Darstellungsprogramm eine entsprechende Anpassung der Glyphenpositionen vorzunehmen.

Der Wert ist eine Länge. Das Attribut ist animierbar. Voreinstellung ist die vom Darstellungsprogramm berechnete Textlänge.

Das Attribut ist in SVG tiny nicht verfügbar.

8.2.5 Attribut *lengthAdjust*

Mit dem Attribut *lengthAdjust* mit den möglichen Werten 'spacing' oder 'spacingAndGlyphs' wird angegeben, wie die Anpassung vorzunehmen ist. Im Falle von spacing werden nur die Abstände zwischen den Glyphen angepasst, was die Voreinstellung ist. Im Falle von spacingAndGlyphs wird beides gedehnt beziehungsweise komprimiert.

Das Attribut ist animierbar.

Das Attribut ist in SVG tiny nicht verfügbar.

```
<text x="50" y="80"
 font-size="10"
 textLength="150"
 lengthAdjust="spacing">Hello World</text>
```

8.2.6 Beispiel *text*

Abb. 90 SVG-Beispiel ***text*** - spezifische Attribute

Hier kann ausprobiert werden, welche Attribute vom aktuellen Darstellungsprogramm bereits (korrekt) interpretiert werden: SVG-Beispiel ***text*** - spezifische Attribute[2]

W3C: ***text***[3]

8.3 Element *tspan*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	2
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	2

tspan ist eine Möglichkeit, um Teile eines Textes zu markieren, anders zu dekorieren oder zu positionieren und kann in SVG 1.1 eingesetzt werden (nicht in SVG tiny 1.1). Ohne Positionierungsfunktionalität ist dies Element auch in SVG tiny 1.2 verfügbar.

In SVG 1.1 tritt ***tspan*** innerhalb von ***text*** und ***textPath*** auf, in SVG tiny 1.2 auch innerhalb von ***textArea***. Innerhalb von ***tspan*** können weitere ***tspan*** notiert werden, in SVG tiny 1.2 alle Elemente, die auch im Elternelement notiert werden dürfen, im Falle von ***textArea*** also auch ***tbreak***.

Innerhalb von ***text*** können ***tspan***-Elemente mit entsprechenden Attributen dazu dienen, neue Zeilen zu beginnen. Ein Darstellungsprogramm sollte so konstruierte mehrzeilige Texte am Stück markieren und kopieren können, anders als bei mehreren unabhängigen ***text***-Elementen. Allerdings hat der Autor selbst zu entscheiden, was jeweils in eine Zeile gehört, einen automatischen Zeilenumbruch gibt es innerhalb von ***text*** auch mit ***tspan*** nicht.

8.3.1 Attribute von *tspan*

Entsprechend zu ***text*** hat ***tspan*** in SVG 1.1 die gleichen Attribute *x*, *y*, *dx*, *dy*, *rotate*, *textLength* und *lengthAdjust*. In SVG tiny 1.2 können nur allgemein anwendbare Präsentationsattribute verwendet werden.

Ist ein Attribut nicht angegeben, kommen jeweils, sofern angegeben, die Attribute des Elternelementes zur Anwendung.

2 `http://de.wikibooks.org/wiki/media%3ASVGtext01.svg`
3 `http://www.w3.org/TR/SVG/text.html`

8.3.2 Beispiele *tspan*

Abb. 91 ***tspan*** - spezifische Attribute

Abb. 92 ***text*** und ***tspan*** - spezifische Attribute in beiden verwendet

Beispiel mit Präsentationsattributen (SVG 1.1 und SVG tiny 1.2):

```
<text x="50" y="80"
```

```
    font-size="10" fill="blue">Hello <tspan fill="green">W<tspan fill="red"
    font-size="20">o</tspan>rld</tspan></text>
```

Mehrzeiliges Beispiel (SVG 1.1)

```
<text font-size="10"><tspan
x="50" y="20">Hallo Welt</tspan><tspan
x="50" y="40">Ja, dies ist eine neue Zeile
<tspan dx="20" dy="20">...und noch eine...</tspan></tspan>
</text>
```

Einfacher alle Glyphen eines Wortes drehen (SVG 1.1):

```
<text x="50" y="50" font-size="10"><tspan
rotate="10">Hallo</tspan> <tspan
rotate="-10">Welt</tspan></text>
```

Mit den Autorenangaben zur Textlänge in folgendem Beispiel ist es wahrscheinlich, dass 'Welt' verdichtet wird und 'Hallo' und '!!!' auseinandergezogen. Die Angabe für das innere Element wird von der Angabe des äußeren abgezogen, um das Verhalten des äußeren zu bestimmen. Offenbar sollte also die Angabe für das äußere Element größer als für das innere sein, sonst kann das Darstellungsprogramm zu keiner plausiblen Darstellung gelangen.

```
<text x="50" y="50" font-size="10"
textLength="180" lengthAdjust="spacingAndGlyphs">Hallo
<tspan textLength="30"
lengthAdjust="spacingAndGlyphs">Welt</tspan>!!!</text>
```

Hier kann ausprobiert, welche Attribute vom aktuellen Darstellungsprogramm bereits (korrekt) interpretiert werden: SVG-Beispiel ***tspan*** - spezifische Attribute[4]

Kniffliger wird es natürlich, wenn die Attribute sowohl für ***text*** als auch ***tspan*** notiert werden. Im Darstellungsprogramm muss dann korrekt implementiert sein, welcher Wert gilt. Zudem sollte das Darstellungsprogramm die evenfalls verfügbaren Elemente ***title*** und ***desc*** natürlich nicht direkt als graphische Textausgabe darstellen, obwohl sie im jeweiligen Element drinstehen.

SVG-Beispiel ***text*** und ***tspan*** - spezifische Attribute in beiden verwendet[5]

8.4 Element *tref*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	-

tref ist ein Element, mit dem Textinhalt anderer Elemente wiederverwendet werden kann und ist in SVG 1.1 verfügbar, nicht in tiny. ***tref*** kann innerhalb von ***text***, ***tspan*** und ***textPath*** notiert werden.

4 http://de.wikibooks.org/wiki/media%3ASVGtspan01.svg
5 http://de.wikibooks.org/wiki/media%3ASVGtspan02.svg

Zum Beispiel kann so Text aus den Elementen ***title***, ***desc***, ***metadata*** zur direkten Anzeige als graphischer Inhalt gebracht werden, ohne den Text selbst wiederholen oder als Entität definieren zu müssen.

8.4.1 Attribute von *tref*

Die Attribute von XLink sind bei diesem Element verwendbar. Insbesondere dient das Attribut *href* dazu, ein anderes SVG-Element zu referenzieren, von dem der Text übernommen werden soll. Insbesondere ist *href* auch animierbar, dargestellter Textinhalt kann also ausgetauscht werden, etwa nützlich für Uhren oder Zähler.

Entsprechend zu ***text*** hat ***tref*** in SVG 1.1 die gleichen Attribute *x*, *y*, *dx*, *dy*, *rotate*, *textLength* und *lengthAdjust*. Das Verhalten ist ansonsten entsprechend zu ***tspan***.

8.4.2 Beispiel *tref*

Abb. 93 SVG-Beispiel ***tref***

Beispiel - Wiederverwendung des Dokumenttitels und von Elementinhalten von Elementen innerhalb des Elementes ***defs*** im graphischen Teil:

```
<svg viewBox="0 0 500 300"
    xmlns="http://www.w3.org/2000/svg" version="1.1"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xml:lang="de"
    font-size="10" font-family="sans-serif">
<title id="title">Hallo! Welt?</title>
<defs>
<text id="text1" xml:lang="la">alea iacta est!</text>
<text id="text2" xml:lang="la">timeo danaos et dona ferentes!</text>
</defs>

<text><tref xlink:href="#title"
x="50" y="50" rotate="-10 -5 0 5 10 15 20 15 10 5 0 -5" />
</text>

<text x="50" y="50" xml:lang="la">
<tref xlink:href="#text1" dy="50" rotate="-15" />
<tref xlink:href="#text2" y="150" rotate="15" />
<tref xlink:href="#text1" y="200" rotate="5" />
<tref xlink:href="#text2" y="250" rotate="-5" />
</text>

</svg>
```

SVG-Beispiel ***tref***[6]

8.5 Element *textPath*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	2
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	-

Mit dem Element ***textPath*** ist es möglich, Text entlang eines Pfades anzuordnen, ist in SVG 1.1 verfügbar, nicht in tiny.

8.5.1 Attribut *href* von XLink

Mit dem Attribut *href* von XLink ist der Pfad (siehe entsprechenden Abschnitt zu Pfaden) zu referenzieren, entlang dessen der Text angeordnet werden soll. Der Wert ist also ein Fragmentidentifizierer eines Elementes ***path***.

Ist mehr Text als Pfad vorhanden, wird jedenfalls der überschüssige Text nicht angezeigt.

8.5.2 Attribute *textLength* und *lengthAdjust*

Die schon für ***text*** angegebenen Attribute *textLength* und *lengthAdjust* sind auch für ***textPath*** verwendbar.

8.5.3 Attribut *startOffset*

startOffset gibt an, mit welchem Abstand relativ zum Beginn des Pfades der Text begonnen werden soll. Voreinstellung ist 0, Prozentangaben beziehen sich auf die gesamte Pfadlänge. Interessant kann auch ein negativer Wert sein (damit der Beginn des Textes nicht erscheint, kann hilfreich sein, wenn ein Lauftext animiert werden soll), dies ist nicht explizit ausgeschlossen.

Das Attribut ist animierbar.

8.5.4 Attribut *method*

Mittels *method* mit den möglichen Werten 'align' und 'stretch' wird angegeben, wie sich die Glyphen an den Pfad anpassen sollen. Bei 'align' werden die Glyphen einfach gedreht und verschoben, bei 'stretch' erfolgt auch noch eine Verzerrung des Buchstabens anhand der Krümmung des Pfades. Voreinstellung ist 'align'.

Das Attribut ist animierbar.

6 `http://de.wikibooks.org/wiki/media%3ASVGtref01.svg`

8.5.5 Attribut *spacing*

Mittels *spacing* mit den möglichen Werten 'auto' und 'exact' wird angegeben, wie mit dem Leerraum zwischen den Glyphen umgegangen werden soll. Im Falle von 'exact' ist in der Spezifikation ein Algorithmus angegeben, nach dem vorgegangen werden soll, bei der Voreinstellung 'auto' kann ein Algorithmus verwendet werden, der elegantere Resultate liefert.

Das Attribut ist animierbar.

8.5.6 Beispiele *textPath*

Abb. 94 SVG-Beispiel ***textPath***

```
<g fill="none" stroke="#ccc">
```

```
<path id="Pfad1"
d="M-400,0C-400,-200 -200,-400  0,-400
S400,-200 400,0
200,400 0,400
-400,200 -400,0Z"/>

<path id="Pfad2"
d="M-400,0C-400,200 -200,400  0,400
S400,200 400,0
200,-400 0,-400
-400,-200 -400,0Z"/>

<path id="Pfad3"
d="M-300 -90 300 -90
M-300 -30 300 -30
M-300 30 300 30
M-300 90 300 90" />
</g>

<text font-size="50">
<textPath xlink:href="#Pfad1" id="Text"
textLength="2450"><tspan dy="-10">Text kann
entlang eines Pfades angeordnet werden.
Dazu wird das Element textPath
verwendet.</tspan></textPath>

<textPath xlink:href="#Pfad2"><tref xlink:href="#Text"
textLength="2450" dy="-5" /></textPath>

<textPath xlink:href="#Pfad3"
textLength="2350">Das Element eignet
sich allerdings nur sehr eingeschränkt für
automatische Zeilenumbrüche.</textPath>
</text>
```

SVG-Beispiel ***textPath***[7]

8.6 Elemente *textArea* und *tbreak*

Squiggle (Batik)	-
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	-

Mit dem Element ***textArea*** kann unabhängig vom Element ***text*** Textinhalt in einem vorgegebenen Bereich ausgegeben werden, wo der Text bei Bedarf auch automatisch oder manuell umgebrochen wird.

Mit dem Element ***tbreak*** kann darin ein manueller Zeilenumbruch vorgenommen werden, mittels ***tspan*** kann ein Textfragment anders dekoriert werden.

tbreak ist ein leeres Element ohne spezifische Attribute.

textArea hat einige Attribute, wie folgt erläutert. Es gibt auch einige Eigenschaften, die in SVG tiny 1.2 als Präsentationsattribute zu notieren sind, die spezifisch für ***textArea*** sind.

7 http://de.wikibooks.org/wiki/media%3ASVGtextPath01.svg

8.6.1 Attribute *x* und *y*

Mit den Attributen *x* und *y* kann für ***textArea*** die linke obere Ecke des Textblockes angegeben werden, die Voreinstellung ist '0'. Der Wert ist jeweils eine Koordinate, *x* für die x-Postion, *y* für die y-Position.

Die Attribute sind animierbar.

8.6.2 Attribute *width* und *height*

Mittels *width* und *height* kann Breite und Höhe angegeben werden, *width* für die Breite, *height* für die Höhe. Die Werte sind Längen oder der Wert 'auto'. Bei 'auto' wird die Ausdehnung automatisch berechnet. Die Voreinstellung ist 'auto'. Anders als vielleicht erwartet werden kann, wenn (X)HTML und CSS bereits bekannt sind, bedeutet 'auto' insbesondere für *width* nicht eine implizite Begrenzung durch ein Elternelement oder den Anzeigebereich, bei dem Wert sind sowohl *width* als auch *height* unbegrenzt. Erst wenn für *width* ein konkreter Wert angegeben wird, erfolgt gegebenenfalls ein automatischer Zeilenumbruch, sonst nur wenn einer mittels ***tbreak*** explizit angegeben wird. Ist insbesondere für *height* ein Wert angegeben und reicht der Platz für den Text nicht aus, so wird überstehender Text nicht angezeigt (!).

Die Attribute sind animierbar.

8.6.3 Eigenschaft *line-increment*

Die Eigenschaft *line-increment* gibt die Höhe einer Zeile an. Der Wert ist 'auto', 'inherit' oder eine Zahl. 'auto' ist die Voreinstellung, 'inherit' gibt eine Vererbung vom Elternelement an. Die Zahl gibt die Zeilenhöhe in lokalen Einheiten an. Ein negativer Wert wird ignoriert. Die Eigenschaft ist anwendbar auf ***textArea***, ***tbreak*** und auf ***tspan***, sofern es sich in ***textArea*** befindet.

Die Eigenschaft ist animierbar.

8.6.4 Eigenschaft *text-align*

text-align ist eine Eigenschaft zur Ausrichtung des Textes. Mögliche Werte sind 'start', 'end', 'center' und 'inherit'. Letzteres ist wie immer die Vererbung vom Elternelement. Bei der Schreibrichtung von links nach rechts bedeutet 'start' linksbündiger Text, 'end' rechtsbündiger Text und 'center' horizontal zentrierter Text. Voreinstellung ist 'start'.

Die Eigenschaft ist animierbar.

8.6.5 Eigenschaft *display-align*

display-align ist ebenfalls eine Eigenschaft zur Ausrichtung des Textes. Mögliche Werte sind 'auto', 'before', 'center', 'after', 'inherit'. 'inherit' gibt eine Vererbung vom Elternelement an. Bei der Zeilenfolge von oben nach unten bedeuten 'before' und 'auto' den Text oben ausrichten, 'after' unten ausrichten, 'center' ein vertikales Zentrieren.

Die Eigenschaft ist animierbar.

8.6.6 Beispiele *textArea* und *tbreak*

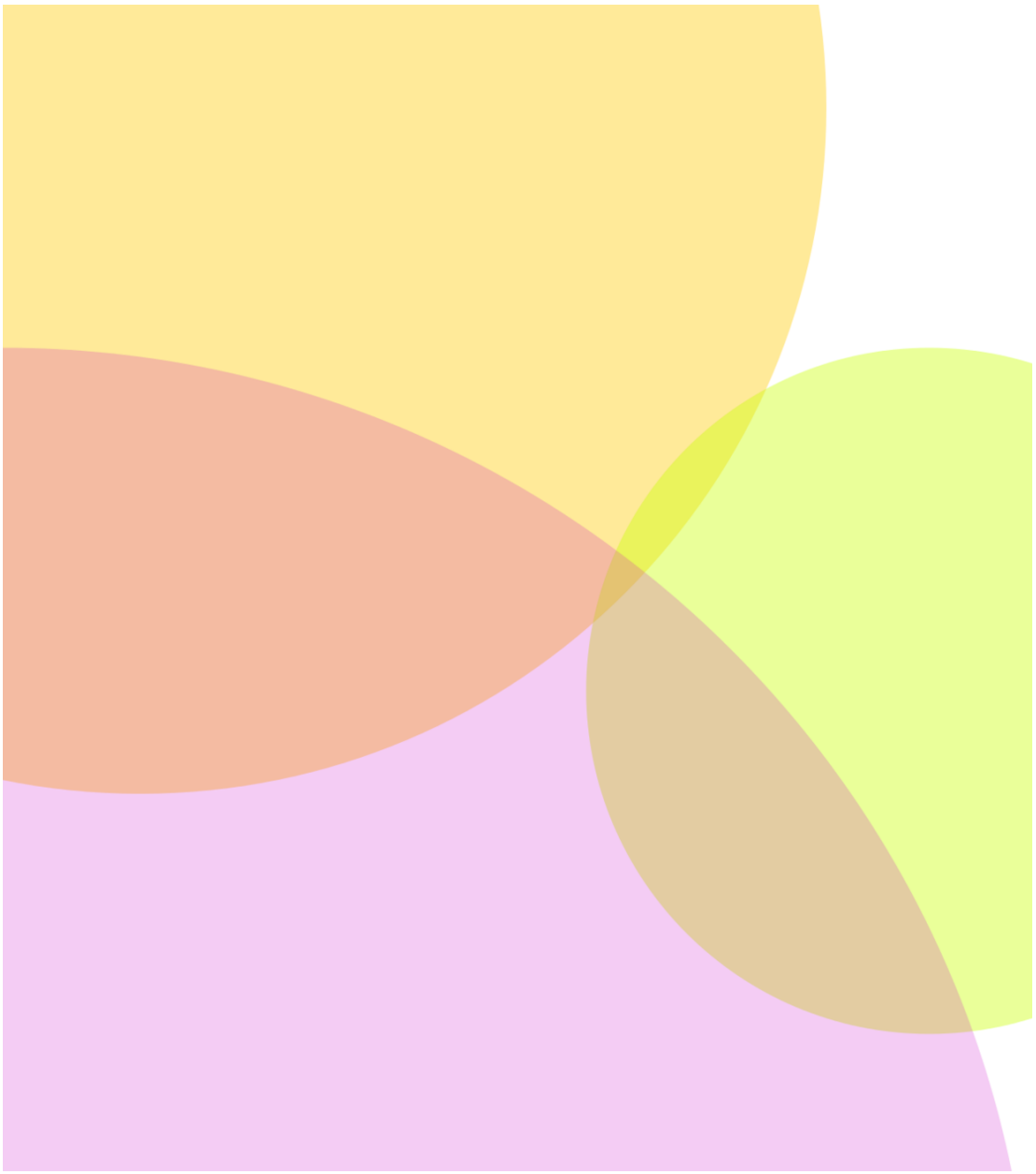


Abb. 95 'Die Kleinsten' von Wilhelm Busch

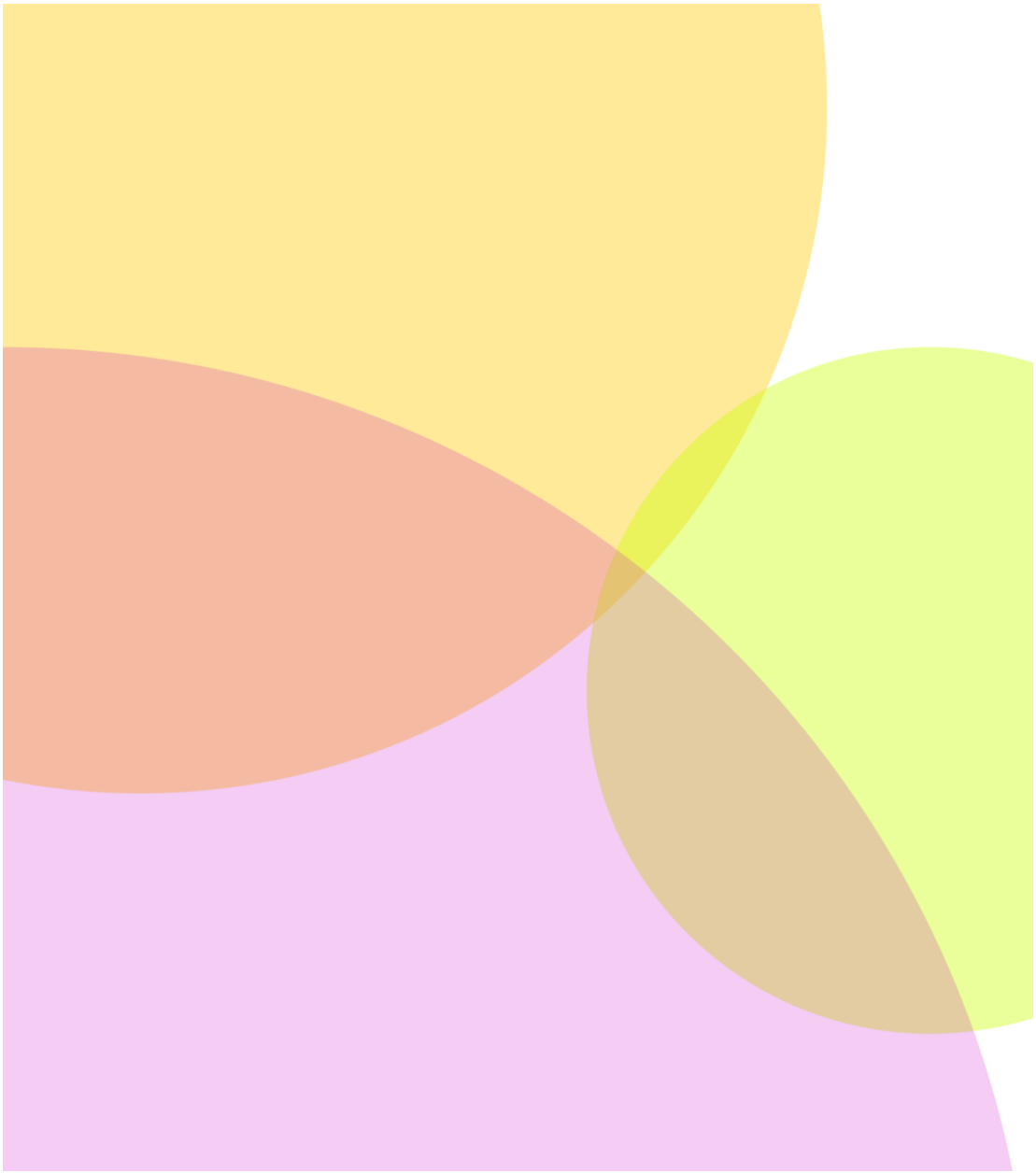


Abb. 96 SVG mit RDFa: 'Die Kleinsten' von Wilhelm Busch

Abb. 97 Beispiel mit automatischem Zeilenumbruch: 'Die Abweisung' von Franz Kafka

Einfaches Beispiel:

```
<textArea x="20" y="20" width="260" height="300"
  text-align="center" display-align="center"
  line-increment="auto"
  font-size="10" font-family="sans-serif"
  fill="#026">
<tspan font-size="20"
 fill="#044">Die Kleinsten</tspan>
<tbreak /><tbreak />
<tbreak />Sag Atome, sage Stäubchen.
<tbreak />Sind sie auch unendlich klein,
<tbreak />Haben sie doch ihre Leibchen
<tbreak />Und die Neigung da zu sein.<tbreak />

<tbreak />Haben sie auch keine Köpfchen,
<tbreak />Sind sie doch voll Eigensinn.
<tbreak />Trotzig spricht das Zwerggeschöpfchen:
<tbreak />Ich will sein so wie ich bin.<tbreak />

<tbreak />Suche nur sie zu bezwingen,
<tbreak />Stark und findig wie du bist.
<tbreak />Solch ein Ding hat seine Schwingen,
<tbreak />Seine Kraft und seine List.<tbreak />

<tbreak />Kannst du auch aus ihnen schmieden
<tbreak />Deine Rüstung als Despot,
<tbreak />Schließlich wirst du doch ermüden,
<tbreak />Und dann heißt es: Er ist tot.<tbreak />

<tbreak /><tspan fill="#028"
```

```
  font-size="6">Wilhelm Busch</tspan>
</textArea>
```

SVG-Beispiel: 'Die Kleinsten' von Wilhelm Busch[8]

Es ist leicht zu erkennen, dass in SVG die wesentliche Struktur des Gedichtes mit Strophen und Zeilen nicht ordentlich ausgezeichnet werden kann, nur eine graphische Repräsentation angeboten werden kann, die wenig bis keine semantische Struktur enthält. Wie im Abschnitt über Barrierefreiheit erläutert, lässt sich dies jedoch durch die Attribute aus RDFa deutlich verbessern. In diesem Fall kann ***tspan*** als Strukturelement verwendet werden und mit dem allgemeinen Attribut *property* angegeben werden, welche semantische Struktur damit repräsentiert wird. Da die für die Auszeichnung von Text eigentlich geläufige Sprache (X)HTML allerdings auch keine Elemente hat, die für Gedichte semantisch relevant wären, muss da im Bedarfsfalle auf Spezialsprachen ausgewichen werden. Immerhin verbessert das insofern die Zugänglichkeit, als rein aus diesen Angaben ohne die graphische Repräsentation eindeutig die Gedichtstruktur entnommen werden kann.

SVG mit RDFa, Beispiel: 'Die Kleinsten' von Wilhelm Busch[9]

SVG-Beispiel mit automatischem Zeilenumbruch: 'Die Abweisung' von Franz Kafka[10]

8 http://de.wikibooks.org/wiki/media%3ASVGtextArea01.svg
9 http://de.wikibooks.org/wiki/media%3ASVGtextArea2.svg
10 http://de.wikibooks.org/wiki/media%3ASVGtextArea03.svg

9 Graphiken formatieren

9.1 Einführung: Eigenschaften und Präsentationsattribute

Während in den Kapiteln über Grundformen, Text als Graphik und Pfade vermittelt wurde, mit welchen Elementen und Attributen Objekte angegeben werden können, die dargestellt werden sollen, so geht es in diesem Kapitel darum anzugeben, wie sie dargstellt werden.

Allen graphisch dargestellten Objekten gemeinsam ist das Konzept, dass die von den angegebenen Pfaden aufgespannten Flächen gefüllt werden können und dass die Pfade selbst als Strich endlicher Breite dargestellt werden können. Da es auch offene Pfade gibt, ist der Pfad oder Strich nicht gleichbedeutend mit dem Rand der aufgespannten Fläche, dennoch wird der dargestellte Pfad oft einfach Rand genannt, Strich oder Weg ist eigentlich die präzisere Bezeichnung.

Füllung (englisch: fill) und der Strich oder Weg (englisch: stroke) haben entsprechend einige Eigenschaften, die angegeben werden können. Text als Graphik hat noch einige Eigenschaften, die spezifisch für diese Arten von Pfaden ist. Einige andere Konstruktionen haben auch noch spezielle Eigenschaften.

In den tiny-Profilen können die Eigenschaften nur als sogenannte Präsentationsattribute genauso wie normale Attribute angegeben werden, bei der Vollversion SVG1.1 sollte dies ebenfalls die bevorzugte Methode sein. Das Verhalten von Attributen und Präsentationsattributen ist aber leicht unterschiedlich, etwa können Eigenschaften von Elternelementen geerbt werden.

Die Begriffe 'Eigenschaft' und 'Präsentationsattribut' deuten schon an, dass Eigenschaften dem Konzept von Cascade Style Sheets [CSS[1]] und XSL entstammen.

Entsprechend gibt es im vollen Profil von SVG 1.1 die Möglichkeit, diese Eigenschaften auch wirklich als Stilvorlagen anzugeben. Die kann mit externen Stilvorlagen erfolgen, die mit Stilvorlagenverarbeitungsanweisungen eingebunden werden. Ebenfalls möglich sind auch Angaben im Element ***style***, welches am besten ziemlich zu Beginn des Dokumentes innerhalb des Elementes ***defs*** auftreten sollte (aber auch in anderen Elementen wie ***svg*** oder ***g***).

Ebenfalls im vollen Profil von SVG 1.1 definiert ist das Attribut *style*, welches bei nahezu jedem Element angegeben werden kann und welches ebenfalls Angaben zu Eigenschaften in der CSS-Notation enthalten kann. Wie auch bei (X)HTML ist allerdings die Verwendung dieses Attributes aus verschiedenen Gründen zu vermeiden. Zum einen können für das Profil tiny zugeschnittene Darstellungsprogramme den Inhalt nicht interpretieren. Bei anderen Darstellungsprogrammen kann es zum Abbruch der Interpretation von weiteren Eigenschaften kommen, wenn es bei der Interpretation einer Eigenschaft Probleme gibt. Zum anderen haben Angaben im Attribut *style* eine relativ hohe Spezifität, können also weder vom Autor noch vom Betrachter einfach mit eigenen Angaben überschrieben

1 `http://de.wikibooks.org/wiki/CSS`

werden, ohne für jeden einzelnen Fall die Spezifität explizit geeignet hoch anzusetzen. Ein weiteres Problem kann bei der Animation von Präsentationsattributen auftreten, zumindest wenn Animation korrekt im Sinne von SMIL implementiert wurde. Leider gibt es immer noch einige Hilfsprogramme, welche das Attribut *style* intensiv nutzen. Autoren ist zu empfehlen, diesen Mangel dann manuell zu beheben, nachdem ein Dokument abgespeichert ist.

Insgesamt kann gesagt werden, dass alles in Form von Präsentationsattributen notiert werden sollte, was inhaltlich relevant ist, also dazu beiträgt, den Inhalt des Bildes zu vermitteln. Präsentationsattribute haben eine niedrige Spezifität und können daher leicht mit Angaben im Element ***style*** oder in einer externen Stilvorlage überschrieben werden, mit denen dann alternative Ansichten des Dokumentes angeboten werden können.

9.2 Eigenschaften zur Füllung

9.2.1 Füllung, *fill*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Abb. 98 Füllfarben mit SVG

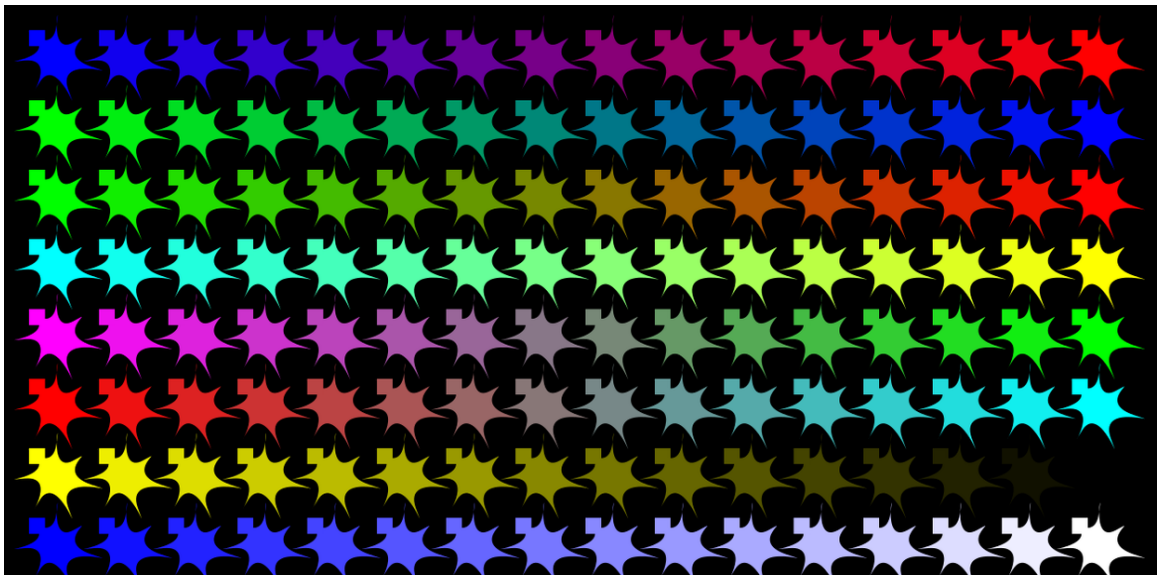


Abb. 99 Vergleich fill und PNG-Farbe

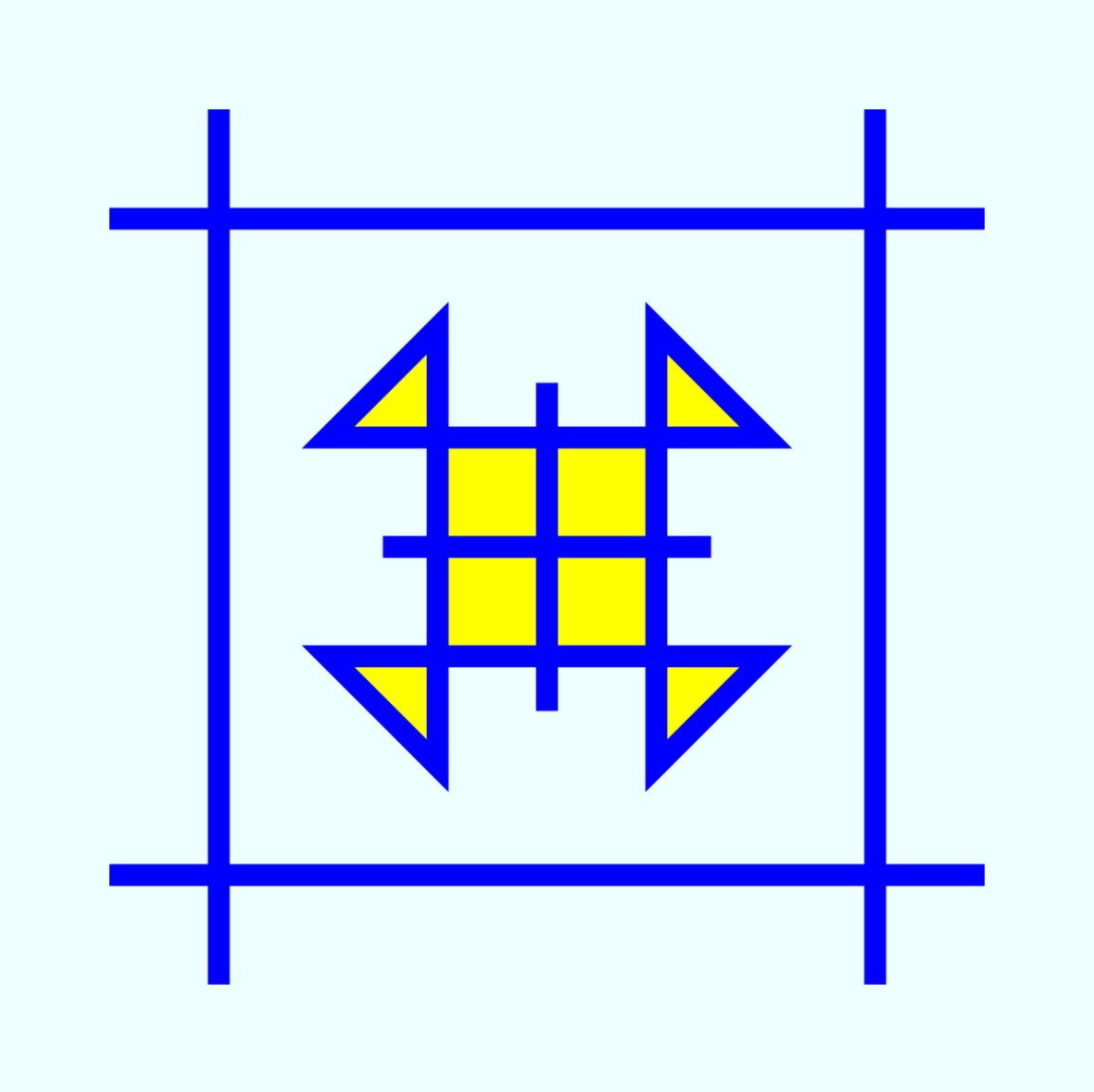


Abb. 100 Gerade Unterpfade und Flächen

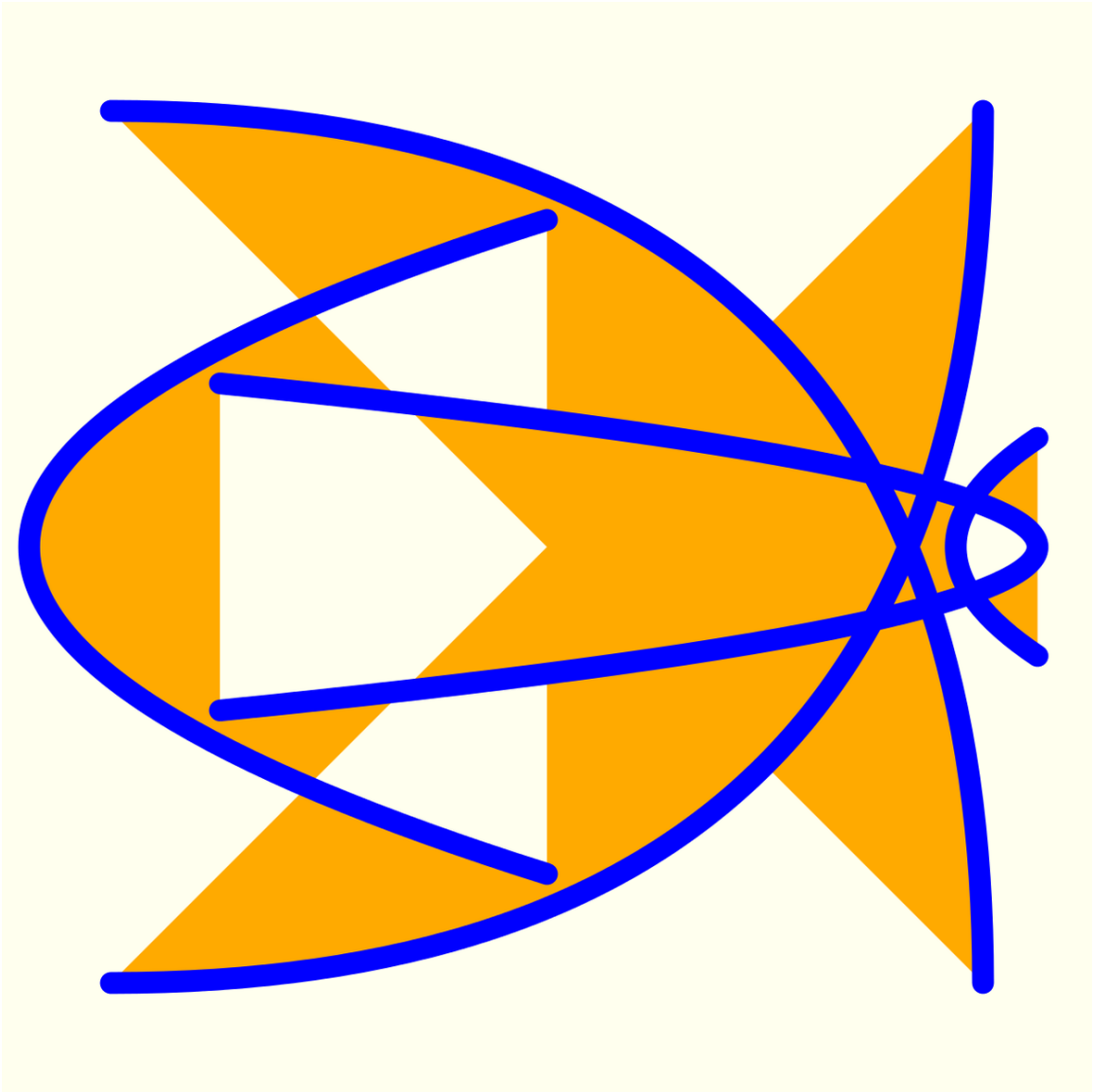


Abb. 101 Offene Unterpfade und Füllung

Bei den Beispielen im Kapitel Zeichnen mit SVG[2] wurden von Pfaden aufgespannte Flächen ohne weitere Angaben immer schwarz gefüllt. Nun mag es sein, dass dies für gewisse Anwendungen sehr nützlich ist – hin und wieder sind jedoch auch andere Farben oder Füllungen wünschenswert. Daher kann mit der Eigenschaft *fill* angegeben werden, wie oder womit eine aufgespannte Fläche gefüllt werden soll.

Die möglichen Werte von *fill* sind sehr reichhaltig und hängen vom Profil ab.

Folgendes sind die Werte mit Erklärung:

'none'

2 `http://de.wikibooks.org/wiki/SVG%2F%20Zeichnen%20mit%20SVG`

keine Füllung

'currentColor'

Es wird der Farbwert von der Eigenschaft *color* übernommen

'inherit'

Es wird der Wert vom Elternelement übernommen (geerbt)

Farbe

Ein Farbwert wie im Folgenden erläutert. Optional kann zusätzlich ein ICC-Farbprofil angegeben werden, wenn es sich nicht um ein tiny-Dokument handelt.

Maldienst

Die URI eines Maldienstes (Farbverlauf, Muster, feste Farbangabe; englisch: paint server), optional gefolgt von einer der anderen Möglichkeiten, falls der Maldienst nicht anwendbar ist. SVG tiny 1.1 hat keine Maldienste, SVG 1.1 zudem keine fest Farbangabe, SVG tiny 1.2 keine Muster. Wie der Maldienst exakt zu notieren ist, wird im entsprechenden Abschnitt erläutert.

Der Initialwert ist 'black' also schwarz und die Füllung wird vererbt. Ist also beim Elternelement ein Wert angegeben, nicht aber beim Kindelement, so gilt beim Kindelement die Angabe vom Elternelement, nicht 'black'.

Als Farbwert sind in den tiny-Profilen nur die numerischen Angaben zulässig und die sechzehn Farbwörter von (X)HTML: black, green, silver, lime, gray, olive, white, yellow, maroon, navy, red, blue, purple, teal, fuchsia, aqua. SVG tiny 1.2 und das volle Profil SVG 1.1 verfügen ferner über die Systemfarben wie von CSS2 definiert, SVG1.1 als Farbangabe, SVG tiny 1.2 als Maldienst, was in dem Falle auf die gleiche Angabe hinausläuft.

Farbwerte können also in dreierlei Form angegeben werden:

- als RGB kodiert in der Form `rgb(rrr,ggg,bbb)` dabei sind rrr,ggg,bbb Farbangaben für die Kanäle rot, grün und blau, Werte jeweils von 0 bis 255. Beziehungsweise `rgb(10%,50%,100%)`, also als Prozentangaben von 0% bis 100%
- hexadezimal angegeben in der Form `#000000` mit jeweils zwei Ziffern von 00 bis ff für wieder jeweils einen Farbkanal oder `#000` mit jeweils einer Ziffer zwischen 0 und f für einen Farbkanal, wobei dann die Ziffer vom Darstellungsprogramm gedoppelt werden, um auf den Wert in der langen Darstellung zu kommen, #abc entspricht also #aabbcc.
- mit dem Namen der Farbe (vgl. CSS[3]), nicht in allen Profilen komplett verfügbar, daher sind andere als die oben genannten Farbwörter eher zu vermeiden.

fill ist animierbar, wobei kontinuierliche Animation nur möglich ist, wenn einem Wert eine numerische Farbangabe zugeordnet werden kann, was insbesondere nicht bei 'none' oder einem Maldienst wie einem Farbverlauf oder einem Muster zutrifft. Aufgrund der Komplexität der möglichen Werte liegen häufig Implementierungsmängel vor, wenn in einer Animation eine bunte Mischung von Farbangaben und nicht interpolierbaren Werten vorgegeben wird. Autoren ist dann dringend zu empfehlen, explizit anzugeben, dass eine diskrete Animation durchgeführt werden soll, statt sich darauf zu verlassen, dass das Darstellungsprogramm dies selbst herausbekommt, wie es laut Spezifikation vorgesehen ist, wenn der Autor keine Angaben macht.

3 `http://de.wikibooks.org/wiki/CSS`

Zur Verdeutlichung der Verwendung von *fill* mit Farbwerten ein kleines Beispiel:

```
<rect x="0" y="0" width="80" height="10" rx="6" ry="10" fill="rgb(255,0,0)" />
<rect x="0" y="0" width="13" height="60" rx="10" ry="7.5" fill="red"/>
<circle cx="0" cy="0" r="20" fill="#C00000"/>
```

Beispiel: Vergleich fill und PNG-Farbe[4].

Die Füllfarbe von Pfaden wird verglichen mit der gleichen Farbe in eingebetteten PNGs. Die PNGs sind links oben über den Pfaden angeordnet - es handelt sich um Quadrate. Sofern ein Farbunterschied zwischen einem Quadrat und der darunterliegenden Form sichtbar wird, werden die Farbangaben im PNG und in der Eigenschaft fill der Pfade unterschiedlich interpretiert, was auf einen Implementierungsfehler hinweist.

Beispiel: Gerade Unterpfade und Flächen[5].

Gerade Pfadsegmente spannen allein keine Fläche auf. Auch mehrere Unterpfade, die nur aus geraden Pfadsegmenten bestehen, spannen keine Fläche auf, die zu füllen wäre.

Anders hingegen, wenn die geraden Pfadsegmente miteinander verbunden werden. Sofern das Resultat nicht wieder nur ein gerader Pfad ist, kann eine Fläche aufgespannt werden, die gefüllt werden kann.

Im konkreten Beispiel gibt es zwei Pfade. Der außenliegende besteht aus vier geraden Unterpfaden, die ein Quadrat aufzuspannen scheinen. Da die Unterpfade aber selbst keine Flächen aufspannen, ergibt sich auch keine Füllung, die ansonsten rot wäre.

Beim inneren Pfad sind einige gerade Pfadsegmente zu einem Pfad verbunden. Dieser Pfad ergibt einige füllbare Flächen, die hier mit einer gelben Füllung versehen sind. Zwei weitere gerade, unverbundene Unterpfade, die ein mittiges Kreuz bilden, spannen keine neuen Flächen auf und ändern nichts an der Füllung.

Beispiel: Offene Unterpfade und Füllung[6].

Zur Füllung werden offene Unterpfade immer so geschlossen, also ob der Endpunkt mit dem Anfangspunkt verbunden wäre.

Ein Pfad mit mehreren gekrümmten Unterpfaden kann auch füllbare Flächen aufspannen, die sich gegenseitig überlagern können, so dass sich ein interessanter Wechsel von Innen und Außen ergeben kann.

Der hier blau dargestellte Pfad besteht aus mehreren offenen Unterpfaden, die sich gegenseitig überschneiden und so ein etwas komplizierteres Muster von Innen (orange) und Außen (gelb-weiß) produzieren.

4 http://de.wikibooks.org/wiki/media%3ASVGfill01.svg
5 http://de.wikibooks.org/wiki/media%3ASVGfill02.svg
6 http://de.wikibooks.org/wiki/media%3ASVGfill03.svg

9.2.2 Füllregel, *fill-rule*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

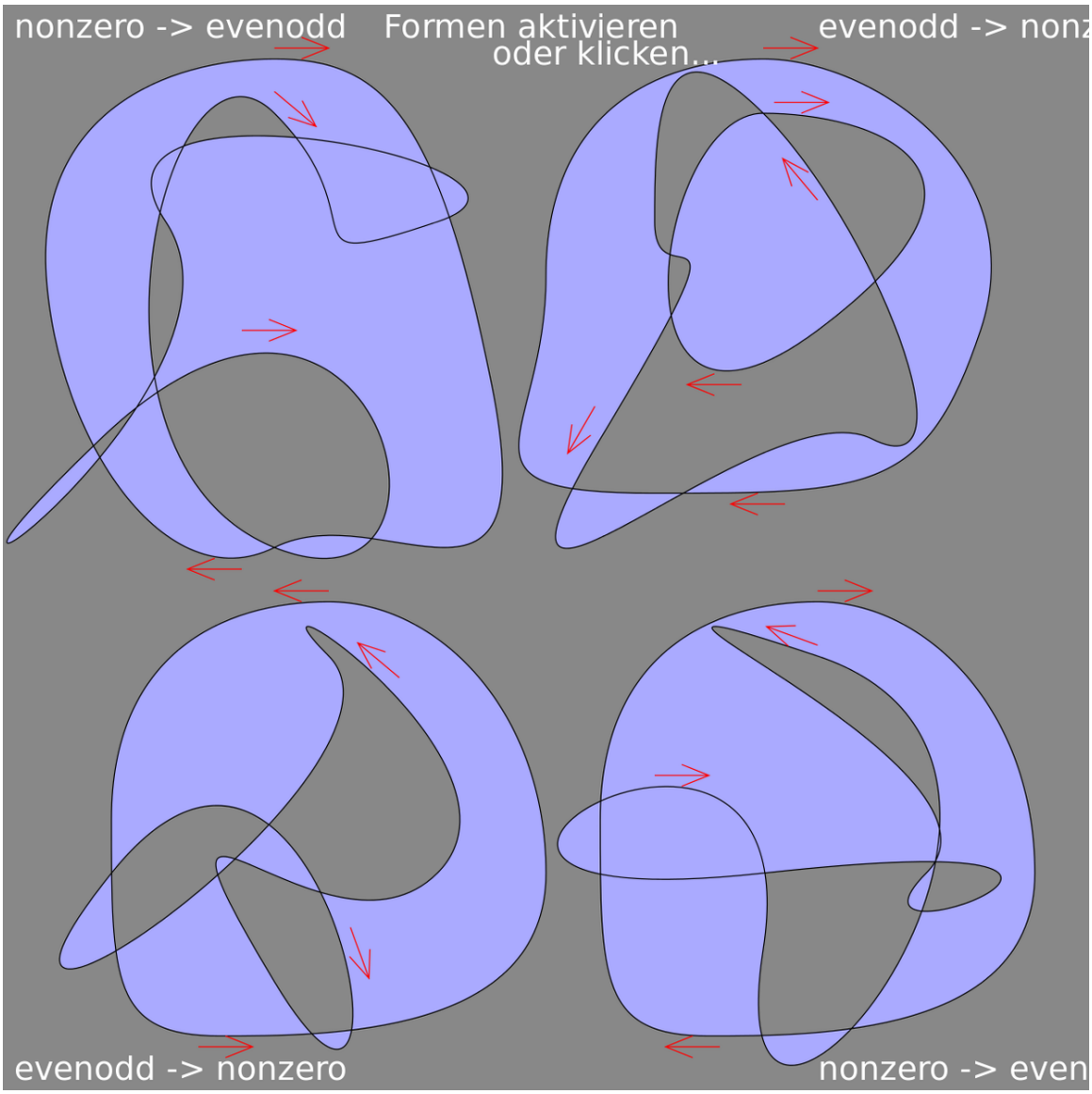


Abb. 102 Beispiele zur Eigenschaft *fill-rule*

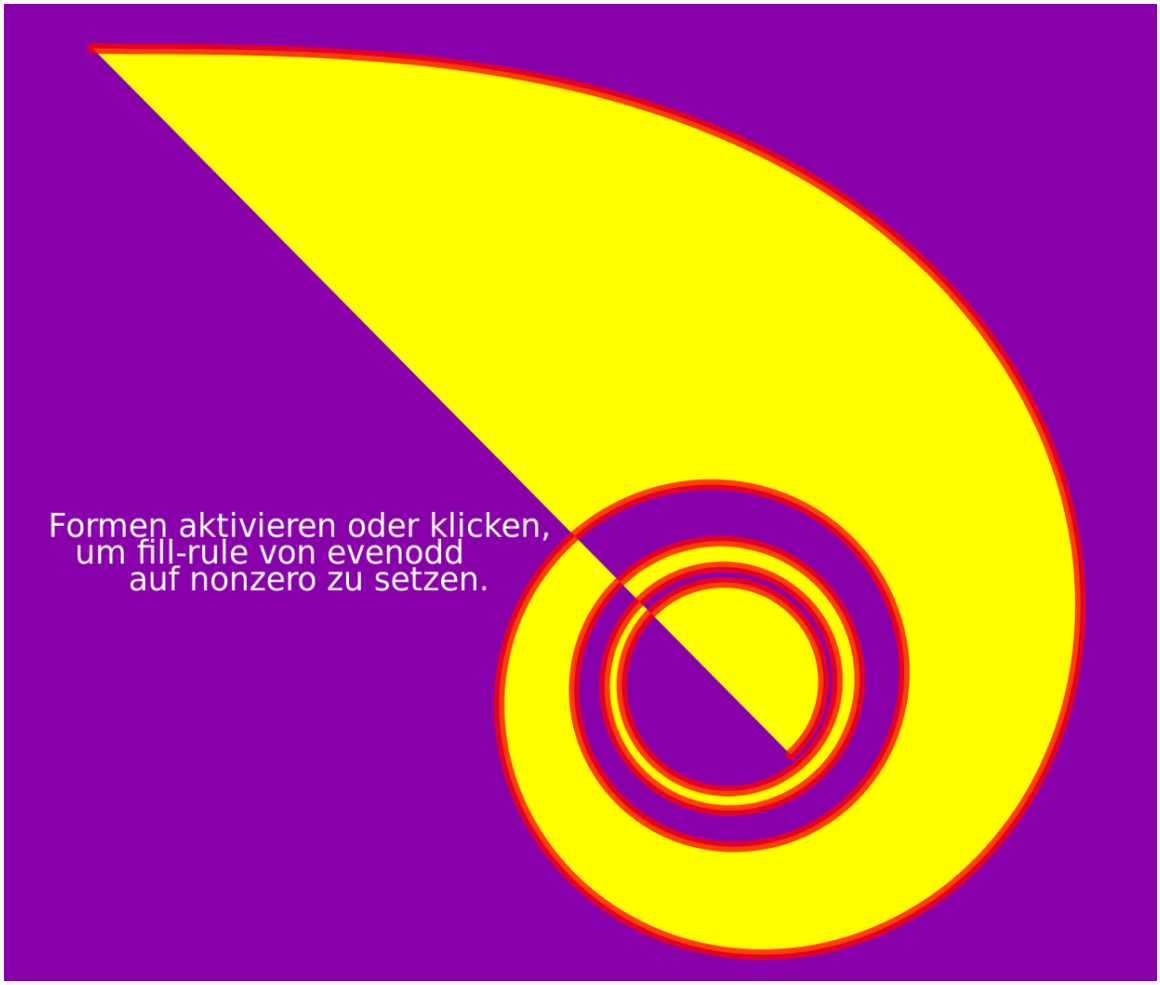


Abb. 103 Beispiel Klothoide - nicht geschlossener Pfad und *fill-rule*

SVG hat also ein Konzept von Innen und Außen für Formen.

Überschneidet sich ein Pfad selbst, ist die Interpretation von Innen und Außen nicht mehr eindeutig. Mit der Eigenschaft *fill-rule* kann das Verhalten festgelegt werden.

Mögliche Werte sind 'nonzero', 'evenodd' und 'inherit'. Der Initialwert ist 'nonzero', die Eigenschaft wird vererbt und ist ferner animierbar.

Umschließt zum Beispiel ein Unterpfad alle anderen und ist die Umlaufrichtung gegensinnig, entsteht durch einen inneren Pfad ein Loch. Ist die Umlaufrichtung gleichsinnig, passiert dies nur mit *fill-rule*="evenodd".

Im anderen Falle für *fill-rule*="nonzero". kann es also passieren, dass bei gleichsinnigen Umflaufrichtungen sich überschneidender oder ineinanderliegender Unterpfade der Strich auch innerhalb der Fläche auftritt, was bei *fill-rule*="evenodd" nicht passieren wird, wenn alle Unterpfade Flächen aufspannen. Da ist der Strich immer am Rand zwischen Innen und Außen. Bei "evenodd" wechseln sich Innen und Außen bei jedem Rand ab.

Beispiele zur Eigenschaft *fill-rule*[7]

In dem interaktiven Beispiel kann die Füllregel für jede Form gewechselt werden. Sollte das aktuell verwendete Darstellungsprogramm hinsichtlich Animation noch eine Implementierungslücke haben, sind jeweils zwei Formen mit evenodd und zwei mit nonzero als Startwert angegeben.

Bei einer nicht geschlossenen Form ist nicht immer ganz intuitiv einzusehen, was innen und außen ist.

Beispiel Klothoide - nicht geschlossener Pfad und *fill-rule*[8]

Anfang und Ende des Pfades werden miteinander verbunden, dann wird anhand von fill-rule bestimmt, was innen und was außen ist. In diesem Fall ist außen violett und innen gelb. Der eigentliche Pfad ist rot und teiltransparent.

Die präzise Bestimmung des Verhaltens für *fill-rule*:

- Bei offenen Pfadsegmenten werden diese für die Zählung mit einer geraden Verbindung vom Ende zum Anfang geschlossen.
- Für jeden Punkt wird bestimmt, ob er innen oder außen liegt, indem ein Strahl betrachtet wird, der im Unendlichen beginnt und am betrachteten Punkt endet.
- Um Komplikationen zu vermeiden, ist dabei der Strahl jeweils so zu wählen, dass er keine Tangente zu einem Pfadsegment darstellt, also immer eindeutig eine Kreuzung vorliegt, nicht nur ein Tangieren.

'nonzero':

Es werden die Schnittpunkte des Strahles mit dem Pfad gezählt, angefangen mit null im Unendlichen. Jedesmal wenn ein Pfadsegment von links nach rechts den Strahl kreuzt, wird der Zähler um eins erhöht. Wenn ein Pfadsegment von rechts nach links den Strahl kreuzt, wird der Zähler um eins erniedrigt. Ist das Endergebnis null, ist der Punkt außerhalb, sonst innerhalb.

'evenodd':

Es wird die Anzahl der Kreuzungen des Strahles mit Pfadsegmenten gezählt. Ist die Zahl der Kreuzungen ungerade, liegt der Punkt innerhalb, sonst außerhalb.

7 http://de.wikibooks.org/wiki/media%3ASVGfill-rule01.svg
8 http://de.wikibooks.org/wiki/media%3ASVGfill-rule02.svg

9.2.3 Füll-Opazität, *fill-opacity*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

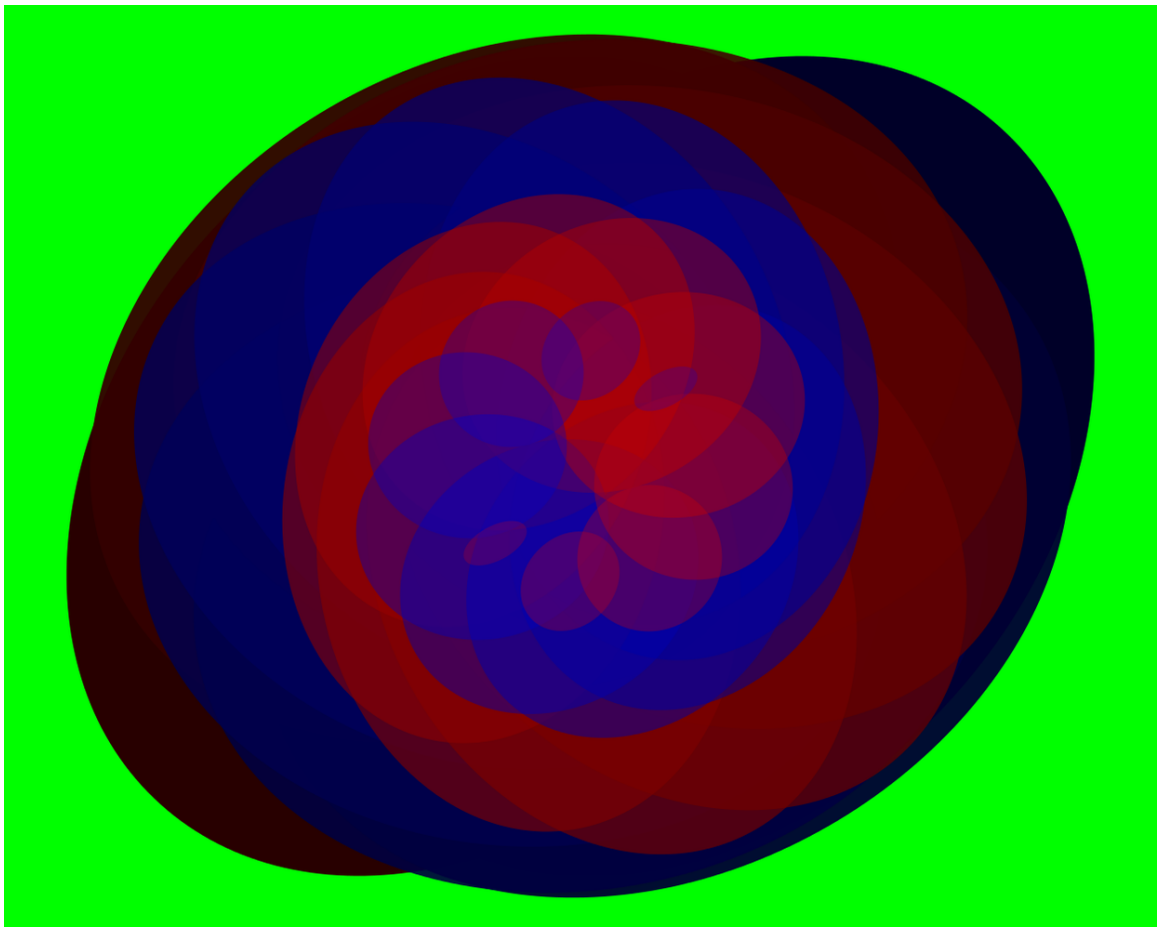


Abb. 104 Beispiel zur Eigenschaft *fill-opacity*

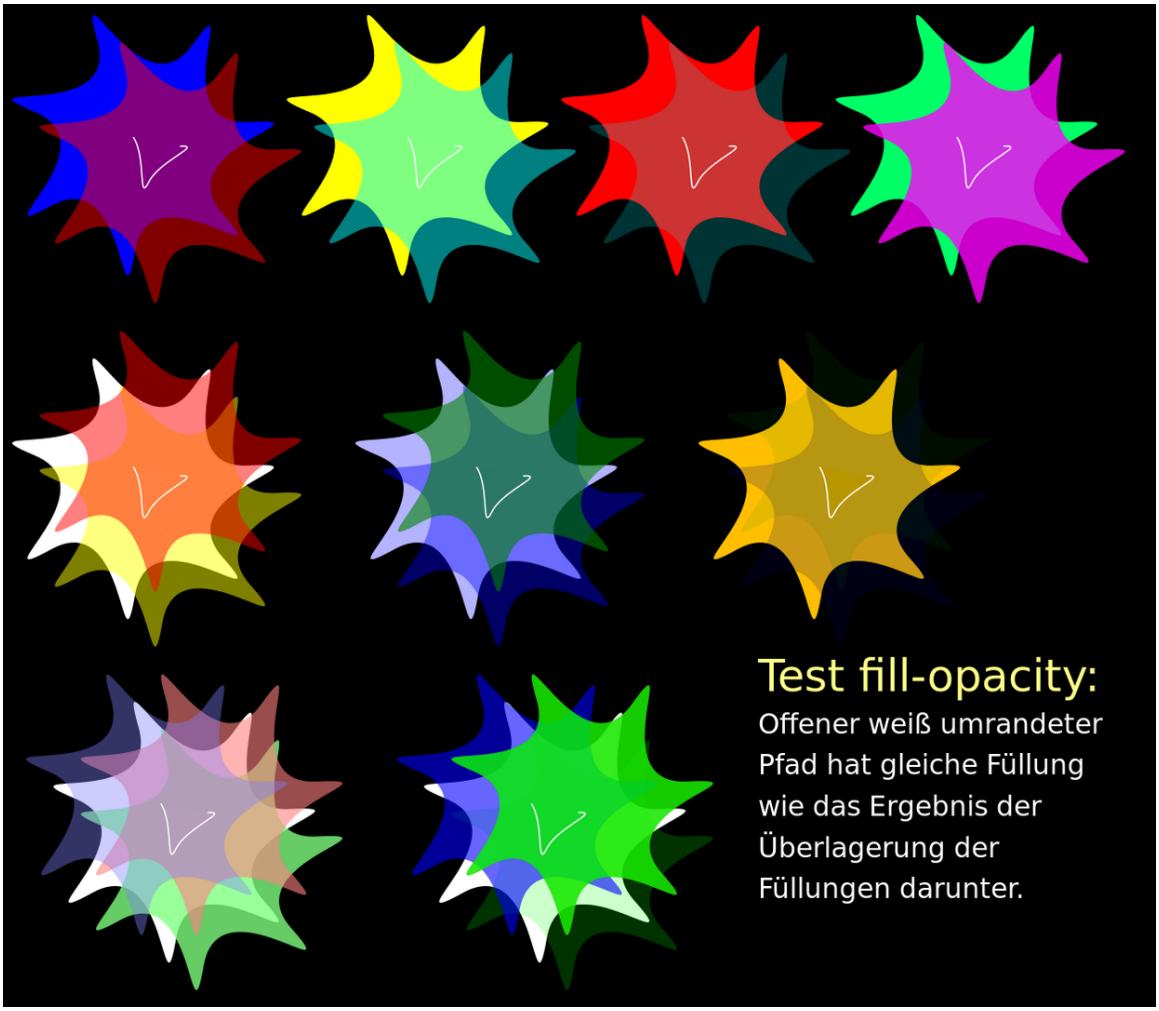


Abb. 105 Test zur Eigenschaft *fill-opacity*

Graphische Objekte können auch teilweise durchsichtig gefüllt sein, dazu dient die Eigenschaft *fill-opacity* mit Werten zwischen 0 und 1. 0 steht für ganz durchsichtig.

Andere Werte außerhalb des Intervalls [0;1] werden zum nächstgelegenen zulässigen Wert gerundet.

Der Initialwert ist 1, also undurchsichtig, die Eigenschaft wird vererbt und ist animierbar.

fill-opacity ist in SVG tiny 1.1 nicht verfügbar.

Die resultierende Farbe von mehreren teildurchsichtigen Elementen kann recht einfach aus den jeweils angegebenen Werten für *fill-opacity* berechnet werden. Ist u die unterliegende Farbe und a die aktuelle Farbe des darüberliegenden Elementes, und t dessen Wert von fill-opacity, so ist die neue Farbe n = (1-t)u + ta.

Beispiel zur Eigenschaft *fill-opacity*[9]

9 `http://de.wikibooks.org/wiki/media%3ASVGfill-opacity01.svg`

Zwei Scharen von Ellipsen, gedreht, mit variierter Füll-Opazität.

Test zur Eigenschaft *fill-opacity*[10]

fill-opacity wird getestet, indem mehrere Formen übereinandergelegt werden, von denen einige teiltransparent sind. Diese Stapel werden mit einer teils offenen Form mit weißem Strich in ihrer Mitte verglichen, welche die gleiche Farbe hat wie die resultierende des Stapels.

9.3 Eigenschaften zum Strich oder Weg

Der Strich, die Darstellung eines Pfades erfolgt mittig zum Pfad, gegebenenfalls also zur Hälfte auf der Füllung, zur anderen Hälfe außerhalb des Objektes.

SVG1.1 und SVG tiny 1.2 sehen keine direkten Möglichkeiten vor, um die Position des Striches relativ zum Pfad zu verschieben, etwa nur innerhalb oder außerhalb der Füllung darstellen. Dies ist für die Vollversion von SVG 1.2 in Vorbereitung, kann derzeit allenfalls mit einigen Tricks auf Umwegen erreicht werden.

9.3.1 Strichfüllung, *stroke*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

10 `http://de.wikibooks.org/wiki/media%3ASVGfill-opacity02.svg`

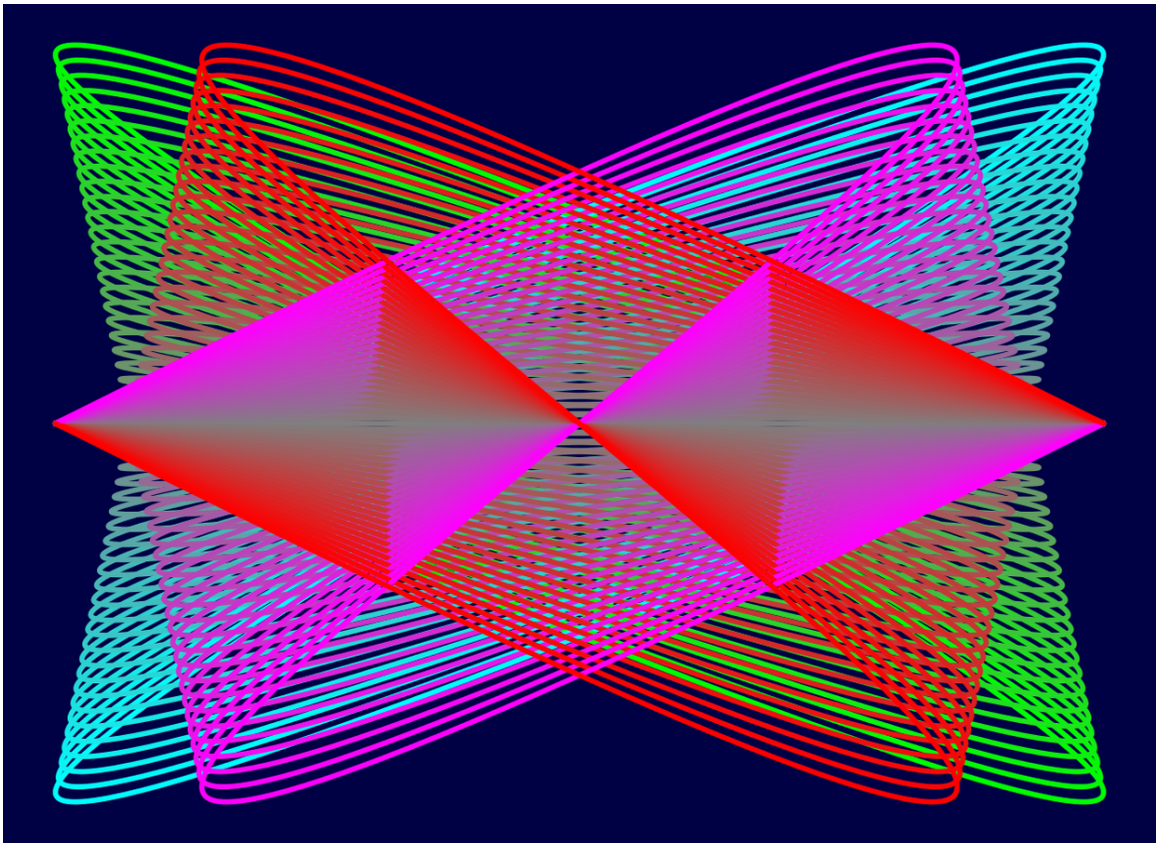


Abb. 106 Zwei Scharen von Kurven werden mit unterschiedlichen Strichfarben gemalt

Neben der Farbangabe für die Füllung kann auch angegeben werden, ob und wie der Strich oder Weg gemalt, beziehungsweise gefüllt werden soll. Dazu dient die Eigenschaft *stroke*.

Die möglichen Werte sind die gleichen wie für *fill*.

Weil allerdings der Initialwert 'none' ist, wird der Strich ohne explizite Angabe nicht dargestellt.

Die Eigenschaft wird vererbt und ist animierbar - mit der gleichen Problematik, wie bereits für *fill* erläutert.

Beispiel: Zwei Scharen von Kurven werden mit unterschiedlichen Strichfarben gemalt[11]

9.3.2 Strichstärken, *stroke-width*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

11 `http://de.wikibooks.org/wiki/media%3ASVGstroke01.svg`

Abb. 107 Strichstärke und -farbe mit SVG

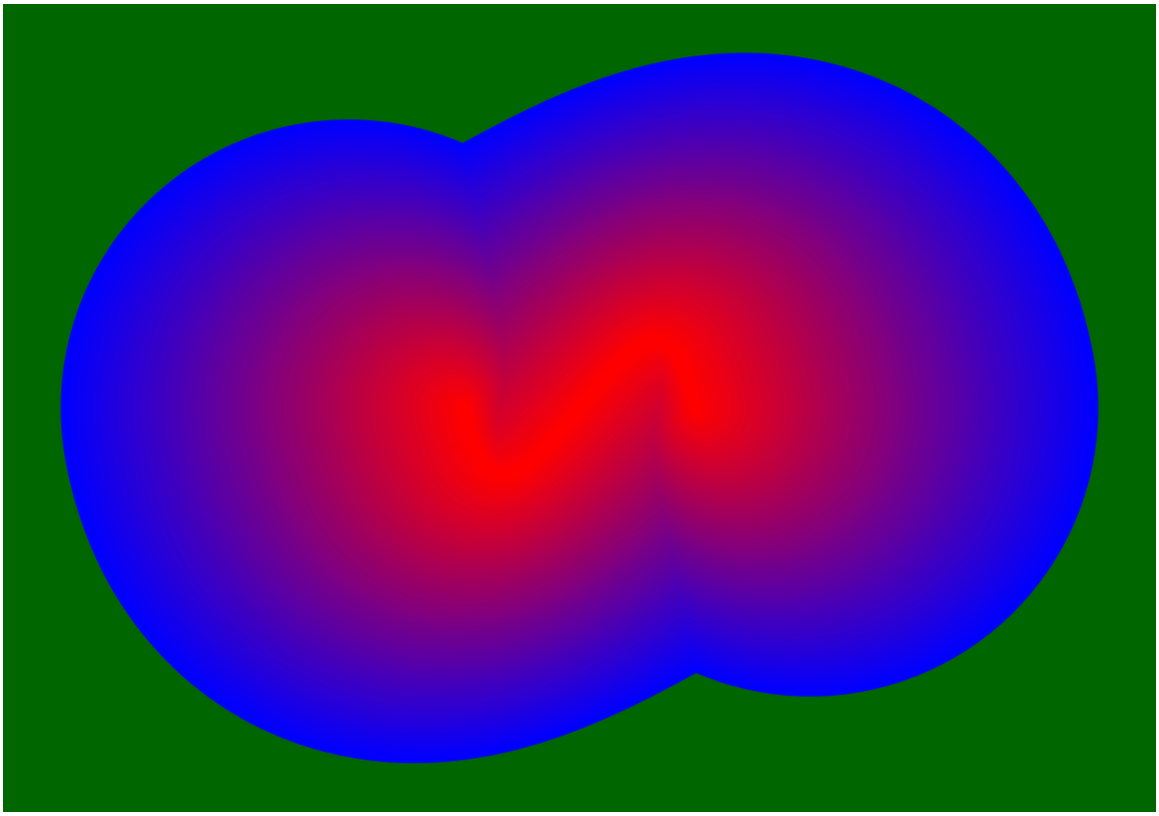


Abb. 108 Eine Schar von Kurven mit abgerundeten Enden wird mit unterschiedlichen Strichfarben gemalt

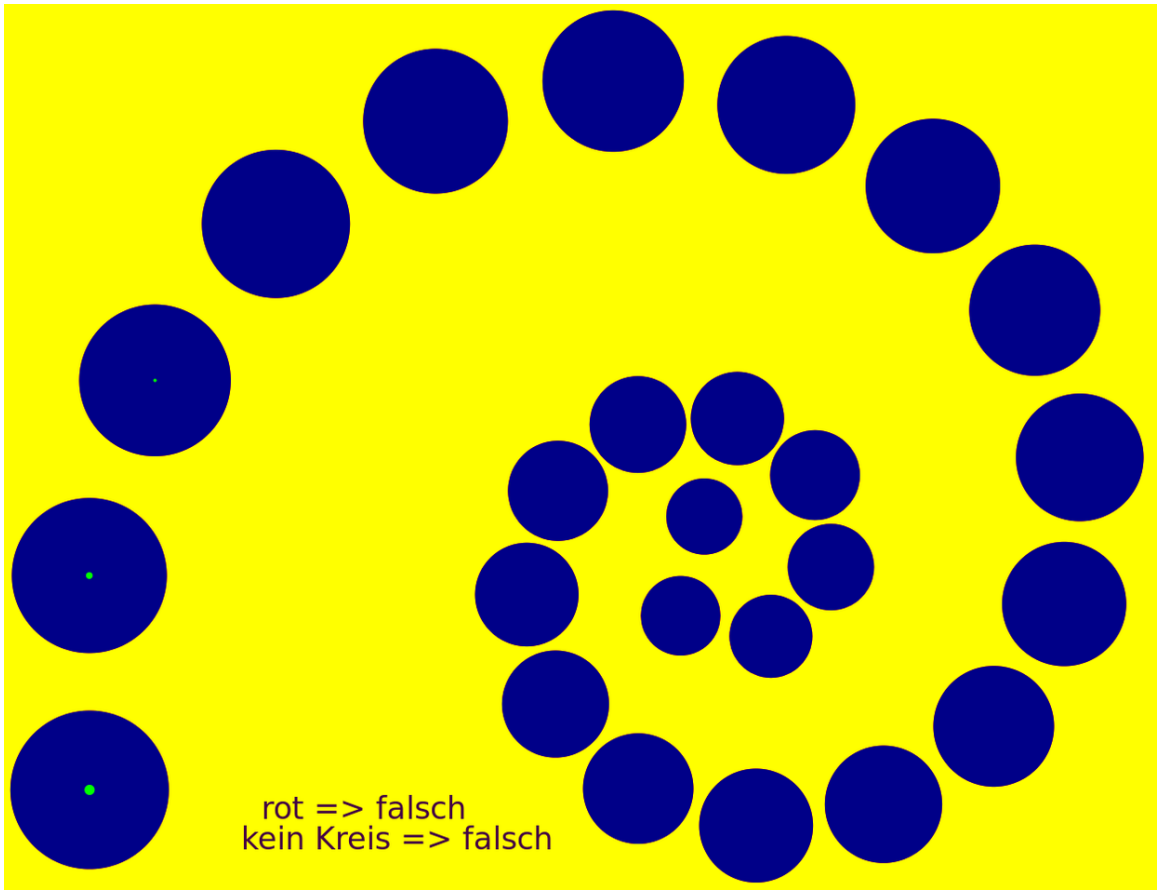


Abb. 109 Kreise und große Strichstärke

Auch die Strichstärke kann angegeben werden - sozusagen die Pinselstärke, mit der ein Pfad gezeichnet wird. Dazu dient die Eigenschaft *stroke-width*.

In SVG 1.1 und tiny 1.2 ist die Strichstärke für einen Pfad immer gleich. Für die Vollversion 1.2 gibt es Planungen und Ideen, die Strichstärke über den Verlauf des Pfades zu ändern. In SVG 1.1 kann dies nur indirekt erreicht werden, indem aus den Rändern eines solchen Striches selbst (gefüllte) Pfade erzeugt werden.

Mögliche Werte von *stroke-width* sind Längen und 'inherit' (vererbt).

Der Initialwert ist 1, die Eigenschaft wird vererbt und ist animierbar.

In folgendem Beispiel hat das obere Rechteck mit einem orangefarbenen Strich (oder in dem Falle auch Rand oder Rahmen, weil ein Rechteck geschlossen ist) der Strichstärke 15. Das untere Rechteck ist mit einem olivfarbenen Rahmen versehen mit der Strichstärke 30. Der Kreis mit goldenem Rand hat schließlich eine Strichstärke von 19.5. Da spezielle Farbwörter verwendet wurden, handelt es sich um ein Dokument in der Vollversion von SVG 1.1.

```
<rect x="0" y="0" width="800" height="100" rx="60" ry="100"
      fill="rgb(73,10,61)" stroke="DarkOrange" stroke-width="15"/>
<rect x="0" y="0" width="103" height="600" rx="100" ry="75"
```

```
      fill="#490A3D" stroke="Olive" stroke-width="30"/>
<circle cx="0" cy="0" r="200"
      fill="MediumVioletRed" stroke="Gold" stroke-width="19.5"/>
```

Weitere Beispiele:

Eine Schar von Kurven mit abgerundeten Enden wird mit unterschiedlichen Strichfarben gemalt.[12].

Durch gleichzeitige Verkürzung der Kurve und Änderung der Farbe entsteht ein stufiger Farbverlauf, der nicht so einfach mit Farbverläufen in SVG zu erreichen ist.

Kreise und große Strichstärke[13].

Solange die Objekte viel größer sind als die Strichstärke, stellt die Darstellung kein praktisches Problem dar.

Überdeckt der Strich mehr als den halben Durchmesser eine Objektes, so kommt es zu einer nicht lokalen Überschneidung der Striche. Dies bedeutet, dass in dem Bereich nichts mehr von der Füllung übrigbleibt.

Einige Darstellungsprogramme haben damit verschiedene Probleme.

Hier werden einige Kreise mit solch breiten Strichstärken entlang einer logarithmischen Spirale angeordnet. Die Kreise haben eine grüne Füllung einen blauen Strich. Werden dahinter angeordnete rote Kreise sichtbar, ist die Implementierung fehlerhaft, ebenso, wenn etwas vom gelben Hintergrund innerhalb eines Kreises sichtbar wird oder die Kreise gar nicht als Kreise erscheinen.

9.3.3 Strich-Opazität, *stroke-opacity*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

12 http://de.wikibooks.org/wiki/media%3ASVGstroke-width01.svg
13 http://de.wikibooks.org/wiki/media%3ASVGstroke-width02.svg

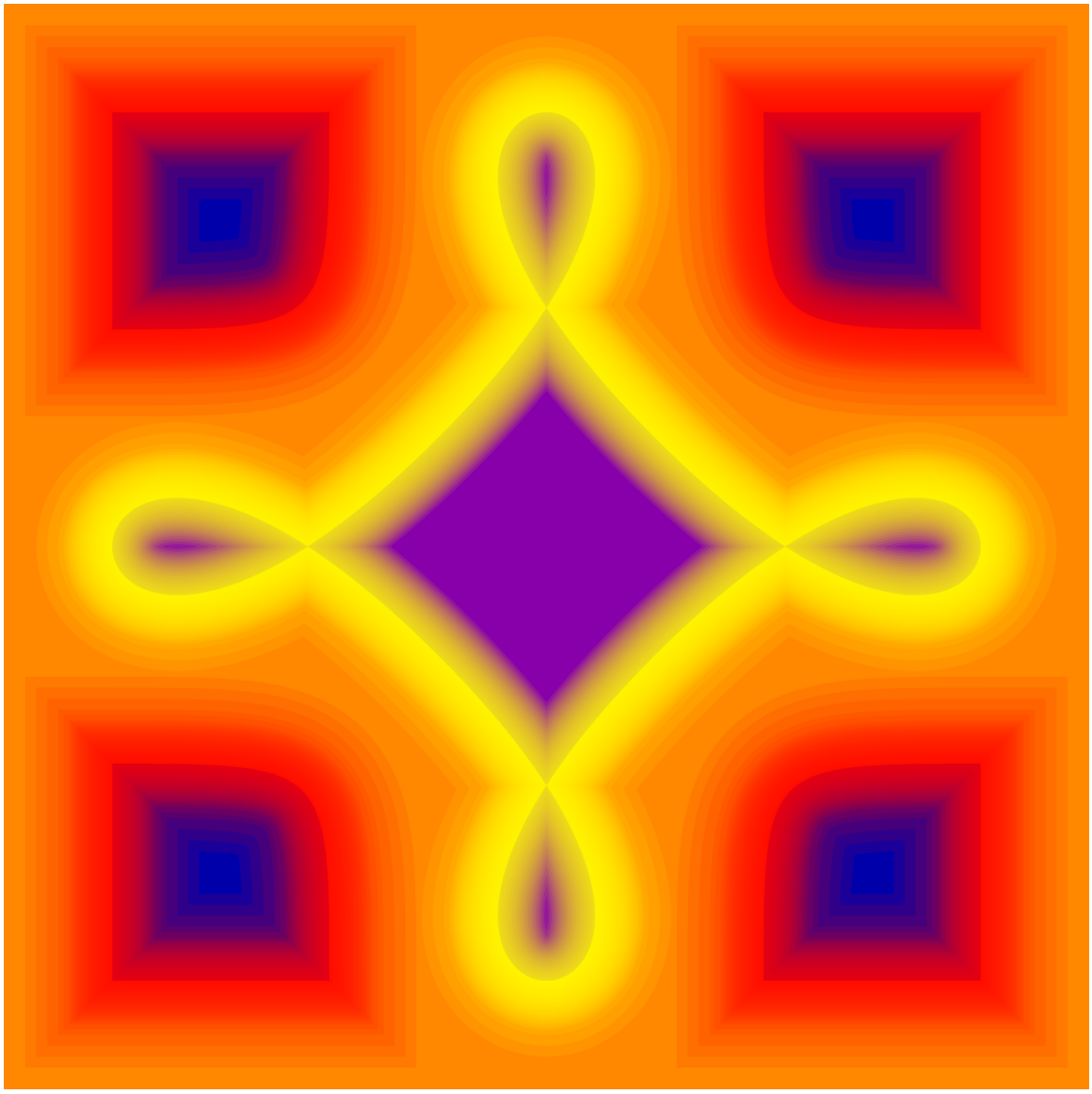


Abb. 110 Beispiel zur Verwendung von *stroke-opacity*

Pfade können auch teilweise durchsichtig gemalt sein, dazu dient die Eigenschaft stroke-opacity mit Werten zwischen 0 und 1. 0 steht für ganz durchsichtig. Andere Werte außerhalb des Intervalls [0;1] werden zum nächstgelegenen zulässigen Wert gerundet.

Der Initialwert ist 1, also undurchsichtig, die Eigenschaft wird vererbt und ist animierbar.

stroke-opacity ist in SVG tiny 1.1 nicht verfügbar.

Beispiel zur Verwendung von *stroke-opacity*[14].

14 `http://de.wikibooks.org/wiki/media%3ASVGstroke-opacity01.svg`

Zwei Pfade mit vierzähliger Drehsymmetrie und Spiegelsymmetrie werden referenziert und jeweils einmal gefüllt und dann mehrmals ohne Füllung, aber mit teiltransparentem Strich mit schmaler werdender Linienbreite übereinandergelegt. So entsteht der Eindruck eines Farbverlaufes senkrecht zu Strich. (Man beachte auch, dass die Eigenschaft *stroke-opacity* selbst nur einmal im Element ***svg*** angegeben ist und von dort aus vererbt wird.)

9.3.4 Strichenden, *stroke-linecap*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

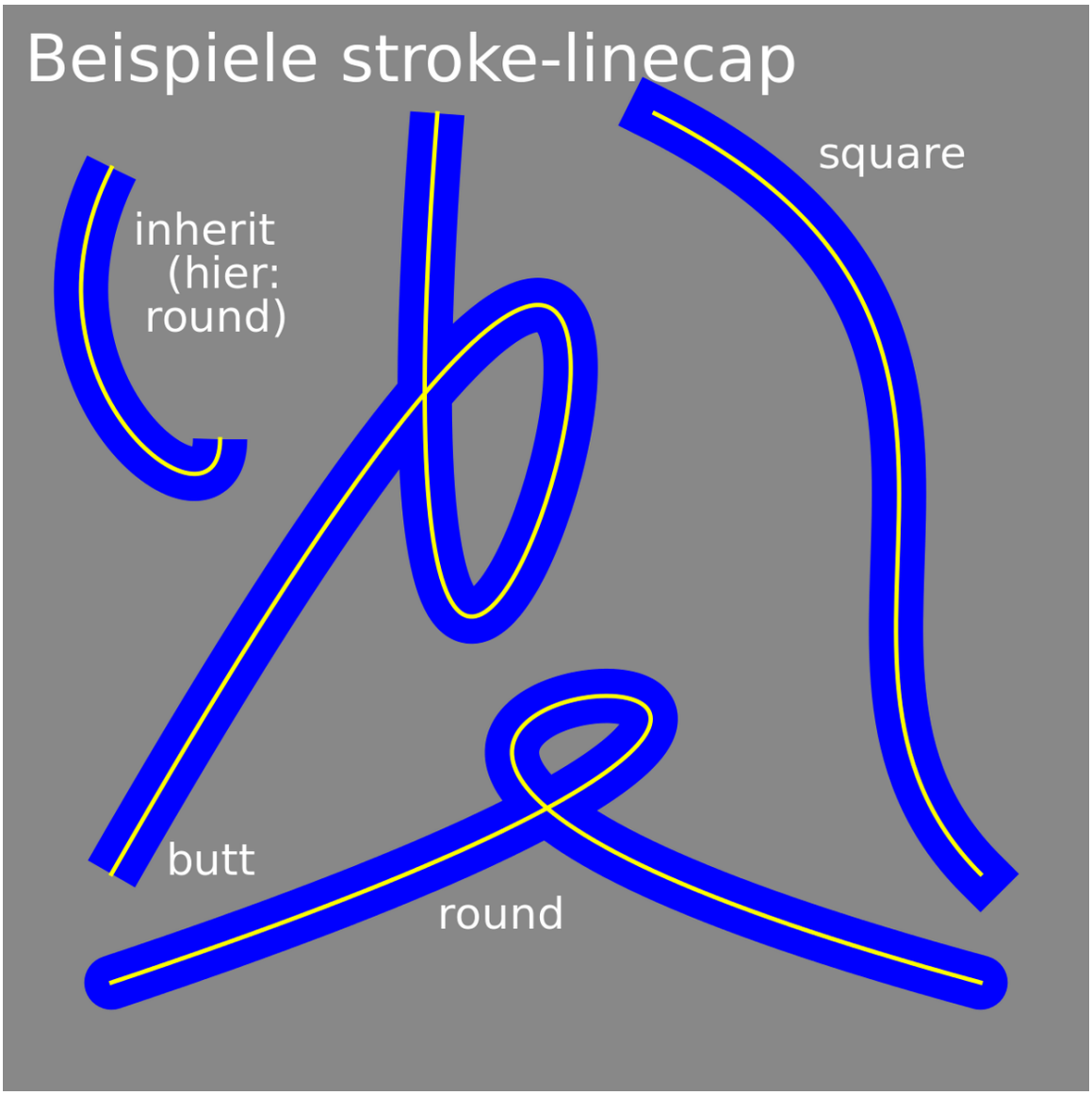


Abb. 111 Beispiel zur Verwendung von stroke-linecap

Mit der Eigenschaft *stroke-linecap* wird festgelegt, wie die Strich- oder Pfadenden von offenen Pfaden oder Unterpfaden darzustellen sind.

Mögliche Werte sind:

inherit

geerbt

round

Halbkreis als Kappe

square

halbes Quadrat als Kappe

butt

keine Kappe

Etwas ausführlicher:

Bei round werden die Pfadenden mit runden Kappen versehen, deren Durchmesser der Strichstärke entspricht.

Bei butt wird keine Kappe an den Pfadenden angefügt.

Bei square werden halbe Quadrate mit der Kantenlänge entsprechend der Strichstärke an den Enden angefügt.

Beim Wert inherit erfolgt eine Vererbung vom Elternelement.

Der Initialwert ist 'butt', die Eigenschaft wird vererbt und ist animierbar.

Beispiel zur Verwendung von stroke-linecap[15].

9.3.5 Strichecken, *stroke-linejoin*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

15 `http://de.wikibooks.org/wiki/media%3ASVGstroke-linecap01.svg`

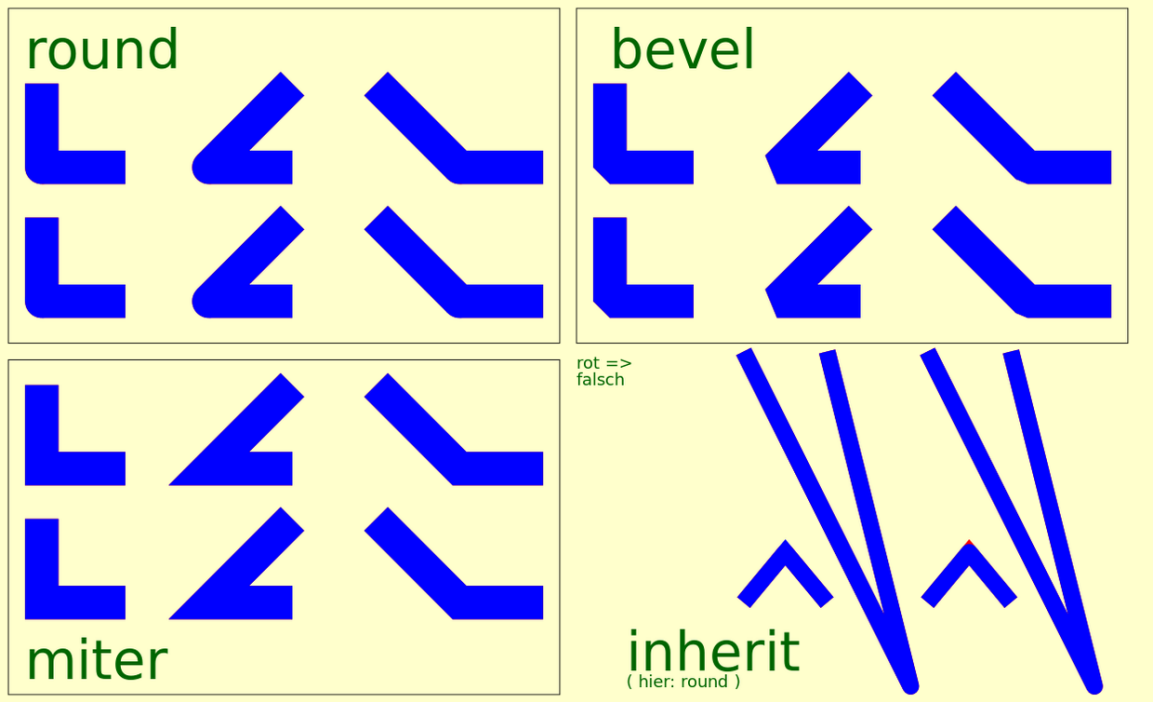


Abb. 112 Beispiel und Test zu *stroke-linejoin*

Dort, wo Pfadesegmente so zusammentreffen, dass Knicke oder Ecken entstehen, kann mit der Eigenschaft *stroke-linejoin* festgelegt werden, wie dies dargestellt werden soll.

Der Wortlaut der Spezifikation impliziert, dass dies auch anwendbar ist, wenn ein Kurvensegment selbst eine Ecke aufweist, was prinzipiell bei einem kubischen Pfadsegment nicht ausgeschlossen ist (ohne es nachgerechnet zu haben, liegt die Vermutung nahe, dass bei einer solchen Ecke der Winkel zwischen den Tangenten immer 0 Grad ist).

Folgende Werte sind möglich:

inherit

geerbt

round

Ecke auf der spitzen Seite abgerundet

miter

Bis zum Schnittpunkt der Tangenten wird der Strich verlängert, die Ecke läuft also spitz zu, eine Gehrung entlang der Winkelhalbierenden der Tangenten der beiden zusammentreffenden Kurvensegmente.

bevel

Statt einer Spitze wie bei miter wird die Ecke mit einer stumpfen Kante beendet, welche senkrecht zur Winkelhalbierenden der Tangenten der beiden zusammentreffenden Kurvensegmente steht.

Initialwert ist 'miter', die Eigenschaft wird vererbt und ist animierbar.

In den Spezifikationen ist lediglich eine Abbildung angegeben, so dass keine weiteren Details schriftlich nachzulesen sind. Immerhin sind die Implementierungen trotzdem einheitlich.

Durch detaillierte Experimente und Nachforschungen ergibt sich jedoch folgendes:

Bei der inneren Ecke laufen die Kanten der Striche immer spitz zusammen, das Verhalten bei der äußeren Ecke hängt vom Wert von stroke-linejoin ab.

- round - die Ecke wird abgerundet. Der Radius der Rundung ist die halbe Linienstärke.
- miter - sofern stroke-miterlimit groß genug ist, laufen die Kanten zu einer Spitze zusammen.
- bevel - die Ecke wird abgestumpft, dazu werden die Kanten bis zum Segmentende ausgeführt und die äußeren Kanten werden dann durch einer gerade Kante miteinander verbunden. Diese Verbindungslinie ist dann senkrecht zur Winkelhalbierenden.

Beispiel und Test zu *stroke-linejoin*[16].

Bei diesen Beispielen wird die konkrete Form des Striches mit einer Kombination von gefüllten Pfaden und Basisformen nachempfunden. Dabei liegt einmal die Nachempfindung oben und einmal der Strich. Die untere rote Version wird daher immer komplett von der oberen blauen Form verdeckt. Wird etwas Rotes sichtbar, liegt ein Implementierungsfehler vor.

9.3.6 Strich-Gehrungsbegrenzung, *stroke-miterlimit*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5
Konqueror (KSVG)	3.2
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

16 `http://de.wikibooks.org/wiki/media%3ASVGstroke-linejoin01.svg`

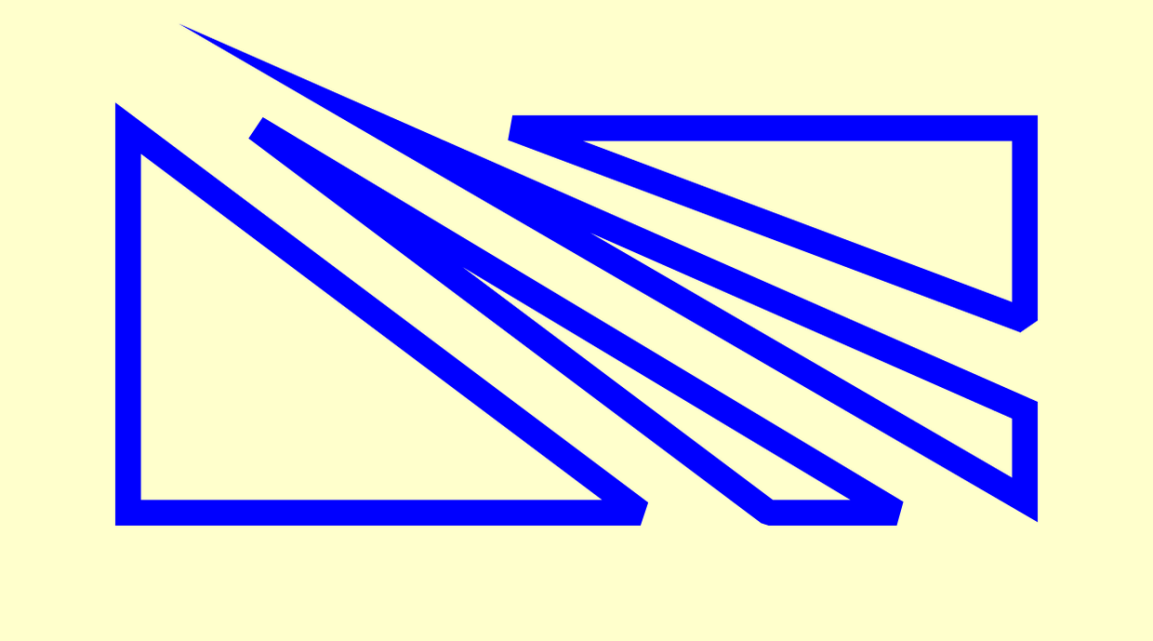


Abb. 113 Einfache Beispiele - einige Dreiecke mit verschiedenen Werten

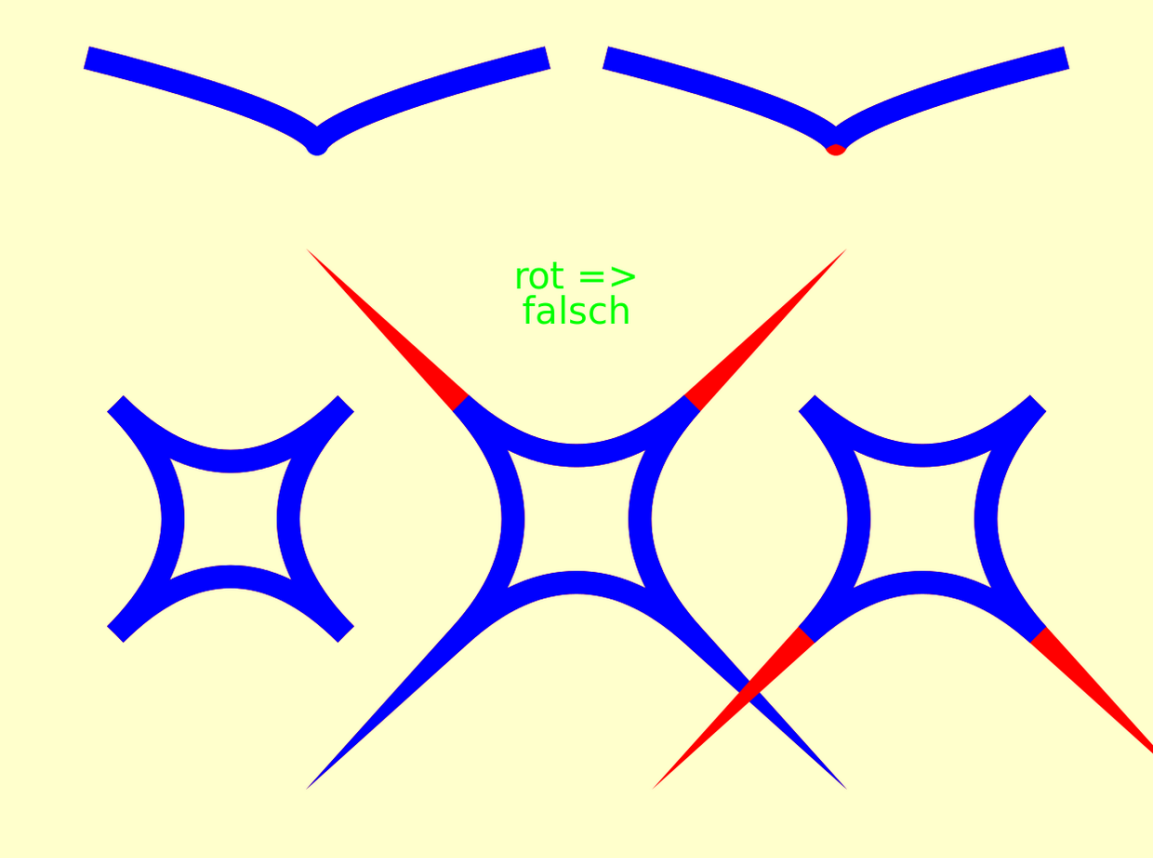


Abb. 114 Test von Implementierungen anhand kritischer Fälle

Bei sehr spitzen Winkeln entsteht bei *stroke-linejoin* 'miter' das Problem, dass die Ecken sehr lang werden, dies wird mit der Eigenschaft *stroke-miterlimit* eingegrenzt. Der Wert ist eine Zahl größer oder gleich 1 oder 'inherit'. Wenn eine Ecke so spitz wird, dass das *stroke-miterlimit* überschritten wird, wird auf den Typ 'bevel' gewechselt.

Die Eigenschaft wird vererbt und ist animierbar. Der Initialwert ist 4.

Werte kleiner als 1 führen in SVG tiny 1.2 dazu, dass der angegebene Wert ignoriert wird, effektiv also der ererbte Wert, beziehungsweise wenn es den nicht gibt der Initialwert gilt. In SVG 1.1 stellt ein Wert kleiner als 1 einen Fehler dar, der zum Abbruch der Darstellung des Dokumentes führen sollte (manche Programme ignorieren diese Implementierungsregel, ein Autor kann sich also nicht darauf verlassen, dass solch ein Fehler unmittelbar sichtbar wird).

Ist m die Länge der Spitze vom äußersten Punkt der Ecke bis zur inneren Ecke, so ergibt sich mittels

'miter' = m / *stroke-width* = $1/\sin(\theta/2)$

ein direkter Zusammenhang zwischen der Länge der Spitze, der Strichstärke, und dem Winkel θ, unter dem die Tangenten der Ecke aufeinanderzulaufen. Bei dem so ermittelten Wert 'miter' wird die Ecke noch spitz dargestellt, wenn dieser Wert nicht größer als *stroke-miterlimit* ist.

Einfache Beispiele - einige Dreiecke mit verschiedenen Werten[17].

In den ***desc***-Elementen stehen nähere Beschreibungen, was wie berechnet wurde. Je nach Winkel und *stroke-miterlimit* sind einige Ecken abgeflacht oder nicht. Insgesamt ist jeweils ein Dreieck zu sehen mit keiner, einer, zwei oder drei spitzen Ecken.

Test von Implementierungen anhand kritischer Fälle[18].

Beispiele zum Testen vom Implementierungen. Es ist jeweils ein Vergleichspfad angegeben. Das Resultat ist dabei immer, dass der blaue Pfad den roten komplett verdeckt. Rot weist auf einen Implementierungsfehler hin.

Einige Implementierungsprobleme, die auf schlechter Rundungsgenauigkeit beruhen, kann ein Autor natürlich leicht umgehen, indem entweder *stroke-linejoin* gleich auf bevel besetzt wird oder aber *stroke-miterlimit* deutlich größer gewählt wird als für die konkrete Form erforderlich ist. Einigen Problemen können Autoren nicht ohne weiteres entgehen, viele davon treten aber in der Praxis eher selten auf.

17 `http://de.wikibooks.org/wiki/media%3ASVGstroke-miterlimit01.svg`
18 `http://de.wikibooks.org/wiki/media%3ASVGstroke-miterlimit02.svg`

9.3.7 Strichmuster, *stroke-dasharray*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

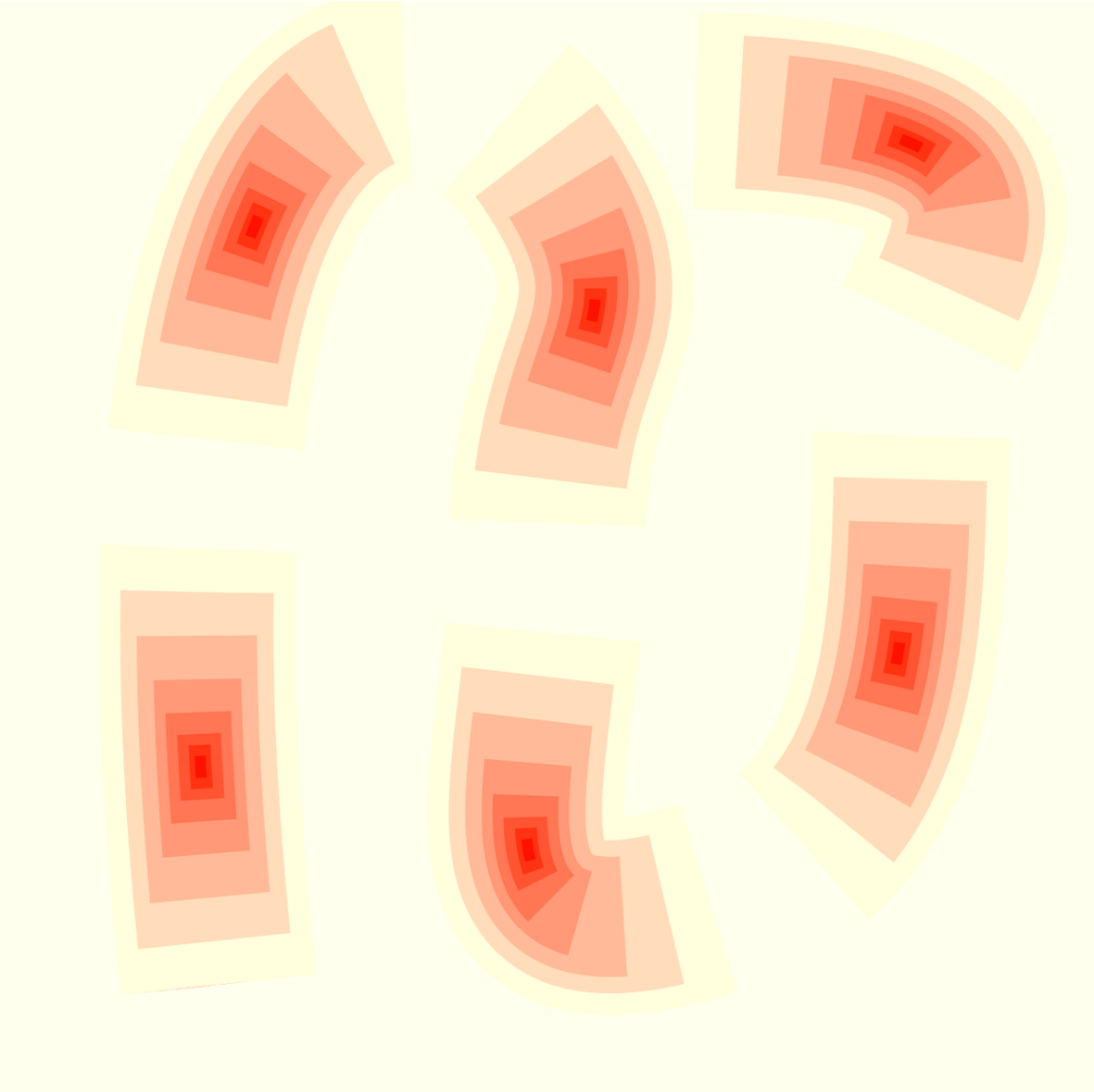


Abb. 115 Beispiel *stroke-dasharray*

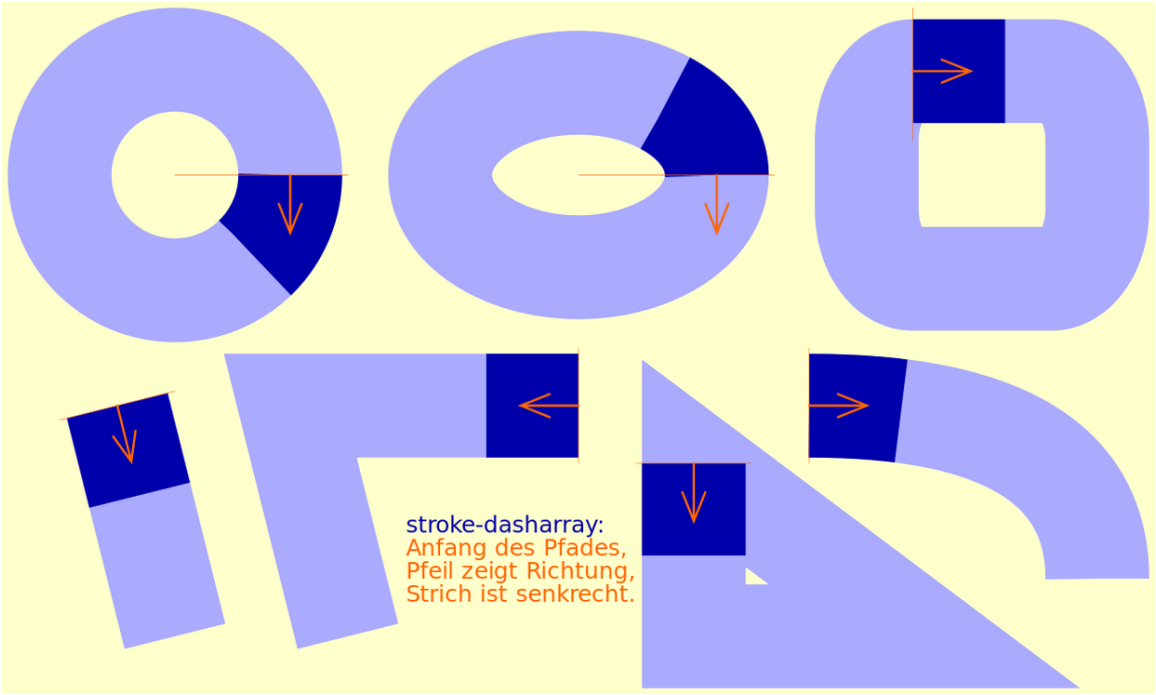


Abb. 116 Beispiel (2) *stroke-dasharray*

Mittels *stroke-dasharray* kann ein Pfad auch recht flexibel gestrichelt werden (grobe Übersetzung: Strich-Strichelaufstellung oder Wegstrichelanordnung). Die Spezifikation selbst nennt es ein Muster von Strichen und Aussparungen. Abwechselnd wird also ein Stück des Striches dargestellt, dann ein Stück ausgepart. Obgleich das recht eindeutig formuliert ist, wird dies zumeist allerdings anders implementiert, wobei die dargestellten Teile wie Unterpfade dargestellt werden, auf die auch *stroke-linecap* angewendet wird. Das gibt zwar die Spezifikation der Eigenschaft nicht her, ist aber mehr oder weniger einheitlich so implementiert (und ist daher als durchgehend falsch anzusehen, kann sogar dazu führen, dass im Falle von *stroke-linecap* square Teile des Musters aus der Strichbreite herausragen, dies also keinesfalls mehr ein Muster des Striches ist).

Auch Wechselwirkungen mit anderen Eigenschaften zum Strich lassen sich zwar teilweise indirekt erschließen, sind aber nicht explizit formuliert. Daher stellt SVG tiny 1.2 explizit klar, dass dies Verhalten noch nicht eindeutig definiert ist und Autoren sich nicht darauf verlassen sollten. Problematisch sind insbesondere die Wechselwirkungen mit *stroke-linecap* vom Typ round oder square, die Wechselwirkung mit *stroke-linejoin* und was genau passiert, wenn mehrere Unterpfade vorliegen. Recht unproblematisch ist das Verhalten bei einfachen Pfaden ohne Knicke und Ecken und mit *stroke-linecap* 'butt'.

Mögliche Werte sind:

inherit

geerbt

none

kein Gestrichel, also Strich komplett durchgezogen

Liste von Längen

Ergibt ein Strichmuster wie angegeben

Die Listenwerte sind in SVG 1.1 jeweils mit einem Komma zu separieren (optional zusätzlicher Leerraum). Bei SVG tiny 1.2 ist auch eine Separation nur mit Leeraum erlaubt.

Ein Wert wie "90,10,50,20" bedeutet also, erst ein Strich der Länge 90, dann eine Aussparung der Länge 10, dann ein Strich der Länge 50, dann eine Aussparung von 20, dann wiederholt sich die Abfolge mit einem Strich der Länge 90 und so weiter. Bei einer ungeraden Anzahl von Listeneinträgen wird die Liste verdoppelt. "50" bedeutet also das gleiche wie "50,50", also abwechselnd ein Strich der Länge 50 und eine Aussparung der Länge 50. Analog: "1,2,3" entspricht "1,2,3,1,2,3". Negative Längen sind in SVG1.1 ein Fehler (Abbruch der Darstellung des Dokumentes). In SVG tiny 1.2 bewirkt dies effektiv nur, dass die Eigenschaft für das Element nicht interpretiert wird, der Strich also durchgezogen wird. Ist hingegen die Summe aller Längen 0, so ist dies gleichbedeutend mit 'none'.

Der Initialwert ist 'none', die Eigenschaft wird vererbt und ist auch animierbar (allerdings nicht additiv; *calcMode* paced ergibt ebenfalls keinen Sinn).

Beispiel *stroke-dasharray*[19]

Ein referenzierter Pfad wird mehrmals mit verschiedener Dicke und anderem Muster übereinandergemalt, um eine interessantere Strukturierung des Strichmusters zu erzeugen. Die Verschiebung der kleineres Strichel lässt sich dann auch noch eleganter mit der Eigenschaft *stroke-dashoffset* lösen. Relevant für den gewünschten Effekt ist zum einen, dass bei allen Pfaden die Summe aus Strichel und Aussparung gleich bleibt und zu anderen bei den kleineren Pfaden mit einer geeignet großen Aussparung begonnen wird, um die Strichel der verschiedenen Pfade relativ zueinander zu zentrieren oder geeignet relativ zueinander zu verschieben.

Beispiel (2) *stroke-dasharray*[20]

Mittels stroke-dasharray wird ermittelt, wo begonnen wird, einen Pfad zu zeichnen. Dies ist in den Spezifikationen genau angegeben. Daher kann diese Datei verwendet werden, um zu testen, ob alles richtig implementiert ist.

Liegt das dunkelblaue stroke-dasharray nicht unter dem jeweiligen orangen Teil, liegt ein Fehler vor. Beginnt das Strichmuster nicht exakt mit dem dünnen orangen Strich, liegt ebenfalls ein Fehler vor.

9.3.8 Strichmusterversatz, *stroke-dashoffset*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

19 `http://de.wikibooks.org/wiki/media%3ASVGstroke-dasharray01.svg`
20 `http://de.wikibooks.org/wiki/media%3ASVGstroke-dasharray02.svg`

Abb. 117 Beispiel *stroke-dashoffset*

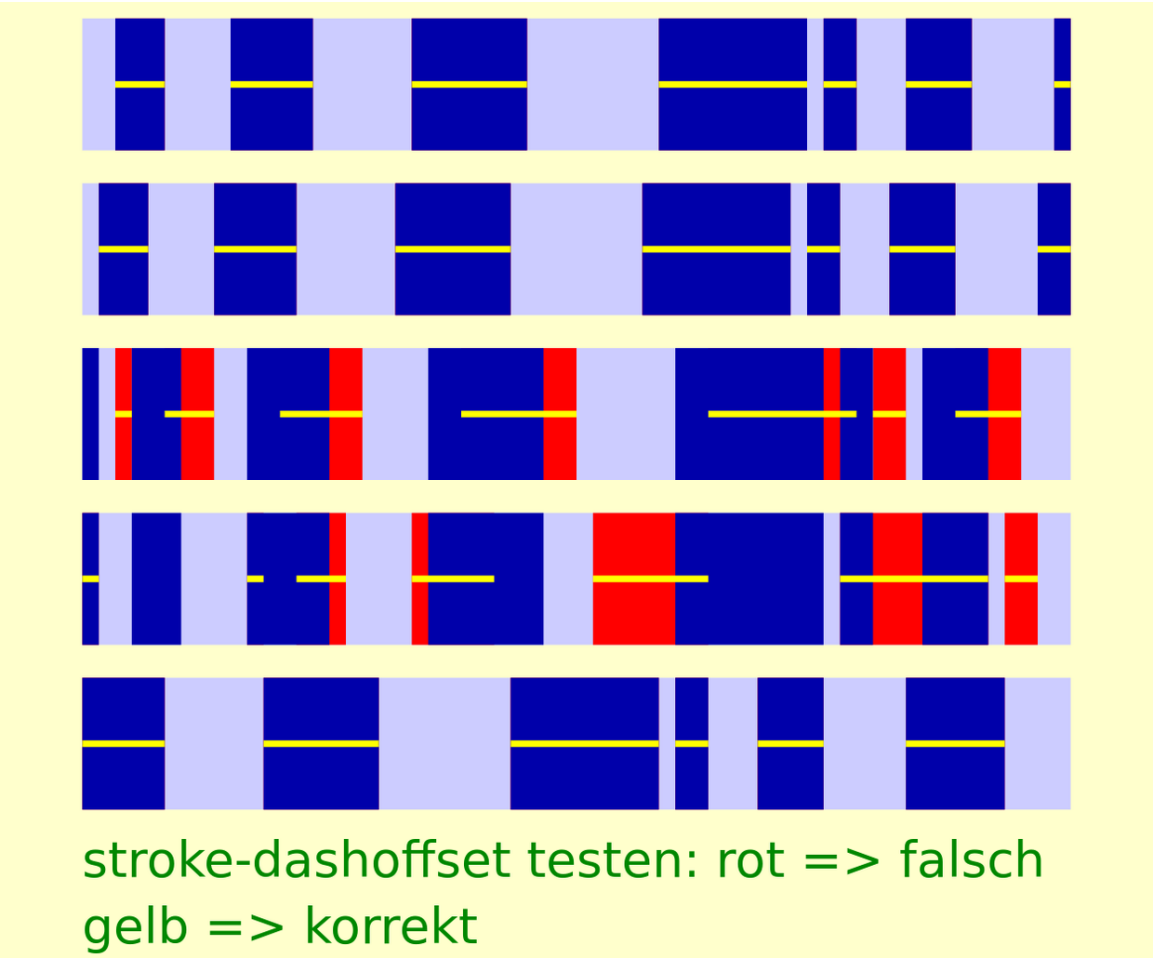


Abb. 118 Beispiel (2) *stroke-dashoffset*

Mit der Eigenschaft *stroke-dashoffset* kann die Anfangsposition des Strichmusters relativ zum Anfang des Pfades verschoben werden.

Der Wert ist inherit oder eine Länge.

Der Initialwert ist 0, die Eigenschaft wird vererbt und ist animierbar.

Für *stroke-dasharray*="100,50" und *stroke-dashoffset*="10" bedeutet dies, dass der erste Strich die Länge 100-10=90 hat. Bei *stroke-dashoffset*="100" beginnt die Darstellung mit einer Aussparung der Länge 50.

Negative Werte sind auch erlaubt, da es sich um ein periodisches Muster ohne Anfang und Ende handelt. *stroke-dashoffset*="-10" bedeutet dann in diesem Falle, dass die Darstellung mit einer Aussparung der Länge 10 beginnt. *stroke-dashoffset*="-100" beginnt mit einem Strich der Länge 50, weil die Aussparung nur 50 lang ist.

Beispiel *stroke-dashoffset*[21].

Ein ähnliches Beispiel wie zuvor für *stroke-dasharray*, nur wird hier die die Zentrierung der Strichel relativ zueinander mit 'stroke-dashoffset*statt mit einer komplizierteren Abfolge* von Werten in *stroke-dasharray* erreicht, was etwas einfacher und eleganter ist. Testweise mag der geneigte Leser auch einmal die jeweiligen Werte für *stroke-dashoffset* so verändern, dass statt einer Zentrierung eine leichte Asymmetrie des Musters entsteht.

Beispiel (2) *stroke-dashoffset*[22].

stroke-dashoffset wird für verschiedene Werte getestet. Dazu werden die Muster für verschiedene Linien mit Pfaden vergleichen, die entsprechende Unterpfade aufweisen.

Eine Lage Vergleichspfade liegt in rot unter dem Muster, eine Lage dünne gelbe Pfade darüber.

Wird etwas Rotes sichtbar, weist dies auf einen Implementierungsfehler hin, ebenso, wenn die dunkelblauen Muster nicht immer exakt unter den gelben Markierungen liegen.

9.3.9 Vektoreffekt, vector-effect

Squiggle (Batik)	-
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

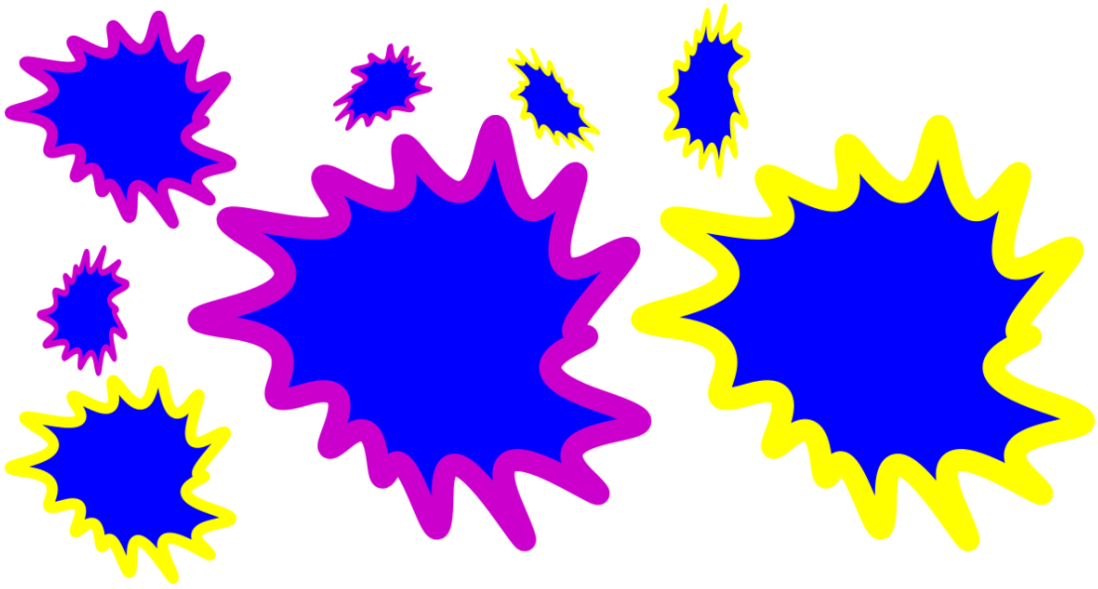


Abb. 119 Beispiel *vector-effect*

21 http://de.wikibooks.org/wiki/media%3ASVGstroke-dashoffset01.svg
22 http://de.wikibooks.org/wiki/media%3ASVGstroke-dashoffset02.svg

Mit SVG tiny 1.2 wurde die Eigenschaft *vector-effect* eingeführt, die demzufolge in SVG 1.1 noch nicht verfügbar ist. Für SVG tiny 1.2 ist auch nur ein Effekt definiert. Weitere sind für zukünftige Versionen von SVG in Vorbereitung.

Mögliche Werte sind:

inherit

geerbt

none

kein Vektoreffekt, Verhalten wie in SVG 1.1

non-scaling-stroke

Angaben zur Strichstärke werden nicht mittransformiert, was zur Folge hat, dass die Strichstärke immer gleich bleibt, auch bei Vergrößerung des Bildes oder bei Verwendung des Attributes *transform*, welches dann zwar noch die Pfaddaten transformiert, nicht aber die Strichstärke, auch Scherungen oder unterschiedliche Skalierungen verzerren die Strichstärke nicht, wie dies im Falle des Wertes 'none' passiert. Die Angabe der Strichbreite bezieht sich also auf das Ausgabekoordinatensystem, nicht auf das lokale Koordinatensystem.

Der Initialwert ist 'none', die Eigenschaft wird nicht vererbt und ist animierbar.

Beispiel *vector-effect*[23]

Die Eigenschaft *vector-effect* wird mit dem Wert 'inherit' auf einen Pfad angewendet. Dieser wird einige Male referenziert und dann transformiert. Diese Kopien haben ebenfalls *vector-effect* gesetzt, entweder auf 'non-scaling-stroke' (magenta Strich) oder 'none' (gelber Strich). *stroke-width* ist auf 5 gesetzt. Dies ist unabhängig von den Transformationen, wenn 'non-scaling-stroke' vorliegt. Daher haben jene Pfade mit magenta Strich alle die gleiche Strichbreite, die von denen mit gelbem Strich variieren. Zudem ändert sich die Strichbreite der Pfade mit magenta Strich nicht, wenn das Dokument vergrößert oder verkleinert wird.

9.4 Eigenschaften des Anzeigebereiches

Ein häufig auftretendes Problem in SVG 1.1 liegt vor, wenn der an sich transparente Anzeigebereich eines Dokumentes eigentlich eine definierte Farbe bekommen soll. Die Lösung besteht in der Praxis meist darin, ein Rechteck ziemlich zu Beginn des Dokumentes einzufügen, welches die passende Füllung hat und deutlich größer ist als der typisch zu erwartende Anzeigebereich des Darstellungsprogrammes.

Mit SVG tiny 1.2 werden neue Eigenschaften eingeführt, mit denen dies Problem elegant gelöst werden kann. In SVG 1.1 stehen diese Eigenschaften noch nicht zur Verfügung.

Die Eigenschaften sind anwendbar auf alle Elemente, die einen (neuen) Anzeigebereich erzeugen. Dies sind in SVG tiny 1.2 die Elemente ***svg***, ***animation***, ***image***, ***video*** und ***foreignObject***.

23 `http://de.wikibooks.org/wiki/media%3ASVGvector-effect01.svg`

9.4.1 Füllung des Anzeigebereiches, viewport-fill

Squiggle (Batik)	-
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

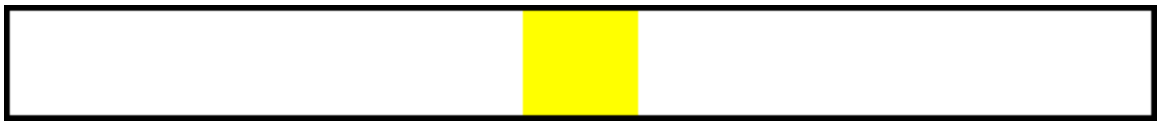


Abb. 120 Beispiel *viewport-fill*

Mittels *viewport-fill* kann der Anzeigebereich mit einer Farbe gefüllt werden.

Mögliche Werte sind:

inherit

geerbt

none

keine Füllung wie in SVG 1.1

currentColor

die aktuelle, mit der Eigenschaft *color* festgelegte Farbe

Farbe

Farbangaben wie für *fill* beschrieben

Der Initialwert ist 'none', die Eigenschaft wird nicht vererbt und ist animierbar.

Beispiel *viewport-fill*[24]

Der Anzeigebereich wird mittels *viewport-fill* hellblau gefüllt. Der Anzeigebereich wird mit einem Rechteck mit schwarzem Strich angedeutet. Der eigentliche Inhalt ist ein gelbes Rechteck, welches bei den gängigen Seitenverhältnissen von Anzeigebereichen von Darstellungsprogrammen zum großen Teil außerhalb des Anzeigebereiches des Dokumentes liegen wird. Weil aber das Seitenverhältnis des Anzeigebereiches des Dokumentes zu erhalten ist, wird so im Anzeigebereich des Darstellungsprogrammes auch etwas dargestellt, was nicht mehr im Anzeigebereich des Dokumentes liegt. In solch einem Falle wird auch der zusätzlich angezeigte Bereich mittels *viewport-fill* mit einer Füllung versehen. Das liegt bei diesem Dokument an dem doch sehr ungewöhnlichen Seitenverhältnis und daran, dass die nicht explizit gesetzten Höhe und Breite jeweils einem Wert von 100% entsprechen, also dem gesamten Anzeigebereich des Darstellungsprogrammes, wenn das Dokument sonst nirgends eingebettet wird.

24 `http://de.wikibooks.org/wiki/media%3ASVGviewport-fill01.svg`

9.4.2 Füll-Opazität des Anzeigebereiches, viewport-fill-opacity

Squiggle (Batik)	-
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

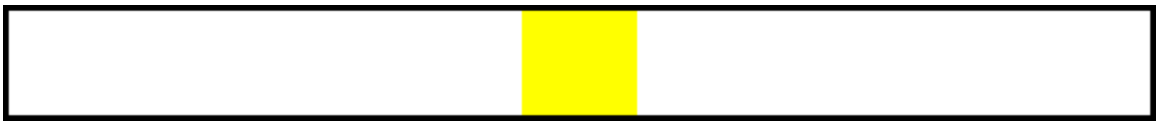


Abb. 121 Beispiel *viewport-fill-opacity*

Der Abzeigebereich kann auch teildurchsichtig gefüllt werden, dazu dient die Eigenschaft *viewport-fill-opacity*.

Mögliche Werte sind:

inherit

geerbt

Zahl

Zahl aus dem Intervall 0 bis 1 einschließlich dieser Werte. Zahlen außerhalb dieses Intervalles werden auf den nächstgelegenen korrekten Wert verändert.

Der Initialwert ist '1', die Eigenschaft wird nicht vererbt und ist animierbar.

Beispiel *viewport-fill-opacity*[25]

Der Anzeigebereich wird mittels *viewport-fill* hellblau gefüllt und durch einen Wert von 0.3 für *viewport-fill-opacity* teilweise durchsichtig.

Die konkrete Farbwirkung hängt also davon ab, welcher Hintergrund im Darstellungsprogramm für den Anzeigebereich verwendet wird. Dies hängt davon ab, ob das Dokument in ein anderes eingebettet ist oder falls nicht, welche Hintergrundfarbe am Darstellungsprogramm als Voreinstellung eingestellt ist. Bei den meisten kann dies vom Nutzer eingestellt werden.

Der Anzeigebereich wird mit einem Rechteck mit schwarzem Strich angedeutet. Der eigentliche Inhalt ist ein gelbes Rechteck, siehe Beispiel zu *viewport-fill*.

25 `http://de.wikibooks.org/wiki/media%3ASVGviewport-fill-opacity01.svg`

9.5 Eigenschaft zur Farbangabe

9.5.1 Farbe, color

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

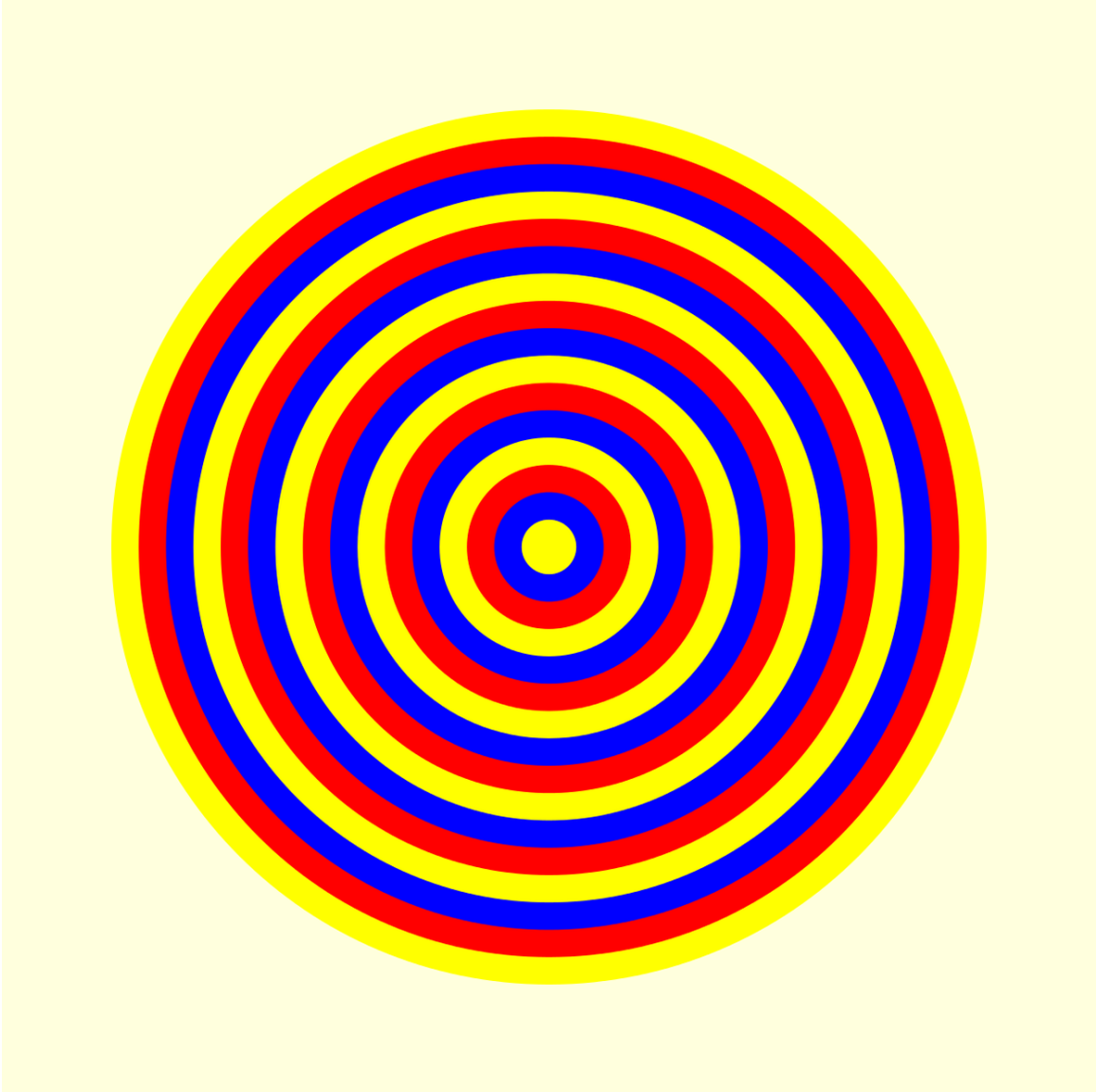


Abb. 122 SVG-Beispiel color und currentColor

Mit der Eigenschaft *color* kann die aktuelle Farbe angegeben werden. Dies kann dann als Farbangabe mittels des Farbwertes 'currentColor' verwendet werden.

Mögliche Werte sind:

inherit

geerbt

Farbe

Farbwert, wie bereits für *fill* erläutert

Der Initialwert ist vom Darstellungsprogramm abhängig, das ist dann also sofern vom Autor nicht anders gesetzt, die vom Nutzer des Darstellungsprogrammes als bevorzugt eingestellte Farbe oder eben die vom Darstellungsprogramm voreingestellte Farbe, wenn der Nutzer diese nicht verändert hat.

Dei Eigenschaft wird vererbt und ist animierbar. Bei einer Animation dieser Eigenschaft bedeutet das insbesonderen, dass der Wert für 'currentColor' sich entsprechend der Änderung von *color* verändert, womit interessante Effekte erreicht werden können.

Eine interessante Anwendung von *color* und 'currentColor' ergibt sich, wenn von vielen Objekte gleichzeitig die Farbe geändert werden können soll, diese aber nicht einfach gruppiert werden können. Da es nur eine solche Eigenschaft gibt, ist das Konzept so nicht beliebig erweiterbar. Für eine statische Änderung kann dies bei der Interpretation von CSS-Selektoren jeweils durch Verwendung von Klassen erfolgen, denen per CSS Eigenschaften zugeordnet werden können. Allerdings ermöglicht es es SMIL-Animation in SVG nicht, Klassen selbst zu animieren. Da CSS-Notation in SVG tiny 1.2 nicht vorgesehen ist, gibt es dort das Element ***solidColor*** mit entsprechenden Eigenschaften, um als Maldienst referenzierbare eigene Farbangaben machen zu können. Da dies auch Animation mit einschließt, ist dies Modell der CSS-Notation mit Klassen überlegen (wobei es inzwischen auch einen Arbeitsentwurf für Animation in CSS3 gibt).

Beispiel zur Verwendung:

```
<g color="blue" fill="yellow">
<animate attributeName="color"
  values="#00a;#f0a;#a0f;#00a" dur="60s"
  repeatDur="indefinite" />
<animate attributeName="fill"
  values="#080;#ff0;#0ff;#080" dur="61s"
  repeatDur="indefinite" />
<circle r="160" />
<circle r="150" fill="red" />
<circle r="140" fill="currentColor" />
<circle r="130" />
<circle r="120" fill="red" />
<circle r="110" fill="currentColor" />
<circle r="100" />
<circle r="90" fill="red" />
<circle r="80" fill="currentColor" />
<circle r="70" />
<circle r="60" fill="red" />
<circle r="50" fill="currentColor" />
<circle r="40" />
<circle r="30" fill="red" />
<circle r="20" fill="currentColor" />
<circle r="10" />
</g>
```

SVG-Beispiel color und currentColor[26].

9.6 Eigenschaften zur Anzeige, Sichtbarkeit und Durchsichtigkeit

Eigenschaften zur Anzeige oder Sichtbarkeit sind bei statischen Dateien eher langweilig, weil nicht angezeigte oder nicht sichtbare Objekte bei einer graphischen Repräsentation nicht besonders viel hermachen. Anders wird es bei nichtstatischen Dateien, die etwa Ereignisse auswerten und animiert sind.

Teilweise durchsichtige Objekte können hingegen auch bereits in statischen Dateien interessant sein. Darunterliegende Objekte scheinen so durch und es können Effekte einfach erreicht werden, die bei beliebigen Pfaden ansonsten einen nicht unerheblichen Aufwand für den Autor bedeuten würden.

9.6.1 Anzeige, display

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

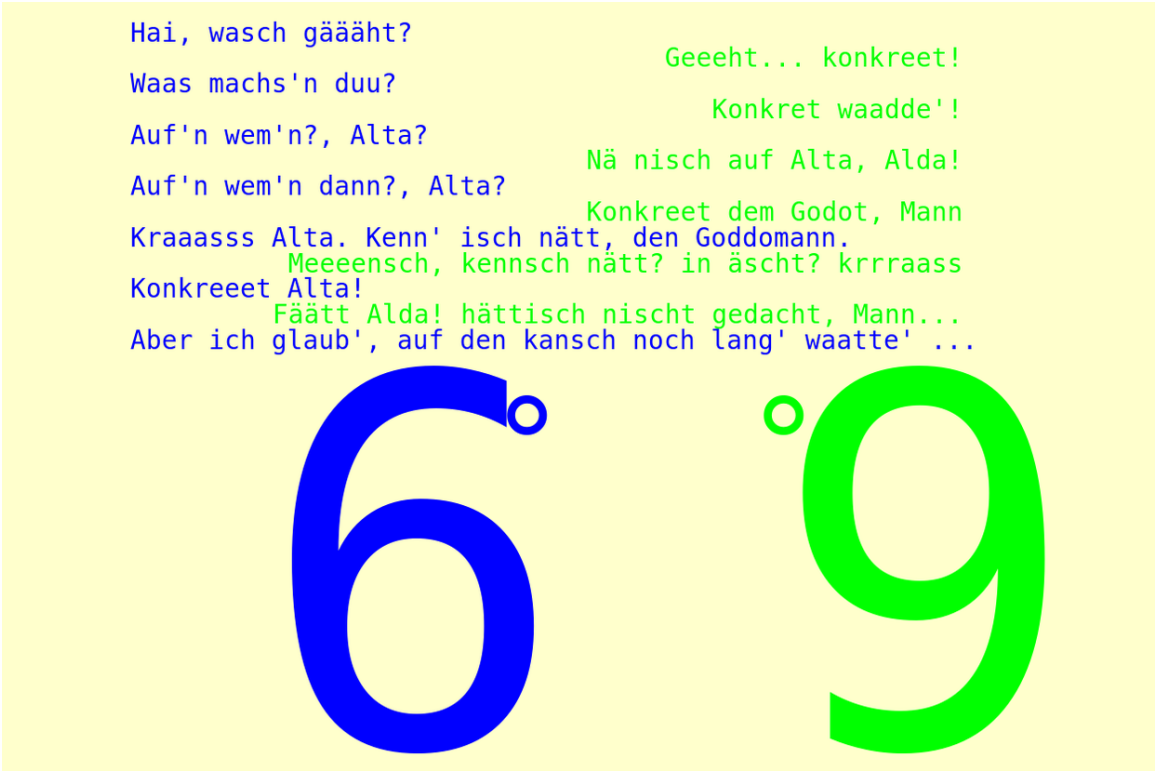


Abb. 123 Minimal-Comic mit Anspielung auf Becketts 'Warten auf Godot'

26 http://de.wikibooks.org/wiki/media%3ASVGcolor01.svg

Ob ein graphisches Element überhaupt direkt angezeigt wird oder nicht, hängt zum einen davon ab, wo es sich befindet - innerhalb eines Elementes ***defs*** wird es zum Beispiel nicht direkt angezeigt werden, kann aber referenziert werden. Dort, wo es generell angezeigt werden kann, kann dies mit der Eigenschaft *display* beeinflusst werden.

Wird für ein Gruppenelement (***g***, ***svg***, ***a***, ***switch***, ***foreignObject***) die Anzeige unterbunden, so gilt dies für alle Kindelemente, unabhängig vom individuellen Wert des jeweiligen Elementes. Ein Element mit unterbundener Anzeige ist auch nicht Empfänger von Ereignissen.

Der konkrete Wert von *display* eines Elternelementes hat allerdings keinen Einfluss darauf, dass ein Element zum Beispiel mit ***use*** referenziert werden kann.

Andersherum kann bei Elementen, die in SVG nicht zur direkten Anzeige gedacht sind, nicht mittels *display* eine Anzeige erzwungen werden. Um dies zu erreichen, müsste die Kennzeichnung des Dokumentes so geändert werden, dass es nur noch als allgemeines XML-Dokument erkannt wird und nicht mehr als SVG-Dokument.

Die Eigenschaft ist in CSS ganz allgemein definiert und enthält daher einige Werte, deren Bedeutung sich für SVG nicht voneinander unterscheidet.

Mögliche Werte sind:

inherit

geerbt

none

keine Darstellung oder Anzeige

inline | block | list-item | run-in | compact | marker | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption

Genau einer der Werte ist anzugeben, um eine Anzeige oder Darstellung zu bewirken.

Der Initialwert ist 'inline', die Eigenschaft wird nicht vererbt, ist aber animierbar.

Beispiele zu Verwendung:

```
<g font-size="15" font-family="fantasy">
<animate attributeName="display" values="inline;none"
dur="2s" repeatDur="indefinite" />
<circle r="100" fill="#fc0" />
<text x="10" y="20">Blinker </text>
<text x="10" y="40">
Unwichtiges<tspan display="none"> kann
gestrichen werden</tspan>!</text>
</g>
```

Der Text 'Blinker' ist also abwechselnd eine Sekunde sichtbar, eine Sekunde unsichtbar. Zweifellos kein wichtiges Beispiel, weswegen da auch der Hinweis 'Unwichtiges!' ausreicht, statt eines ganzen Satzes, nach dem Motto: Was gestrichen ist, kann auch nicht durchfallen!

```
<text x="10" y="20" font-size="15" font-family="fantasy">
```

```
<tspan><animate attributeName="display" values="inline;none"
repeatDur="indefinite" keyTimes="0;0.1" dur="10s" begin="0s" />0</tspan>
<tspan display="none"><animate attributeName="display" values="inline;none"
repeatDur="indefinite" keyTimes="0;0.1" dur="10s" begin="1s" />1</tspan>
<tspan display="none"><animate attributeName="display" values="inline;none"
repeatDur="indefinite" keyTimes="0;0.1" dur="10s" begin="2s" />2</tspan>
<tspan display="none"><animate attributeName="display" values="inline;none"
repeatDur="indefinite" keyTimes="0;0.1" dur="10s" begin="3s" />3</tspan>
<tspan display="none"><animate attributeName="display" values="inline;none"
repeatDur="indefinite" keyTimes="0;0.1" dur="10s" begin="4s" />4</tspan>
<tspan display="none"><animate attributeName="display" values="inline;none"
repeatDur="indefinite" keyTimes="0;0.1" dur="10s" begin="5s" />5</tspan>
<tspan display="none"><animate attributeName="display" values="inline;none"
repeatDur="indefinite" keyTimes="0;0.1" dur="10s" begin="6s" />6</tspan>
<tspan display="none"><animate attributeName="display" values="inline;none"
repeatDur="indefinite" keyTimes="0;0.1" dur="10s" begin="7s" />7</tspan>
<tspan display="none"><animate attributeName="display" values="inline;none"
repeatDur="indefinite" keyTimes="0;0.1" dur="10s" begin="8s" />8</tspan>
<tspan display="none"><animate attributeName="display" values="inline;none"
repeatDur="indefinite" keyTimes="0;0.1" dur="10s" begin="9s" />9</tspan>
</text>
```

Ein simpler Zähler von 0 bis 9 mit Wiederholung.

Beispiel: Minimal-Comic mit Anspielung auf Becketts 'Warten auf Godot'[27].

Anwendung der Eigenschaft *display* mit Animation.

Das Dokument ist so angelegt, dass bei einer kompletten Implementierungslücke hinsichtlich Animation eine statische Version des Comics verfügbar ist.

9.6.2 Sichtbarkeit, visibility

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

27 http://de.wikibooks.org/wiki/media%3ASVGdisplay01.svg

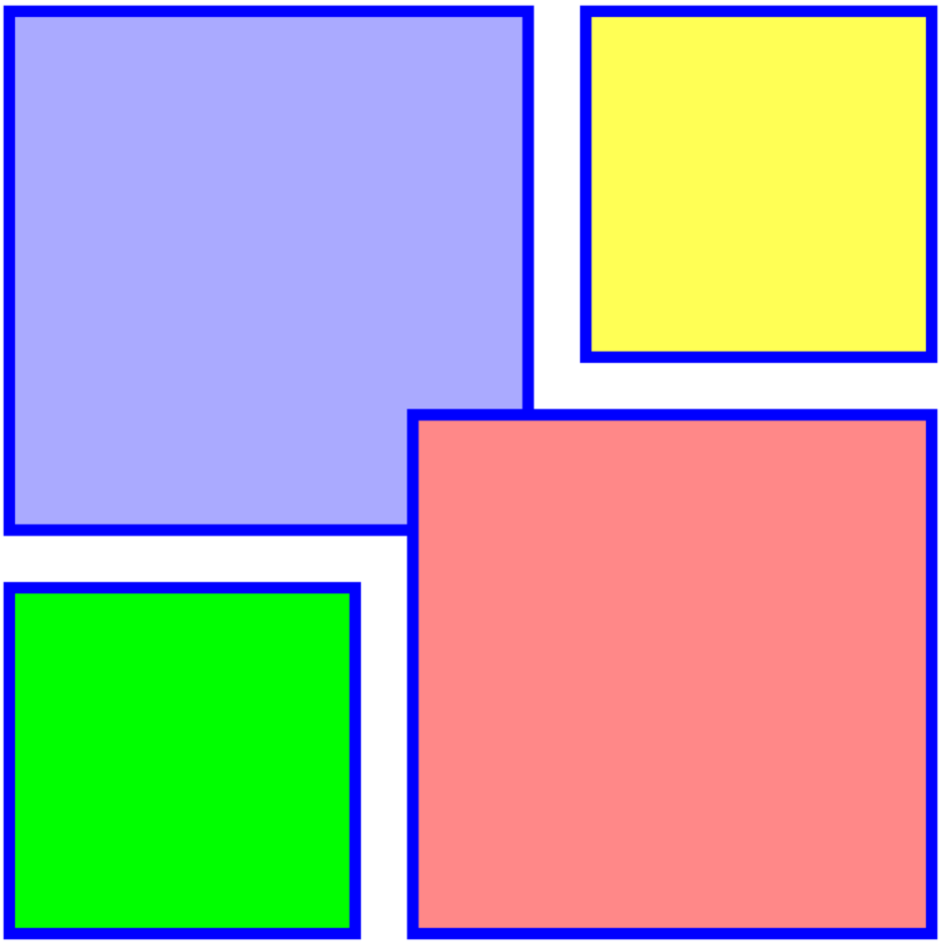


Abb. 124 Verwendung von *visibility*

Anders als *display* hat *visibility* nur Einfluss auf die Sichtbarkeit von Elementen, diese Werte aber nicht komplett aus der Darstellung entfernt, was etwa Einfluss auf Elemente wie ***tspan*** hat, weil auch unsichtbare Glyphen die aktuelle Textposition verändern, nicht dargestellte Glyphen tun dies hingegen nicht. Grob gesagt bleibt der Platz für unsichtbare Element reserviert, für nicht dargestellte nicht. Nicht sichtbare Elemente können trotzdem Ziel von Ereignissen sein, so dass ein unsichtbares Element platziert werden kann, um Ereignisse abzufangen, die zum Beispiel für sichtbare Elemente unerwünscht wären.

In SVG tiny 1.2 gibt es auch die Möglichkeit hörbarer Inhalte. Wie der Name der Eigenschaft bereits impliziert, hat die Frage der Sichtbarkeit nichts mit der der Hörbarkeit zu tun, insofern hat die Eigenschaft *visibility* auch keinen Einfluss auf hörbare Elemente.

Mögliche Werte sind:

inherit

geerbt

visible

sichtbar

hidden | collapse

Genau einer der Werte ist anzugeben,dann ist das Objekt nicht sichtbar.

Initialwert ist 'visible', die Eigenschaft wird vererbt und ist animierbar.

Beispiel zu Verwendung:

```
<text x="10" y="40" font-size="15"
      font-family="fantasy">Beispiel
 <tspan visibility="hidden">zur Verwendung von</tspan>
 visibility</text>
```

Beispiel: Verwendung von *visibility*[28].

Von fünf (!) Quadraten kann die Füllung durch Klicken oder Aktivieren verändert werden.

Das kleine Quadrat in der Mitte ist anfänglich nicht sichtbar, kann aber Ziel von Klicks sein oder kann aktiviert werden. Dies bewirkt eine Änderung der Füllung der großen Quadrate. Gleichzeitig bewirkt dies aber auch eine Änderung der Sichtbarkeit und zudem wird für 20s die Möglichkeit deaktiviert, Ziel von Ereignissen zu sein.

Beim Quadrat links unten bewirkt ein Ereignis anfangs das Setzen der Füllung auf 'none' für 5s. Weil *pointer-events* auf 'visiblePainted' gesetzt ist, lässt sich dies nur durch ein abermaliges Ereignis verlängern, wenn entweder der Strich angeklickert oder aktiviert wird oder aber die 5s gewartet wird, bis die Füllung wieder einen anderen Wert erhält.

Beim Quadrat rechts oben bewirkt ein Ereignis anfangs das Setzen der Sichtbarkeit auf 'hidden' für 5s. Weil *pointer-events* auf 'painted' gesetzt ist, lässt sich dies jederzeit durch ein weiteres Ereignis verlängern.

9.6.3 Opazität, opacity

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	1.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

28 http://de.wikibooks.org/wiki/media%3ASVGvisibility01.svg

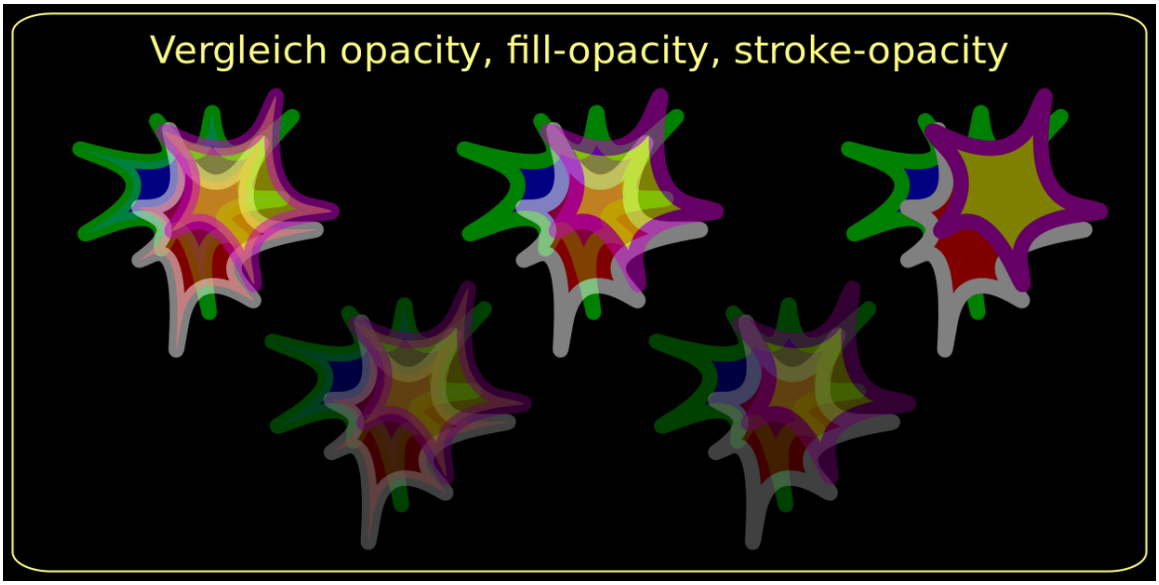


Abb. 125 Vergleich opacity, fill-opacity, stroke-opacity

Während *fill-opacity* und *stroke-opacity* nur den Grad der Durchsichtigkeit der Füllung, beziehungsweise des Striches individueller Elemente beeinflussen, wird mit *opacity* die Durchsichtigkeit des gesamten Elementes oder einer ganzen Gruppe von Elementen beeinflusst. Diese Angabe zur Durchsichtigkeit wird verarbeitet, nachdem bereits die Füll- und Stricheigenschaften und gegebenenfalls die Anordnung in der Gruppe verarbeitet wurden.

Während *visibility* angibt, ob das Element oder die Gruppe überhaupt sichtbar ist, geht es bei *opacity* nur darum, wie durchsichtig es ist. Auch komplett durchsichtige Elemente können so das Ziel von Ereignissen sein, die auf sichtbare Elemente wirken, sofern sie nicht mittels *visibility* explizit unsichtbar gemacht werden oder gar mittels *display* gar nicht dargestellt werden.

Mögliche Werte sind 'inherit' für einen geerbten Wert oder eine Zahlenangabe aus dem Intervall 0 bis 1; 0 für komplett durchsichtig, 1 für komplett undurchsichtig. Werte außerhalb des Intervalles werden auf den nächsten zulässigen Wert, also 0 beziehungsweise 1 korrigiert, bevor die Eigenschaft angewendet wird.

Der Initialwert ist 1, die Eigenschaft wird nicht vererbt und ist animierbar.

opacity ist in SVG tiny 1.1 nicht verfügbar und ist in SVG tiny 1.2 darauf beschränkt, auf ***image***-Elemente angewendet zu werden. Diese Einschränkung hängt damit zusammen, dass die Anwendung dieses Attributes eine größere Menge Rechnerei zur Folge haben kann, die zumindest für Mobiltelephone eine Herausforderung sein kann. Wenn es also für eine Anwendung weniger rechenintensive Umsetzungen gibt, sind diese sicher auch in der Vollversion von 1.1 dem Einsatz von *opacity* vorzuziehen.

Beispiel: Vergleich opacity, fill-opacity, stroke-opacity[29].

29 `http://de.wikibooks.org/wiki/media%3ASVGopacity01.svg`

Zum Vergleich der Eigenschaften *opacity*, *fill-opacity* und *stroke-opacity* werden fünf Gruppen von Formen jeweils unterschiedlich mit diesen Eigenschaften ausgestattet. Links oben sind *fill-opacity* und *stroke-opacity* jeweils einzeln auf jede Form angewendet. In der Mitte oben ist *opacity* jeweils einzeln auf jede Form angewendet. Rechts oben ist *opacity* auf die ganze Gruppe angewendet. Links unten sind *fill-opacity* und *stroke-opacity* jeweils einzeln auf jede Form angewendet und *opacity* auf die ganze Gruppe. Rechts unten ist *opacity* jeweils einzeln auf jede Form angewendet und *opacity* auf die ganze Gruppe.

9.6.4 Überlauf, overflow

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	3.2
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

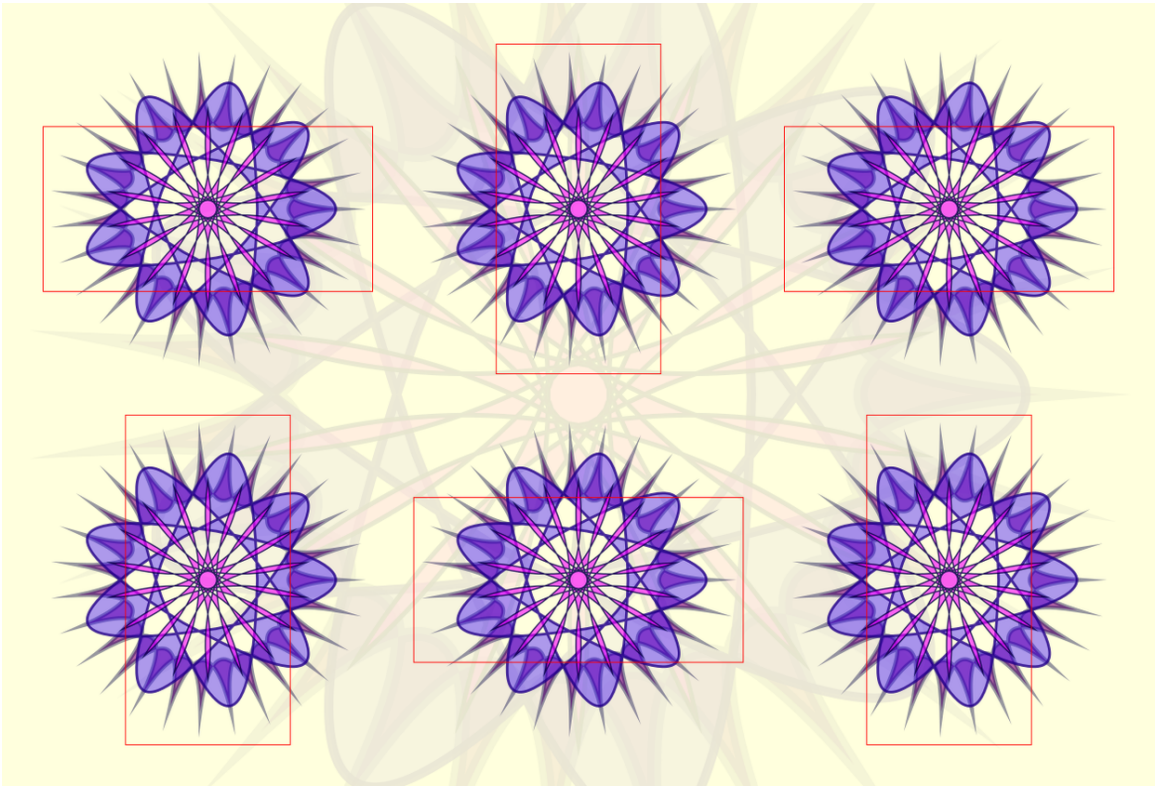


Abb. 126 Ein Symbol steht über

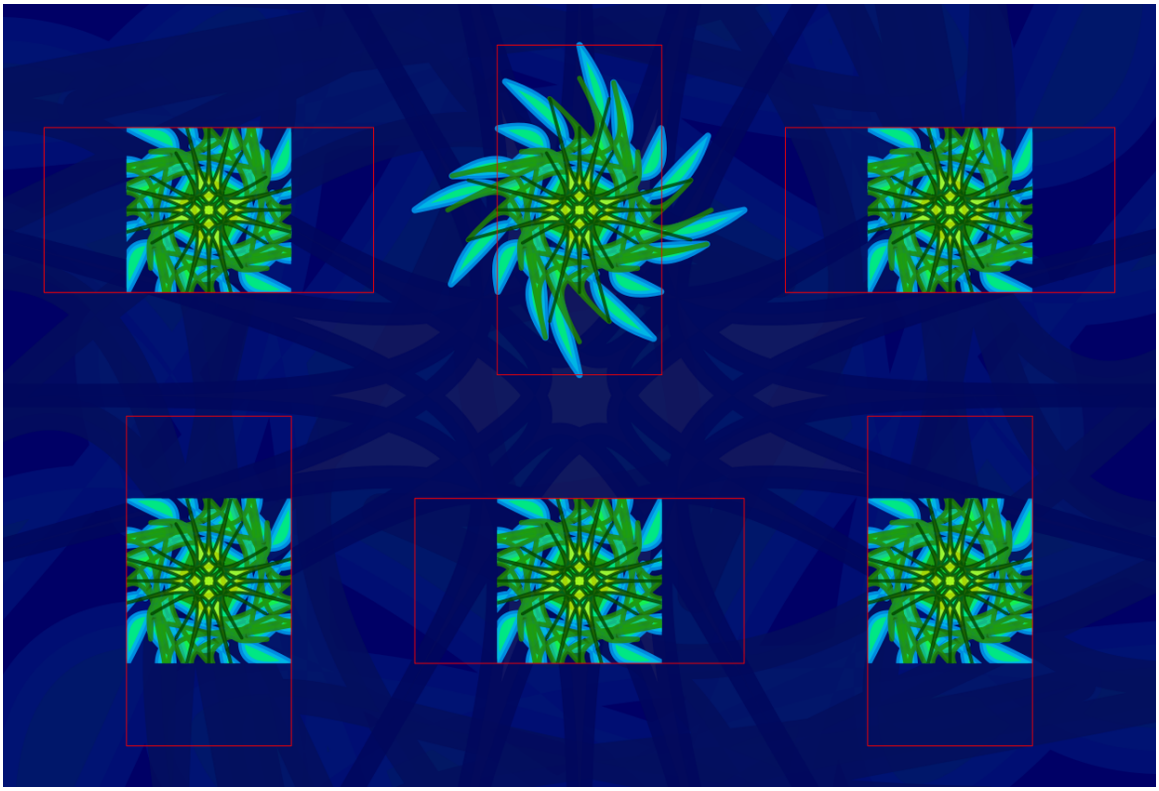


Abb. 127 Ein Symbol steht über (2)

Mit der Eigenschaft *overlow* kann angegeben werden, ob überstehende oder überlaufende Teile eines Elementes angezeigt werden sollen oder nicht. Dies gilt für Elemente, die einen eigenen Anzeigebereich öffnen, außerdem auch bei Markierungen und Mustern, wobei das Verhalten bei Mustern unterspezifiziert und uneinheitlich implementiert ist, es ist nicht eindeutig geregelt, wie überstehender Inhalt bei einem Muster miteinander wechselwirkt. Die Situation ist bei Mustern also derzeit zu vermeiden.

Die Eigenschaft ist nicht in SVG tiny 1.1 und tiny 1.2 verfügbar.

Elemente, die in der Vollversion einen eigenen Anzeigebereich eröffnen, sind: ***svg***, ***symbol***, ***image*** und ***foreignObject***.

Mögliche Werte sind:

inherit

geerbt

hidden

Überstehende Bereiche werden nicht angezeigt.

scroll

Überstehende Bereiche werden nicht angezeigt. Es gibt allerdins auch keine Rollbalken, wie man vom Namen her und von CSS her erwarten könnte.

visible

Überstehende Bereiche werden angezeigt, sofern sie in einem anzuzeigenden Bereich liegen

auto

Überstehende Bereiche werden angezeigt, sofern sie in einem anzuzeigenden Bereich liegen

Handelt es sich beim betroffenen Element um das ***svg***-Hauptelement, welches in einem anderen XML-Dokument mit Stilvorlageninterpretation eingebettet ist, entspricht der Ausschnitt im Falle von 'hidden' und 'scroll' dem des anfänglichen Anzeigebereiches. Ansonsten wird die Eigenschaft verwendet, wie in CSS definiert. Wird ist für das XML-Dokument keine Stilvorlageninterpretation vorgesehen oder das ***svg***-Hauptelement ist nicht eingebettet, so wird die Eigenschaft ignoriert und der Ausschnitt ergibt sich aus dem anfänglichen Anzeigebereich.

Der Initialwert ist in CSS 'visible', allerdings wird dies durch die Stilvorlage des SVG-Darstellungsprogrammes überschrieben, dort ist der Wert 'hidden'. Die Eigenschaft wird nicht vererbt und ist animierbar.

Ein Symbol steht über[30]:

Einige Beispiele für das Zusammenspiel von ***symbol*** mit der Eigenschaft *overflow*, Überstehen bedingt durch *preserveAspectRatio* 'slice'.

Ein Symbol steht über (2)[31]:

Einige Beispiele für das Zusammenspiel von ***symbol*** mit der Eigenschaft *overflow*, wenn die Pfade über die viewBox überstehen.

Der Anzeigebereich ist jeweils rot gerandet angedeutet. Die beiden Beispiele rechts haben keine Anzeige außerhalb des Anzeigebereiches, bei den vier anderen wird der Überlapp dargestellt.

9.7 Texteigenschaften

9.7.1 Allgemeine und spezielle Eigenschaften auf Text angewendet

Da Text in SVG ebenfalls als Graphik dargestellt wird, kann Text ebenfalls mit den Eigenschaften zur Füllung und zum Strich versehen werden. In SVG tiny 1.1 ist allerdings nicht vorgesehen, dass Eigenschaften zum Strich auch auf Text wirkt, weil die verwendeten Schriften auf Mobiltelephonen dies zur Zeit der Herausgabe der Spezifikation vermutlich noch nicht hergegeben haben oder aber dies die Prozessoren von Mobiltelephonen zu der Zeit zu stark gefordert haben mag.

Sofern Eigenschaften vererbbar sind können Texte nicht nur direkt durch Angabe der Präsentationsattribute in Elementen wie ***text*** oder ***tspan*** notiert werden, sondern auch in dem übergeordneten Element zum Beispiel der Gruppierung ***g*** angegeben werden. So können Füllung, Strich, Schriftart und Zeichenhöhe angegeben werden:

30 http://de.wikibooks.org/wiki/media%3ASVGoverflow01.svg
31 http://de.wikibooks.org/wiki/media%3ASVGoverflow02.svg

```
<g fill="red" fill-opacity="0.5"
stroke="blue" stroke-width="0.2"
stroke-linejoin="round" stroke-linecap="round"
stroke-opacity="0.75"
font-family="fantasy" font-size="25">
<text x="10" y="50">Hallo</text>
<text x="40" y="60">Welt<tspan font-size="50">!</tspan></text>
</g>
```

Neben *font-family* und *font-size* gibt es einige weitere Eigenschaften, die nur auf Text als Graphik wirken und nicht auf anderen Elemente.

9.7.2 Auswahl der Schriftart

Schriftfamilie, font-family

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

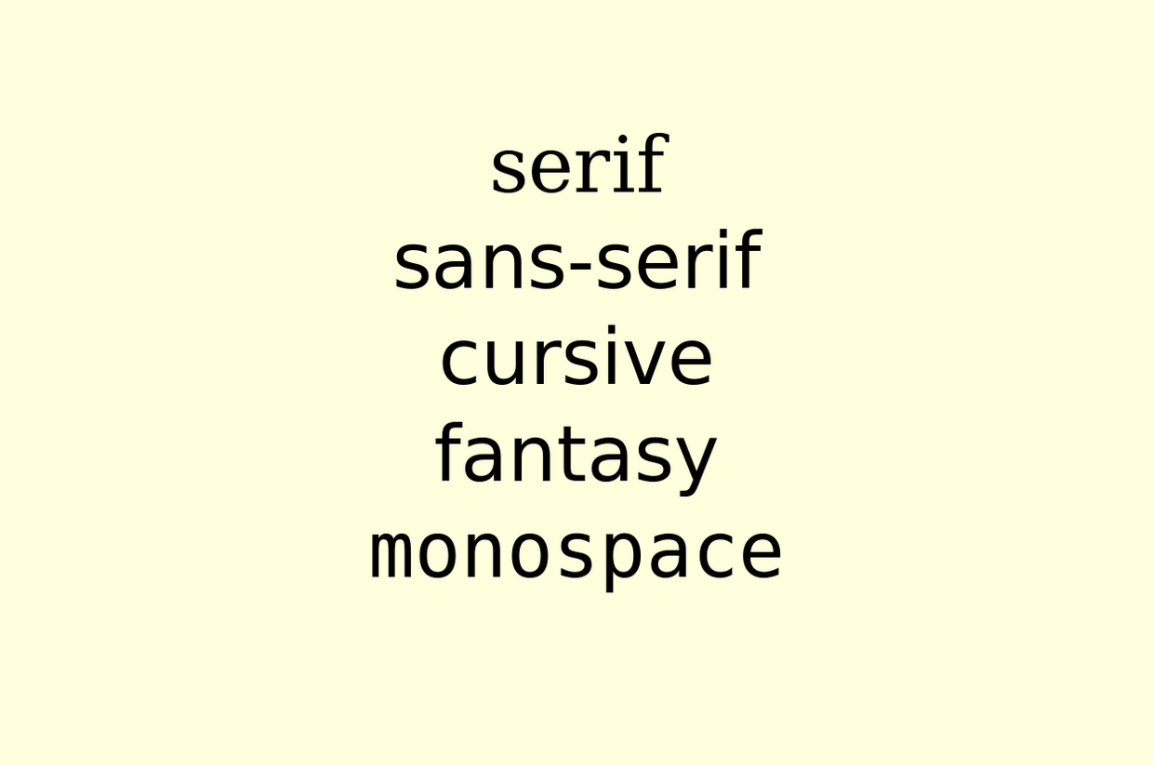


Abb. 128 Beispiel generische Schriftfamilien

Mit der Eigenschaft *font-family* wird die Schriftart angegeben. Der Wert von *font-family* ist eine mit Komma separierte Liste von Namen von Schriftarten oder generischen Schriftfamilien oder das

Schlüsselwort 'inherit' für vererbt. Generische Schriftfamilien sind gekennzeichnet durch folgende Schlüsselwörter, die nicht in Anführungszeichen zu setzen sind: serif, sans-serif, cursive, fantasy, and monospace. Jedes Darstellungsprogramm sollte eine Schriftart für jede dieser generischen Schriftfamilien verfügbar haben, nicht aber zwangsläufig eine spezielle Schriftart.

Enthält der Name einer Schriftart Leerzeichen, so sollte der Name in Anführungszeichen stehen.

Findet das Darstellungsprogramm in der ersten angegebenen Schriftart einen Glyphen nicht, wird in der nächsten gesucht und so weiter. Die letzte Angabe in der Liste sollte daher immer eine generische Schriftfamilie sein, um sicherzustellen, dass überhaupt was dargestellt wird.

Der Initialwert ist vom Darstellungsprogramm abhängig (zum Beispiel, was der Nutzer dort eingestellt hat). Die Eigenschaft wird vererbt und ist animierbar.

Folgende Angabe wäre also suboptimal, weil die generische Schriftfamilie fehlt und der Name der Schriftart Leerzeichen enthält, aber nicht in Anführungszeichen steht (die vom Präsentationsattribut gelten nicht, gemeint sind zusätzliche innerhalb der Liste):

```
<text x="300" y="150" font-family="Comic Sans MS">Weitere Projekte</text>
```

Wichtig ist hierbei also, korrekte Angaben zu machen anzugeben, so kann die Angabe "'Comic Sans MS'" zwar zum richtigen Ergebnis führen - dies muss jedoch nicht sein, wenn dieser Zeichensatz zum Beispiel nicht auf diesem Computer installiert ist oder eine Glyphe benötigt wird, die darin nicht enthalten ist. Daher sollte von der Möglichkeit Gebrauch gemacht werden, eine Liste anzugeben:

```
<text x="300" y="150"
 font-family="Frutiger,'Comic Sans MS',Arial,Helvetica,sans-serif">Weitere
Projekte</text>
```

oder vielleicht besser:

```
<text x="300" y="150"
font-family="'Frutiger Capitalis','Alexa Regular','Comic Strip','Comic Sans
MS',fantasy"
>Weitere Projekte</text>
```

weil 'Comic Sans MS' zur fantasy-Schriftfamilie gehört und dies dem Autor wichtig sein könnte.

Beispiel generische Schriftfamilien[32]

Vorstellung der fünf generischen Schriftfamilien. Welche Schriftart die jeweilige Schriftfamilie repräsentiert, kann der Nutzer in der Regel beim Darstellungsprogramm selbst auswählen.

- serif - Schriftart mit Serifen, also kleine Füßchen, altbekannt ist zum Beispiel die Schriftart 'Times'.
- sans-serif -Schriftart ohne Serifen, also keine Füßchen, altbekannt ist zum Beispiel die Schriftart 'Helvetica'.
- cursive - schräggestellt, wird oft aus einer anderen Schriftart durch Scherung gewonnen.
- fantasy - stark dekorative, spielerische Schriftart.
- monospace - nichtproportionale Schriftart, altbekannt ist zum Beispiel die Schriftart 'Courier'.

32 http://de.wikibooks.org/wiki/media%3ASVGfont-family01.svg

Schriftgröße, font-size

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

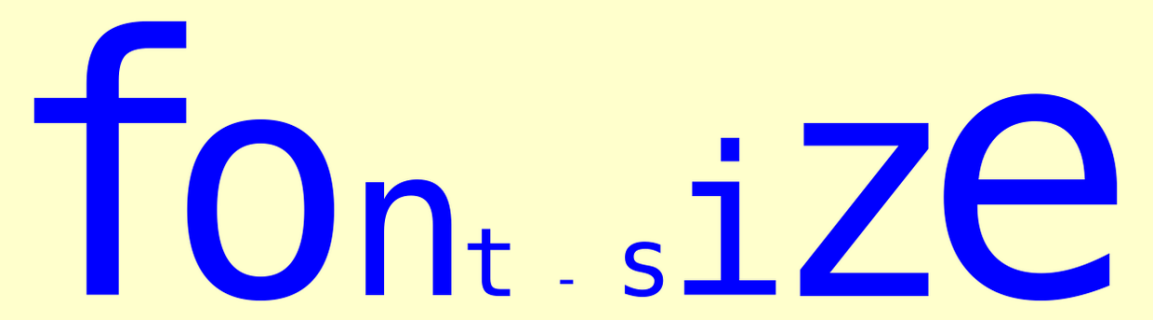


Abb. 129 Beispiel Schriftgröße

Die Schriftgröße wird durch die Eigenschaft *font-size* angegeben. Der mögliche Wert hängt vom verwendeten Profil ab.

inherit

geerbt

absolute Größe

Eines der Schlüsselwörter: xx-small, x-small, small, medium, large, x-large, xx-large

relative Größe

Eines der Schlüsselwörter: larger,smaller

Längenangabe

Eine nicht negative Länge

Prozentangabe

Zum Beispiel 120%, relativ zur Schriftgröße des Elternelementes (nicht in den tiny-Profilen)

Der Initialwert ist medium. Die Eigenschaft wird vererbt und ist animierbar.

Bei den tiny-Profilen sind wie gehabt Einheiten bei der Längenangabe nicht möglich, die Zahl ist also in lokalen Einheiten. Aufgrund der Skalierungsfunktionen von SVG kommt der Schriftgröße in SVG eine etwas andere Bedeutung zu als bei Angaben für (X)HTML-Dokumente. Bei CSS+(X)HTML sind eher absolute oder relative Größen, Prozentangaben und Längeneinheiten wie em oder ex sinnvoll, damit die vom Nutzer als optimal eingestellte Schriftgröße Berücksichtigung findet oder diese nicht ungeeignete Angaben einfach überschreibt. Indes in SVG sind eher lokale Einheiten angemessen, da es sich um Text als Graphik handelt und SVG zudem explizit die Möglichkeit hat, die Darstellung zu skalieren und SVG auch eher suboptimal für langen Fließtext ist. Bei einer Gaphik

hat eher Priorität, dass der Autor einschätzen kann, wie groß der Text relativ zu den restlichen Inhalten wird. In Elementen wie ***text*** wird der Text auch nicht automatisch umgebrochen, also haben Autor und Betrachter kaum eine Chance, flexible Schriftgrößen sinnvoll zu nutzen. Bei Fließtext in (X)HTML kommt es eher auf optimale Lesbarkeit von längeren Textpassagen ohne Kunststücke seitens des Betrachters an.

So wird die Eigenschaft notiert:

```
<text x="300" y="150" font-size="12.3">Weitere Projekte</text>
```

Beispiel Schriftgröße[33]

Die Schriftgröße wird für jeden Buchstaben des Wortes font-size geändert. In der Mitte beim Bindestrich liegt der kleinste Wert.

Schriftstil, font-style

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

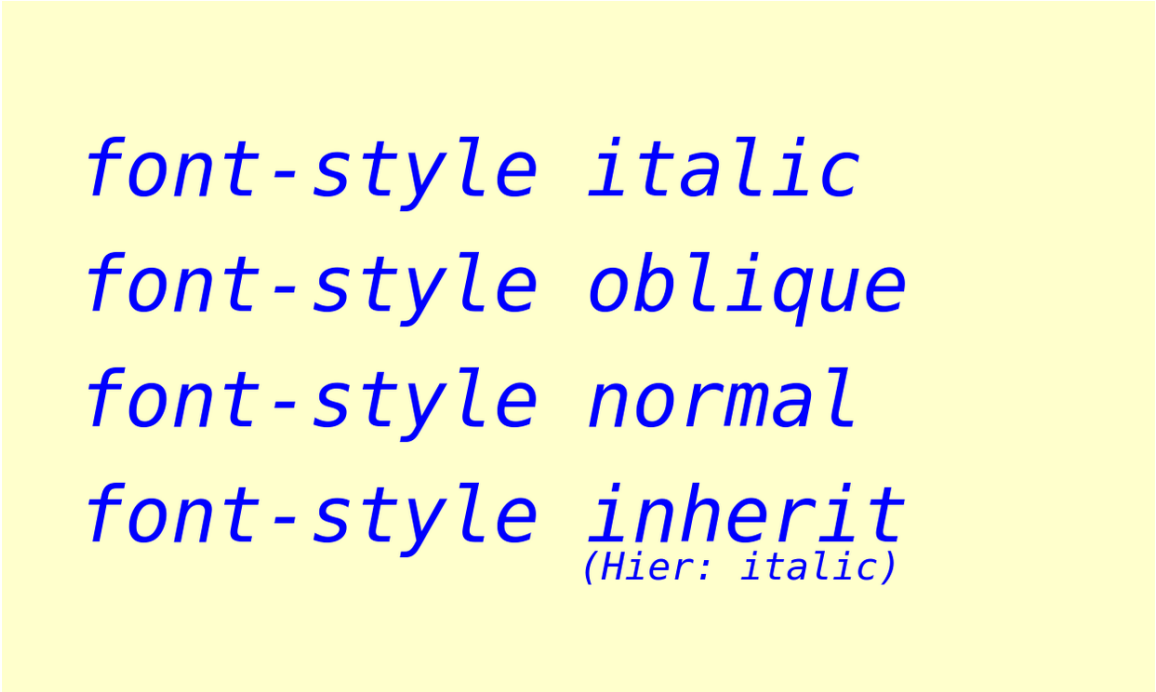


Abb. 130 Beispiel Schriftstil

33 http://de.wikibooks.org/wiki/media%3ASVGfont-size01.svg

Mittels der Eigenschaft *font-style* wird der Schriftstil ausgewählt, damit ist innerhalb einer Schriftfamilie auswählbar, ob die normale Variante verwendet wird, eine kursive oder eine geneigte.

Mögliche Werte sind:

inherit

geerbt

normal

'normal' im Gegensatz zu kursiv oder geneigt

italic

kursiv

oblique

geneigt

Jede Schriftfamilie sollte mindestens einen Eintrag für eine normale Variante und eine geneigte haben. Gibt es keine für kursiv, so kann auch die für oblique verwendet werden. Begriffe mit ähnlicher Bedeutung unter Experten für Schriftarten für geneigt oder oblique sind auch slanted oder incline (sofern jemand auf diese Begriffe in diesem Zusammenhang stoßen sollte).

Der Initialwert ist 'normal', die Eigenschaft wird vererbt und ist animierbar.

Beispiel Schriftstil[34]

Der Schrifstil wird für ein paar Zeilen Text wie angegeben notiert.

Schriftvariante, font-variant

Squiggle (Batik)	-
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

34 `http://de.wikibooks.org/wiki/media%3ASVGfont-style01.svg`

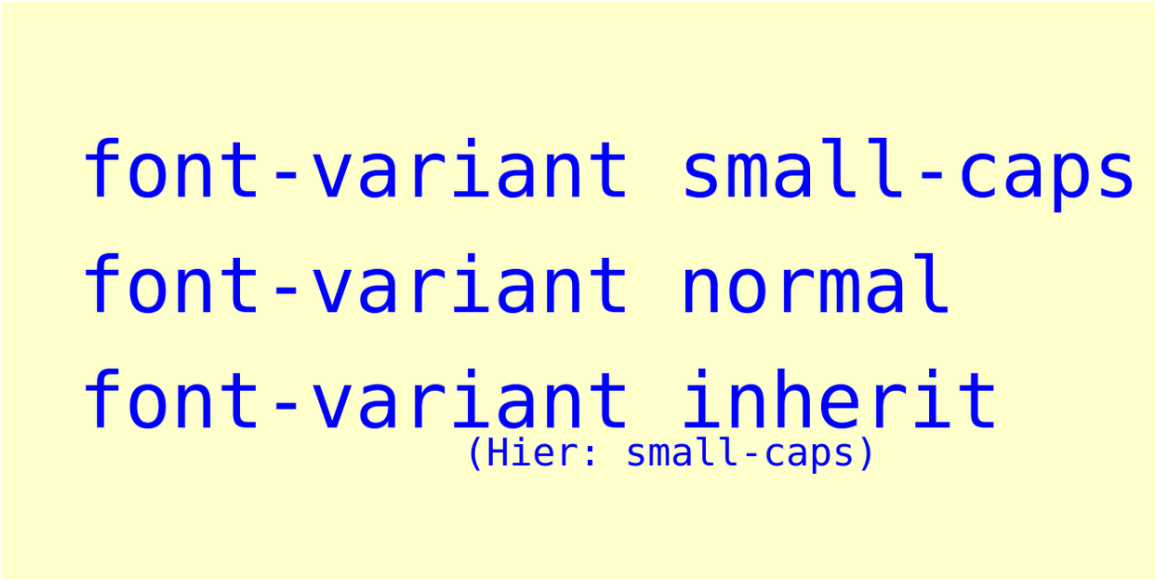


Abb. 131 Beispiel Schriftvariante

Mit der Eigenschaft *font-variant* kann angegeben werden, ob Kapitälchen verwendet werden sollen oder nicht. Glyphen in der Kaptälchenvariante sehen ähnlich wie Großbuchstaben aus, sind bei Kleinbuchstaben aber etwas kleiner und eventuell etwas anders proportioniert als Großbuchstaben.

Mögliche Werte sind:

inherit

geerbt

normal

keine Kapitälchen

small-caps

mit Kapitälchen

Der Initialwert ist 'normal', die Eigenschaft wird vererbt und ist animierbar.

Die Eigenschaft ist in SVG tiny 1.1 nicht verfügbar.

Beispiel Schriftvariante[35]

Die Schriftvariante wird für ein paar Zeilen Text wie angegeben notiert.

35 http://de.wikibooks.org/wiki/media%3ASVGfont-variant01.svg

Schriftgewichtung, font-weight

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

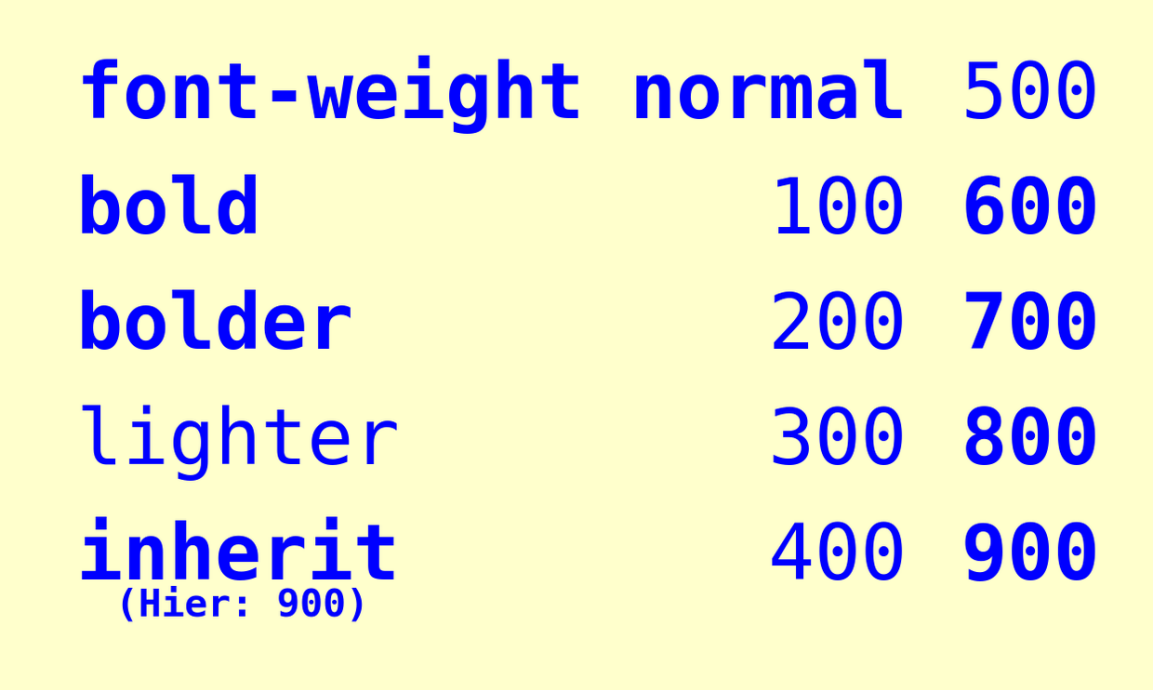


Abb. 132 Beispiel Schriftgewichtung

Die Gewichtung der Schrift kann mit der Eigenschaft *font-weight* angegeben werden. Im Großen und ganzen geht es darum, wie fett die Schrift dargstellt wird, das ist in SVG nicht zu verwechseln mit der Strichstärke, da im Allgemeinen die Glyphen vorrangig aus Füllungen bestehen.

Mögliche Werte sind:

inherit:geerbt

100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900

genau eine der Zahlen ist anzugeben, eine kleine Zahl bedeutet ein kleines Gewicht, also ein 'mageres' Aussehen der Glyphen, eine große Zahl entsprechend ein großes Gewicht oder fette Glyphen.

normal

wie 400

bold

wie 700

bolder

es wird eine Zahl größer angenommen als die geerbte, 900 bleibt unverändert

lighter

es wird eine Zahl kleiner angenommen als die geerbte, 100 bleibt unverändert

Der Initialwert ist 'normal', die Eigenschaft wird vererbt und ist animierbar.

Je nach Schriftfamilie müssen nicht alle Gewichte verfügbar sein, es wird dann das dem angegebenen Wert am nächsten liegende verfügbare Gewicht gewählt.

Beispiel Schriftgewichtung[36]

Die Schriftgewichtung wird für ein paar Zeilen Text wie angegeben notiert.

Schriftdehnung, font-stretch

Squiggle (Batik)	1.7
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

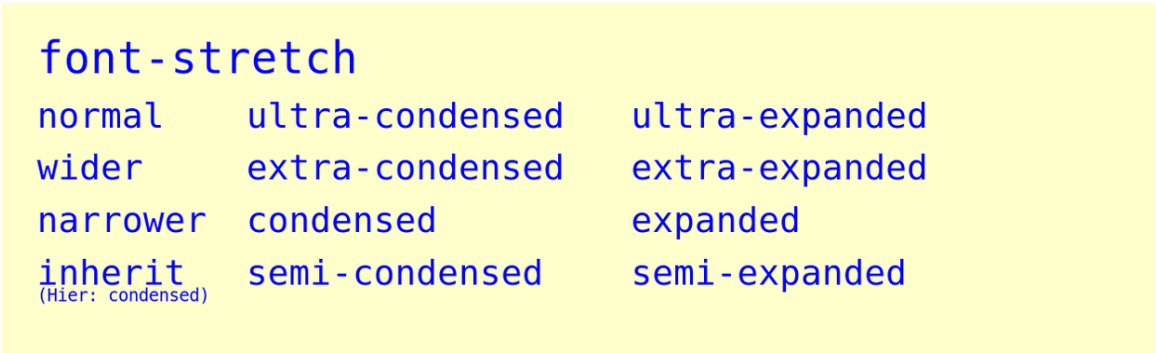


Abb. 133 Beispiel Schriftdehnung

Die Eigenschaft *font-stretch* entscheidet darüber, ob die Darstellung von Text normal, komprimiert oder gedehnt dargestellt wird.

Mögliche Werte sind:

normal

weder Dehnung noch Komprimierung

wider

36 http://de.wikibooks.org/wiki/media%3ASVGfont-weight01.svg

gedehnter als geerbt

narrower

komprimierter als geerbt

ultra-condensed

maximal komprimiert

extra-condensed

stark komprimiert

condensed

komprimiert

semi-condensed

schwach komprimiert

semi-expanded

schwach gedehnt

expanded

gedehnt

extra-expanded

stark gedehnt

ultra-expanded

maximal gedehnt

Der Initialwert ist 'normal', die Eigenschaft wird vererbt und ist animierbar.

Die Eigenschaft ist in SVG tiny 1.1 und tiny 1.2 nicht verfügbar.

In der Reihenfolge steht 'normal' zwischen 'semi-condensed' und 'semi-expanded'. Mit den relativen Angaben wird jeweils zum nächsten Wert relativ zu geerbten gesprungen, es sei denn, es gibt in der Reihe keinen nächsten Wert mehr, dann wird der geerbte verwendet.

Beispiel Schriftdehnung[37]

Die Schriftdehnung wird für ein paar Zeilen Text wie angegeben notiert.

37 `http://de.wikibooks.org/wiki/media%3ASVGfont-stretch01.svg`

Schriftanpassung, font-size-adjust

Squiggle (Batik)	-
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

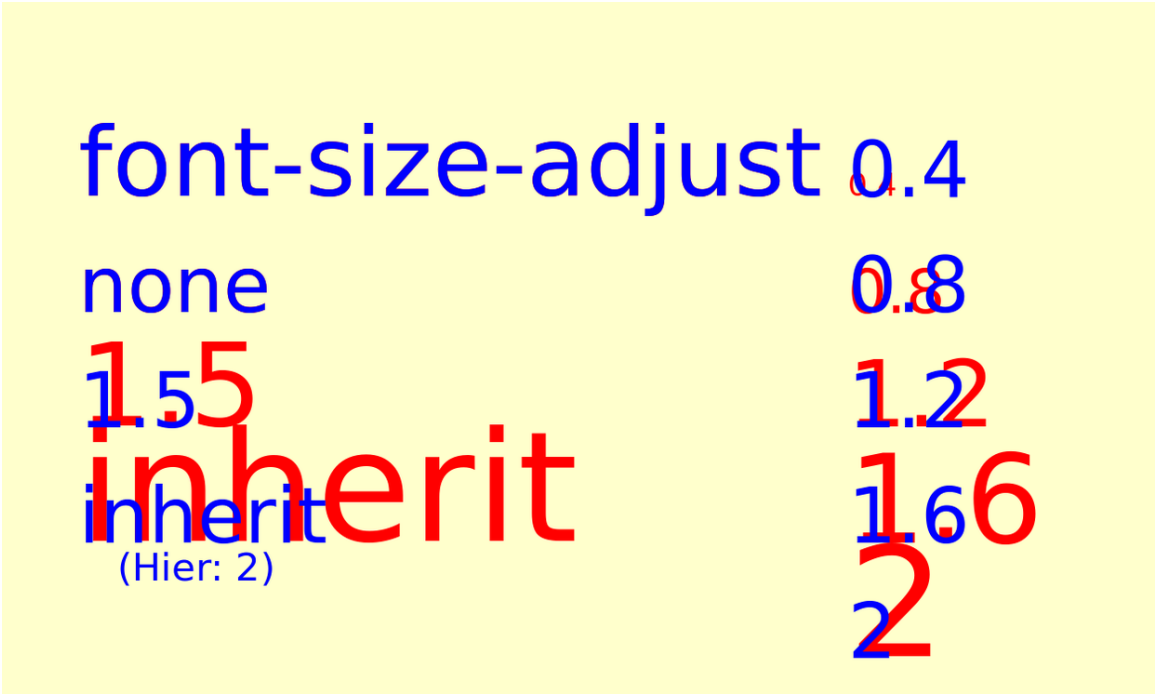


Abb. 134 Beispiel Schriftanpassung

Anpassung des Aspektwertes eine Schrift (zu einer anderen). Die Höhe oder Größe von Glyphen aus verschiedenen Schriftarten kann verschieden ausfallen, wird eine davon bei einzelnen Glyphen als Alternative zur ersten Wahl verwendet, weil es für die Glyphe kein Zeichen in der Schriftart gibt, so wird eine aus der Alternative genommen. Durch Angabe von *font-size-adjust* kann nun die Erscheinung der Alternative mit der ersten Wahl harmonisiert werden.

Mögliche Werte sind:

inherit

geerbt

none

beibehalten, nichts anpassen

Zahl

Aspektwert, sinnvollerweise größer als 0; 1 entspricht none, typische Werte liegen um 1 herum

Der Initialwert ist 'none', die Eigenschaft wird vererbbar und ist animierbar.

Die Eigenschaft ist in SVG tiny 1.1 und tiny 1.2 nicht verfügbar.

Der Aspektwert ist das Verhältnis der Höhen.

Sei c die anzuwendende Schriftgröße, a der Aspektwert der angegebenen Schrift (erste Wahl), a' der der Alternative, s die (für die erste Wahl) angegebene Schriftgröße, so ist

c = s (a/a')

Beispiel Schriftanpassung[38]

Der Schriftgrößenanpassung wird für ein paar Zeilen Text wie angegeben notiert.

Es wird angenommen, dass die primär angegebene Schriftart 'Kukkuluttumuffiei' nicht verfügbar ist. Als Ersatz ist 'serif' angegeben. Entsprechend font-size-adjust ist die Schriftgröße zu skalieren. Dies ist mit einem dahinterliegenden roten Vergleich angedeutet. Entpricht die Schriftgröße nicht dem roten Vergleich, ist dies ein Hinweis auf einen Implementierungsfehler.

Schriftart, font

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

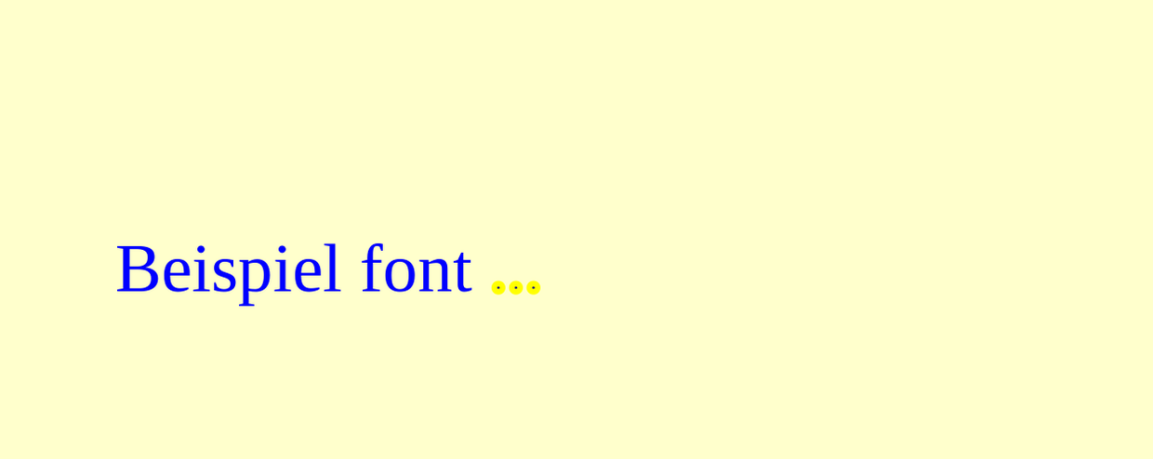


Abb. 135 Beispiel Schriftart

38 http://de.wikibooks.org/wiki/media%3ASVGfont-size-adjust01.svg

Die Eigenschaft *font* faßt alle Eigenschaften zur Auswahl der Schriftart zusammen, die zuvor beschrieben wurde. Diese Eigenschaft kann nicht als Präsentationsattribut auftreten, allenfalls als Eigenschaft in CSS-Notation oder in einer Animation mit *attributeType*="CSS".

Die genaue Syntax ist der Spezifikation für CSS2.0 (!) zu entnehmen.

In Kurzschreibweise sieht der mögliche Wert wie folgt aus:

```
[ [ 'font-style' || 'font-variant' || 'font-weight' ]?

'font-size' [ / 'line-height' ]? 'font-family' ] |

caption | icon | menu | message-box|

small-caption | status-bar | inherit
```

'|' steht jeweils zwischen Alternativen, von denen genau eine gewählt werden muss. '||' steht jeweils zwischen Angaben, von denen mindestens eine gewählt werden muss. Werden mehrere gewählt, ist die Reihenfolge egal. Optionale Angaben sind mit einem '?' dahinter gekennzeichnet. Die eckigen Klammern dienen nur der Gruppierung.

Die Eigenschaft wird vererbt und ist animierbar. Der Initialwert ergibt sich aus denen der einzelnen Eigenschaften. Die ebenfalls angebbare Eigenschaft *line-height* hat auf Text in SVG keinen Einfluss, ebenfalls angebbare Optionen für Systemschriftarten brauchem vom Darstellungsprogramme nicht interpretiert werden.

Beispiel Schriftart[39]

Die Eigenschaft font wird auf einen kurzen Beispieltext angewendet.

Der Text endet zudem mit drei Punkten, welche sich in einem tspan befinden, welches ein style-Attribut hat, mit welchem fill auf dunkelblau und stroke auf gelb festgelegt werden.

9.7.3 Layout-Eigenschaften

unicode-bidi

Squiggle (Batik)	1.7
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

39 http://de.wikibooks.org/wiki/media%3ASVGfont01.svg

Die Eigenschaft *unicode-bidi* ergibt die Möglichkeit, einen Bereich mit veränderter Schreibrichtung zu kennzeichnen. Generell wird die Schreibrichtung automatisch anhand der verwendeten Unicode-Zeichen bestimmt, diese Eigenschaft zusammen mit *direction* dienen dazu, gezielt davon abzuweichen, ist also eher für spezielle Problemfälle gedacht, nicht zur durchgehenden Anwendung.

Mögliche Werte:

inherit

geerbt

normal

keine Änderung oder besondere Aktion

embed

Öffnet einen neuen Bereich, in dem die Eigenschaft *direction* wirkt, entsprechend der Angabe zu *direction* wird zu Beginn und am Ende des Bereiches ein Richtungszeichen impliziert

bidi-override

Der automatische Algorithmus wird komplett ignoriert und die Reihenfolge erfolgt strikt nach der Angabe zu *direction*

Der Initialwert ist 'normal', die Eigenschaft wird nicht vererbt und ist nicht animierbar.

Die Eigenschaft ist in SVG tiny 1.1 nicht verfügbar.

direction

Squiggle (Batik)	1.7
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

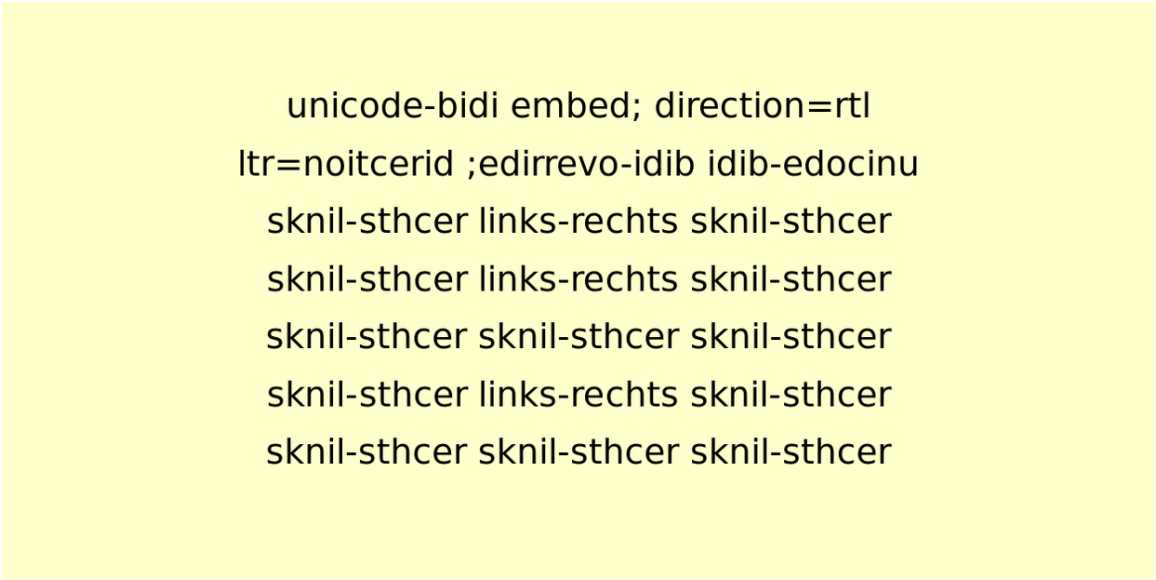


Abb. 136 Beispiel zu unicode-bidi und direction

direction wird immer zusammen mit *unicode-bidi* verwendet und bestimmt die Schreibrichtung, beziehungsweise überschreibt gegebenenfalls die automatisch bestimmte Schreibrichtung und ist daher eher für Problemfälle gedacht, nicht zur durchgehenden Anwendung.

Mögliche Werte:

inherit

geerbt

ltr

von links nach rechts

rtl

von rechts nach links

Der Initialwert ist 'ltr', die Eigenschaft ist vererbbar und ist nicht animierbar.

Die Eigenschaft ist in SVG tiny 1.1 nicht verfügbar.

Beispiel zu unicode-bidi und direction[40]

unicode-bidi und direction werden ausprobiert. Wo die Reihenfolge umgedreht wird, ist diese im Quelltext umgedreht, so dass der Text lesbar bleibt. Bei lateinischer Schrift ist bei 'embed' keine Umkehr zu erwarten, weil die Schreibrichtung wieder automatisch festgelegt wird. 'embed' sollte also nur bei Schrifen wirken, wo die Richtung nicht eindeutig festliegt.

40 `http://de.wikibooks.org/wiki/media%3ASVGbidirectionality01.svg`

writing-mode

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

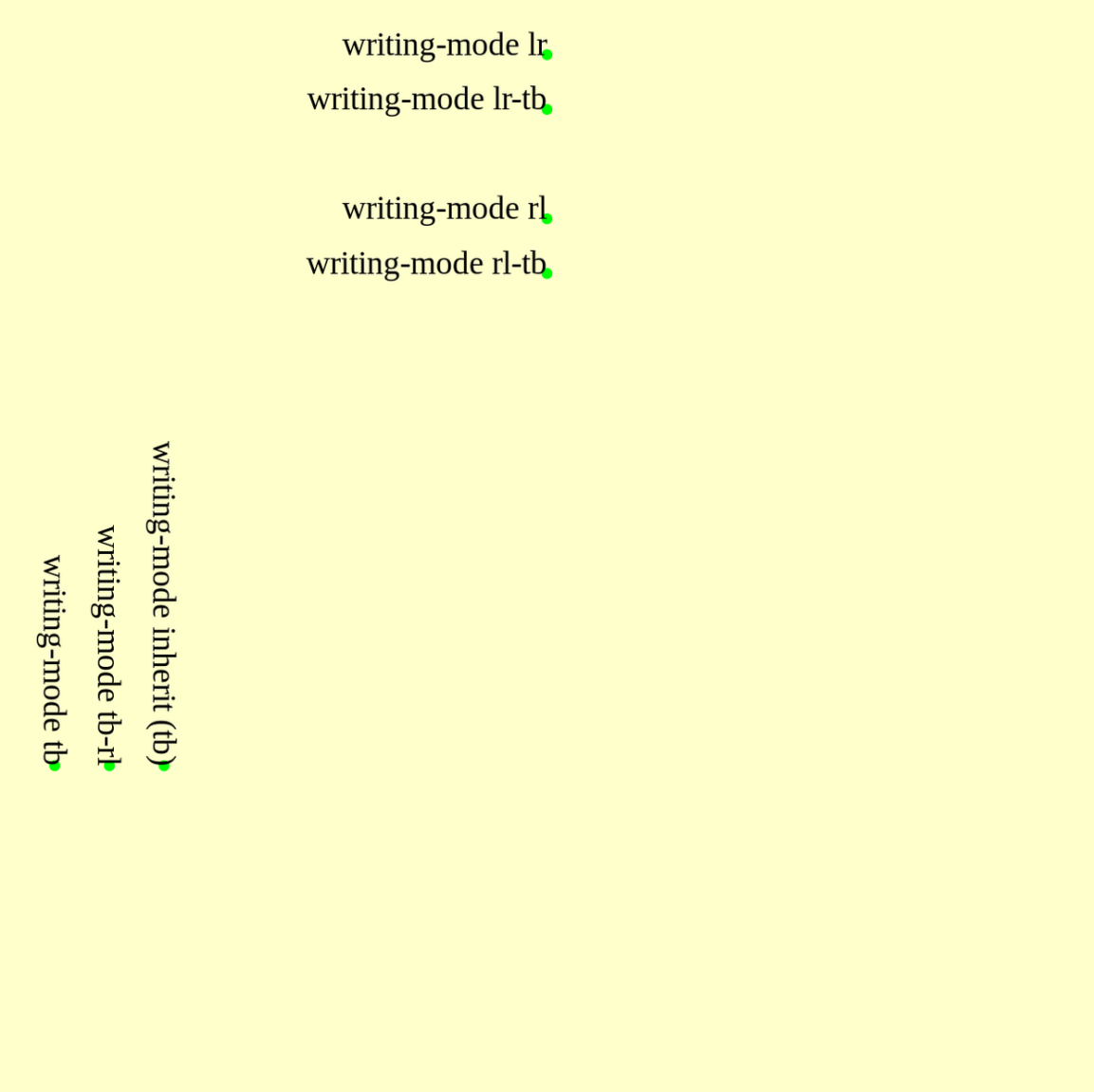


Abb. 137 Beispiel zu writing-mode

Die Eigenschaft *writing-mode* ist spezifisch für das Element ***text*** in der Vollversion von SVG 1.1 und bestimmt die Schreibrichtung. Innerhalb von anderen Textelementen wie ***tspan*** ist die Eigenschaft nicht anwendbar, dort sind dann gegebenenfalls *unicode-bidi* und *direction* zu verwenden, um besondere Probleme zu beheben.

Mögliche Werte:

inherit

geerbt

lr-tb

übliche Schreibrichtung für lateinische Schriften, zeilenweise von links nach rechts und die nächste Zeile darunter

lr

übliche Schreibrichtung für lateinische Schriften, von links nach rechts

rl-tb

übliche Schreibrichtung für arabische oder häbräische Schriften, zeilenweise von rechts nach links und die nächste Zeile darunter

rl

übliche Schreibrichtung für arabische oder häbräische Schriften, von rechts nach links

tb-rl

übliche Schreibrichtung für chinesische oder japanische Schriften, zeilenweise von oben nach unten, die nächste Zeile rechts daneben

tb

übliche Schreibrichtung für chinesische oder japanische Schriften von oben nach unten

Der Initialwert ist 'lr-tb', die Eigenschaft wird vererbt und ist nicht animierbar.

Da es keine dem W3C bekannten Kulturen gibt, die von unten nach oben schreiben oder im Kreis oder auf Spiralen etc, gibt es dafür auch keine weiteren Werte. Gegebenenfalls ist das dann mit Tricks indirekt zu realisieren, wenn es notwendig ist.

Da SVG 1.1 auch keinen automatischen Zeilenumbruch kennt, gibt es da hinsichtlich der Unterschiede für den Zeilenvorschub auch keinen Unterschied zwischen den Kurz- und Langversionen der Werte. Dies kann indes einen Effekt haben, wenn eine alternative Textversion des Dokumentes erstellt werden soll, entweder direkt vom Darstellungsprogramm oder auch durch eine Transformation in ein anderes Format. Das betreffende Programm könnte einen solchen Hinweis auswerten und den Text mit der gewünschten Zeilenfortschrittsrichtung darstellen.

Beispiel zu writing-mode[41]

Der writing-mode wird für ein paar Zeilen Text wie angegeben notiert.

Der Textanker ist auf end gesetzt und wird als grüner Punkt angedeutet. Weil sich bei lateinischen Glyphen bei horizontaler Textausrichtung die Richtung automatisch ergibt, ist dort die Textrichtung nicht umgedreht, aber die Wirkung von text-anchor end.

41 http://de.wikibooks.org/wiki/media%3ASVGwriting-mode01.svg

glyph-orientation-vertical

Squiggle (Batik)	1.7
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Die Glyphen eines Textes können in einer bestimmten Weise orientiert werden (nur in der Vollversion SVG 1.1). *glyph-orientation-vertical* bestimmt die Orientierung, wenn die Schreibrichtung vertikal ist.

Mögliche Werte sind:

inherit

geerbt

auto

automatisch bestimmen

Winkel

Winkel in Grad

Die Winkelangabe beschränkt sich auf die Werte 0, 90, 180, und 270 und ist in Grad gemeint. Andere Werte werden zum nächsten zulässigen Winkel gerundet. Angaben zur Rundung von 45, 135 und 225 liegen nicht vor, das scheint eine Lücke in der Spezifikation zu sein.

Die Winkelangabe versteht sich immer relativ zur Referenzrichtung, die ohne die Angabe des Attributes vorläge und ist in Uhrzeigerrichtung gemeint.

Der Initialwert ist 'auto', die Eigenschaft wird vererbt, ist aber nicht animierbar.

glyph-orientation-horizontal

Squiggle (Batik)	1.7
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

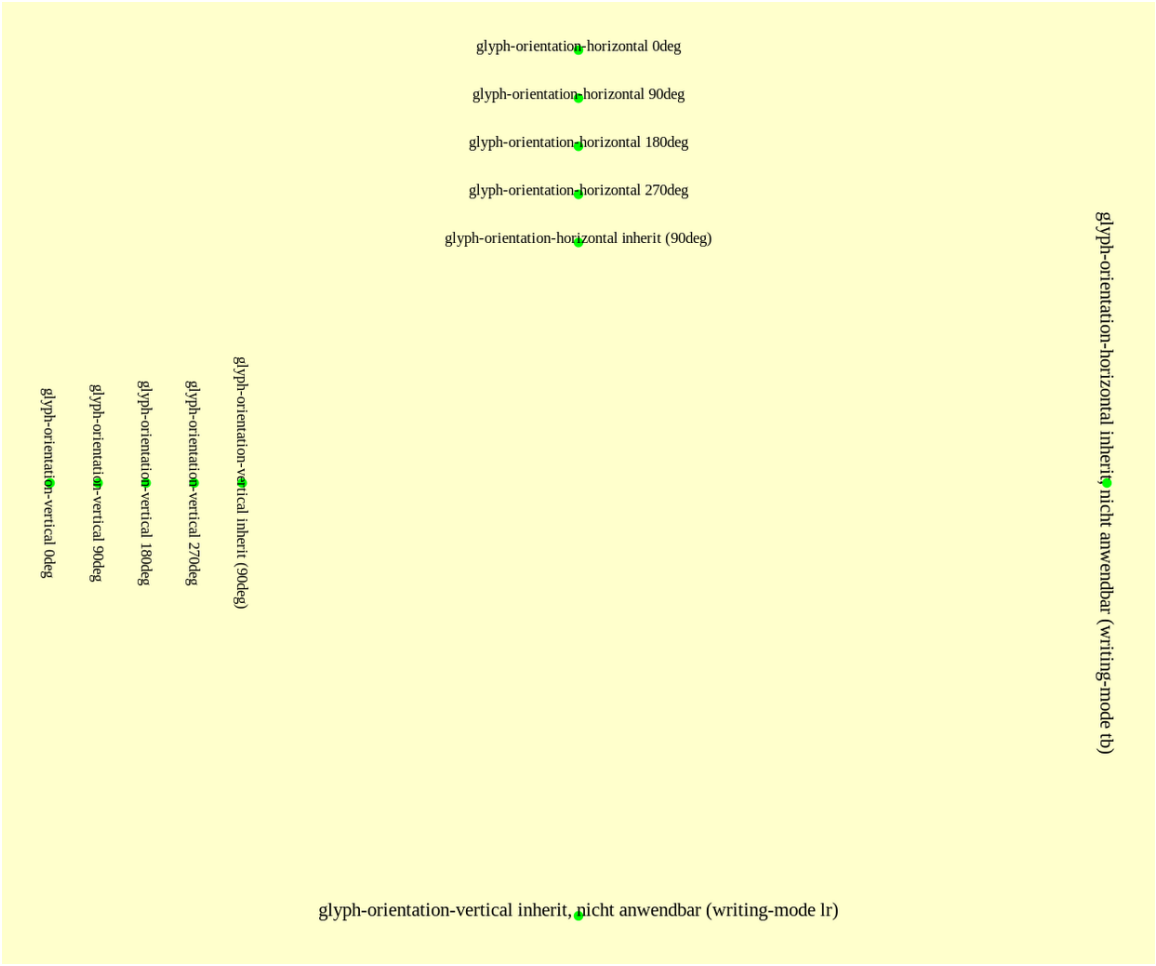


Abb. 138 Beispiel zu glyph-orientation-horizontal und glyph-orientation-vertical

Die Glyphen eines Textes können in einer bestimmten Weise orientiert werden (nur in der Vollversion SVG 1.1). *glyph-orientation-horizontal* bestimmt die Orientierung, wenn die Schreibrichtung horizontal ist.

Mögliche Werte sind:

inherit

geerbt

Winkel

Winkel in Grad

Die Winkelangabe beschränkt sich auf die Werte 0, 90, 180, und 270 und ist in Grad gemeint. Andere Werte werden zum nächsten zulässigen Winkel gerundet. Angaben zur Rundung von 45, 135 und 225 liegen nicht vor, das scheint eine Lücke in der Spezifikation zu sein.

Die Winkelangabe versteht sich immer relativ zur Referenzrichtung, die ohne die Angabe des Attributes vorläge und ist in Uhrzeigerrichtung gemeint.

Der Initialwert ist '0', die Eigenschaft wird vererbt, ist aber nicht animierbar.

Beispiel zu glyph-orientation-horizontal und glyph-orientation-vertical[42]

glyph-orientation-* wird für ein paar Zeilen Text wie angegeben notiert. Der Textanker ist auf middle gesetzt und wird als grüner Punkt angedeutet.

9.7.4 Eigenschaften zur Textanordnung

Textanker, text-anchor

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

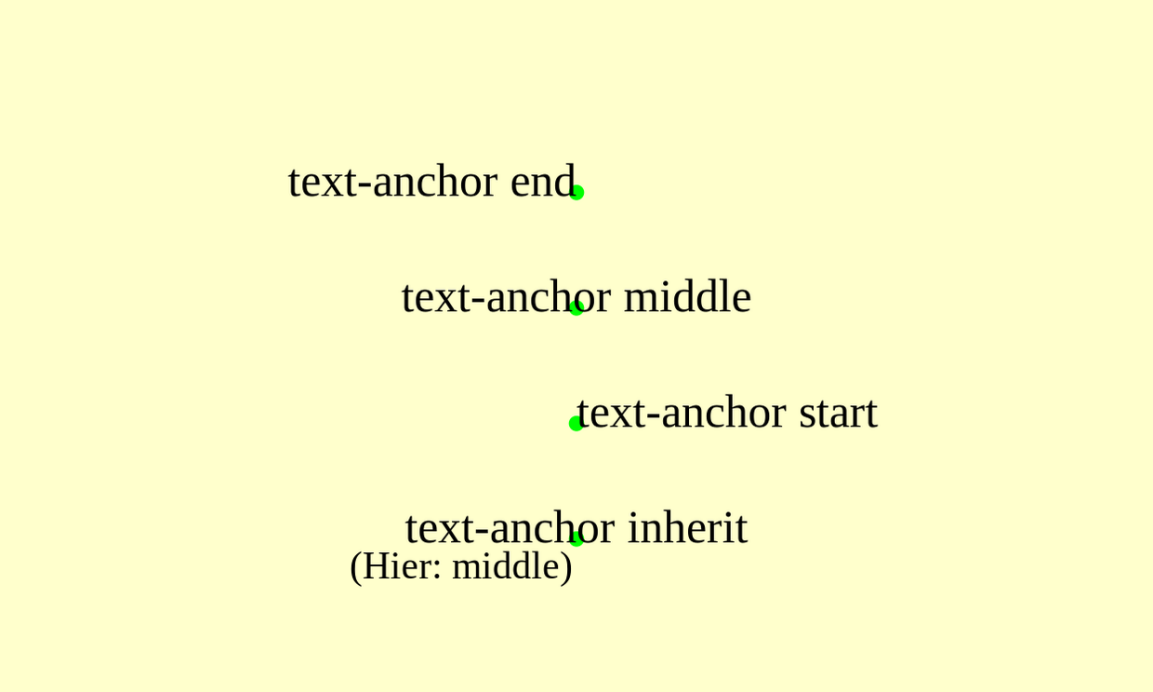


Abb. 139 Beispiel *text-anchor*

42 http://de.wikibooks.org/wiki/media%3ASVGglyph-orientation01.svg

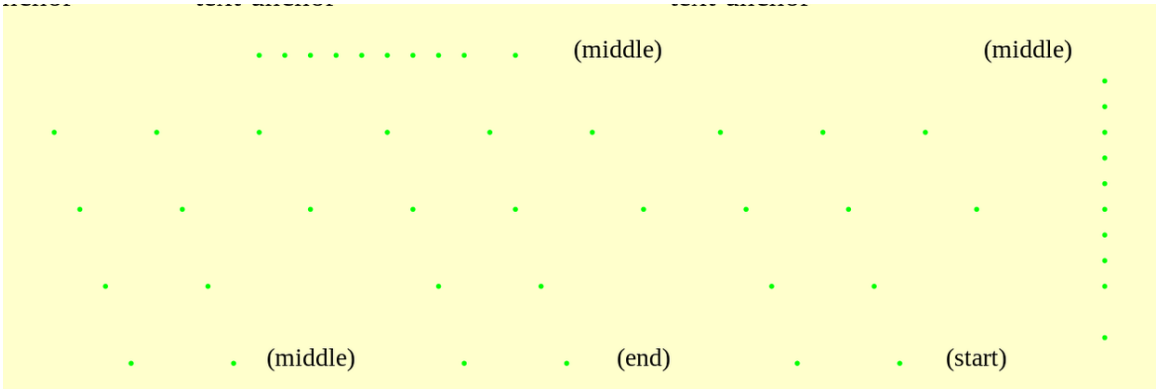


Abb. 140 Beispiel *text-anchor* (2)

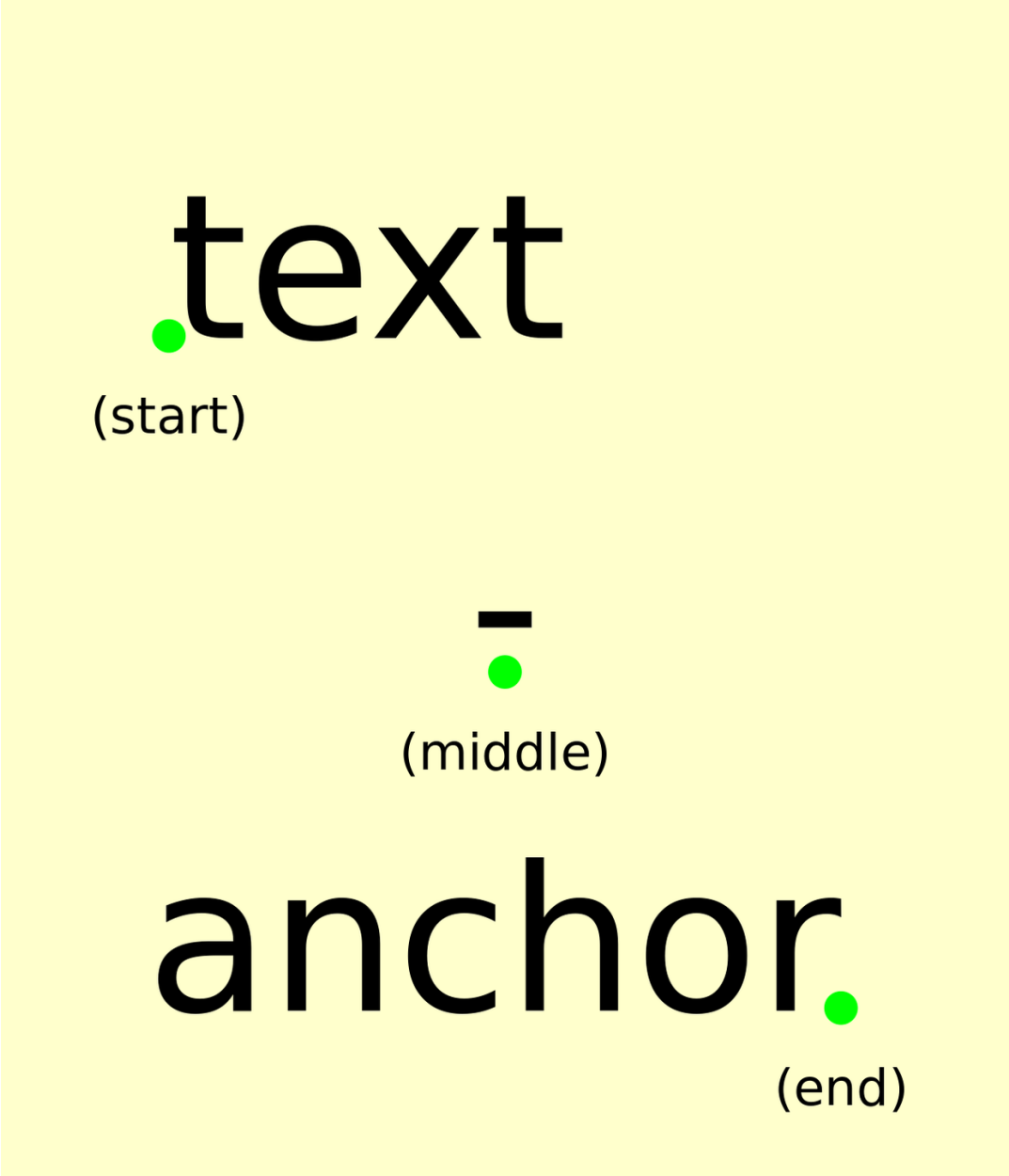


Abb. 141 Beispiel *text-anchor* und ***tspan***

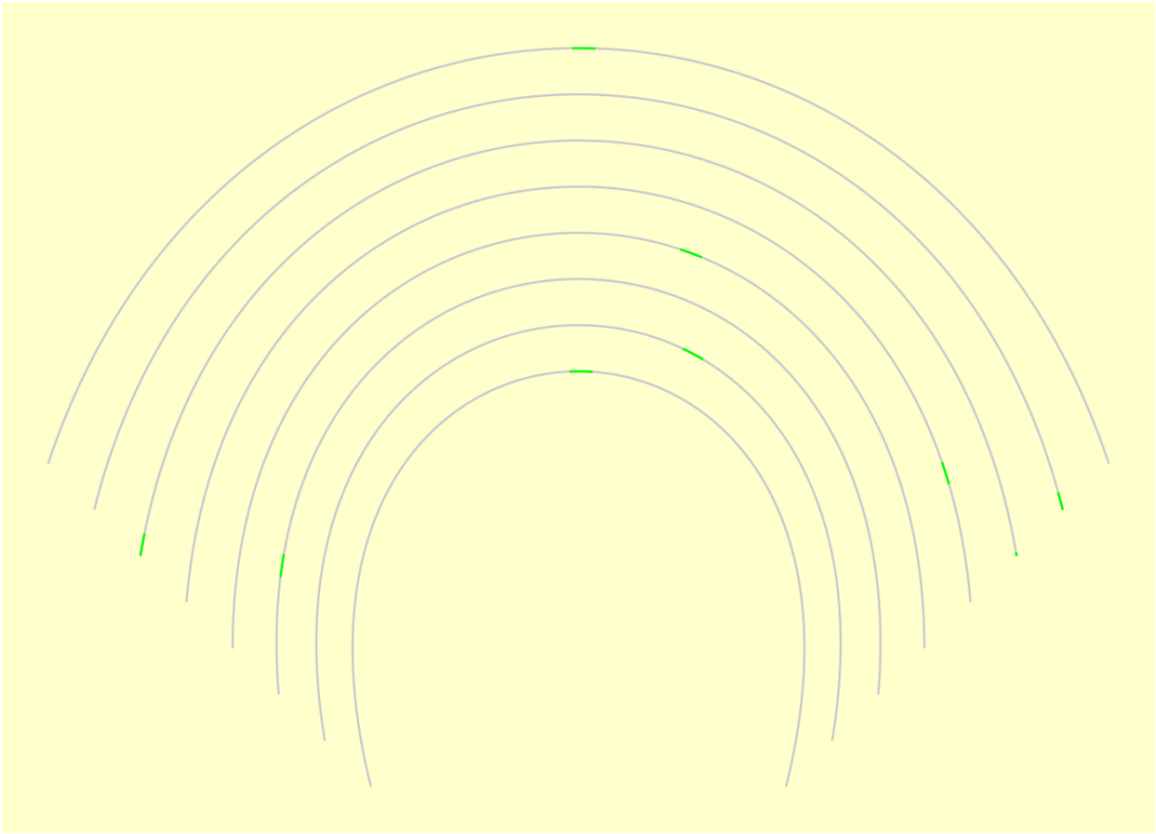


Abb. 142 Beispiel *text-anchor* und ***textPath***

Mit der Eigenschaft *text-anchor* kann angegeben werden, wie die Positionsangaben eines absolut positionierten Textsegmentes im Detail zu verstehen sind.

Mögliche Werte sind:

inherit

geerbt

start

Der Anfang des Textsegmentes ist bei der aktuellen Textposition.

end

Das Ende des Textsegmentes ist bei der aktuellen Textposition.

middle

Das Textsegment wird so angeordnet, dass die Mitte an der aktuellen Textposition erscheint.

Der Initialwert ist 'start', die Eigenschaft wird vererbt und ist animierbar.

Die Eigenschaft erleicht die Anordnung von Text relativ zu anderen Objekten, ohne dass es erforderlich wäre, die genaue Länge des Textes zu kennen. Mit dieser Eigenschaft kann einfach links oder rechts von einem Objekt Text angeordnet werden, wobei die aktuelle Textposition sich weiter vom Objekt entfernt, also es nicht überschreibt. Mit 'middle' kann der Text auch einfach mittig zu einem

Objekt (darunter, darüber, darin) angeordnet werden, ohne die genaue Länge des Textsegmentes kennen zu müssen.

Beispiel *text-anchor*[43]

Der Textanker wird für ein paar Zeilen Text wie angegeben notiert. Der Textanker selbst ist als grüner Punkt angedeutet.

Beispiel *text-anchor* (2)[44]

Der Textanker wird für ein paar teils einzeln angeordnete Glyphen wie angegeben notiert. Der Textanker selbst ist als grüner Punkt angedeutet.

Beispiel *text-anchor* und ***tspan***[45]

Der Textanker wird für Teile des Begriffes 'text-anchor' in mehreren Elementen ***tspan*** wie angegeben notiert. Der Textanker selbst ist als grüner Punkt angedeutet.

Beispiel *text-anchor* und ***textPath***[46]

Der Textanker wird für etwas Text wie angegeben notiert. Der Text wird mit ***textPath*** angeordnet. Die Kombination mit dem Wert von *startOffset* entscheidet über die Anordnung des Textes entlang des jeweiligen Pfades. Der Textanker selbst ist als grüner Teil des Pfades angedeutet.

Für 'start' beginnt der Text um den Wert von *startOffset* im Pfad verschoben.

Für 'middle' beginnt der Pfad verschoben um *startOffset* minus der halben Textlänge. Für ein *startOffset* von 50% ergibt sich also eine mittige Anordnung des Textes relativ zum Pfad.

Für 'end' beginnt der Pfad verschoben um *startOffset* minus der ganzen Textlänge. Für ein *startOffset* von 100% ergibt sich also eine Anordnung des Textes am Ende des Pfades, so dass also das Ende des Textes mit dem des Pfades zusammenfällt.

Grundlinie, dominant-baseline

Squiggle (Batik)	-
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

43 http://de.wikibooks.org/wiki/media%3ASVGtext-anchor01.svg
44 http://de.wikibooks.org/wiki/media%3ASVGtext-anchor02.svg
45 http://de.wikibooks.org/wiki/media%3ASVGtext-anchor03.svg
46 http://de.wikibooks.org/wiki/media%3ASVGtext-anchor04.svg

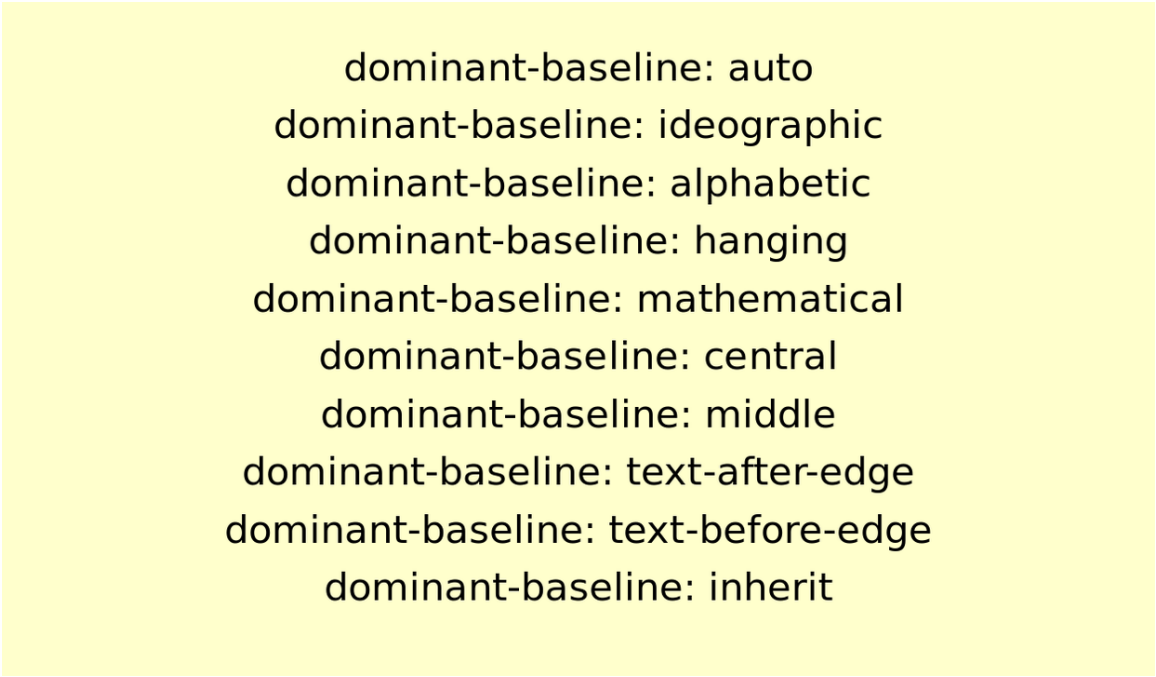


Abb. 143 Beispiel *dominant-baseline*

In verschiedenen Kulturen oder auch Anwendungsgebieten für Schrift gibt es verschiedene Ideen, wo die Grundlinie relativ zu den Glyphen verlaufen sollte. Schriftfamilien haben oft Tabellen, in denen für die jeweilige Schrift angegeben ist, wo für welche Art von Anwendung die Grundlinie anzunehmen ist.

Mit der Eigenschaft *dominant-baseline* kann die Wahl der Grundlinie beeinflusst werden.

Für Details über die Tabellen und die verschiedenen Modelle für Schriften in verschiedenen Kulturen sei auf die entsprechende Fachliteratur verwiesen.

Mögliche Werte sind:

inherit

geerbt

auto

automatisch bestimmen

use-script

Ein Skript oder *writing-mode* wird verwendet, um die Grundlinie zu bestimmen

no-change

Verwendung der Angaben des Text-Elternelementes

reset-size

Skalierung der Grundlinientabelle relativ zur *font-size*

ideographic

Verwendung des ideographischen Modells, typisch für chinesich, japanisch, koreanisch oder vietnamesisch

alphabetic

Verwendung des alphabetische Modells, typisch für alphabetische Schriften

hanging

Verwendung des hängenden Modells, typisch für einige indische Schriften wie bengalisch, Gurmukhi, Devanagari

mathematical

Für mathematische Symbole

central

Mittig zur em-Zelle

middle

Um eine halbe x-Höhe relativ zur alphabetischen Grundlinie verschoben

text-after-edge

Endecke der em-Zelle

text-before-edge

Anfangsecke der em-Zelle

Der Initialwert ist 'auto', die Eigenschaft wird nicht vererbt und ist animierbar. Die Eigenschaft ist verfügbar in der Vollversion von SVG 1.1.

Beispiel *dominant-baseline*[47]

dominant-baseline wird für ein paar Zeilen Text wie angegeben notiert.

Relative Anordnung der Grundlinie, alignment-baseline

Squiggle (Batik)	-
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

47 `http://de.wikibooks.org/wiki/media%3ASVGdominant-baseline01.svg`

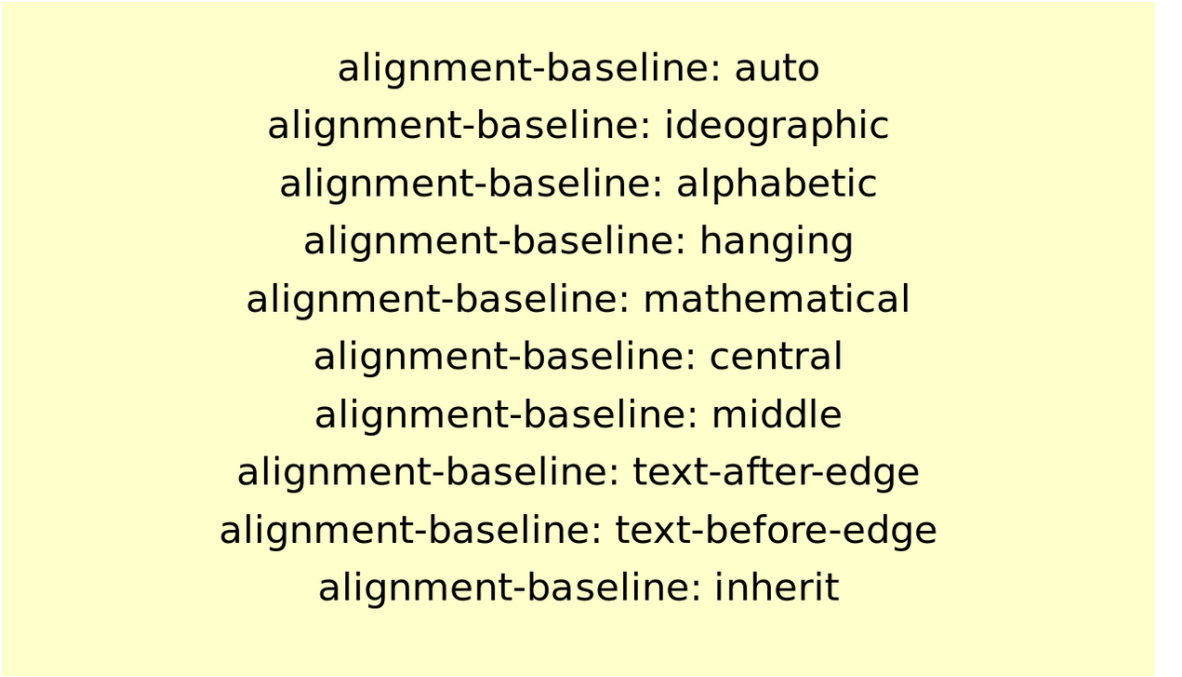


Abb. 144 Beispiel *alignment-baseline*

Die Eigenschaft gibt die Grundlinienposition relativ zum Elternelement an und ist anwendbar auf die Elemente ***tspan***, ***tref***, ***altGlyph***, ***textPath***.

Mögliche Werte sind:

inherit

geerbt

auto

automatisch bestimmen

baseline

wie beim Elternelement

before-edge

Anfangsecke der em-Zelle

text-before-edge

Anfangsecke der em-Zelle

after-edge

Endecke der em-Zelle

text-after-edge

Endecke der em-Zelle

ideographic

Verwendung des ideographischen Modells, typisch für chinesich, japanisch, koreanisch oder vietnamesisch

alphabetic

Verwendung des alphabetische Modells, typisch für alphabetische Schriften

hanging

Verwendung des hängenden Modells, typisch für einige indische Schriften wie bengalisch, Gurmukhi, Devanagari

mathematical

Für mathematische Symbole

central

Mittig zur em-Zelle

middle

Um eine halbe x-Höhe relativ zur alphabetischen Grundlinie verschoben

Der Initialwert ist 'auto', die Eigenschaft wird nicht vererbt und ist animierbar. Die Eigenschaft ist verfügbar in der Vollversion von SVG 1.1.

Beispiel *alignment-baseline*[48]

alignment-baseline wird für ein paar Zeilen Text wie angegeben notiert.

Verschiebung der Grundlinie, baseline-shift

Squiggle (Batik)	1.7
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	3.2
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

$$\text{Weg: } c(u) = (c_x, c_y)$$

$$\text{Weglänge}(f) = {}_0\!\int_f du\ ((dc_x/du)_2 + (dc_y/du)_2)_{1/2}$$

Abb. 145 Beispiel *baseline-shift*

baseline-shift dient dazu, die Grundlinie für Text zu verschieben, zum Beispiel, um einen Index oder einen Exponenten zu schreiben. Die Grundlinie bezieht sich auf das umgebende Elternelement, wo

48 http://de.wikibooks.org/wiki/media%3ASVGalignment-baseline01.svg

diese auch bereits mit der Eigenschaft verschoben sein darf. Die Eigenschaft ist anwendbar für die Elemente ***tspan***, ***tref***, ***altGlyph***, ***textPath***

Die Eigenschaft ist nicht in SVG tiny 1.1 oder tiny 1.2 verfügbar.

Mögliche Werte sind:

inherit

geerbt

baseline

Grundlinie, unverschoben

sub

Index, tiefergestellt, also nach unten verschoben

super

Exponent, hochgestellt, also nach oben verschoben

Prozentangabe

bezieht sich auf die Schrifthöhe, positiv bei lateinischer Schrift nach oben, 0% ist die Grundlinie

Längenangabe

Absolute Angabe der Verschiebung, positiv bei lateinischer Schrift nach oben, 0 ist die Grundlinie

Der Initialwert ist baseline. Die Eigenschaft wird nicht vererbt und ist animierbar.

Beispiel *baseline-shift*[49]

Anhand des Integrals zur Bestimmung der Weglänge eines Pfades wird die Verwendung der Eigenschaft baseline-shift gezeigt.

9.7.5 Eigenschaften zum Glyphenabstand

Unterschneidung, kerning

Squiggle (Batik)	1.7
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

49 `http://de.wikibooks.org/wiki/media%3ASVGbaseline-shift01.svg`

AYAv Taffljipj Kv; kerning -10
AYAv Taffljipj Kv; kerning -5
AYAv Taffljipj Kv; kerning 0
AYAv Taffljipj Kv; kerning 5
AYAv Taffljipj Kv; kerning 10
AYAv Taffljipj Kv; kerning 20
AYAv Taffljipj Kv; kerning inherit (5)
AYAv Taffljipj Kv; kerning auto

Abb. 146 Beispiel *kerning*

Mit der Eigenschaft *kerning* wird ein Bereich der Typographie behandelt, bei dem es um den Abstand zwischen Glyphen (eines Wortes oder einer Abkürzung) geht, wobei dieser Abstand davon abhängig gemacht wird, welche Glyphen benachbart sind.

Ohne Unterschneidung hat jede Glyphe eine eigene (rechteckige) Zelle. Wird aus Glyphen ein Wort gebildet, werden diese Zellen aneinandergereiht.

Je nachdem, welche Buchstaben nebeneinanderstehen, ergibt das manchmal recht große weiße Flächen zwischen den Glyphen, die sich störend auf den Lesefluß auswirken. Zum Beispiel bei den Kombinationen WA, VA, TJ oder LT zu erkennen. Bei einer anderen Kombination wie WV, UH, XZ oder HL treten hingegen keine großen weißen Flächen auf. Je nach Schriftart kann es also Tabellen geben, die angeben, bei welcher Kombination von zwei Buchstaben jeweils wie viel freier Raum zwischen den beiden eingespart werden kann.

Mit der Eigenschaft *kerning* kann nun dies Verhalten an- oder abgeschaltet werden.

Folgende Wert sind möglich:

inherit

geerbt

auto

Sofern vorhanden, wird die zur Schriftart gehörige Tabelle verwendet

Längenangabe

Die zur Schriftart gehörige Tabelle wird nicht verwendet, die angegebene Länge verlängert (positiver Wert) oder verkürzt (negativer Wert) die Zelle der Glyphe.

Der Initialwert ist 'auto', die Eigenschaft wird vererbt und ist animierbar.

Die Eigenschaft ist nicht in SVG tiny 1.1 oder tiny 1.2 verfügbar.

Beispiel *kerning*[50]

kerning wird für eine Phantasiezeichenfolge 'AYAv Taffljipj Kv' wie angegeben notiert.

Glyphenabstand, letter-spacing

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

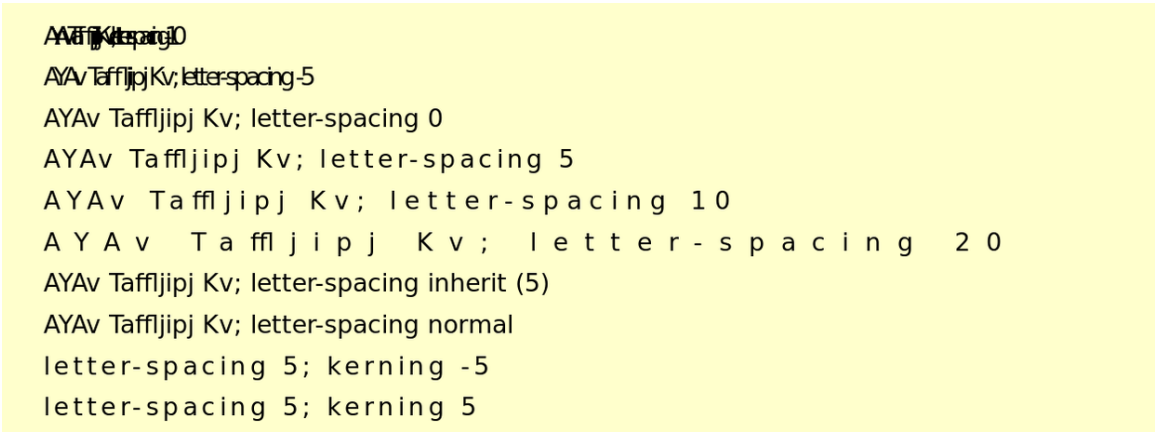


Abb. 147 Beispiel *letter-spacing*

Zusätzlich zur Unterschneidung kann mittels *letter-spacing* eine Angabe zum Abstand zwischen Glyphen notiert werden.

Mögliche Werte sind:

inherit

geerbt

normal

Der Abstand ergibt sich aus den Angaben der Schriftart

Längenangabe

Die angegebene Länge wird zur Unterschneidung addiert. Auch negative Werte sind zulässig.

Der Initialwert ist 'normal', die Eigenschaft wird vererbt und ist animierbar.

Die Eigenschaft ist nicht in SVG tiny 1.1 oder tiny 1.2 verfügbar.

50 http://de.wikibooks.org/wiki/media%3ASVGkerning01.svg

Beispiel *letter-spacing*[51]

letter-spacing wird für eine Phantasiezeichenfolge 'AYAv Taffljipj Kv' wie angegeben notiert. Zwei weitere Beispiele sind zur Kombination mit kerning ergänzt.

Wortabstand, word-spacing

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

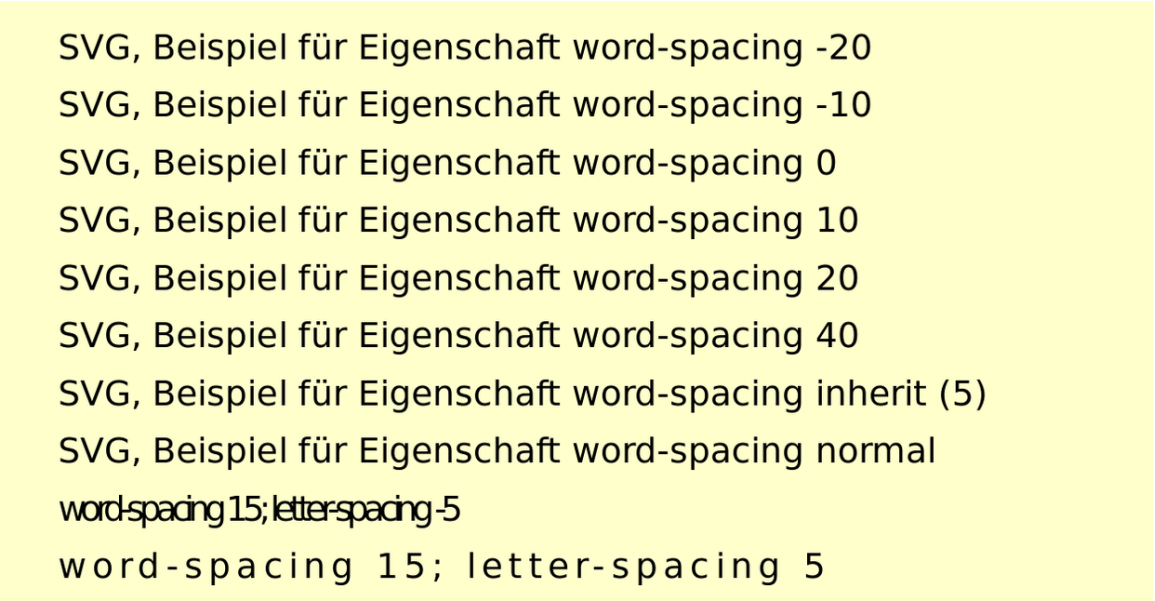


Abb. 148 Beispiel *word-spacing*

Zur besseren Lesbarkeit wird zwischen verschiedenen Wörtern ein Leerzeichen eingefügt. Mit der Eigenschaft *word-spacing* kann festgelegt werden, wie groß solch ein Abstand zwischen zwei Wörtern dargestellt werden soll.

Mögliche Werte sind:

inherit

geerbt

normal

Der Abstand ergibt sich aus den Angaben der Schriftart

51 `http://de.wikibooks.org/wiki/media%3ASVGletter-spacing01.svg`

Längenangabe

Die angegebene Länge wird zur Unterschneidung addiert. Auch negative Werte sind zulässig.

Der Initialwert ist 'normal', die Eigenschaft wird vererbt und ist animierbar.

Die Eigenschaft ist nicht in SVG tiny 1.1 oder tiny 1.2 verfügbar.

Beispiel *word-spacing*[52]

word-spacing wird für ein paar Wörter wie angegeben notiert. Zwei weitere Beispiele sind zur Kombination mit letter-spacing ergänzt.

9.7.6 Eigenschaft zur Textdekoration

text-decoration

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

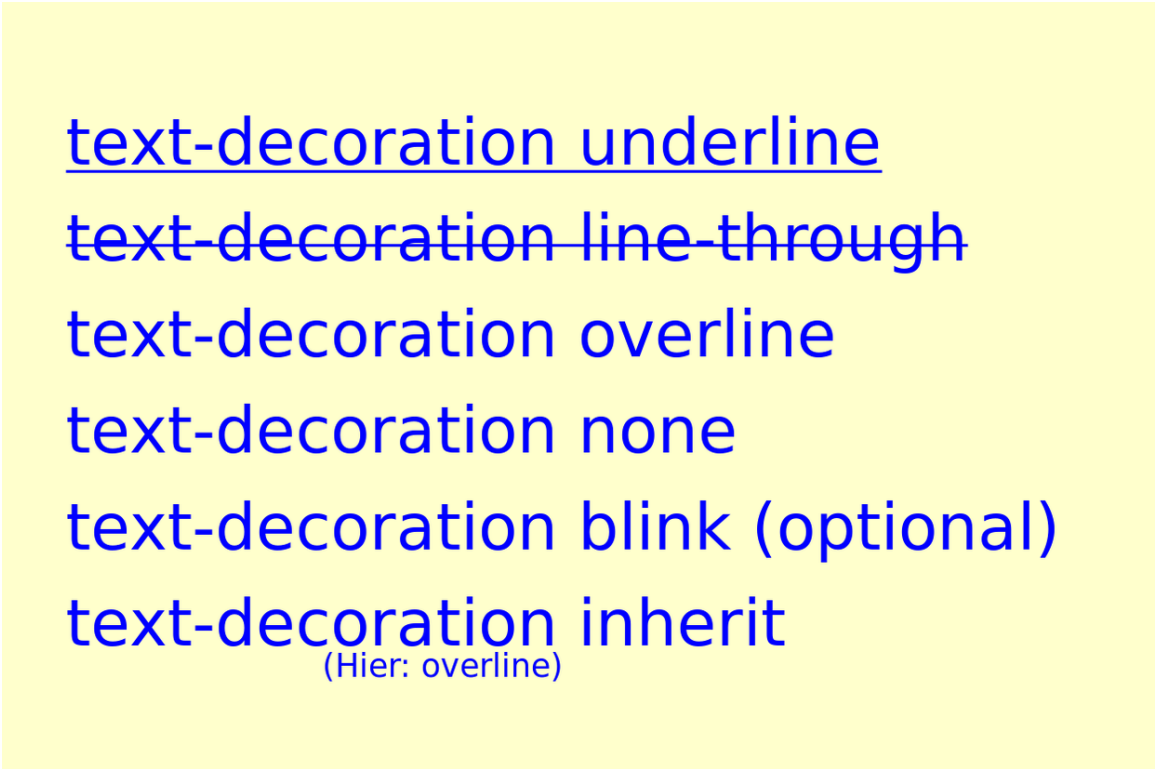


Abb. 149 Beispiel *text-decoration*

52 http://de.wikibooks.org/wiki/media%3ASVGword-spacing01.svg

Die Eigenschaft *text-decoration* dient dazu, Text zu dekorieren, mit Strichen unter oder über dem Text oder durchgestrichenem Text oder einem (optionalen) Blinken.

Mögliche Werte sind:

inherit

geerbt

none

keine Dekoration

underline

unterstrichen

overline

'überstrichen', also ein Strich über dem Text

line-through

durchgestrichen (mittig)

blink

blinkend; dies ist allerdings optional, es ist nicht notwendig, dass ein Darstellungsprogramm dies interpretiert, wenn nicht, ist der Effekt der gleiche wie für 'none'

Der Initialwert ist 'none', die Eigenschaft wird nicht vererbt und ist animierbar. Hinsichtlich der Vererbung gilt allerdings, dass falls bei einem Element ***text*** notiert, die Dekoration auf den gesamten Text angewendet wird, also zum Beispiel auf alle Elemente ***tspan***, ***tref***, ***textPath*** darin, sofern bei diesen nicht anders angegeben ist. Entsprechendes gilt, wenn die Eigenschaft für ***tspan*** oder ***textPath*** gesetzt ist und darin kommen weitere Elemente wie ***tspan***.

Ist die Eigenschaft hingegen bei einem umgebenden Element ***g*** gesetzt, so vererbt sich die Eigenschaft nur, wenn *text-decoration* explizit auf 'inherit' gesetzt ist.

Füll- und Stricheigenschaften, die für den Text gelten, gelten auch für die graphischen Repräsentationen, welche durch *text-decoration* hinzukommen.

Die Eigenschaft ist in SVG tiny 1.1 und tiny 1.2 nicht verfügbar.

Beispiel *text-decoration*[53]

Die Textdekoration wird für ein paar Zeilen Text wie angegeben notiert. Derzeit (Ende 2009) haben offenbar einige Darstellungsprogramme mit der Vererbung dieser Eigenschaft. Sofern keine Implementierungsfehler aufgedeckt werden sollen, kann es also vorteilhaft sein, diese Eigenschaft nur direkt im betroffenen Element zu setzen.

53 `http://de.wikibooks.org/wiki/media%3ASVGtext-decoration01.svg`

9.8 Wiedergabeeigenschaften

9.8.1 Farbinterpolation, color-interpolation

Mit der Eigenschaft *color-interpolation* kann angegeben werden, wie bei Animationen, Teildurchsichtigkeiten oder Farbverläufen zwischen den angegebenen Werten interpoliert werden soll.

Verfügbar sind Interpolationen im Standard-Farbraum (sRGB) und in einem linearisierten Farbraum (linearRGB). Die Umrechnung erfolgt gemäß der folgenden Regeln für die drei Farbkanäle R, G, B:

```
R[sRGB] = R[sRGB-8bit] / 255

G[sRGB] = G[sRGB-8bit] / 255

B[sRGB] = B[sRGB-8bit] / 255

Falls R[sRGB], G[sRGB] oder B[sRGB] kleiner oder gleich 0.04045
ist, gilt:

R[linearRGB] = R[sRGB] / 12.92

G[linearRGB] = G[sRGB] / 12.92

B[linearRGB] = B[sRGB] / 12.92

sonst falls R[sRGB], G[sRGB] oder B[sRGB] größer als 0.04045 ist,
gilt:

R[linearRGB] = ((R[sRGB] + 0.055) / 1.055) ^ 2.4

G[linearRGB] = ((G[sRGB] + 0.055) / 1.055) ^ 2.4

B[linearRGB] = ((B[sRGB] + 0.055) / 1.055) ^ 2.4

R[linearRGB-8bit] = R[linearRGB] * 255

G[linearRGB-8bit] = G[linearRGB] * 255

B[linearRGB-8bit] = B[linearRGB] * 255
```

Beachtenswert ist jedenfalls, dass der Standardfarbraum und auch sonstige übliche Farbräume nicht den gesamten sichtbaren Bereich des Spektrums abdecken, sondern nur einen mehr oder weniger

großen Teil. Ohne Kalibrierung kann die Darstellung von Farben auf verschiedenen Monitoren oder mit verschiedenen Druckern auch deutlich variieren. Die Vollversion von SVG 1.1 bietet durchaus auch die Möglichkeit, ein anderes Farbprofil anzugeben, nicht jedoch diese Eigenschaft zur Interpolation.

Mögliche Werte für *color-interpolation* sind:

inherit

geerbt

auto

Das Darstellungsprogramm kann selbst entscheiden, dem Autor ist es egal

sRGB

Interpolation erfolgt im Standard-Farbraum

linearRGB

Interpolation erfolgt im linearisierten Farbraum

Der Initialwert ist 'sRGB', die Eigenschaft wird vererbt und ist animierbar. Die Eigenschaft ist in SVG tiny 1.1 und tiny 1.2 nicht verfügbar.

9.8.2 Farbinterpolation für Filter, color-interpolation-filters

Speziell für die SVG-Filter gibt es eine eigene Eigenschaft zur Farbinterpolation *color-interpolation-filters*. Diese hat die gleichen möglichen Werte wie *color-interpolation*.

Der Initialwert ist allerdings 'linearRGB', die Eigenschaft wird ebenfalls vererbt und ist animierbar.

Die Eigenschaft ist in SVG tiny 1.1 und tiny 1.2 nicht verfügbar.

9.8.3 Farbwiedergabe, color-rendering

Je nachdem, ob Qualität oder Geschwindigkeit wichtig ist, kann ein Autor mittels der Eigenschaft *color-rendering* einen entsprechenden Hinweis geben.

Mögliche Werte sind:

inherit

geerbt

auto

Mit leicht größerer Gewichtung der Qualität soll das Darstellungsprogramm selbst ein Optimum zwischen Qualität und Geschwindigkeit finden.

optimizeSpeed

Geschwindigkeit ist wichtiger als Qualität. Interpolation kann im eigenen Farbraum stattfinden.

optimizeQuality

Qualität ist wichtiger als Geschwindigkeit.

Der Initialwert ist 'auto', die Eigenschaft wird vererbt und ist animierbar.

9.8.4 Formwiedergabe, shape-rendering

Einmal abgesehen von den generellen Mindestanforderungen an die Genauigkeit der Darstellung kann ein Darstellungsprogramm auch noch höhere Genauigkeit bieten. Mittels *shape-rendering* kann ein Autor einen entsprechenden Hinweis geben, ob Geschwindigkeit oder Qualität wichtig ist.

Mögliche Werte sind:

inherit

geerbt

auto

Mit leicht größerer Gewichtung der geometrischen Präzision soll das Darstellungsprogramm selbst ein Optimum zwischen Qualität und Geschwindigkeit finden.

optimizeSpeed

Geschwindigkeit ist wichtiger als Qualität, also zum Beispiel eher keine Kantenglättung.

crispEdges

Ein hoher Kontrast soll erreicht werden, gegebenenfalls auch indem nahezu horizontale und vertikale Linien entsprechend der Auflösung des Anzeigegerätes gerade ausgerichtet werden, Kantenglättung kann abgeschaltet werden.

geometricPrecision

Geometrische Präzision ist wichtiger als hoher Kontrast oder hohe Geschwindigkeit.

Der Initialwert ist 'auto', die Eigenschaft wird vererbt und ist animierbar.

Die Eigenschaft ist in SVG tiny 1.1 nicht verfügbar.

9.8.5 Textwiedergabe, text-rendering

Ein Hinweis zur Qualität von als Graphik wiedergegebenem Text kann mit der Eigenschaft *text-rendering* erfolgen.

Mögliche Werte sind:

inherit

geerbt

auto

Mit leicht größerer Gewichtung der geometrischen Präzision soll das Darstellungsprogramm selbst ein Optimum zwischen Qualität und Geschwindigkeit finden.

optimizeSpeed

Geschwindigkeit ist wichtiger als Qualität, also zum Beispiel eher keine Kantenglättung.

optimizeLegibility

Gute Lesbarkeit soll erreicht werden Kantenglättung kann abgeschaltet werden und bei mehreren verfügbaren Schriftarten die am besten lesbare gewählt werden.

geometricPrecision

Geometrische Präzision ist wichtiger als gute Lesbarkeit oder hohe Geschwindigkeit.

Der Initialwert ist 'auto', die Eigenschaft wird vererbt und ist animierbar.

Die Eigenschaft ist in SVG tiny 1.1 nicht verfügbar.

9.8.6 Bildwiedergabe, image-rendering

Oftmals wird in SVG auch Pixelgraphik zusammen mit dem anderen Inhalt skaliert oder sonstwie weiterverarbeitet. Mit der Eigenschaft *image-rendering* kann der Autor einen Hinweis geben, was dabei wichtig ist.

Mögliche Werte sind:

inherit

geerbt

auto

Mit leicht größerer Gewichtung der Qualität soll das Darstellungsprogramm selbst ein Optimum zwischen Qualität und Geschwindigkeit finden.

optimizeSpeed

Geschwindigkeit ist wichtiger als Qualität. Interpolation kann im eigenen Farbraum stattfinden.

optimizeQuality

Qualität ist wichtiger als Geschwindigkeit.

Der Initialwert ist 'auto', die Eigenschaft wird vererbt und ist animierbar.

Die Eigenschaft ist in SVG tiny 1.1 nicht verfügbar.

9.8.7 Zwischenspeicher und Wiedergabe, buffered-rendering

Für länger unverändert bleibende Bereiche eines Dokumentes kann es sich für das Darstellungsprogramm lohnen, diese zwischenzuspeichern. Der Autor kann mit der Eigenschaft *buffered-rendering* einen Hinweis geben, ob sich dies lohnt oder nicht.

Mögliche Werte sind:

inherit

geerbt

auto

Das Darstellungsprogramm sollte selbst ein Optimum zwischen Geschwindigkeit und optimaler Speicherverwaltung finden.

dynamic

Der betroffene Bereich wird häufig verändert.

static

Der betroffene Bereich wird nicht häufig verändert, eine Zwischenspeicherung kann sich lohnen.

Der Initialwert ist 'auto' die Eigenschaft wird nicht vererbt und ist animierbar.

Die Eigenschaft ist in SVG 1.1 nicht verfügbar.

9.9 Eigenschaften zur Interaktivität

9.9.1 Zeigerereignisse, pointer-events

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

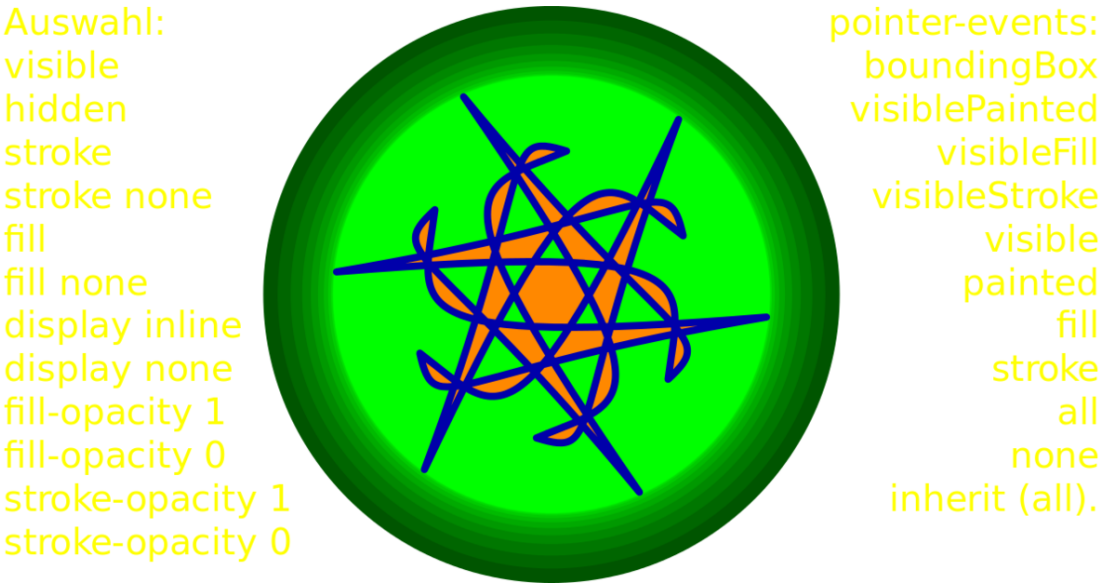


Abb. 150 Beispiel zu *pointer-events* (SVG tiny 1.2)

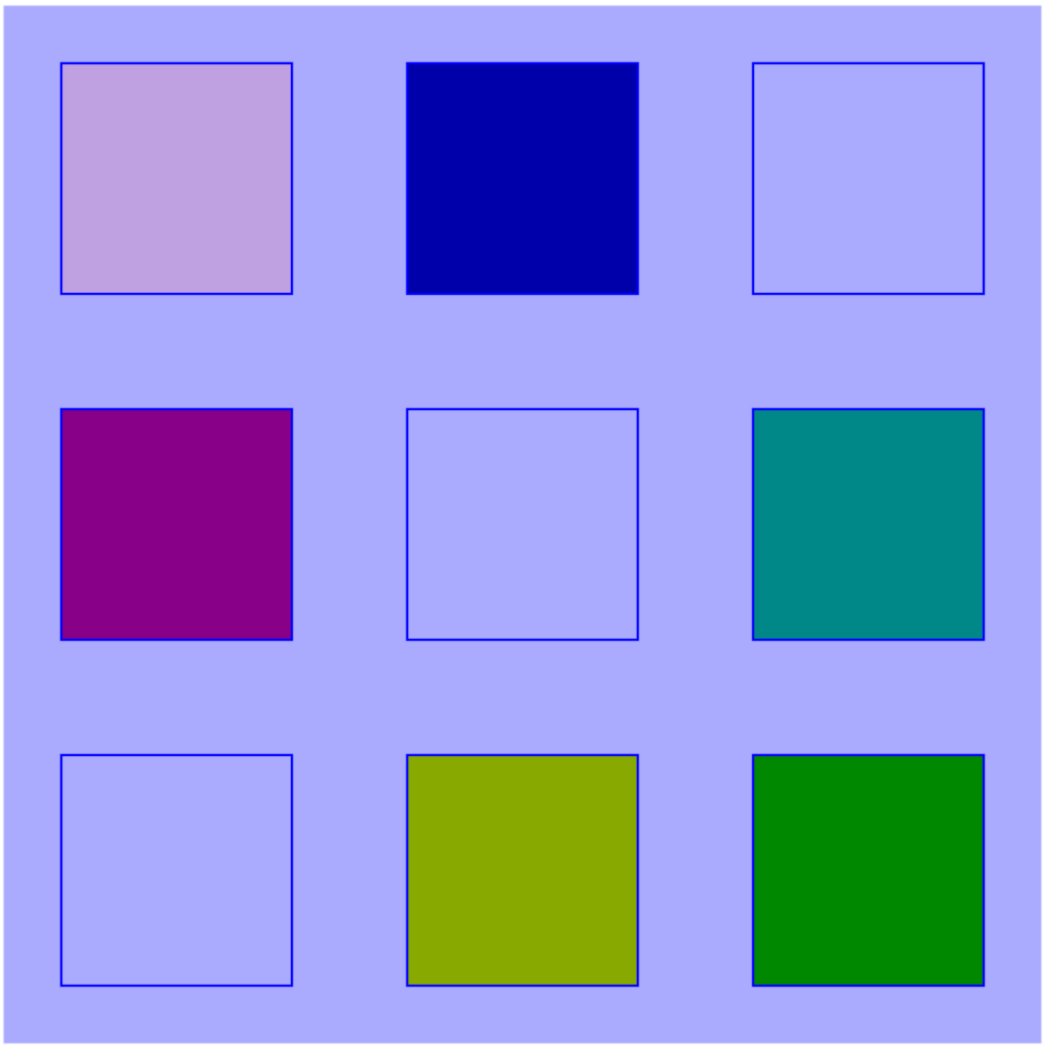


Abb. 151 Test zu *pointer-events*

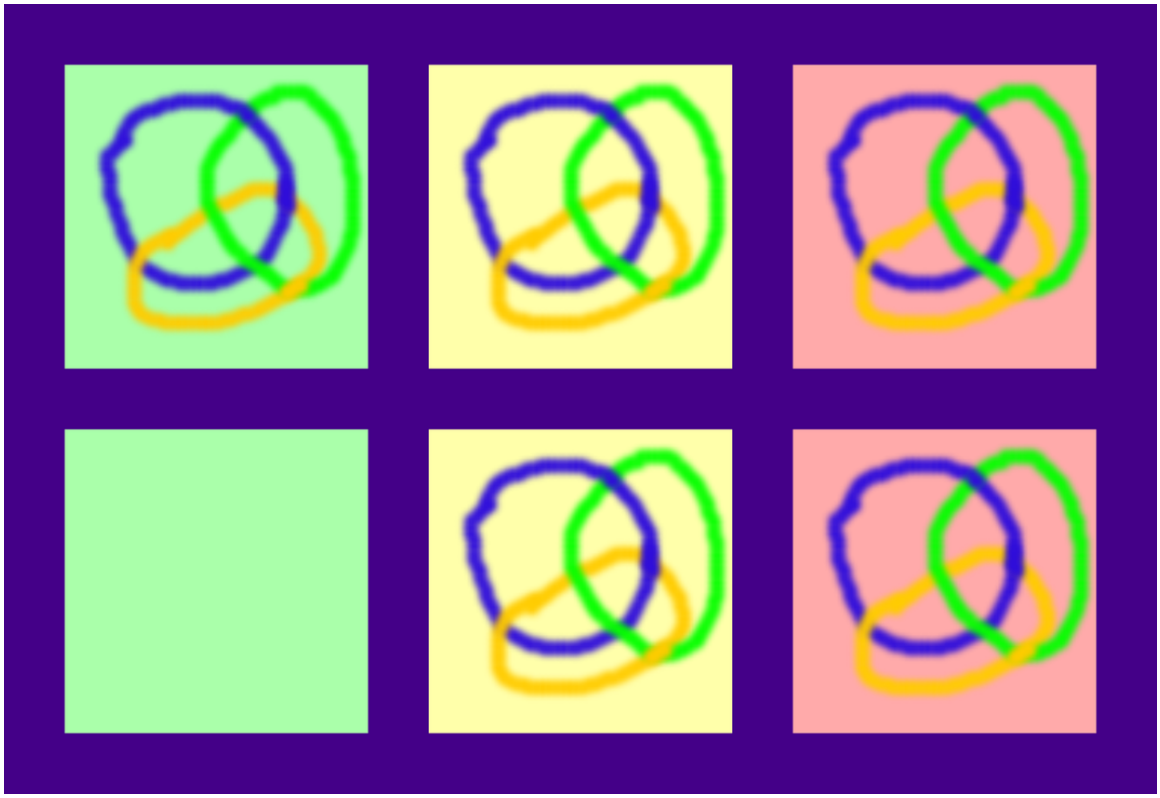


Abb. 152 *pointer-events* und Bilder

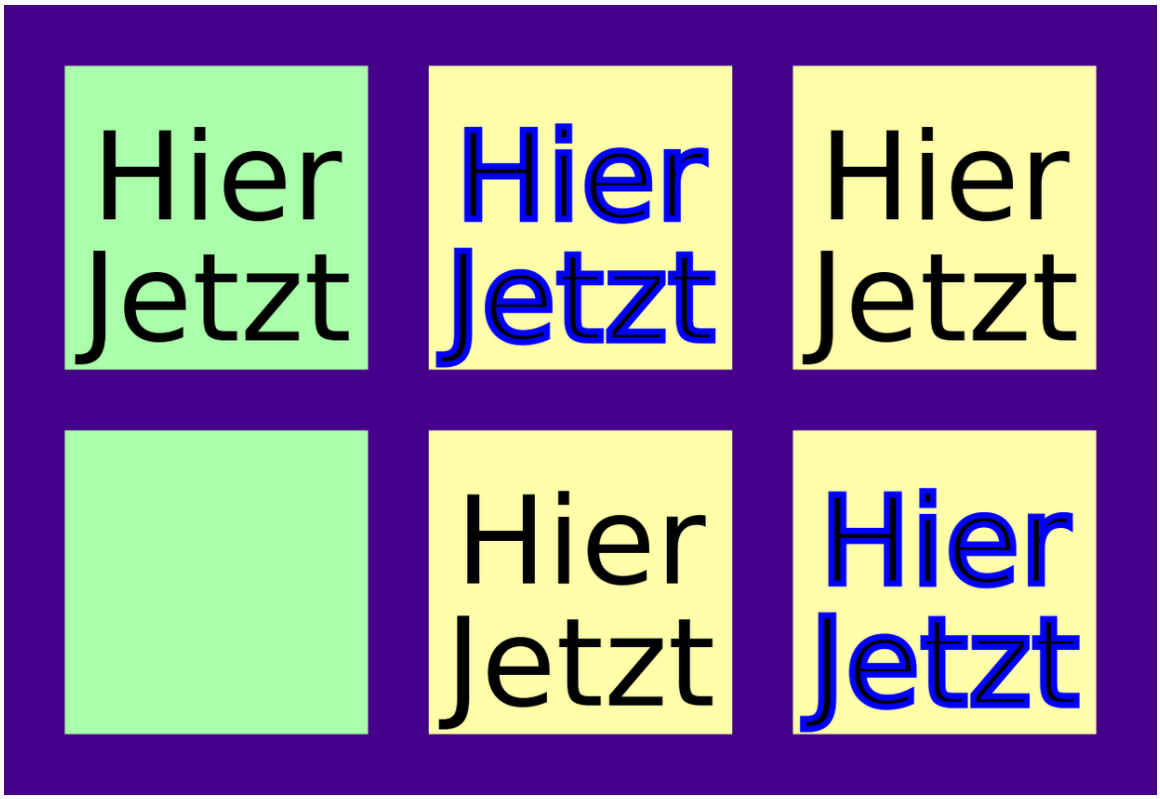


Abb. 153 *pointer-events* und Text

Je nach Anwendung kann der Autor entscheiden, welche Elemente Ziel von Ereignissen sein können. Dies kann mit der Eigenschaft *pointer-events* angegeben werden.

Mögliche Werte sind:

inherit

geerbt

none

as Element kann nicht Ziel eines Ereignisses sein.

visiblePainted

Sichtbar gemalt, also *fill* beziehungsweise *stroke* nicht auf 'none' und *visibility* auf 'visible', was in dem Sinne sichtbar ist, kann Ziel eines Ereignisses sein.

visibleFill

Füllung sichtbar, also *visibility* auf 'visible', dann kann die Füllung (egal mit welchem Wert) Ziel eines Ereignisses sein.

visibleStroke

Strich sichtbar, also *visibility* auf 'visible', dann kann der Strich (egal mit welchem Wert) Ziel eines Ereignisses sein.

visible

sichtbar, also *visibility* auf 'visible', dann können Füllung oder Strich (egal mit welchem Wert) Ziel eines Ereignisses sein.

painted

gemalt, also *fill* beziehungsweise *stroke* nicht auf 'none', was in dem Sinne sichtbar ist, kann Ziel eines Ereignisses sein. Der Wert von *visibility* ist egal.

fill

Die Füllung kann Ziel eines Ereignisses sein. Die Werte von *fill* und *visibility* sind egal.

stroke

Der Strich kann Ziel eines Ereignisses sein. Die Werte von *stroke* und *visibility* sind egal.

all

Das Element kann Ziel eines Ereignisses sein. Die Werte von *fill*, *stroke* und *visibility* sind egal.

boundingBox

Das Element kann Ziel eines Ereignisses sein, wenn der Zeiger über der umgrenzenden Zelle des Elementes ist. Dies ist im lokalen Koordinatensystem das kleinste entlang der lokalen x- und y-Achsen ausgerichtete Rechteck, welches das Element umschließt (ist die Höhe oder Breite des Elementes 0, gibt es solch eine Zelle nicht, sonst schon). Dieser Wert ist in SVG tiny 1.2 verfügbar, nicht in SVG 1.1.

Der Initialwert ist 'visiblePainted', die Eigenschaft wird vererbt und ist animierbar.

Die Eigenschaft ist nicht verfügbar in SVG tiny 1.1.

Zu beachten ist dabei, dass Angaben zur Durchsichtigkeit keinen Einfluss auf die Auswertung von Zeigerereignissen haben. Auch komplett durchsichtige Elemente können also Ziel von Ereignissen sein.

Da ferner im Malermodell obenliegende Elemente die Ereignisse abfangen, erreichen diese darunterliegende Elemente nicht. Ein durchsichtiges, obenliegendes Element kann also verhindern, dass darunterliegende Elemente Ziel von Ereignissen werden, obgleich sie sichtbar sein mögen.

Für Pixelbilder sind die Werte 'visibleFill', 'visibleStroke' und 'visible' als gleich anzusehen.

Ebenso sind für Pixelbilder 'fill', 'stroke' and 'all' als gleich anzusehen.

Hinsichtlich der Durchsichtigkeit gelten für Pixelbilder allerdings etwas andere Regeln als für sonstige Elemente. Voll durchsichtige Pixel nehmen eine Sonderrolle ein. Solche Pixel gelten in diesem Sinne als nicht gemalt und nicht sichtbar.

Beispiel zu *pointer-events*[54] (SVG tiny 1.2)

Ein Pfad mit sechzähliger diskreter Drehsymmetrie und mit Überschneidungen dient als Testelement zum Ausprobieren des Verhaltens von *pointer-events*.

In der Auswahl kann durch Klicken oder Aktivieren bestimmt werden, ob und wie der Pfad dargestellt wird. Auf der anderen Seite gibt es eine weitere Auswahl für pointer-events.

Ein erfolgreiches Ereignis ('mouseover' und 'mouseout' des Pfades selbst) ergibt eine einfache Farbanimation des dahinterliegenden Kreises.

Der anfängliche und geerbte Wert für *pointer-events* ist hier 'all'.

Test zu *pointer-events*[55]

Die kleinen Quadrate sind alle so aufgebaut, dass sie aus verschiedenen Gründen selbst nicht Ziel von Ereignissen sein können (einmal abgesehen von einigen blauen Rändern). Klicken oder aktivieren hat also immer eine Aktivierung einer Farbanimation des großen blauen Quadrates zur Folge.

pointer-events und Bilder[56]

Ein referenziertes, teilweise transparentes PNG-Bild wird verwendet, um pointer-events bei einem Bild zu testen.

54 `http://de.wikibooks.org/wiki/media%3ASVGpointer-events01.svg`
55 `http://de.wikibooks.org/wiki/media%3ASVGpointer-events02.svg`
56 `http://de.wikibooks.org/wiki/media%3ASVGpointer-events03.svg`

Bei den hellgrün hinterlegten Quadraten führt ein Klick oder eine Aktivierung immer zu einem Start einer einfachen Farbanimation des Quadrates. Bei den hellgelb hinterlegten Quadraten führt ein Klick oder eine Aktivierung zu einem Start einer einfachen Farbanimation des Quadrates, wenn ein komplett transparenter Bereich Ereigniszieles ist. Bei den rosa hinterlegten Quadraten führt ein Klick oder eine Aktivierung nicht zu einem Start einer einfachen Farbanimation des Quadrates.

pointer-events und Text[57]

Ein referenzierter Text wird verwendet, um pointer-events bei einem Text zu testen.

Bei Text erfolgt die Erfassung eines Ereignisses auf Basis der Zelle eines Glyphen. Eine Differenzierung zwischen Füllung und Strich oder innen und außen findet nicht statt.

Bei den hellgrün hinterlegten Quadraten führt ein Klick oder eine Aktivierung immer zu einem Start einer einfachen Farbanimation des Quadrates. Bei den hellgelben nur am Rand des Quadrates außerhalb der Zelle eines Glyphen.

9.9.2 Positionsmarke, cursor

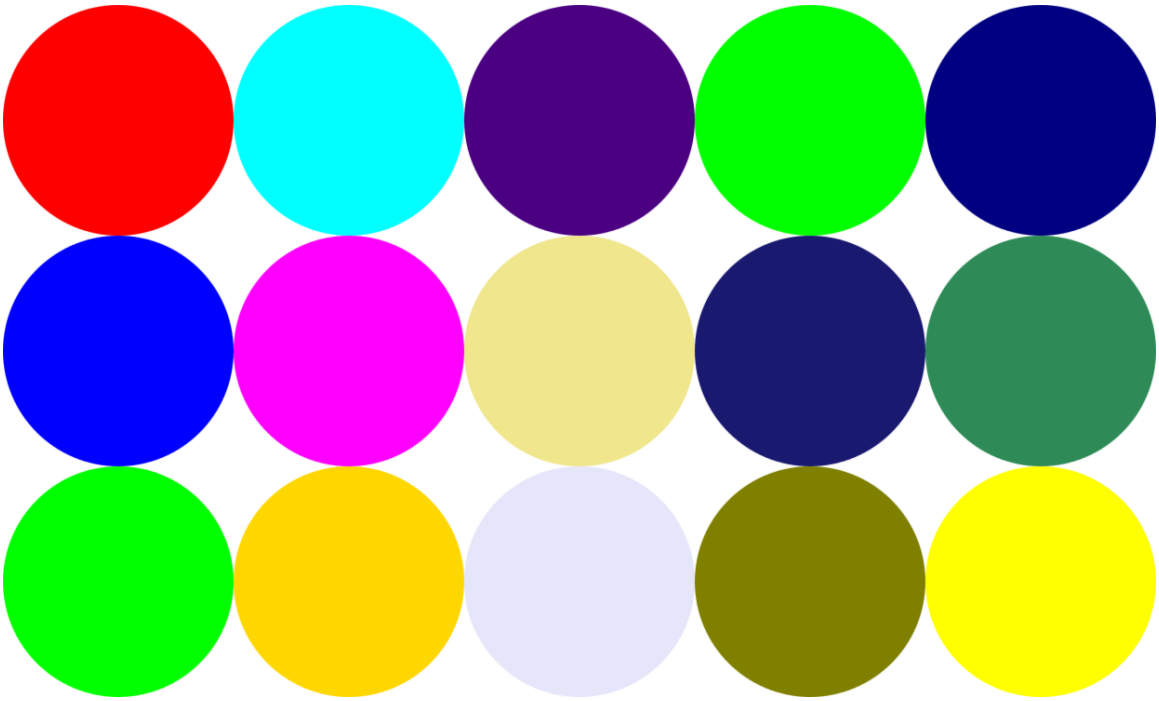


Abb. 154 Positionsmarke

Mit der Eigenschaft *cursor* kann das Markierungssymbol für die Position eine Zeigegerätes wie einer Maus ausgewählt werden.

Mögliche Werte sind:

inherit

57 `http://de.wikibooks.org/wiki/media%3ASVGpointer-events04.svg`

geerbt

auto

Das Darstellungsprogramm bestimmt das Aussehen je nach aktueller Situation

crosshair

Fadenkreuz, ähnlich einem '+'

default

Voreinstellung des Darstellungsprogrammes, of ein Pfeil

pointer

Zeiger, der auf einen Verweis hindeutet

move

Zeigt an, dass etwas bewegt wird

e-resize

Größenänderung nach rechts

w-resize

Größenänderung nach links

n-resize

Größenänderung nach oben

s-resize

Größenänderung nach unten

ne-resize

Größenänderung nach oben-rechts

nw-resize

Größenänderung nach oben-links

se-resize

Größenänderung nach unten-rechts

sw-resize

Größenänderung nach unten-links

text

Zeigt an, dass Text ausgewählt werden kann, oft ähnlich 'I'

wait

Zeigt an, dass das Programm beschäftigt ist und abgewartet werden soll, oft eine Uhr oder eine Sanduhr

help

Zeigt an, dass Hilfe zum betreffenden Objekt verfügbar ist, häufig ein Fragezeichen

URI-Liste und Ersatzwert

Eine Liste von eigenen Symbolen kann angegeben werden, einschließlich eines Ersatzes in Form einer der anderen vorherigen Werte

Der Initialwert ist 'auto', die Eigenschaft wird vererbt und ist animierbar. Die Eigenschaft ist nicht in SVG tiny 1.1 oder tiny 1.2 verfügbar.

Die Angabe mit einer URI-Liste sieht zum Beispiel so aus:

cursor='url("Zeiger.svg#Hilfe"), url("HilfeZeiger.svg"), url("HilfeZeiger.cur"), url("HilfeZeiger.csr"), url("HilfeZeiger.gif"), url("HilfeZeiger.png"), url("HilfeZeiger.jpeg"), help'

Der jeweiligen URI voran geht also die Zeichenkette 'url(' und hinter der URI folgt das Zeichen ')' - dazwischen steht die URI, gegebenenfalls in doppelten oder einfachen Anführungszeichen. Es handelt sich um eine mit Kommata separierte Liste mit zusätzlichem optionalen Leerraum.

Wenn die erste Wahl nicht darstellbar ist, wird die zweite genommen und so weiter bis zuletzt eines der generischen Symbole angeben ist, für welches immer eine Darstellung möglich sein sollte. Da ein Darstellungsprogramm für SVG immer auch SVG-Dokumente oder Fragmente darstellen können sollte, ebenso wie PNGs oder JPEGs, sollten diese Alternativen eigentlich immer funktionieren. Indes, es gibt Implementierungslücken, weswegen auch die Angabe exotischerer Formate hilfreich sein kann.

Mittels des SVG-Elementes ***cursor*** kann plattformunabhängig ein Symbol definiert werden, welches von einem SVG-Darstellungsprogramm interpretiert werden muss.

Beispiel:

```
<circle cx="50" cy="50"  r="50" fill="red" cursor="crosshair" />
<circle cx="50" cy="150" r="50" fill="blue" cursor="hand" />
<circle cx="50" cy="250" r="50" fill="green" cursor="wait" />

<circle cx="150" cy="50"  r="50" fill="Cyan" cursor="help" />
<circle cx="150" cy="150" r="50" fill="Fuchsia" cursor="text" />
<circle cx="150" cy="250" r="50" fill="Gold" cursor="move" />

<circle cx="250" cy="50"  r="50" fill="Indigo" cursor="n-resize" />
<circle cx="250" cy="150" r="50" fill="Khaki" cursor="s-resize" />
<circle cx="250" cy="250" r="50" fill="Lavender" cursor="e-resize" />

<circle cx="350" cy="50"  r="50" fill="Lime" cursor="w-resize" />
<circle cx="350" cy="150" r="50" fill="MidnightBlue" cursor="ne-resize" />
<circle cx="350" cy="250" r="50" fill="Olive" cursor="se-resize" />

<circle cx="450" cy="50"  r="50" fill="Navy" cursor="sw-resize" />
<circle cx="450" cy="150" r="50" fill="SeaGreen" cursor="auto" />
<circle cx="450" cy="250" r="50" fill="Yellow" cursor="default" />
```

9.10 Stilvorlagen

Wie bereits erwähnt, können in der Vollversion von SVG 1.1 auch Stilvorlagen eingesetzt werden.

Zum einen können dafür Stilvorlagenverarbeitungsanweisungen verwendet werden. Dann können innerhalb des Elementes auch Stilvorlagen innerhalb des Elementes ***style*** notiert werden. Wie bereits erwähnt, wenig zu empfehlen ist der Einsatz des Attributes *style*, auch können darin keine Selektoren oder Pseudoklassen notiert werden.

Wenn ein Darstellungsprogramm allerdings überhaupt die Angabe von Stilvorlagen interpretiert, dann auch alle Möglichkeiten, diese zu notieren. Es liegt also zum Beispiel eine fehlerhafte Implementierung vor, wenn Angaben im Attribut *style* interpretiert würden, nicht aber solche in externen Stilvorlagen.

Ähnlich wie für (X)HTML definiert SVG auch für das Attribut *class* eine Sonderrolle bezüglich der CSS-Notation. Eine vereinfachte Notation wie zum Beispiel circle.nr1 für einen Kreis mit einem Attribut *class*, in dessen Wertliste eine Klasse 'nr1' notiert ist, ist also zulässig. Die allgemeine CSS-Schreibweise dazu ist entsprechend circle[class~="nr1"].

Auch andere Selektoren von CSS 2.0(!) können für SVG-Dokumente in der Vollversion 1.1 verwendet werden.

9.10.1 Einbindung externer Stilvorlagen mit Stilvorlagenverarbeitungsanweisungen

Eine SVG-Datei mit externen Stilvorlagen könnte zum Beispiel so aussehen:

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet href="css1.css" type="text/css" title="Stil 1"
   alternate="yes" media="all" ?>
<?xml-stylesheet href="css2.css" type="text/css" title="Stil 2"
   alternate="yes" media="all" ?>
<?xml-stylesheet href="css0.css" type="text/css" title="Kein CSS"
   media="all" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg viewBox="-150 -150 300 300"
  xmlns="http://www.w3.org/2000/svg" version="1.1"
  stroke="black" xml:lang="de">
<title>Beispiel für ein SVG-Dokument mit Kreisen</title>

<circle id="erster" class="nr1" r="120" fill="red" />
<circle class="nr2" r="110" fill="blue" />
<circle class="nr3" r="100" fill="green" />
<circle class="nr4" r="90" fill="yellow" />
<circle class="nr1" r="80" fill="red" />
<circle class="nr2" r="70" fill="blue" />
<circle class="nr3" r="60" fill="green"/>
<circle class="nr4" r="50" fill="yellow" />
<circle class="nr1" r="40" fill="red" />
<circle class="nr2" r="30" fill="blue" />
<circle class="nr3" r="20" fill="green" />
<circle class="nr4" r="10" fill="yellow" />

</svg>
```

Mittels der Stilvorlangenverarbeitungsanweisungen werden also alternative Stilvorlagen angeboten, die im Darstellungsprogramm auswählbar sein sollten.

Die CSS-Datei css0.css könnte so aussehen:

Das ist also eine leere Datei, das wäre dann die primäre Variante, bei der die angegebenen Präsentationsattribute nicht überschrieben werden.

Die CSS-Datei css1.css könnte so aussehen:

```
circle {stroke-width: 2; stroke-dasharray: 10,10}
circle.nr1 {fill:#f40; stroke: #0dd}
circle.nr2 {fill:#20a; stroke: #f80}
circle.nr3 {fill:#0a2; stroke: #a0a}
circle.nr4 {fill:#ef2; stroke: #006}
```

Die Angaben in dieser Variante sind spezifischer als die Präsentationsattribute, überschreiben diese also.

Die CSS-Datei css2.css könnte so aussehen:

```
circle {stroke-width: 3; stroke-dasharray: 10,10}
circle.nr4 {fill:#f40; stroke: #0dd}
circle.nr3 {fill:#20a; stroke: #f80}
circle.nr2 {fill:#0a2; stroke: #a0a}
circle.nr1 {fill:#ef2; stroke: #006}

circle#erster {stroke-width: 5; stroke-dasharray:20,20;
fill:#fd0; stroke:#02a}
```

9.10.2 Verwendung des Elementes *style*

Mittels der CSS-Importregel können externe Stilvorlagen auch innerhalb des Elementes ***style*** angegeben werden. Ein Beispiel (Inhalt der Datei css1.css etwa wie im vorherigen Abschnitt):

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg viewBox="-150 -150 300 300"
  xmlns="http://www.w3.org/2000/svg" version="1.1"
  stroke="black">
<title>Beispiel für ein SVG-Dokument mit Kreisen</title>
<defs>
  <style type="text/css">
    @import "css1.css";
  </style>
</defs>

<circle class="nr1" r="120" fill="red" />
<circle class="nr2" r="110" fill="blue" />
<circle class="nr3" r="100" fill="green" />
<circle class="nr4" r="90" fill="yellow" />
<circle class="nr1" r="80" fill="red" />
<circle class="nr2" r="70" fill="blue" />
<circle class="nr3" r="60" fill="green"/>
<circle class="nr4" r="50" fill="yellow" />
<circle class="nr1" r="40" fill="red" />
<circle class="nr2" r="30" fill="blue" />
<circle class="nr3" r="20" fill="green" />
<circle class="nr4" r="10" fill="yellow" />
```

```
</svg>
```

Mit dem Attribut *type* wird angegeben, welche Stilvorlagensprache verwendet wird, hier als 'text/css', was für CSS steht. *type* ist nicht animierbar.

Zudem kann das Attribut *media* angegeben werden, mit welchem festgelegt werden kann, für welche Medien die Stilvorlage gilt. Dies entspricht dem gleichnamigen Attribut in der Stilvorlagenverarbeitungsanweisung.

Mögliche Werte sind:

all

Alle, Voreinstellung

aural

Hörbare Ausgabe

braille

ertastbare Braille-Ausgabe

embossed

Braille-Drucker

handheld

Kleingerät wie etwa Mobiltelephone

print

Druckausgabe

projection

Zur Projektion

screen

Computermonitore

tty

Medien mit sehr begrenzter Zellenausgabe

tv

Fernsehausgabe, geringe Auflösung, kaum rollbar, aber Ton verfügbar

Ein weiteres Attribut ist *title*, mit dem kann der Stilvorlage ein Titel gegeben werden.

Alternativ können die CSS-Eigenschaften auch direkt innerhalb des Elementes ***style*** notiert werden, was den Vorteil hat, dass das Dokument nicht von externen Dateien abhängig ist, die Stilvorlage kann also nicht versehentlich verlorengehen. Allerdings sind so die Stilvorlagen nicht effektiv wiederverwendbar.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg viewBox="-150 -150 300 300"
  xmlns="http://www.w3.org/2000/svg" version="1.1"
  stroke="black">
<title>Beispiel für ein SVG-Dokument mit Kreisen</title>
<defs>
<style type="text/css">
circle {stroke-width: 3; stroke-dasharray: 10,10}
circle.nr4 {fill:#f40; stroke: #0dd}
circle.nr3 {fill:#20a; stroke: #f80}
circle.nr2 {fill:#0a2; stroke: #a0a}
circle.nr1 {fill:#ef2; stroke: #006}

circle#erster {stroke-width: 5; stroke-dasharray:20,20;
fill:#fd0; stroke:#02a}
</style>
</defs>

<circle id="erster" class="nr1" r="120" fill="red" />
<circle class="nr2" r="110" fill="blue" />
<circle class="nr3" r="100" fill="green" />
<circle class="nr4" r="90" fill="yellow" />
<circle class="nr1" r="80" fill="red" />
<circle class="nr2" r="70" fill="blue" />
<circle class="nr3" r="60" fill="green"/>
<circle class="nr4" r="50" fill="yellow" />
<circle class="nr1" r="40" fill="red" />
<circle class="nr2" r="30" fill="blue" />
<circle class="nr3" r="20" fill="green" />
<circle class="nr4" r="10" fill="yellow" />

</svg>
```

Zu beachten ist dabei, dass der Inhalt von ***style*** (vom XML-parser) interpretiert wird. Zeichen wie insbesondere &, < und >, die mit der XML-Notation im Konflikt stehen, sind zu maskieren.

Beispiel:

```
<defs>
<style type="text/css">

g.Bsp circle {stroke:#f08}

g.Bsp &gt; circle {stroke:yellow}

</style>
</defs>

<g class="Bsp">
<circle r="50" fill="red" />
<circle r="40" fill="blue" />
<g>
<circle r="30" fill="green" />
<circle r="20" fill="#0a5" />
</g>
<g>
```

Alternativ kann auch die allgemeine Schreibweise aus SGML und XML verwendet werden, um anzudeuten, dass der Inhalt nicht vom XML-parser interpretiert werden soll:

```
<defs>
```

```
<style type="text/css">
<![CDATA[
g.Bsp circle {stroke:#f08}

g.Bsp > circle {stroke:yellow}
]]>
</style>
</defs>

<g class="Bsp">
<circle r="50" fill="red" />
<circle r="40" fill="blue" />
<g>
<circle r="30" fill="green" />
<circle r="20" fill="#0a5" />
</g>
<g>
```

9.10.3 Verwendung des Attributes *style*

Auf die Probleme mit dem Attribut *style* wurde bereits eingegangen. Die Schreibweise und die Möglichkeiten und auch die Probleme entsprechen denen beim Einsatz in (X)HTML.

Das Attribut selbst ist nicht animierbar. Der Attributwert ist eine Liste von Eigenschaften mit Wertzuweisungen, die jeweils voneinander mit einem Semikolon getrennt sind.

Beispiel:

```
<circle r="100" fill="#808" stroke="#0a0"
style="fill:magenta;stroke:green;stroke-dasharray:10,10;stroke-width:20" />
```

Durch Angabe der Präsentationsattribute wird sichergestellt, dass der Kreis auch bunt angezeigt wird, wenn der Inhalt von *style* etwa von Darstellungsprogrammen ignoriert wird, die auf das tiny-Profil ausgelegt sind. Weil die im style-Attribut notierten Eigenschaften eine höhere Spezifität haben, überschreiben diese die Präsentationsattribute. Weil sie auch eine höhere Spezifität haben als die meisten Selektoren, die innerhalb des Elementes ***style*** oder in einer externen Stilvorlage angegeben werden können, erweist sich die Verwendung dieses Attributes als recht sperrig für Autoren und Nutzer. Ein Nutzer müsste praktisch in seiner eigenen Stilvorlage direkt dieses Element mit der !important-Regel überschreiben. Es wäre also erforderlich, das einzelne Dokument zu analysieren und dann eine spezifische Stilvorlage dafür anzulegen.

9.10.4 CSS-Pseudoklassen

Sofern das Darstellungsprogramm CSS interpretiert, ist auch eine Interpretation von Pseudoklassen wie :hover, :active, :focus oder :first-child, :visited, :link und :lang erforderlich.

Die Angaben sind dann entweder in einer externen Stilvorlage oder innerhalb des Elementes ***style*** zu notieren.

Beispiel mit Pseudoklasse :hover

```
<defs>
        <linearGradient id="verlauf1" x1="50%" x2="50%" y1="0%" y2="100%">
```

```
                <stop offset="0" stop-color="black"/>
                <stop offset="1" stop-color="white"/>
        </linearGradient>
        <linearGradient id="verlauf2" x1="50%" x2="50%" y1="0%" y2="100%">
                <stop offset="0" stop-color="red"/>
                <stop offset="1" stop-color="white"/>
        </linearGradient>
        <linearGradient id="verlauf3" x1="50%" x2="50%" y1="0%" y2="100%">
                <stop offset="0" stop-color="navy"/>
                <stop offset="1" stop-color="white"/>
        </linearGradient>

<style type="text/css">
#r1{fill:url(#verlauf1)}
#r2{fill:url(#verlauf2)}
#r3{fill:url(#verlauf3)}

#r1:hover{fill:url(#verlauf2)}
#r2:hover{fill:url(#verlauf3)}
#r3:hover{fill:url(#verlauf1)}
</style>
</defs>

<rect width="300" height="100" x="50" y="20" ry="10" rx="10" id="r1" />
<rect width="300" height="100" x="50" y="220" ry="10" rx="10" id="r2"/>
<rect width="300" height="100" x="50" y="420" ry="10" rx="10" id="r3"/>
```

9.10.5 Gemeinsamkeiten und Abweichungen: 'CSS für SVG' und 'CSS allgemein'

Generell werden die Regeln von CSS übernommen, wenn das CSS auf SVG angewendet wird. Eine Ausnahme ist die Angabe von Längen (und Winkeln). Während CSS allgemein die Angabe von Einheiten vorsieht, sind diese bei Anwendung auf SVG optional. Längen ohne Einheitsangaben werden auf das lokale Koordinatensystem bezogen, in Einheiten von 1. Winkelangaben werden als in Grad angegeben angenommen.

Erwähnt wurde auch bereits der spezielle Selektor '.' für Listenbestandteile im Wert des Attributes *class* innerhalb von CSS, welches auf SVG angewendet wird. Dies deckt sich mit dem Selektor für CSS auf (X)HTML angewendet. Dieser Selektor ist nicht allgemein verfügbar für CSS angewendet auf XML, wie bereits in CSS erläutert, sofern das jeweilige Format nicht wie SVG und (X)HTML spezifiziert, welches Attribut mit dem Selektor '.' ausgewertet werden soll.

Wenn ein Darstellungsprogramm über eine akustische Ausgabe verfügt und entsprechende Eigenschaften von CSS2.0 für akustische Präsentationen interpretiert, so können diese auch in SVG notiert werden. Sie haben eine Bedeutung für Elemente, die Text enthalten. Präsentationsattribute gibt es zu diesen Eigenschaften allerdings nicht.

9.10.6 Stilvorlagen mit XSL

Mit XSL (erweiterbare Stilvorlagensprache; englisch: eXtensible Stylesheet Language) können prinzipiell auch Stilvorlagen angeboten werden. XSL gehört wie SVG zur XML-Sprachfamilie. XSL wird insbesondere auch bereits vor jeglicher Darstellung mit gesonderten Programmen verwendet werden, um Dokumente von einem Format in ein anderes zu transformieren (XSLT, wobei T für Transformation steht). Darstellungsprogramme können jedoch ebenfalls eine Möglichkeit haben, Stilvorlagen in diesem Format zu interpretieren. Auch dann kann von einem eigenen Format zum

Beispiel in ein SVG-Dokument transformiert werden. XSL kann aber auch ähnlich wie CSS verwendet werden, um den Inhalt zu dekorieren - oder in einer Mischung von beiden, ein SVG-Dokument in ein anderes SVG-Dokument zu verwandeln, welches anders dargestellt wird als das Dokument ohne die Stilvorlage.

XSL selbst zu erkären, liegt außerhalb des Themenbereiches dieses Buches, daher gibt es hier nur ein einfaches Beispiel. Gegeben sei folgendes SVG-Dokument:

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="style.xsl" type="text/xsl" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg"
     version="1.1"
     viewBox="0 0 300 200"
     xml:lang="de">
<title>Beispiel SVG mit XSL</title>
<desc>
Beispiel zur Verwendung von XSL in SVG.
Anders als bei CSS ist es mit XSL auch möglich, Attribute zu ändern
oder ganze Elemente auszutauschen.
</desc>

 <rect class="hintergrund"
       x="0" y="0" width="300" height="200"
       fill="#ffa"/>

  <circle cx="160" cy="100" r="30" fill="yellow"/>

  <rect x="25" y="25" width="60" height="30"/>

  <rect id="r" x="25" y="85" width="60" height="30"/>

  <rect class="bsp" x="25" y="145" width="60" height="30"/>

  <ellipse cx="250" cy="100" rx="10" ry="20"/>

  <g id="anders" />
</svg>
```

Mittels einer XML-Stilvorlagenverarbeitungsanweisung wird hier auf das Dokument 'style.xsl' verwiesen. XSL wird typisch in eigenständigen Dateien notiert, auf welche solch eine Anweisung verweist. Das Dokument 'style.sxl' mit der Vorlage könnte zum Beispiel folgenden Inhalt haben, wobei bereits in Kommentaren grob zur Information angegeben ist, was wozu gut ist:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:svg="http://www.w3.org/2000/svg">
  <xsl:output
    method="xml"
    encoding="utf-8"/>
  <!-- Wurzelelement kopieren und Vorlagen anwenden -->
  <xsl:template match="/svg:svg">
    <xsl:copy>
      <xsl:copy-of select="@*"/>
      <xsl:apply-templates/>
    </xsl:copy>
  </xsl:template>

  <!-- title kopieren -->
  <xsl:template match="svg:title">
    <xsl:copy>
```

```
      <xsl:value-of select="." />
    </xsl:copy>
  </xsl:template>
  <!-- desc kopieren -->
  <xsl:template match="svg:desc">
    <xsl:copy>
      <xsl:value-of select="." />
    </xsl:copy>
  </xsl:template>

<!-- Kreise -->
  <xsl:template match="svg:circle">
    <xsl:copy>
      <xsl:copy-of select="@*"/>
      <xsl:attribute name="fill">red</xsl:attribute>
      <xsl:attribute name="stroke">blue</xsl:attribute>
      <xsl:attribute name="stroke-width">3</xsl:attribute>
      <xsl:attribute name="r">50</xsl:attribute>
      <xsl:attribute name="cx">160</xsl:attribute>
      <xsl:attribute name="cy">70</xsl:attribute>
    </xsl:copy>
  </xsl:template>

<!-- Ellipsen -->
  <xsl:template match="svg:ellipse">
    <xsl:copy>
      <xsl:copy-of select="@*"/>
      <xsl:attribute name="fill">#ff0</xsl:attribute>
      <xsl:attribute name="stroke">#00a</xsl:attribute>
      <xsl:attribute name="stroke-width">10</xsl:attribute>
      <xsl:attribute name="rx">20</xsl:attribute>
      <xsl:attribute name="ry">80</xsl:attribute>
    </xsl:copy>
  </xsl:template>

  <!-- Rechtecke -->
  <xsl:template match="svg:rect">
    <xsl:copy>
      <xsl:copy-of select="@*"/>
      <xsl:attribute name="fill">#f0f</xsl:attribute>
      <xsl:attribute name="stroke">#ff0</xsl:attribute>
      <xsl:attribute name="stroke-width">3</xsl:attribute>
    </xsl:copy>
  </xsl:template>

  <!-- Element mit Identifizierer r -->
  <xsl:template match="id('r')">
    <xsl:copy>
      <xsl:copy-of select="@*"/>
      <xsl:attribute name="fill">#0ff</xsl:attribute>
      <xsl:attribute name="stroke">#f80</xsl:attribute>
      <xsl:attribute name="stroke-width">1</xsl:attribute>
    </xsl:copy>
  </xsl:template>

  <!-- Elemente der Klasse bsp -->
  <xsl:template match="*[@class='bsp']">
    <xsl:copy>
      <xsl:copy-of select="@*"/>
      <xsl:attribute name="fill">#080</xsl:attribute>
      <xsl:attribute name="stroke">#f02</xsl:attribute>
      <xsl:attribute name="stroke-width">5</xsl:attribute>
      <xsl:attribute name="width">180</xsl:attribute>
    </xsl:copy>
  </xsl:template>

  <!-- Elemente der Klasse hintergrund -->
  <xsl:template match="*[@class='hintergrund']">
```

```
    <xsl:copy>
      <xsl:copy-of select="@*"/>
      <xsl:attribute name="fill">#eee</xsl:attribute>
      <xsl:attribute name="stroke">#666</xsl:attribute>
      <xsl:attribute name="stroke-width">1</xsl:attribute>
    </xsl:copy>
  </xsl:template>

  <!-- Element mit Identifizierer 'anders' austauschen -->
  <xsl:template match="id('anders')">
          <svg:g fill="#cff" stroke="#404" stroke-linejoin="round" stroke-width="2"
                transform="translate(160 70) scale(0.5)">
          <svg:path d="M-100,0Q0,0 0,-100 0,0 100,0 0,0 0,100 0,0 -100,0Z"/>
          <svg:path d="M-50,-50Q0,0 50,-50 0,0 50,50 0,0 -50,50 0,0 -50,-50Z"
 fill-opacity="0.5"/>
          <svg:path d="M-50,0Q0,0 0,-50 0,0 50,0 0,0 0,50 0,0 -50,0Z"
 fill-opacity="0.3333"/>
          <svg:path d="M-25,-25Q0,0 25,-25 0,0 25,25 0,0 -25,25 0,0 -25,-25Z"
 fill-opacity="0.25"/>
          <svg:path d="M-25,0Q0,0 0,-25 0,0 25,0 0,0 0,25 0,0 -25,0Z"
 fill-opacity="0.2"/>
          </svg:g>
  </xsl:template>

</xsl:stylesheet>
```

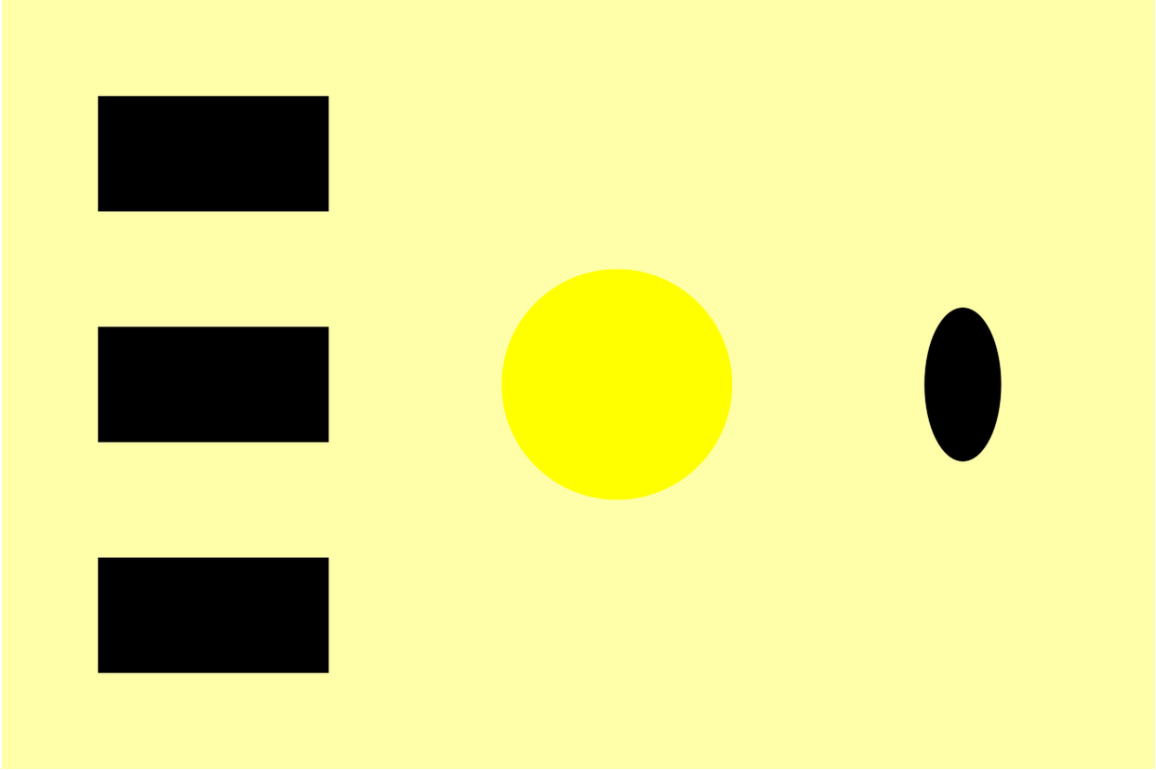


Abb. 155 Beispiel SVG mit XSL

Prinzipiell kann XSL auch direkt im SVG-Dokument notiert werden. Dabei ist davon auszugehen, dass dies etwa einen Validator überfordert, der auf DTDs basiert, denn anders als bei CSS taucht so wirklich eine andere XML-Sprache im Dokument auf. SVG schreibt andererseits aber explizt nicht vor, wie die Sprache auszusehen hat, welche innerhalb des Elementes ***style*** zu notieren ist. Es ist

explizit bei der Definition des Elementes angegeben, dass die Sprache der Stilvorlage dies festlegt. Da es sich bei XSL um XML handelt, gibt es da auch keine technischen Probleme bei der Notation.

Allerdings wird trotzdem typisch eine XML-Stilvorlagenverarbeitungsanweisung anzugeben sein, welche auf das XSL-Fragment verweist, damit interpretierwillige Darstellungsprogramme Notiz von der Vorlage nehmen. Das Problem dabei ist, dass der dabei notwendige Fragmentidentifizierer in einem als dafür geeigneten Attribut notiert sein muss. Das Attribut *xml:id* eignet sich dafür einerseits als generischer Identifizierer. Er ist andererseits allerdings in älteren (Versionen von) Darstellungsprogrammen noch nicht implementiert, welche dann eine Fehlermeldung ausgeben werden, weil sie die Stilvorlage nicht finden können. Somit ist es in der Praxis vorzuziehen, wie oben beschrieben vorzugehen und auf ein externes Dokument zu verweisen.

Obiges Beispiel mit wie beschrieben integriertem XSL ist zum Beispiel mit einer aktuellen Version von Opera nachvollziehbar, dort sind bereits alle erforderlichen Bestandteile samt *xml:id* implementiert:

Beispiel SVG mit XSL[58].

Beispiel für die Transformation eines allgemeinen XML-Dokumentes in ein SVG-Dokument, wobei Inhalte von XML-Elementen verwendet werden, um per XSLT die Opazität von Flächen in der SVG-Ausgabe festzulegen:

Wahlbeteiligung einer fiktiven Wahl in Österreich[59].

58 `http://de.wikibooks.org/wiki/media%3ASVG_Stilvorlage_mit_XSL01.svg`
59 `http://de.wikibooks.org/wiki/SVG%2F%20XSLT`

10 Farben

10.1 Farbwerte in SVG

Wie bereits für die Eigenschaft *fill* erläutert, können Farbwerte in SVG in verschiedener Form angegeben werden. Immer verfügbar ist die numerische Angabe, entweder in hexadezimaler Form (#abc oder #abcdef) oder in dezimaler Form (rgb(1,2,3)) oder als Prozentwerte (rgb(1%,2%,3%)). Dazu gibt es die sechzehn Farbwörter von (X)HTML: black, green, silver, lime, gray, olive, white, yellow, maroon, navy, red, blue, purple, teal, fuchsia, aqua (siehe unten für die entsprechenden numerischen Angaben).

Das volle Profil von SVG 1.1 und auch SVG tiny 1.2 verfügen über die ursprünglich in CSS2 definierten Schlüsselwörter für Systemfarben.

In SVG tiny 1.2 wurden zudem feste Farben als Maldienst neu eingeführt. Damit kann der Autor selbst Schlüsselwörter beziehungsweise einen Maldienst für oft verwendete Farben definieren.

Im vollen Profil von SVG 1.1 sind ferner einige weitere Farben mit Namen benannt (wie ursprünglich in CSS definiert), diese Schlüsselwörter müssen dann allerdings nicht bei einem Darstellungsprogramm für das tiny-Profil funktionieren.

10.1.1 Hexadezimaler Farbwert

Zur Angabe eines hexadezimalen Farbwertes wird eine Notation mit drei Farbkanälen, rot, grün, blau im Standardfarbraum sRGB verwendet. Die Angabe beginnt mit einem # gefolgt von hexadezimalen Ziffern.

Dies ist entweder exakt eine Ziffer für jeden Farbkanal, also 3 Ziffern, jeweils eine Möglichkeit von folgenden: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, A, b, B, c, C, d, D, e, E, f, F. Große und kleine Buchstaben bedeuten jeweils das gleiche, im dezimalen System sind a bis f die Zahlen 10 bis 15.

Alternativ kann auch eine Notation mit zwei von den Ziffern pro Kanal gewählt werden, deckt dann dezimal den Bereich von 0 bis 255 ab. Die Konversion von der einziffrigen Notation in der mit zwei Ziffern pro Kanal erfolgt einfach durch Verdopplung der einzelnen Ziffer, #fff ist also gleich #ffffff oder auch weiß oder #abc ist gleich #aabbcc.

10.1.2 Dezimaler Farbwert

Dezimale Angaben aus dem gleichen Bereich von 0 bis 255 für jeden Kanal sind ebenfalls verfügbar. Es gibt wieder die drei Kanäle des Standardfarbraumes sRGB. Die Notation beginnt mit 'rgb(' und endet mit ')' dazwischen befinden sich die Angaben zu den drei Kanälen, jeweils mit Komma separiert, also zum Beispiel rgb(255,255,255) entspricht #fff in hexadezimaler Schreibweise oder auch weiß.

10.1.3 Prozentualer Farbwert

Statt einer dezimalen Angabe ist auch die Angabe von drei Prozentwerten möglich, jeweils im Bereich von 0% bis 100%, die Angabe des Symboles '%' ist wichtig, damit das Darstellungsprogramm zwischen Prozentangaben und dezimalen Angaben unterscheiden kann. Es darf nicht gemischt werden, also ein Kanal in dezimaler Notation und einer in prozentualer Notation ist nicht erlaubt. Ansonsten ist die Notation für prozentuale Angaben die gleiche wie für dezimale, Beispiel: rgb(17.1345%,0.57%,99.2%). Anders als bei den anderen Notationen sind bei Prozentangaben auch nichtganzzahlige Angaben erlaubt.

10.2 Farbprofil

Mit dem Element ***color-profile*** kann ein ICC-Farbprofil (International Color Consortium) festgelegt werden. Die Funktionalität dieses Elementes entspricht der CSS-Regel @color-profile.

10.2.1 Attribut *href* von XLink

Mit dem Attribut wird eine Ressource für ein ICC-Farbprofil referenziert.

Für dieses Element ist dieses Attribut nicht animierbar. Neben diesem Attribut sind auch die anderen von XLink verfügbar.

10.2.2 Attribut *local*

Alternativ zur Referenzierung einer externen Ressource mit *href* von XLink kann auch der Fragmentidentifizierer einer lokalen Ressource angegeben werden. Dies kann mit dem Attribut *local* passieren. Sind beide Möglichkeiten angegeben, wird erst die lokale Ressource durchsucht, wenn dort das Farbprofil nicht gefunden wird, wird auch noch die externe Ressource durchsucht.

Der Wert des Attributes ist eine Zeichenkette, welche einen Fragmentidentifizier im Sinne von ICC darstellt.

Das Attribut ist nicht animierbar.

10.2.3 Attribut *name*

Das Attribut gibt den Namen an, der als erster Parameter bei der Verwendung eines ICC-Profiles bei der Angabe eines Farbwertes (zum Beispiel für *fill* und *stroke etc) verwendet wird.*

Der Wert ist eine Zeichenkette. Das Attribut ist nicht animierbar.

Der Name 'sRGB' ist reserviert und nicht erlaubt.

10.2.4 Attribut *rendering-intent*

Das Attribut legt die beabsichtigte Darstellungsart fest. Möglich sind die Werte:

auto

Voreinstellung, das Darstellungsprogramm entscheidet automatisch

perceptual

Das Gamut wird verändert, um die Verhältnisse zwischen Farbwerten zu erhalten

relative-colorimetric

Angaben innerhalb des Gamuts bleiben erhalten, außerhalb wird an den Rand des erlaubten Bereiches verschoben

saturation

Die Farben werden so konvertiert, dass die Sättigung erhalten bleibt

absolute-colorimetric

Schaltet Punkt-für-Punkt-Anpassung ab, nicht empfohlen.

Das Attribut ist nicht animierbar.

10.2.5 Eigenschaft *color-profile*

Die Eigenschaft ist anwendbar für Elemente, die Pixelgraphik referenzieren, wie ***image*** und gibt das Farbprofil an, welches verwendet werden soll, um das Bild darzustellen.

Mögliche Werte sind:

inherit

geerbt

auto

Voreinstellung. Hat das Bild ein eigenes Farbprofil, wird dies verwendet, sonst sRGB.

sRGB

Farbprofil sRGB, wie auch anderweitig in SVG verwendet

Name

Angabe des Namens eines Farbprofils

IRI

IRI zu einem Farbprofil

Das Attribut ist animierbar. Die Eigenschaft wird vererbt.

10.3 Systemfarben oder Systemmaldienste

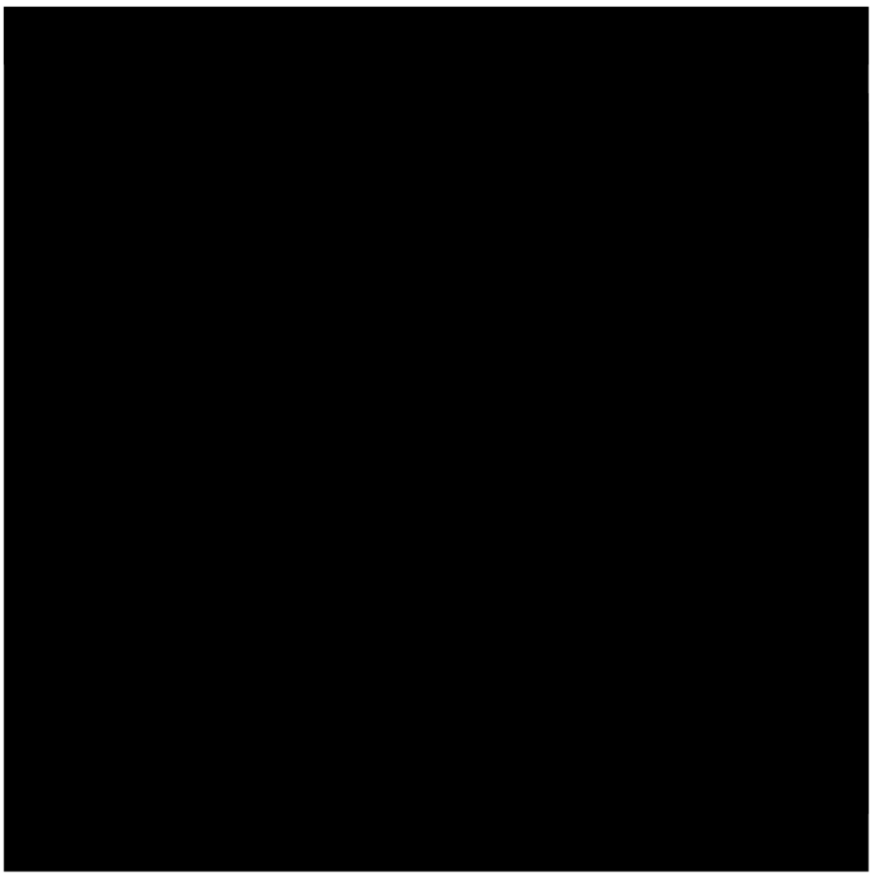


Abb. 156 Beispiel Systemfarben

Squiggle (Batik)	1.7
Opera (Presto)	9.0
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.0
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	-

In SVG 1.1 können Systemfarben mit Schlüsselwörtern angegeben werden. Ob die Systemfarben auch in SVG tiny 1.1 verfügbar sind, ist nicht eindeutig festgelegt. Die gleichen Schlüsselwörter werden allerdings in SVG tiny 1.2 als Maldienste bereitgestellt. Es gibt folgende Schlüsselwörter mit der Angabe, wo die entsprechenden Farben auftauchen:

ActiveBorder

Rand des aktiven Fensters.

ActiveCaption

Titel des aktiven Fensters.

AppWorkspace

Hintergrundfarbe der Schnittstelle bei mehrfachen Dokumenten.

Background

Arbeitsflächenhintergrund.

ButtonFace

Oberflächenfarbe von dreidimensionalen Anzeigelementen.

ButtonHighlight

Dunkler Schatten von dreidimensionalen Anzeigelementen (Kanten an der lichtabgewandten Seite).

ButtonShadow

Farbe des Schattens von dreidimensionalen Anzeigelementen.

ButtonText

Text auf Druckknöpfen.

CaptionText

Text der Überschrift, Größeneinstellungsbereich, Rollbalkenpfeilbereich.

GrayText

Deaktivierter, ausgegrauter Text.

Highlight

Ausgewählte Punkte einer Kontrolliste.

HighlightText

Text in ausgewählten Punkten einer Kontrolliste.

InactiveBorder

Inaktiver Fensterrand.

InactiveCaption

Inaktiver Fenstertitel.

InactiveCaptionText

Farbe des Textes eines inaktiven Fenstertitels.

InfoBackground

Hintergrundfarbe für Kontrollelemente von Nutzerhilfen.

InfoText

Textfarbe für Kontrollelemente von Nutzerhilfen.

Menu

Menü Hintergrund.

MenuText

Text in Menüs.

Scrollbar

Grauer Bereich eines Rollbalkens.

ThreeDDarkShadow

Dunkler Schatten von dreidimensionalen Anzeigelementen.

ThreeDFace

Overflächenfarbe von dreidimensionalen Anzeigelementen.

ThreeDHighlight

Herausstellungsfarbe von dreidimensionalen Anzeigelementen.

ThreeDLightShadow

Farbe des Lichtes von dreidimensionalen Anzeigelementen (die dem Licht zugewandten Kanten).

ThreeDShadow

Dunkler Schatten von dreidimensionalen Anzeigelementen.

Window

Fensterhintergrund.

WindowFrame

Fensterrahmen.

WindowText

Text im Fenster.

Anwendungsbeispiel:

```
fill="Window" stroke="WindowFrame"
```

Beispiel Systemfarben[1]

1 http://de.wikibooks.org/wiki/media%3ASVGSystemcolorexample.svg

10.4 Maldienst feste Farbe, *solidColor*

Squiggle (Batik)	1.7
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

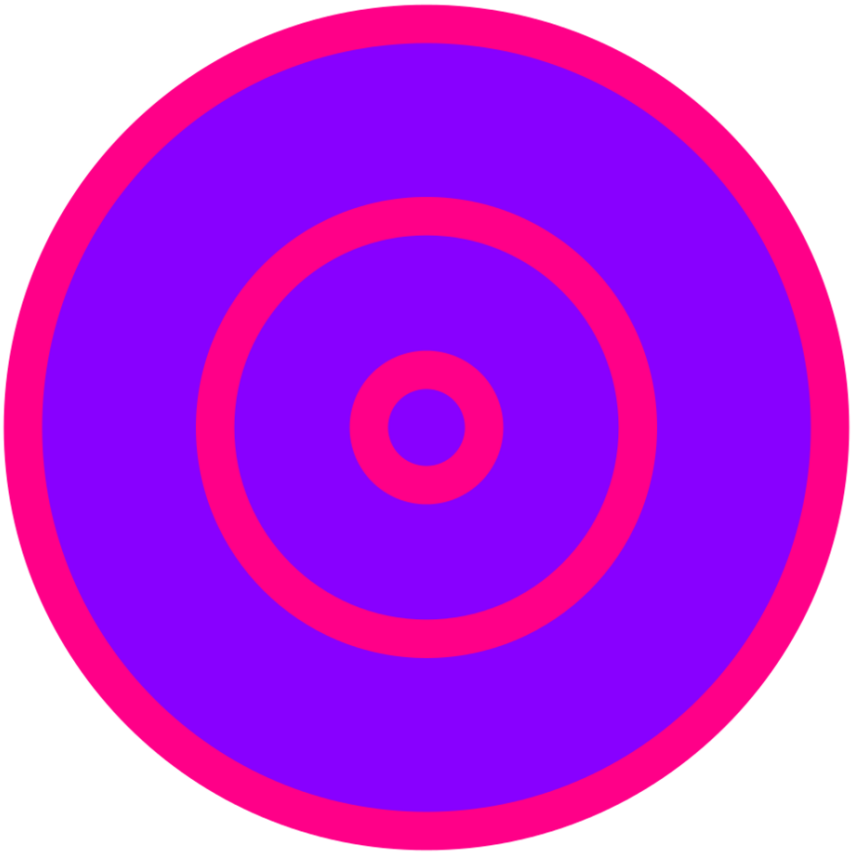


Abb. 157 Beispiel feste Farben

Mit dem Element ***solidColor*** bietet SVG tiny 1.2 eine Möglichkeit, Schlüsselwörter für Farben (einschließlich Opazität) festzulegen, um sie dann zu verwenden. Bei Bedarf können also auch die in der Vollversion von SVG 1.1 verfügbaren Farbnamen so definiert und verfügbar gemacht werden.

Die Farbe wird mit der Eigenschaft *solid-color* festgelegt, die Opazität mit der Eigenschaft *solid-opacity*. Mit dem Attribut *id* beziehungsweise *xml:id* wird das Element ***solidColor*** mit einem Fragmentidentifizierer versehen und kann so als Maldienst referenziert werden.

10.4.1 Eigenschaft feste Farbe, *solid-color*

solid-color legt eine feste Farbe fest. Mögliche Werte sind:

inherit

geerbt

currentColor

aktuelle Farbe, mit *color* festgelegt

Farbangabe

Eine Farbangabe in numerischer Form oder mit einem Farbnamen

Der Wert wird nicht vererbt, ist aber animierbar. Der vorgegebene Wert ist black (schwarz).

10.4.2 Eigenschaft feste Opazität, *solid-opacity*

solid-opacity legt eine feste Opazität fest. Mögliche Werte sind:

inherit

geerbt

Opazitätsangabe

eine Zahl zwischen 0 und 1 einschließlich, beziehungsweise ein Wert außerhalb dieses Intervalls wird vor der Darstellung auf den nächstgelegenen zulässigen Wert konvertiert. 0 ist komplett durchsichtig, 1 komplett undurchsichtig.

Der Wert wird nicht vererbt, ist aber animierbar. Der vorgegebene Wert ist 1 (undurchsichtig oder komplett opak).

10.4.3 Beispiel feste Farbe

Beispiel zur Anwendung:

```
<defs>
    <solidColor xml:id="UiUiUi" solid-color="#fc8" solid-opacity="0.8367">
    <animate attributeName="solid-color" dur="13s" repeatDur="indefinite"
             values="#fc8;#c40;#4c0;#fc8"/>
    </solidColor>
    <solidColor xml:id="OiOiOi" solid-color="#c8f" solid-opacity="0.7367"/>
    <solidColor xml:id="EiEiEi" solid-color="#cf8" solid-opacity="0.6367"/>
</defs>

<g color="#f08" fill="#80f">
<circle r="280" fill="url(#EiEiEi)" />
<circle r="260" fill="url(#OiOiOi)" />
```

```
<circle r="240" fill="url(#UiUiUi)" />
<circle r="220" fill="currentColor" />
<circle r="200" />
<circle r="180" fill="url(#EiEiEi)" />
<circle r="160" fill="url(#OiOiOi)" />
<circle r="140" fill="url(#UiUiUi)" />
<circle r="120" fill="currentColor" />
<circle r="100" />
<circle r="80" fill="url(#EiEiEi)" />
<circle r="60" fill="url(#OiOiOi)" />
<circle r="60" fill="url(#UiUiUi)" />
<circle r="40" fill="currentColor" />
<circle r="20" />
</g>
```

Beispiel feste Farben[2]

Konzentrische Kreise werden verschieden eingefärbt, wofür geerbte Werte für fill verwendet werden, currentColor und feste Farben, die mit solidColor definiert sind. Eine feste Farbe wird animiert.

10.5 Farbnamen in der Vollversion von SVG 1.1

Squiggle (Batik)	1.7
Opera (Presto)	9.0
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.0
Konqueror (KSVG)	3.2
Safari (Webkit)	3
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	1

In der Vollversion von SVG 1.1 können statt der numerischen Angaben auch folgende Farbnamen verwendet werden (Groß- oder Kleinschreibung ist nicht signifkant, kann beliebig gemischt werden):

2 http://de.wikibooks.org/wiki/media%3ASVGsolidColor01.svg

Farbname	Beispiel	Komponenten (Rot,Grün, Blau)					
		Hexadezimal			Dezimal		
AliceBlue		F0	F8	FF	240	248	255
AntiqueWhite		FA	EB	D7	250	235	215
Aqua		00	FF	FF	0	255	255
Aquamarine		7F	FF	D4	127	255	212
Azure		F0	FF	FF	240	255	255
Beige		F5	F5	DC	245	245	220
Bisque		FF	E4	C4	255	228	196
Black		00	00	00	0	0	0
BlanchedAlmond		FF	EB	CD	255	235	205
Blue		00	00	FF	0	0	255
BlueViolet		8A	2B	E2	138	43	226
Brown		A5	2A	2A	165	42	42
BurlyWood		DE	B8	87	222	184	135
CadetBlue		5F	9E	A0	95	158	160
Chartreuse		7F	FF	00	127	255	0
Chocolate		D2	69	1E	210	105	30
Coral		FF	7F	50	255	127	80
Cornflower		64	95	ED	100	149	237
Cornsilk		FF	F8	DC	255	248	220
Crimson		DC	14	3C	220	20	60
Cyan		00	FF	FF	0	255	255
DarkBlue		00	00	8B	0	0	139
Dark Cyan		00	8B	8B	0	139	139
DarkGoldenrod		B8	86	0B	184	134	11
DarkGray		A9	A9	A9	169	169	169
DarkGreen		00	64	00	0	100	0
DarkKhaki		BD	B7	6B	189	183	107
DarkMagenta		8B	00	8B	139	0	139
DarkOliveGreen		55	6B	2F	85	107	47
DarkOrange		FF	8C	00	255	140	0
DarkOrchid		99	32	CC	153	50	204
DarkRed		8B	00	00	139	0	0
DarkSalmon		E9	96	7A	233	150	122
DarkSeaGreen		8F	BC	8F	143	188	143
DarkSlateBlue		48	3D	8B	72	61	139
DarkSlateGray		2F	4F	4F	47	79	79
DarkTurquoise		00	CE	D1	0	206	209
DarkViolet		94	00	D3	148	0	211
DeepPink		FF	14	93	255	20	147
DeepSkyBlue		00	BF	FF	0	191	255
DimGray		69	69	69	105	105	105
DodgerBlue		1E	90	FF	30	144	255

Firebrick	B2	22	22	178	34	34
FloralWhite	FF	FA	F0	255	250	240
ForestGreen	22	8B	22	34	139	34
Fuchsia	FF	00	FF	255	0	255
Gainsboro	DC	DC	DC	220	220	220
GhostWhite	F8	F8	FF	248	248	255
Gold	FF	D7	00	255	215	0
Goldenrod	DA	A5	20	218	165	32
Gray, Grey	80	80	80	128	128	128
Green	0	80	0	0	128	0
GreenYellow	AD	FF	2F	173	255	47
Honeydew	F0	FF	F0	240	255	240
HotPink	FF	69	B4	255	105	180
IndianRed	CD	5C	5C	205	92	92
Indigo	4B	00	82	75	0	130
Ivory	FF	FF	F0	255	255	240
Khaki	F0	E6	8C	240	230	140
Lavender	E6	E6	FA	230	230	250
LavenderBlush	FF	F0	F5	255	240	245
LawnGreen	7C	FC	00	124	252	0
LemonChiffon	FF	FA	CD	255	250	205
LightBlue	AD	D8	E6	173	216	230
LightCoral	F0	80	80	240	128	128
LightCyan	E0	FF	FF	224	255	255
LightGoldenrod	FA	FA	D2	250	250	210
LightGreen	90	EE	90	144	238	144
LightGrey	D3	D3	D3	211	211	211
LightPink	FF	B6	C1	255	182	193
LightSalmon	FF	A0	7A	255	160	122
LightSeaGreen	20	B2	AA	32	178	170
LightSkyBlue	87	CE	FA	135	206	250
LightSlateGray	77	88	99	119	136	153
LightSteelBlue	B0	C4	DE	176	196	222
LightYellow	FF	FF	E0	255	255	224
Lime	00	FF	00	0	255	0
LimeGreen	32	CD	32	50	205	50
Linen	FA	F0	E6	250	240	230
Magenta	FF	00	FF	255	0	255
Maroon	80	00	00	128	0	0
MediumAquamarine	66	CD	AA	102	205	170
MediumBlue	00	00	CD	0	0	205
MediumOrchid	BA	55	D3	186	85	211
MediumPurple	93	70	DB	147	112	219
MediumSeaGreen	3C	B3	71	60	179	113
MediumSlateBlue	7B	68	EE	123	104	238

MediumSpringGreen		00	FA	9A	0	250	154
MediumTurquoise		48	D1	CC	72	209	204
MediumVioletRed		C7	15	85	199	21	133
MidnightBlue		19	19	70	25	25	112
MintCream		F5	FF	FA	245	255	250
MistyRose		FF	E4	E1	255	228	225
Moccasin		FF	E4	B5	255	228	181
NavajoWhite		FF	DE	AD	255	222	173
Navy		00	00	80	0	0	128
OldLace		FD	F5	E6	253	245	230
Olive		80	80	00	128	128	0
OliveDrab		6B	8E	23	107	142	35
Orange		FF	A5	00	255	165	0
OrangeRed		FF	45	00	255	69	0
Orchid		DA	70	D6	218	112	214
PaleGoldenrod		EE	E8	AA	238	232	170
PaleGreen		98	FB	98	152	251	152
PaleTurquoise		AF	EE	EE	175	238	238
PaleVioletRed		DB	70	93	219	112	147
PapayaWhip		FF	EF	D5	255	239	213
PeachPuff		FF	DA	B9	255	218	185
Peru		CD	85	3F	205	133	63
Pink		FF	C0	CB	255	192	203
Plum		DD	A0	DD	221	160	221
PowderBlue		B0	E0	E6	176	224	230
Purple		80	00	80	128	0	128
Red		FF	00	00	255	0	0
RosyBrown		BC	8F	8F	188	143	143
RoyalBlue		41	69	E1	65	105	225
SaddleBrown		8B	45	13	139	69	19
Salmon		FA	80	72	250	128	114
SandyBrown		F4	A4	60	244	164	96
SeaGreen		2E	8B	57	46	139	87
Seashell		FF	F5	EE	255	245	238
Sienna		A0	52	2D	160	82	45
Silver		C0	C0	C0	192	192	192
SkyBlue		87	CE	EB	135	206	235
SlateBlue		6A	5A	CD	106	90	205
SlateGray		70	80	90	112	128	144
Snow		FF	FA	FA	255	250	250
SpringGreen		00	FF	7F	0	255	127
SteelBlue		46	82	B4	70	130	180
Tan		D2	B4	8C	210	180	140
Teal		00	80	80	0	128	128
Thistle		D8	BF	D8	216	191	216

Tomato	FF	63	47	255	99	71	
Turquoise	40	E0	D0	64	224	208	
Violet	EE	82	EE	238	130	238	
Wheat	F5	DE	B3	245	222	179	
White	FF	FF	FF	255	255	255	
WhiteSmoke	F5	F5	F5	245	245	245	
Yellow	FF	FF	00	255	255	0	
YellowGreen	9A	CD	32	154	205	50	

11 Farbverlauf und Muster

11.1 Farbverläufe und Muster

Farbverläufe und Muster werden in der Vollversion von SVG 1.1 als Maldienste angeboten. In SVG tiny 1.2 gibt es eine leicht vereinfachte Variante der Farbverläufe, ebenfalls als Maldienst.

Farbverläufe und Muster werden ganz allgemein von Eigenschaften wie *fill* und *stroke* als Maldienste referenziert. Dazu wird das den Dienst definierende Element mit einem Fragmentidentifizier versehen (Attribut *id* oder in SVG tiny 1.2 auch *xml:id*). Diese werden als funktionale IRI referenziert, in SVG tiny 1.2 nur eine Referenz zu einem Element im selben Dokument. In einer Vollversion gibt es diese Einschränkung auf ein lokales Element nicht.

Farbverläufe stellen eine (zumeist kontinuierliche) Änderung der Farbe über einen Flächenbereich dar, in den aktuellen Spezifikationen gibt es zwei Arten von Farbverläufen, entlang einer vorgegebenen Richtung (linearer Verlauf) oder radial von einem Punkt aus (radialer Verlauf). Möglicherweise wird es in zukünftigen Versionen von SVG weitere Möglichkeiten für Farbverläufe geben, da sich mit der Kombination diesen beiden Möglichkeiten nur sehr schlecht gezielt beliebige Farbverläufe erzielen lassen.

Bei Mustern sind in der Vollversion von SVG 1.1 periodische Flächenfüllungen oder Parkettierungen gemeint, die nur durch Translation erzielt werden. Zwar können so auf indirekte Weise alle möglichen periodischen Parkettierungen oder ebene kristallographische Gruppen dargestellt werden, jedoch meist nicht in einer Weise, welche die Symmetrieeigenschaften der jeweiligen Gruppe ausnutzt. Der Autor muss also bei der jeweiligen Parkettierung selbst im Muster eine rechteckige Einheitszelle identifizieren, aus der sich durch Translationen die gewünschte Parkettierung ergibt. Einige Darstellungsprogramme mögen dann allerdings Darstellungsfehler resultierend von Rundungen am Ausschnitt haben.

Für aperiodische Parkettierungen wie die nach Penrose stellt SVG keine spezifischen Hilfen außer den Elementen ***use*** und ***g*** zur Verfügung.

11.2 Farbverlauf

Farbverläufe werden mit den Elementen ***linearGradient*** und ***radialGradient*** angegeben. Diese sollten immer innerhalb von ***defs*** notiert werden. Erlaubt, aber nicht empfehlenswert ist es, sie direkt in Gruppierungselementen wie ***g***, ***svg*** etc zu notieren.

11.2.1 gerader Farbverlauf, *linearGradient*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Element ***linearGradient*** wird ein gerader Farbverlauf realisiert. Die Richtung wird mit vier Koordinaten festgelegt.

Die vier Koordinaten werden mit den Attributen *x1*, *y1*, *x2*, *y2* festgelegt, mit x jeweils die x-Koordinaten, mit y die y-Koordinaten, 1 steht jeweils für den Anfangspunkt, 2 für den Endpunkt, die Richtung der Farbänderung erfolgt also vom Anfangspunkt in Richtung Endpunkt. In einem Koordinatensystem, in dem x- und y-Richtung gleich skaliert sind, bleibt senkrecht zur Richtung die Farbe jeweils gleich.

Ist *x1* nicht gesetzt, wird der Wert als 0 angenommen.

Ist *y1* nicht gesetzt, wird der Wert als 0 angenommen.

Ist *x2* nicht gesetzt, wird der Wert als 1 angenommen.

Ist *y2* nicht gesetzt, wird der Wert als 0 angenommen.

Alle vier Attribute sind animierbar.

Mit dem Attribut *gradientUnits* wird angegeben, in welchem Koordinatensystem die Koordinaten angegeben werden. Der Wert 'userSpaceOnUse' gibt an, dass das lokale Koordinatensystem (Nutzerkoordinatensystem) verwendet wird, also jenes des Elementes, von dem der Farbverauf referenziert wird. Prozentangaben beziehen sich dann auf Höhe und Breite des aktuellen Anzeigebereiches.

Der Wert 'objectBoundingBox' gibt an, dass das Koordinatensystem aus der das referenzierende Element umgebenden Box resultiert, genauer das Element, auf welches der Farbverlauf angewendet wird (wenn etwa der Farbverlauf von einem ***g***-Element geerbt wird und auf ein Rechteck wirkt, so definiert das Rechteck die umgebende Box, nicht auch die anderen Kindelemente des ***g***-Elementes). Dabei ist (0,0) oder (0%,0%) links oben und (1,1) oder (100%,100%) rechts unten. Die Box ergibt sich, wenn im lokalen Koordinatensystem ein Rechteck um das Element gelegt wird, welches jeweils mit den Seiten zu den lokalen x- und y-Achsen ausgerichtet ist. Die umgebende Box ist die kleinste, welche das Element komplett umgibt. Entscheidend ist dabei die Form, nicht jedoch eventuelle Ausdehnungen des Striches *stroke*.

Ist insbesondere ein Objekt nur eindimensional (eine gerade Linie) und exakt in x-Richtung oder y-Richtung ausgerichtet, gibt es keine umgebende Box mit von 0 verschiedener Fläche und der Farbverlauf ist nicht anwendbar. Für den Fall kann jeweils eine Ersatzwert angegeben

werden, etwa so: *stroke*="url(#MeinVerlauf") red" oder *stroke*="url(#MeinVerlauf") none" oder *stroke*="url(#MeinVerlauf") currentColor".

Ist #MeinVerlauf keine Fragmentidentifizierer für einen Farbverlauf oder ist der Farbverlauf nicht anwendbar, so wird die Farbe rot angenommen (beziehungsweise 'none' oder 'currentColor'). Fehlt der Ersatzwert, wird in diesem Sinne aber benötigt, ist ein entsprechendes Dokument nach SVG 1.1 fehlerhaft und die Darstellung wird abgebrochen. In SVG tiny 1.2 wird dann stattdessen der Wert 'none' angenommen.

Ist das Attribut *gradientUnits* nicht angegeben, wird 'objectBoundingBox' angenommen. Auch dies Attribut ist animierbar.

Mit dem Attribut *gradientTransform* kann der Farbverlauf in der Vollversion von SVG 1.1 transformiert werden. Das Attribut ist animierbar mit dem Element ***animateTransform***. Die Werte sind die gleichen wie für das Attribut *transform*. Durch eine ungleichmäßige Transformation kann erreicht werden, dass die Richtung konstanter Farbe nicht mehr senkrecht zur Richtung des Farbverlaufes ist.

Mit dem Attribut *spreadMethod* wird in der Vollversion von SVG 1.1 angegeben, wie der Farbverlauf jenseits der notierten Koordinaten fortgesetzt wird, falls dort noch Teile des Elementes dargestellt werden müssen.

Mit dem Wert 'pad' wird notiert, dass für die Fortsetzung einfach die letzte Farbe am jeweiligen Ende verwendet wird.

Mit dem Wert 'reflect' wird notiert, dass für die Fortsetzung der Farbverlauf gespiegelt wird, jeweils alternatierend Start-Ende / Ende-Start /Start-Ende / Ende-Start etc.

Mit dem Wert 'repeat' wird notiert, dass für die Fortsetzung der Farbverlauf wiederholt wird, also Start-Ende / Start-Ende / Start-Ende / Start-Ende etc.

Sofern nicht angegeben, wird der Wert 'pad' angenommen.

Das Attribut ist animierbar.

Mit dem Attribut *href* von XLink kann ein anderer Farbverlauf referenziert werden, von dem die Farben des Farbverlaufes übernommen werden, sofern sie beim referenzierenden Element nicht gesetzt sind. Mit diesem Attribut sind auch die anderen Attribute von XLink bei diesem Element verfügbar.

Wird ferner ein ***linearGradient*** referenziert, so werden auch alle Attribute vom referenzierten Farbverlauf übernommen, die beim referenzierenden nicht gesetzt sind.

Diese Funktion ist in der Vollversion von SVG 1.1 verfügbar. Das Attribut ist animierbar.

Als Inhalt des Elementes ***linearGradient*** werden Elemente ***stop*** notiert, um festzulegen, wo im Farbverlauf welche Farbe auftritt.

11.2.2 radialer Farbverlauf, *radialGradient*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Element ***radialGradient*** wird ein radialer Farbverlauf realisiert.

Die Elemente *gradientUnits*, *gradientTransform*, *spreadMethod* und *href* von XLink werden sinngemäß verwendet, wie für ***linearGradient*** beschrieben.

Bei *href* werden dann entsprechend und gegebenenfalls Attribute von einem referenzierten ***radialGradient*** übernommen. *gradientUnits* bezieht sich somit auf die Attribute *cx*, *cy*, *r*, *fx*, *fy* von ***radialGradient***.

Die Attribute *cx*, *cy*, *r* beschreiben den äußeren Kreis des radialen Verlaufs. Dabei ist die Länge *r* der Radius des Kreises und (*cx*,*cy*) die Koordinaten des Mittelpunktes des Kreises.

Alle drei Attribute sind animierbar. In der Vollversion von SVG 1.1 wird für *cx*, *cy* und *r* jeweils 50% angenommen, sofern nicht angegeben. Da in SVG tiny 1.2 keine Prozentwerte möglich sind, wird dort dann entsprechend 0.5 angenommen, was bei *gradientUnits* 'objectBoundingBox' auf das gleiche hinausläuft.

Negative Werte für *r* sind nicht erlaubt und sind in SVG 1.1 ein Fehler, der zum Abbruch der Darstellung führt. In SVG tiny 1.2 führt dies nur dazu, dass der angegebene Wert ignoriert wird, woraus der Wert 0.5 folgt.

Der Kreis ist jeweils das Ende des Farbverlaufes.

Die Attribute *fx*, *fy* definieren Koordinaten für den fokalen Punkt des Farbverlaufes, von wo aus der Verlauf beginnt. Dies gilt nur, sofern der fokale Punkt innerhalb des durch *cx*, *cy* und *r* definierten Kreises liegt. Ansonsten wird eine Korrektur vorgenommen, indem die korrigierten *fx* und *fy* auf den Schnittpunkt der Linie von (*cx*, *cy*) zu (*fx*, *fy*) gelegt werden.

Falls nicht angegeben, wird für *fx* der Wert von *cx* angenommen und für *fy* entsprechend *cy*.

Die Attribute sind animierbar.

In SVG tiny 1.2 sind die Attribute *fx* und *fy* nicht verfügbar, der Anfangspunkt des Farbverlaufes liegt bei SVG tiny 1.2 also immer bei (*cx*, *cy*).

Mit r dem Radius und c = (cx,cy) dem Zentrum des Kreises, f=(fx,fy) dem fokale Punkt ergibt sich also das Folgende.

Falls r > |c - f| so gilt:

Der Radius beim Versatz v (in Anteilen von 1, also die Position einer bestimmten Farbe des Farbverlaufes) ist R = v * r.

Der Mittelpunkt liegt bei P = (1 - v) f + v c.

Sonst findet eine Korrektur statt, statt f wird f' verwendet mit

f' = c + (f - c) r / |f - c|

P = (1 - v) f' + v c.

Als Inhalt des Elementes ***radialGradient*** werden Elemente ***stop*** notiert, um festzulegen, wo im Farbverlauf welche Farbe auftritt.

11.2.3 Farbverlaufsliste, *stop*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Element ***stop*** innerhalb von ***linearGradient*** oder ***radialGradient*** wird jeweils eine Farbe als Stützpunkt für den Farbverlauf angegeben. Die Liste der Elemente ***stop*** definiert somit die im Farbverlauf vorkommenden Farben samt ihrer Position im Farbverlauf und möglicher Teiltransparenzen. Zwischen zwei Stützpunkten wird dann linear interpoliert, um die Zwischenwerte des Farbverlaufes zu erhalten.

Es sollten immer mindestens zwei ***stop***-Elemente pro Farbverlauf angegeben sein. Ist keines angegeben, entspricht dies einem Wert für *fill* beziehungsweise *stroke* von 'none'. Wird nur eines angegeben, so hat der Farbverlauf nur eine konstante Farbe und keinen Verlauf.

Mit dem Attribut *offset* (deutsch: Versatz) wird festgelegt, für welche Stelle im Farbverlauf das ***stop***-Element gelten soll. Der Wert ist eine Zahl, typisch im Bereich 0 bis 1 (in der Vollversion von SVG 1.1 kann es auch eine Prozentangabe sein, dann typisch zwischen 0% und 100%. Die Angabe des Prozentzeichens ist dann wichtig, um die Angaben von der mit Zahlen zu unterscheiden). Bei Werten außerhalb des typischen Bereiches wird auf den nächstgelegenen typischen Wert korrigiert.

Bei einem linearen Farbverlauf kennzeichnet dies die Position entlang des Farbverlaufs, also 0 der Startpunkt und 1 der Endpunkt. Beim radialen Farbverlauf entspricht 0 dem Punkt (fx,fy) und 1 dem Kreis mit Radius r um (cx,cy), siehe auch obige Formeln, mit v als Wert für *offset*.

In SVG 1.1 ist die Angabe von *offset* erforderlich, sonst resultiert daraus ein Fehler, der zum Abbruch der Darstellung führt. In SVG tiny 1.2 wird in dem Falle lediglich der Wert 0 angenommen.

offset ist animierbar.

Bei aufeinanderfolgenden ***stop***-Elementen sind die Werte von *offset* so zu wählen, dass der spätere Wert nicht kleiner als der frühere ist. In der Abfolge im Quelltext müssen die Werte also monoton steigend sein. Trifft dies nicht zu, wird der zu kleine Wert jeweils auf den größten vorhergehenden korrigiert.

Wenn zwei aufeinanderfolgende ***stop***-Elemente den gleichen Wert für *offset* haben, aber andere Farbangaben oder Opazitäten, so ergibt sich ein diskreter Sprung im Verlauf. Der Farbverlauf erfolgt also zunächst bis zu dem ***stop***, welches von den beiden gleichen zuerst notiert ist, springt dann zum zweiten mit gleichem *offset* und setzt sich dann kontinuierlich fort zum nächsten.

Mit der Eigenschaft *stop-color* wird die Farbe für den ***stop*** angegeben.

Mögliche Werte sind:

inherit

geerbt

currentColor

aktuelle Farbe, mit color angegeben

Farbwert

Farbwert wie für *fill* oder *stroke* beschrieben

Die Eigenschaft wird nicht vererbt und ist animierbar, Voreinstellung ist black.

Mit der Eigenschaft *stop-opacity* wird die Opazität oder Teiltransparenz für den ***stop*** angegeben.

Mögliche Werte sind entweder 'inherit' (geerbt also) oder eine Zahl, wobei Zahlen außerhalb des Bereiches 0 bis 1 vor der Darstellung auf den nächstgelegenen korrekten Wert korrigiert werden.

Die Eigenschaft wird nicht vererbt und ist animierbar, Voreinstellung ist 1 also undurchsichtig.

Farbverlauf W3C Referenz[1]

1 `http://www.w3.org/TR/SVG11/pservers.html#Gradients`

11.2.4 Beispiele Farbverlauf

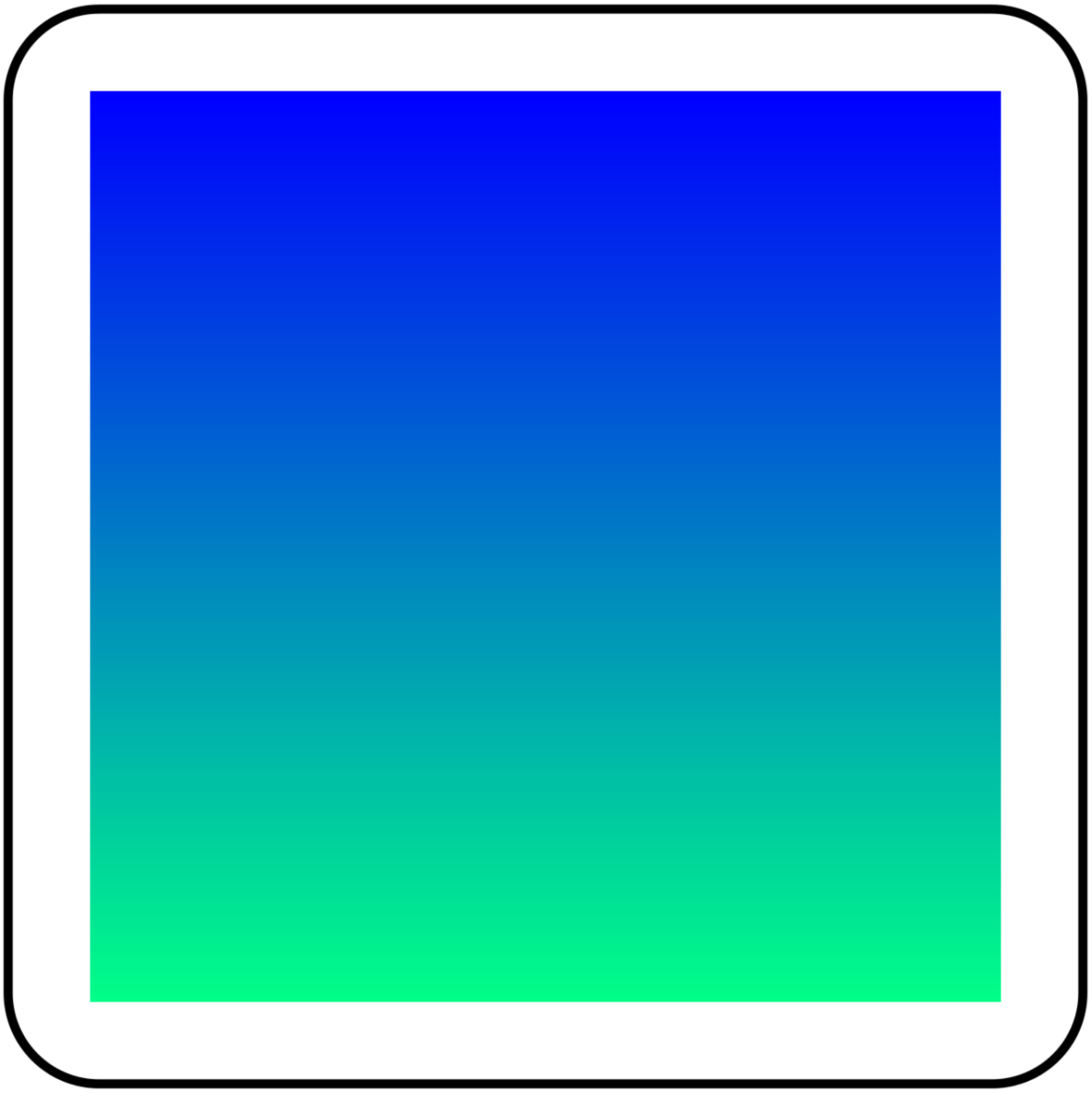


Abb. 158 linearGradient

Einfaches Beispiel linearer Farbverlauf:

```
<defs>
        <linearGradient id="lg" x1="0" y1="0" x2="0" y2="1">
                <stop stop-color="#00f" offset="0" />
                <stop stop-color="#0f8" offset="1" />
        </linearGradient>
</defs>

<rect fill="url(#lg)" width="100" height="100" x="10" y="10" />
```

gradientUnits werden hier impliziert, also als 'objectBoundingBox' angegeben. An sich sind auch die Attribute *x1*, *y1* und *y2* redundant, weil auch deren Werte mit den Voreinstellungen übereinstimmen.

Der Farbverlauf erfolgt bedingt durch das *x2*="0" vertikal. Da bei den beiden ***stop***-Elementen die Eigenschaft *stop-opacity* nicht angegeben ist, ist der Verlauf undurchsichtig.

Bei diesem Beispiel wird auch die Eigenschaft *stop-opacity* verwendet. Durch die Farbverläufe ergibt sich so eine Maskierung des äußeren Randes des Hintergrundes: Farbverlauf als Maskierung[2]

Zwei Rechtecke referenzieren jeweils einen linearen Farbverlauf, einer horizontal, einer vertikal.

Diese verdecken wegen ihrer Teiltransparenz außen teilweise den Hintergrund, bestehend aus konzentrischen roten und blauen Kreisen.

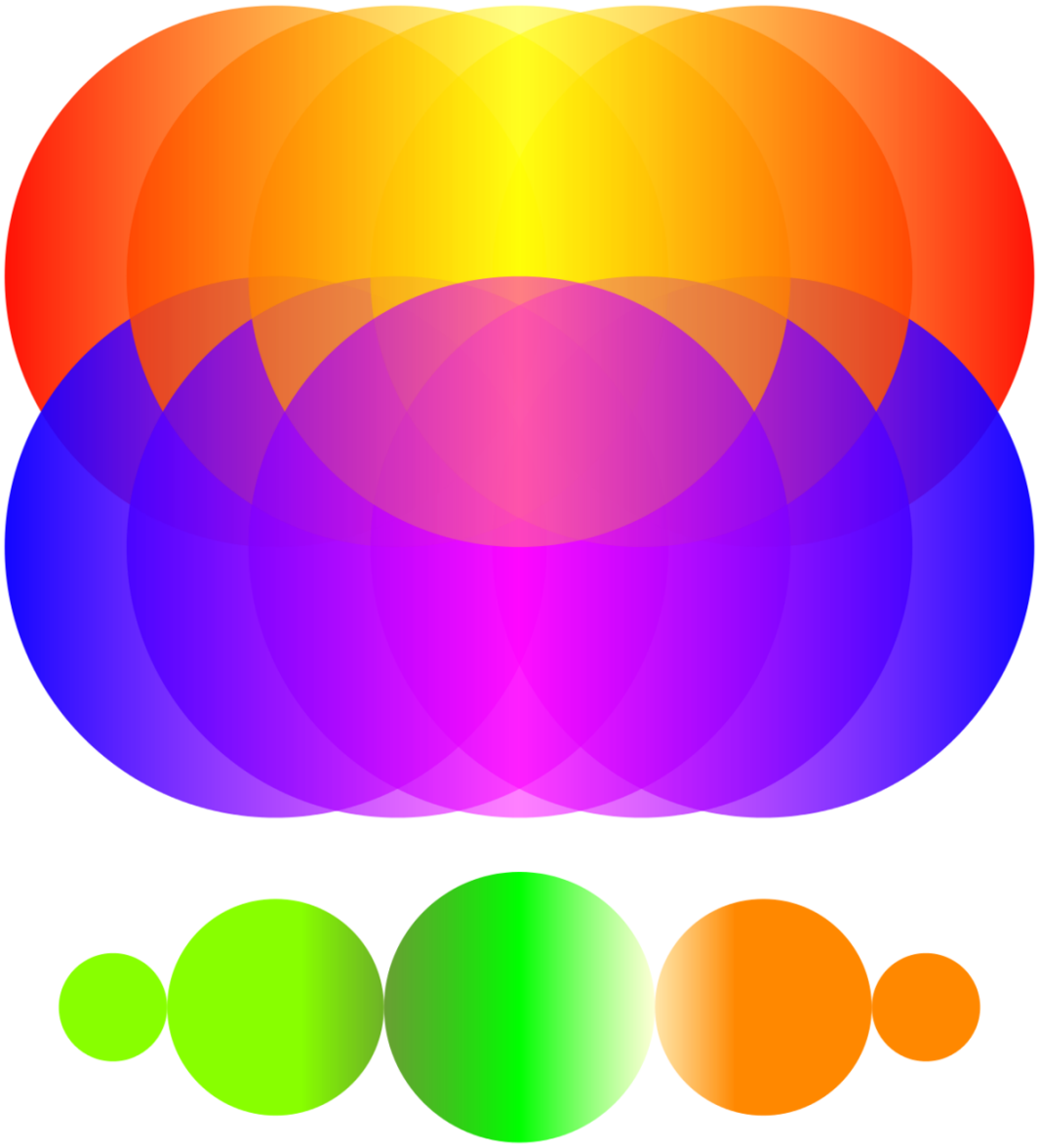


Abb. 159 Beispiel zu gradientUnits userSpaceOnUse

2 `http://de.wikibooks.org/wiki/media%3ASVGlinearGradient01.svg`

Beispiel zu gradientUnits userSpaceOnUse[3]

gradientUnits userSpaceOnUse eignet sich unter anderem gut, wenn ein Verlauf auf mehrere Elemente angewendet werden soll, die nicht zu einer Gruppe zusammengefasst werden können.

Dies wird hier ausgenutzt, indem jeweils auf die oberen und die unteren Kreise ein Verlauf angewendet wird, wobei im Quelltext obere und untere Kreise abwechselnd angeordnet sind. So können die Kreise überlappen und es ist nicht notwendig, für jeden Kreis einen eigenen Verlauf zu definieren.

Auch Transformationen haben keinen Einfluss auf den beim Elternelement spezifizierten Verlauf, was an den Kreisen ganz unten zu sehen ist.

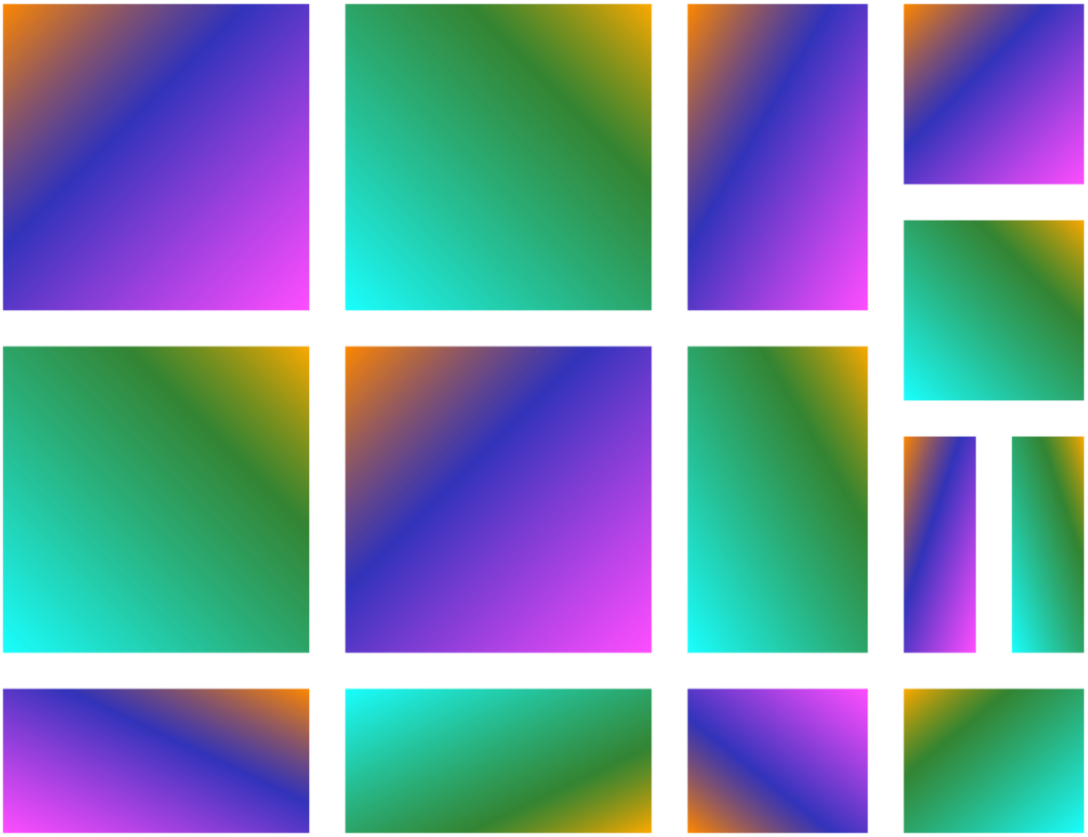


Abb. 160 Beispiel zu gradientUnits objectBoundingBox

Beispiel zu gradientUnits objectBoundingBox[4]

Bei gradientUnits objectBoundingBox bestimmt jeweils das Element, auf welches der Farbverlauf angewendet wird das verwendete Koordinatensystem. Das eignet sich gut, wenn der gleiche Farbverlauf so verwendet werden soll, dass er bei verschiedenen Elementen an verschiedenen Stellen 'gleich'

3 http://de.wikibooks.org/wiki/media%3ASVGlinearGradient02.svg
4 http://de.wikibooks.org/wiki/media%3ASVGlinearGradient03.svg

aussehen soll. Dabei ist 'gleich' relativ zu sehen, wenn die verschiedenen Elemente unterschiedliche Größen und Aspektverhältnisse haben oder transformiert werden.

Das zeigen hier einige Beispiele, bei denen zwei verschiedene Farbverläufe bei verschiedenen Rechtecken zum Einsatz kommen.

Bei der einen Gruppe ist zudem der Farbverlauf direkt in der Elementanfangsmarkierung referenziert, bei der anderen wird der Farbverlauf vom Elternelement geerbt.

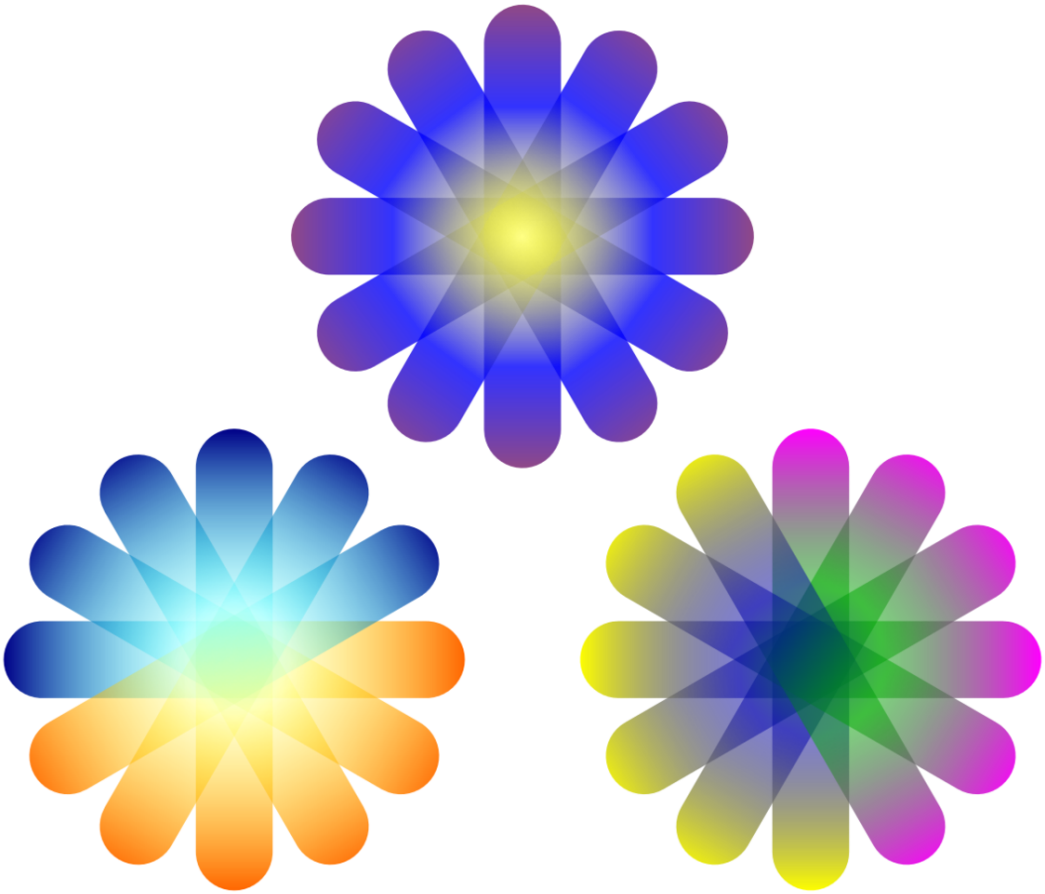


Abb. 161 Beispiel zur Anwendbarkeit von Farbverläufen bei gradientUnits objectBoundingBox

Beispiel zur Anwendbarkeit von Farbverläufen bei gradientUnits objectBoundingBox[5]

Bei eindimensionalen graphischen Objekten, die entlang der lokalen x- oder y-Richtung ausgerichtet sind, ist die Anwendung eines Farbverlaufes mit gradientUnits objectBoundingBox nicht möglich. Dies trifft zu für die Linien links und rechts, wo dann jeweils die angegebene Ersatzfarbe rot verwendet werden muss. Oben hingegen ist die Linie gedreht, spannt somit eine boundingBox auf, deren Fläche von 0 verschieden ist, somit ist dort gradientUnits objectBoundingBox und damit auch der Farbverlauf anwendbar.

5 `http://de.wikibooks.org/wiki/media%3ASVGlinearGradient04.svg`

gradientUnits ist hier nicht explizit gesetzt und ist somit als objectBoundingBox anzunehmen.

Zeigt ein Darstellungsprogramm links und rechts statt der einfachen roten Füllung auch einen (an sich attraktiven) Farbverlauf an (z.B. das adobe plugin oder Opera 10), so ist dies ein Fehler des Darstellungsprogrammes.

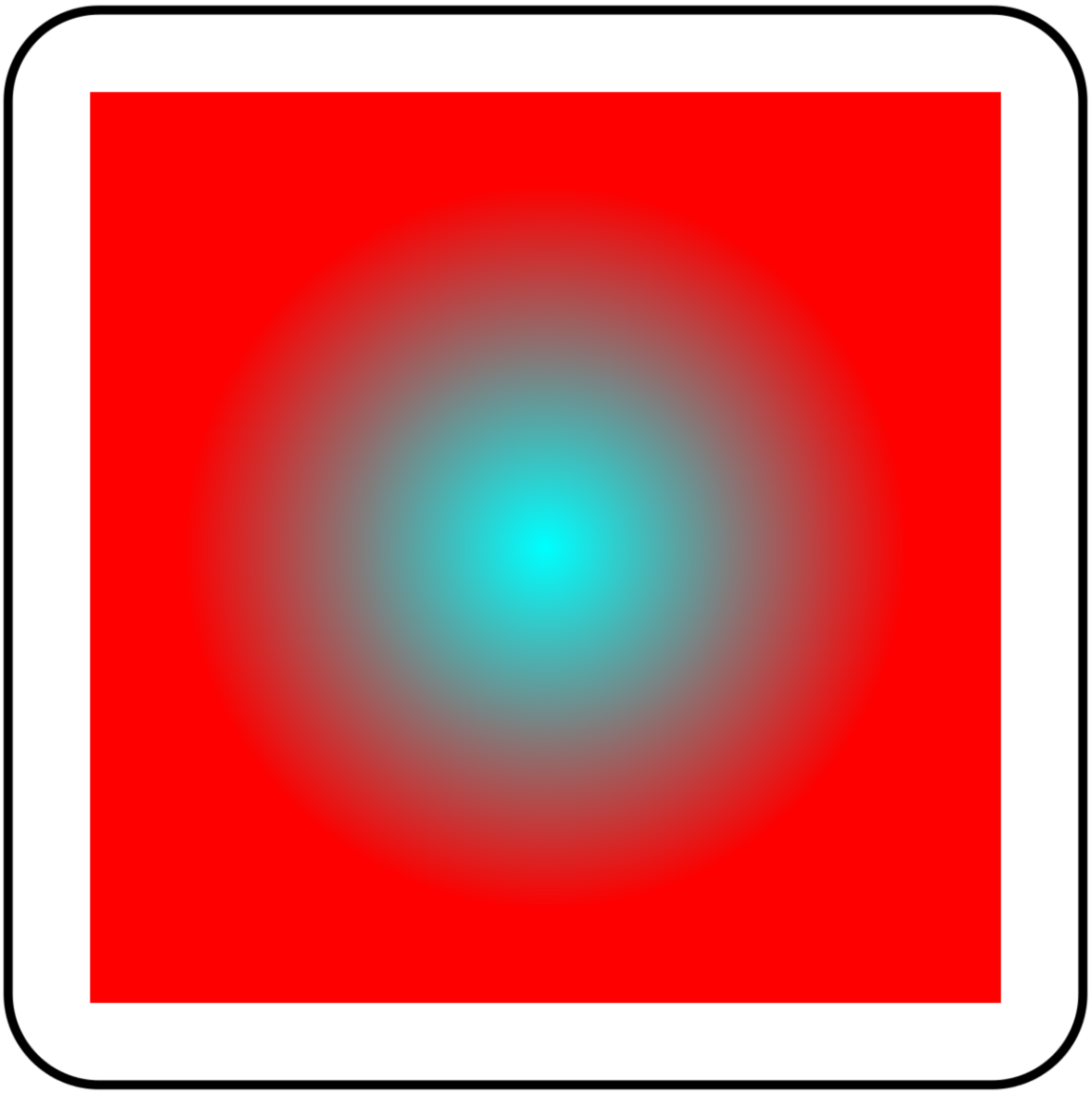


Abb. 162 radialGradient

Einfaches Beispiel radialer Farbverlauf:

```
<defs>
        <radialGradient id="rg" cx="0.5" cy="0.5" r="0.4">
                <stop stop-color="#0FF" offset="0" />
                <stop stop-color="#F00" offset="1" />
        </radialGradient>
</defs>

<rect fill="url(#rg)" width="100" height="100" x="10" y="10" />
```

Auch hier wird *gradentUnits* impliziert. Weil keine ungleichmäßigen Transformationen vorliegen, ergibt sich ein einfacher konzentrischer Farbverlauf.

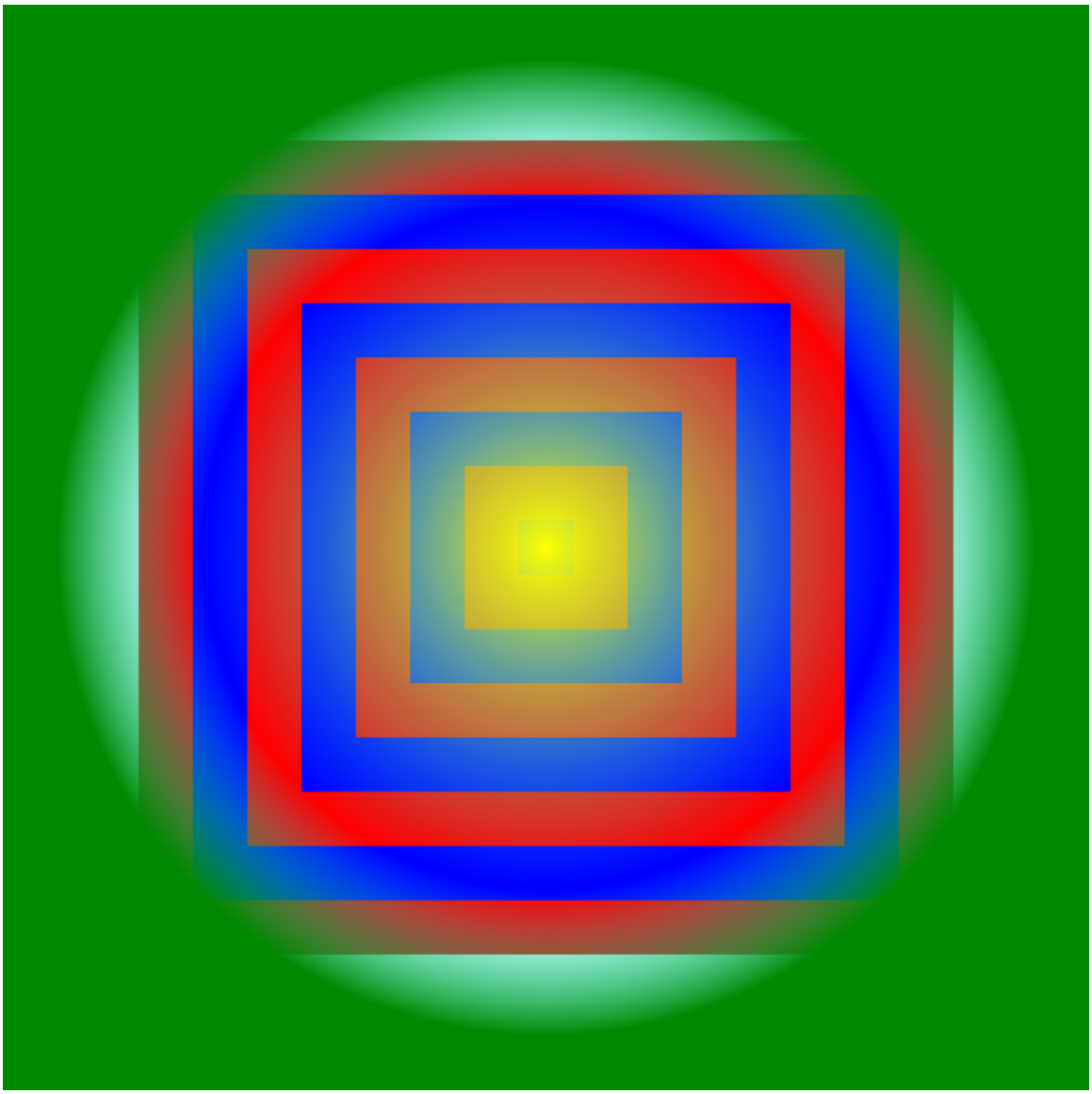


Abb. 163 Beispiel mit teiltransparentem radialem Farbverlauf

Beispiel mit teiltransparentem radialem Farbverlauf[6]

Ein Rechteck referenziert einen radialen Farbverlauf, welcher innen und außen komplett undurchsichtig ist und dazwischen an einer Stelle komplett durchsichtig.

Somit wird der Hintergrund teilweise verdeckt, welcher aus konzentrischen roten und blauen Quadraten besteht.

6 `http://de.wikibooks.org/wiki/media%3ASVGradialGradient01.svg`

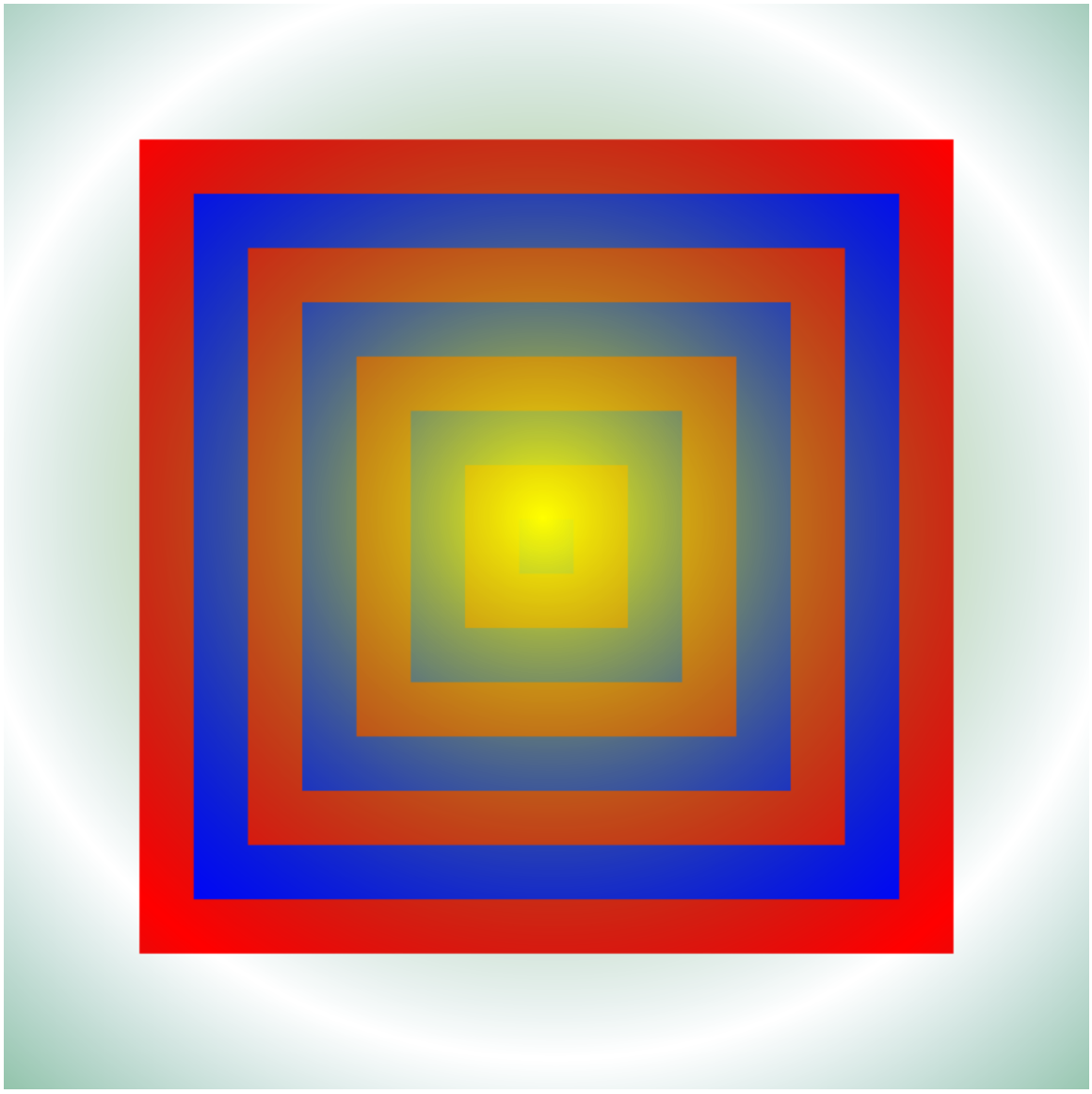


Abb. 164 Beispiel radialer Verlauf mit *gradientUnits* objectBoundingBox und Verzerrung

Beispiel radialer Verlauf mit *gradientUnits* objectBoundingBox und Verzerrung[7]

Ein Rechteck referenziert einen radialen Farbverlauf, welcher innen und außen komplett undurchsichtig ist und dazwischen an einer Stelle komplett durchsichtig.

Somit wird der Hintergrund teilweise verdeckt, welcher aus konzentrischen roten und blauen Quadraten besteht.

Da *gradientUnits* objectBoundingBox verwendet werden, das referenzierende Rechteck aber kein Quadrat ist, ist das Ergebnis eine Ellipse. Weil das referenzierende Element ferner gedreht ist, ist somit auch der Farbverlauf relativ zum Hintergrund gedreht.

7 `http://de.wikibooks.org/wiki/media%3ASVGradialGradient02.svg`

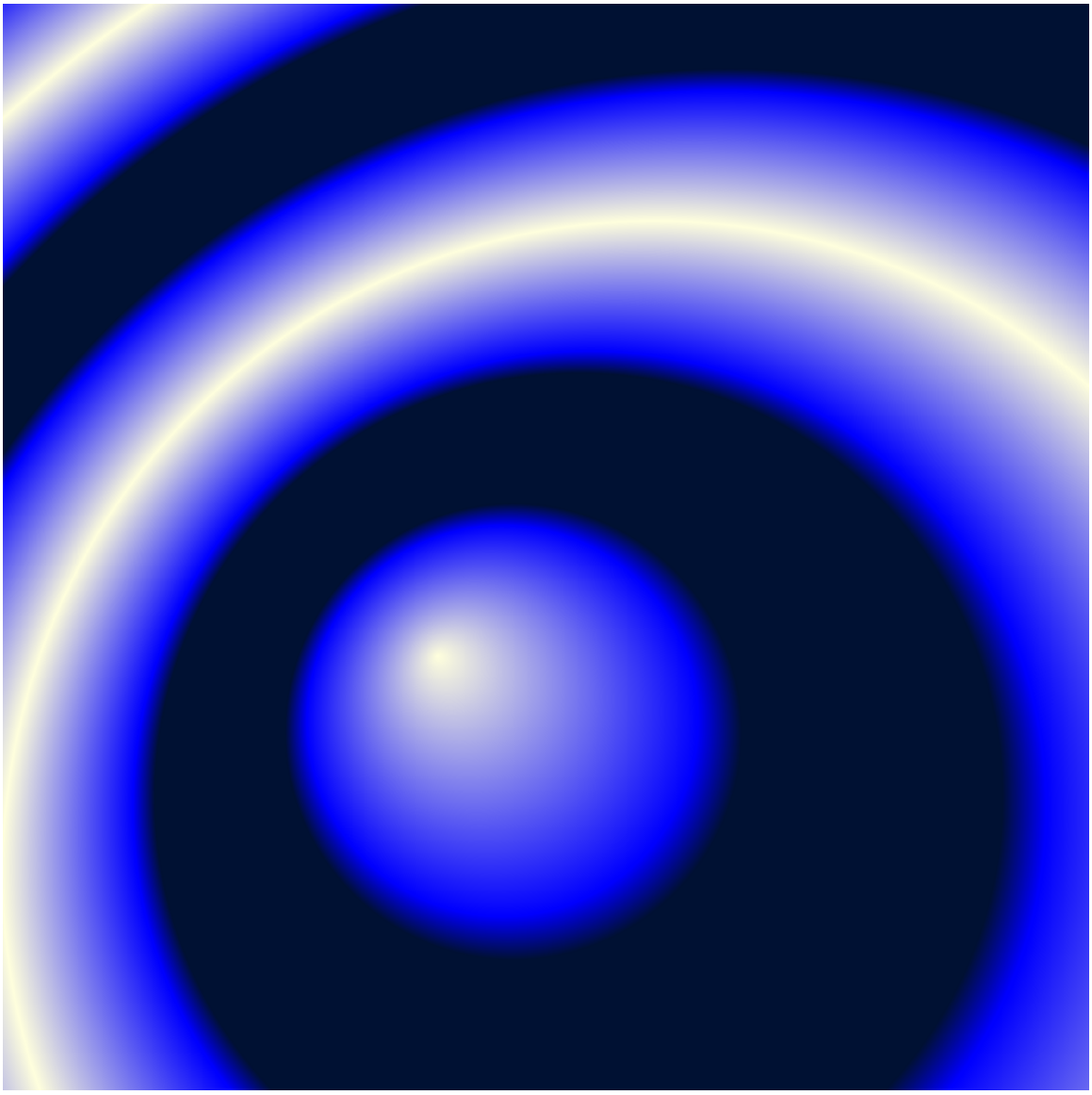


Abb. 165 Beispiel mit fokalem Punkt

Beispiel mit fokalem Punkt[8]

Ein radialer Farbverlauf erweckt den Eindruck eines Beleuchtungseffektes.

8 http://de.wikibooks.org/wiki/media%3ASVGradialGradient03.svg

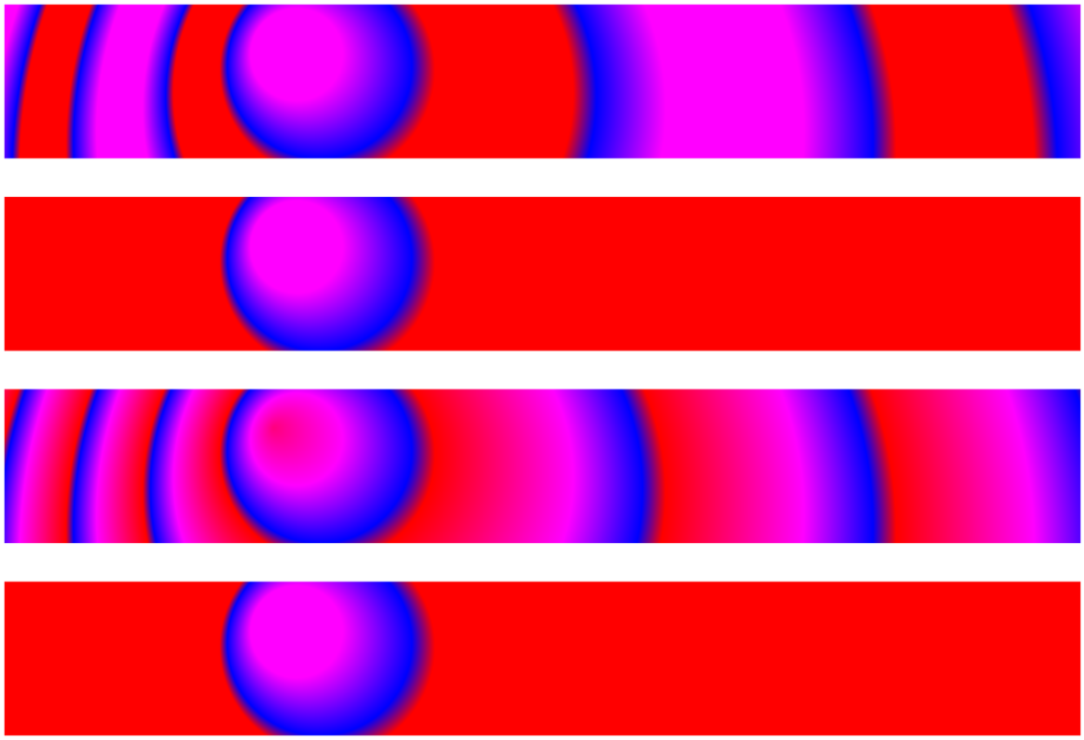


Abb. 166 Beispiel zur Verwendung des Attributes *spreadMethod*

Beispiel zur Verwendung des Attributes *spreadMethod*[9]

Beispiel zur Verwendung des Attributes *spreadMethod* mit den Werten reflect, pad, repeat von oben nach unten und ganz unten ein Beispiel ohne explizite Angabe, was pad entspricht.

9 http://de.wikibooks.org/wiki/media%3ASVGradialGradient04.svg

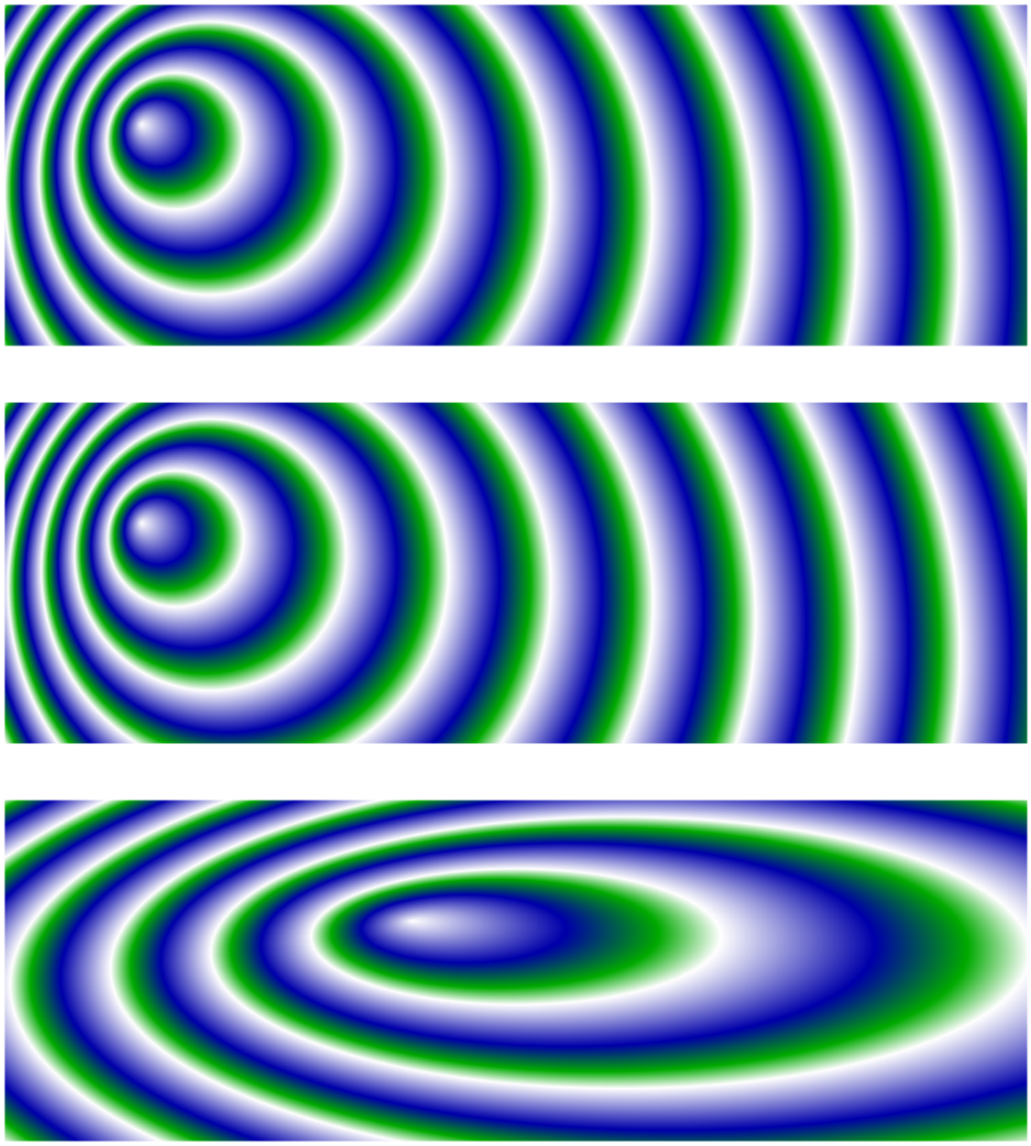


Abb. 167 Beispiel zu *gradientTransform* und Vermeidung von Verzerrungen bei *gradientUnits* objectBoundingBox

Beispiel zu *gradientTransform* und Vermeidung von Verzerrungen bei *gradientUnits* objectBoundingBox[10]

Bei *gradientUnits* objectBoundingBox und einem nicht quadratischen Anzeigebereich kann *gradientTransform* verwendet werden, um die Verzerrung rückgängig zu machen. Dies passiert hier beim mittleren. Zum Vergleich ist oben ein unverzerrter Verlauf mit *gradientUnits* userSpaceO-

10 `http://de.wikibooks.org/wiki/media%3ASVGradialGradient05.svg`

nUse angegeben und unten ein verzerrter, weil der Anzeigebereich nicht quadratisch ist und kein *gradientTransform* verwendet wird.

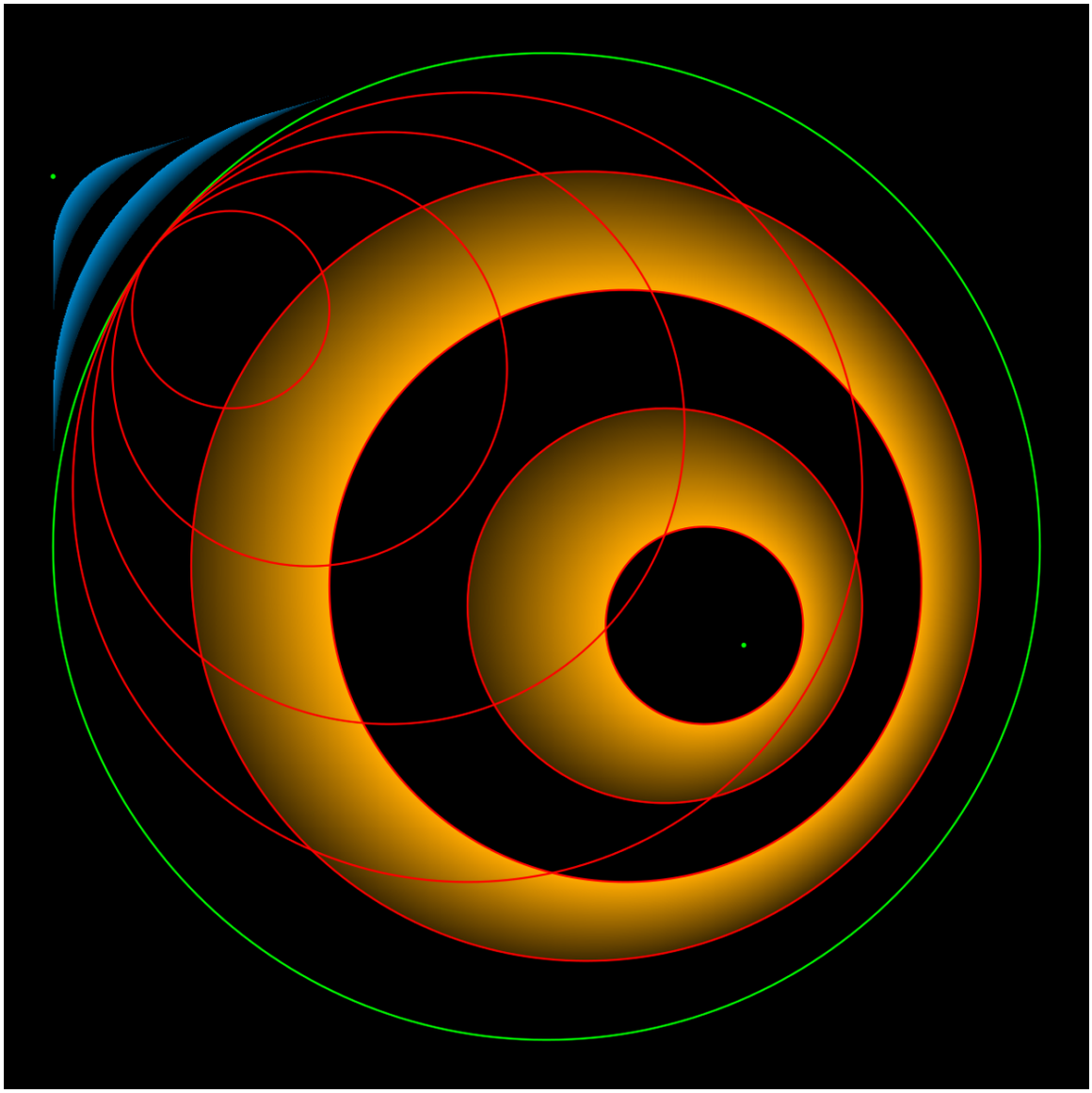


Abb. 168 Test fokaler Punkt mit und ohne Korrektur

Test fokaler Punkt mit und ohne Korrektur[11]

Illustration der Positionierung der *offset*s bei radialem Farbverlauf, einmal mit fokalem Punkt im Kreis und einmal außerhalb mit Korrektur.

Die Ränder zwischen sichtbaren *offset*s und nicht sichtbaren *offset*s sind rot gekennzeichnet, großer Radius und fokale Punkte grün.

11 `http://de.wikibooks.org/wiki/media%3ASVGradialGradient06.svg`

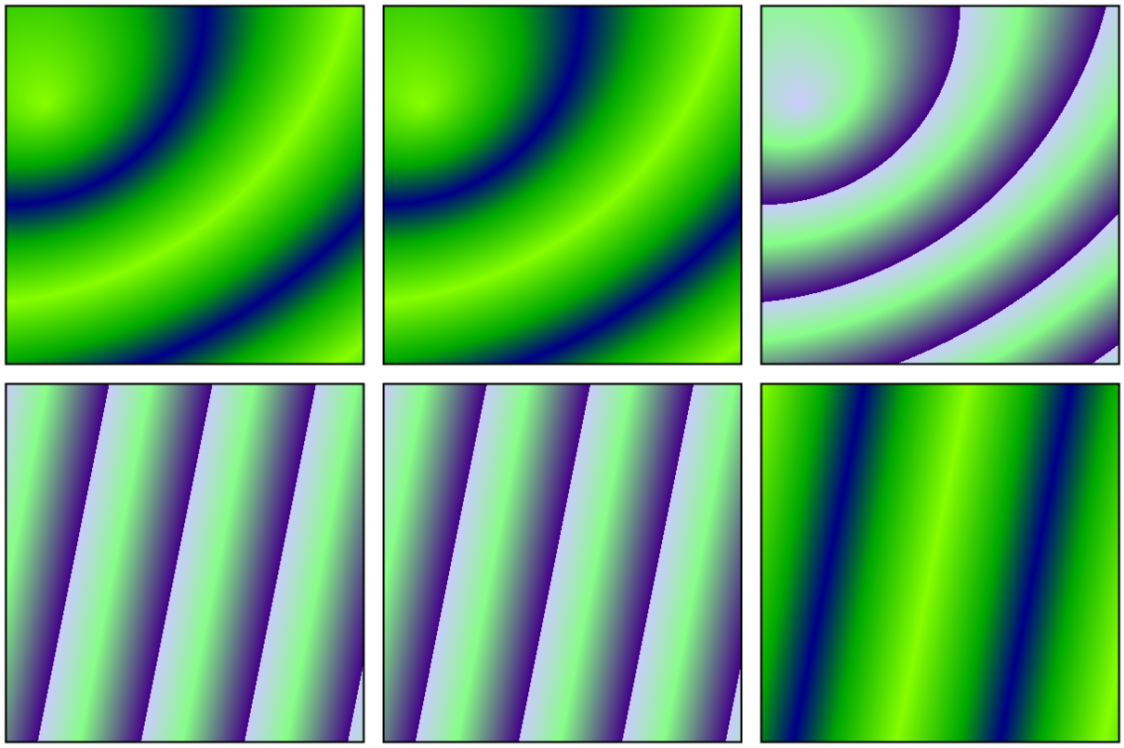


Abb. 169 Wiederverwendung von Farbverläufen

Wiederverwendung von Farbverläufen[12]

Links sind ein radialer und ein linearer Farbverlauf zu sehen. In der Mitte sind diese jeweils mit anderen Farbverläufen referenziert. Rechts sind bei der Referenzierung der lineare und der radiale Farbverlauf vertauscht, also auch die Farben und die ebenfalls angegebene spreadMethod.

12 `http://de.wikibooks.org/wiki/media%3ASVGgradient01.svg`

Abb. 170 Mehrfarbiger linearer Farbverlauf

Mehrfarbiger linearer Farbverlauf[13]

Mehrere Rechtecke mit teiltransparenten Farbverläufen werden übereinandergelegt, um einen mehrfarbigen linearen Farbverlauf zu erreichen.

13 http://de.wikibooks.org/wiki/media%3ASVGgradient02.svg

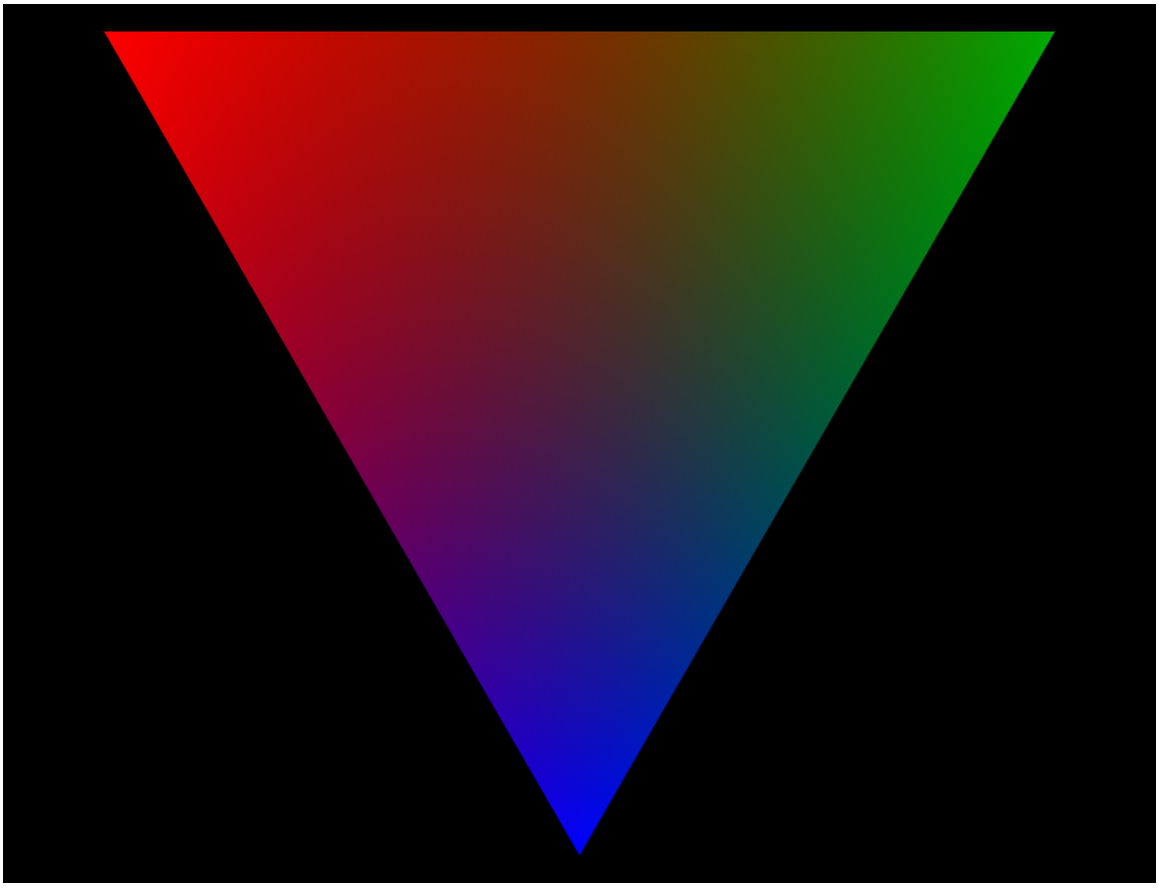


Abb. 171 Farbverlaufsdreieck

Farbverlaufsdreieck[14]

Mehrere Dreiecke mit teiltransparenten Farbverläufen werden übereinandergelegt, um einen mehrfarbigen linearen Farbverlauf zu erreichen. Mit dieser Methode ist es leicht möglich, einen Farbverlauf über eine größere Fläche zu erstellen, wobei die Farben nur an Stützstellen gegeben sind, die auf einem Gitter aus solch gleichseitigen Dreiecken liegen. Es ergibt sich dann jeweils eine lineare Interpolation zwischen den Stützpunkten. Es gibt allerdings mehrere Möglichkeiten für den Farbverlauf innerhalb des Dreieckes, die als Verlauf sinnvoll sind. Ein voll transparenter weißer Endwert führt zum Beispiel eher zu einer pastellartigen Mitte, ein schwarzer zu einer dunkleren Mitte.

Der jeweilige Verlauf selbst führt jeweils von einer Ecke des gleichseitigen Dreiecks zur Mitte der gegenüberliegenden Seite. Die Ecke ist jeweils komplett undurchsichtig und die Seite jeweils komplett durchsichtig. Bei einer Verallgemeinerung eines nicht gleichseitigen Dreiecks wäre der Farbverlauf dann so auszurichten, dass er senkrecht zur Seite verläuft und auf der Höhe der Ecke beginnt (oder endet).

14 `http://de.wikibooks.org/wiki/media%3ASVGgradient03.svg`

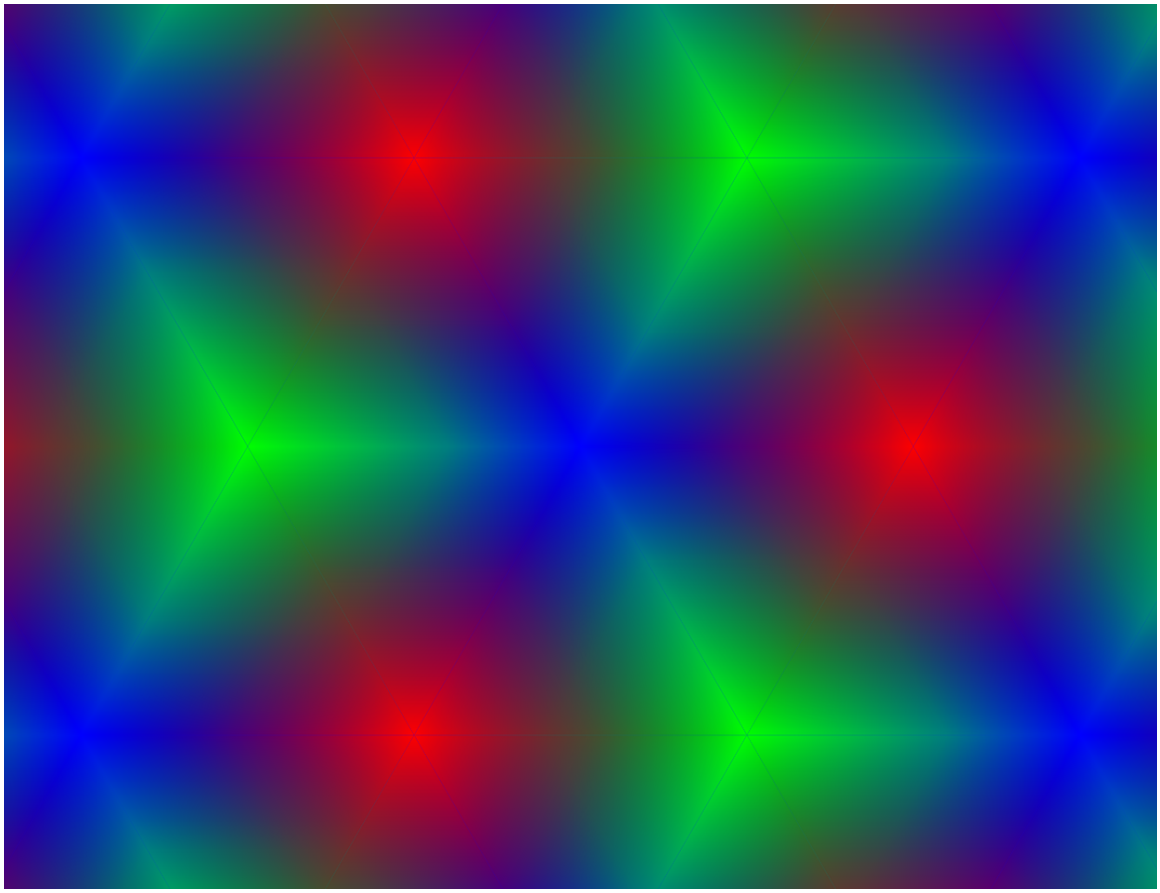


Abb. 172 Kombination linearer Farbverläufe

Kombination linearer Farbverläufe[15]

Durch Variation des vorherigen Dreiecks und Duplikation kann eine Fläche gefüllt werden.

Einmal abgesehen von den üblichen Rundungsfehlern der Notation und von Darstellungsprogrammen, lässt sich so eine Ebene periodisch oder nicht periodisch mit Farbverläufen abdecken, hier gezeigt mit einer angedeuteten periodischen Abdeckung, welche sich dann allerdings etwas besser mit dem Element ***pattern*** erreichen lässt als nur mit dem Element ***use*** wie hier.

Zu sehen sind so oder so charakteristische Strahlen entlang der Achsen der Dreiecke, ich auch ohne Rundungsprobleme bleiben und darauf hinweisen, dass die Interpolationsmethode nicht besonders gut ist und zudem auch nur wenige Stützstellen verwendet werden.

15 `http://de.wikibooks.org/wiki/media%3ASVGgradient08.svg`

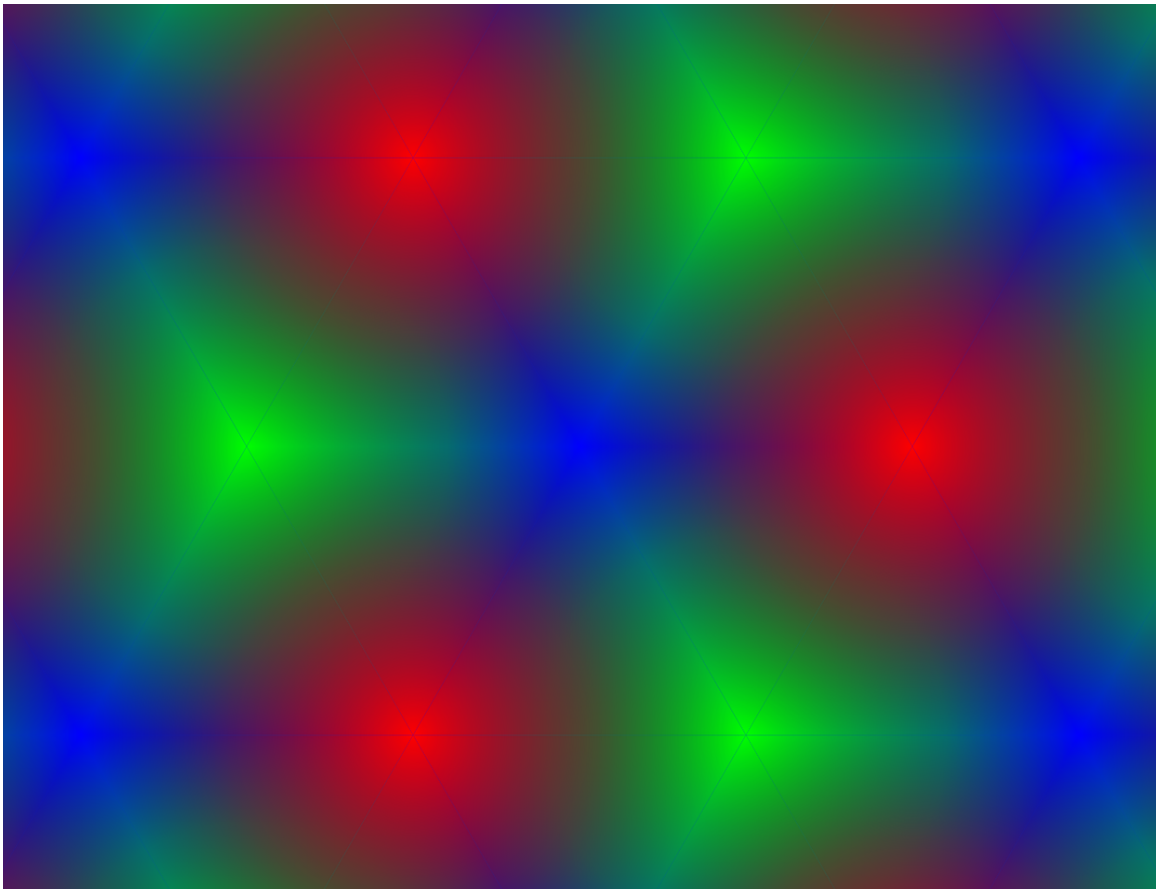


Abb. 173 Kombination radialer Farbverläufe

Dies ändert sich auch nicht signifikant durch Kombination radialer Farbverläufe[16]

Die Struktur ist bis auf die Farbverläufe die gleiche wie beim vorherigen Beispiel. Der jeweilige Verlauf führt jeweils von einer Ecke des gleichseitigen Dreiecks als Zentrum des raldialen Veraufs bis zu den beiden anderen Ecken als Abstand für den Radius.

16 `http://de.wikibooks.org/wiki/media%3ASVGgradient07.svg`

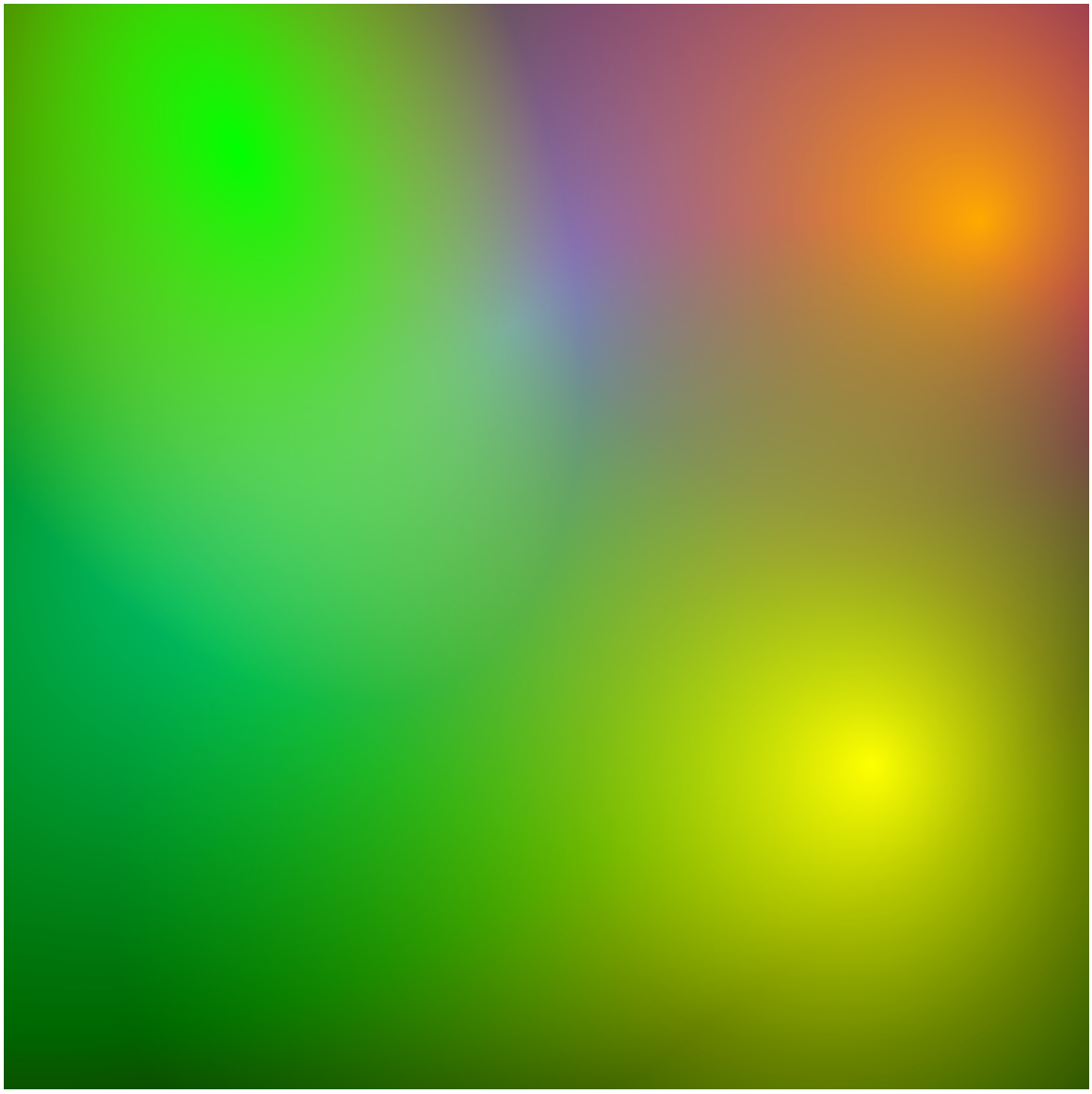


Abb. 174 Ungleichmäßige Kombination radialer Farbverläufe

Ungleichmäßige Kombination radialer Farbverläufe[17]

Durch ungleichmäßige Kombination radialer Farbverlauf kann ebenfalls ein komplizierterer Verlauf erzeugt werden. Es ist allerdings nicht unbedingt trival, für einen beliebigen gewünschten Verlauf die optimale Kombination von Einzelverläufen zu berechnen. Prinzipiell ist dies allerdings durch geschickte Kombination diverser radialer und linearer Verläufe möglich.

17 `http://de.wikibooks.org/wiki/media%3ASVGgradient04.svg`

Abb. 175 Superposition ebener Wellen als Farbverlauf

Superposition ebener Wellen als Farbverlauf[18]

Eine Überlagerung von zwei ebenen Wellen, in der Art kann zum Beispiel Interferenz bei Wasserwellen beobachtet werden.

18 http://de.wikibooks.org/wiki/media%3ASVGgradient05.svg

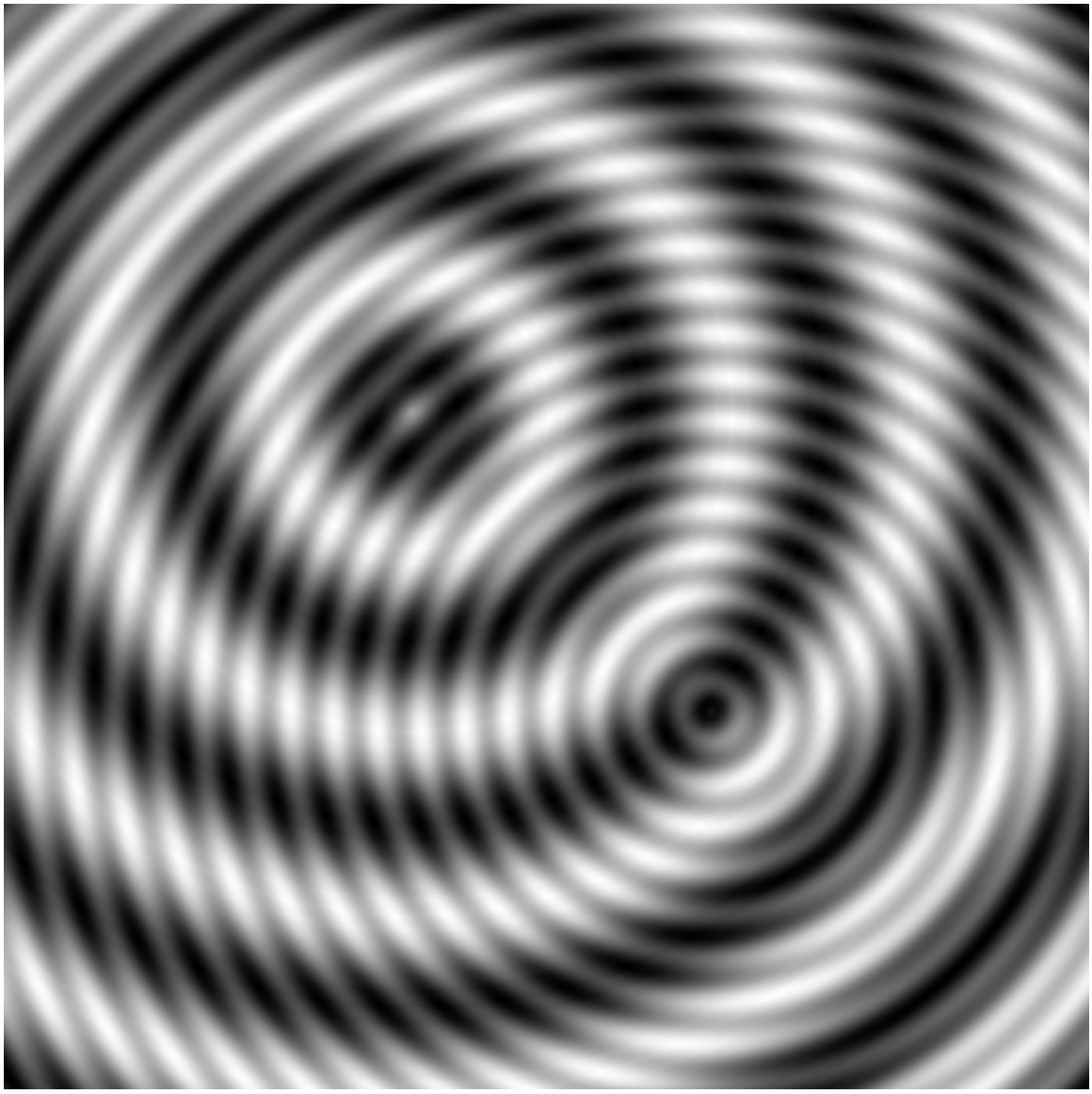


Abb. 176 radiale Wellen als Farbverlauf

radiale Wellen als Farbverlauf[19]

Superposition zweier radialer Wellen. Eine läuft ein, eine aus.

19 http://de.wikibooks.org/wiki/media%3ASVGgradient06.svg

11.3 Muster, *pattern*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5
Konqueror (KSVG)	3.2
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Element ***pattern*** wird ein Muster bereitgestellt, welches dann in der Vollversion von SVG 1.1 als Maldienst verwendet werden kann.

Wie bei Farbverläufen wird empfohlen, ***pattern*** innerhalb von ***defs*** zu notieren. Andere Gruppierungselemente als Elternelemente sind aber auch möglich.

Das Element öffnet ein eigenes Koordinatensystem, der mögliche Inhalt ist der gleiche wie für andere Gruppierungselemente, ähnlich wie bei ***defs*** wird der Inhalt aber nicht direkt dargestellt. ***pattern*** ist immer als Maldienst über einen Fragmentidentifizierer zu referenzieren.

Eigenschaften werden vom Elternelement von ***pattern*** aus geerbt, nicht vom referenzierenden Element aus.

Die Eigenschaft *overflow* ist für ***pattern*** per Voreinstellung 'hidden'. Im Bedarfsfalle muss die Eigenschaft explizit gesetzt werden, wenn dies geändert werden soll. Allerdings ist für einen anderen Wert das Verhalten weder spezifiziert noch einheitlich noch typischer Weise sinnvoll implementiert, was größere Probleme mit sich bringt, wenn komplizierterer Inhalt von ***pattern*** eine andere Symmetrie hat als ein an den x- und y-Richtungen ausgerichtetes Rechteck.

Mit den Attributen *x* und *y* und den Längen *width* und *height* werden Position (links oben) und Größe (Höhe und Breite) des Musters angegeben. Alle Attribute sind animierbar.

Die Attribute werden in *patternUnits* angegeben und *patternTransform* wirkt auf das so festgelegte Koordinatensystem.

Mit dem Attribut *patternUnits* wird also angegeben, in welchem Koordinatensystem diese Koordinaten angegeben werden. Der Wert 'userSpaceOnUse' gibt an, dass das lokale Koordinatensystem (Nutzerkoordinatensystem) verwendet wird, also jenes des Elementes, von dem das Muster referenziert wird. Prozentangaben beziehen sich dann auf Höhe und Breite des aktuellen Anzeigebereiches.

Der Wert 'objectBoundingBox' gibt an, dass das Koordinatensystem aus der das referenzierende Element umgebenden Box resultiert, genauer das Element, auf welches der Farbverlauf angewendet wird (wenn etwa das Muster von einem ***g***-Element geerbt wird und auf ein Rechteck wirkt, so definiert das Rechteck die umgebende Box, nicht auch die anderen Kindelemente des ***g***-Elementes). Dabei ist (0,0) oder (0%,0%) links oben und (1,1) oder (100%,100%) rechts unten. Die Box ergibt sich, wenn im lokalen Koordinatensystem ein Rechteck um das Element gelegt wird, welches jeweils mit den Seiten zu den lokalen x- und y-Achsen ausgerichtet ist. Die umgebende Box ist die kleinste, welche das Element komplett umgibt, entscheidend ist dabei die Form, nicht jedoch eventuelle Ausdehnungen des Striches *stroke*.

Ist insbesondere ein Objekt nur eindimensional (eine gerade Linie) und exakt in x-Richtung oder y-Richtung ausgerichtet, gibt es keine umgebende Box mit von 0 verschiedener Fläche und das Muster ist nicht anwendbar.

Ist das Attribut *patternUnits* nicht angegeben, wird 'objectBoundingBox' angenommen. Auch dies Attribut ist animierbar.

Mit dem Attribut *patternContentUnits* wird das Koordinatensystem für den Inhalt von ***pattern*** angegeben. Werte und Bedeutung sind entsprechend denen von *patternUnits*, also 'userSpaceOnUse' für das lokale Koordinatensystem, 'objectBoundingBox' für die umgebende Box. Das Attribut hat keinen Effekt, wenn das Attribut *viewBox* angegeben ist, welches dann entsprechend das Koordinatensystem impliziert.

Wenn keines der beiden Attribute angegeben ist, gilt 'userSpaceOnUse'.

viewBox enthält wie bereits für andere Elemente beschrieben eine Liste von vier Zahlen, minimales x, minimales y, Breite und Höhe.

Sowohl *patternContentUnits* als auch *viewBox* ist animierbar.

Mit dem Attribut *patternTransform* kann das Muster transformiert werden. Das Attribut ist animierbar mit dem Element ***animateTransform***. Die Werte sind die gleichen wie für das Attribut *transform*.

Mit dem Attribut *href* von XLink kann ein anderes Muster im gleichen Dokument referenziert werden. Mit diesem Attribut sind auch die anderen Attribute von XLink bei diesem Element verfügbar. Es werden alle Attribute vom referenzierten Muster übernommen, die beim referenzierenden nicht gesetzt sind. Gleiches gilt für die Kindelemente, falls das referenzierende Element selbst keine hat.

Das Attribut ist animierbar.

Mit dem Attribut *preserveAspectRatio* wird festgelegt, wie die viewBox in den Anzeigebereich eingepaßt wird, ob zum Beispiel das Aspektverhältnis erhalten bleibt. Voreinstellung ist der Wert 'xMidYMid meet', das Attribut ist animierbar und hat ansonsten die gleichen Werte wie bereits für die Verwendung bei anderen Elementen beschrieben.

W3C pattern Referenz[20]

20 `http://www.w3.org/TR/SVG/pservers.html#PatternElement`

11.3.1 Muster - Beispiele

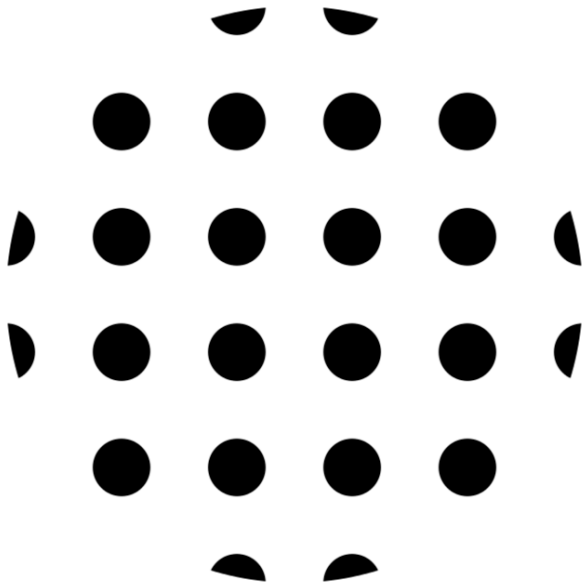


Abb. 177 Kreis mit Muster

Die Attribute und die Verwendung von Mustern ähnlich technisch stark denen der Farbverläufe.

Zunächst erst einmal wieder ein einfaches Beispiel:

```
<defs>
<pattern id="test" width="20" height="20"  patternUnits="userSpaceOnUse">
<circle cx="10" cy="10" r="5"/>
</pattern>
</defs>
<circle cx="100" cy="100" r="50" fill="url(#test) none"/>
```

Der Inhalt von ***pattern*** ist ein einfacher Kreis, damit definiert ist, worauf sich die Längen und Koordinaten des Kreises beziehen, öffnet ***pattern*** einen eigenen Anzeigebereich und hat eine definierte Größe, was von den jeweiligen Attributwerten abhängt.

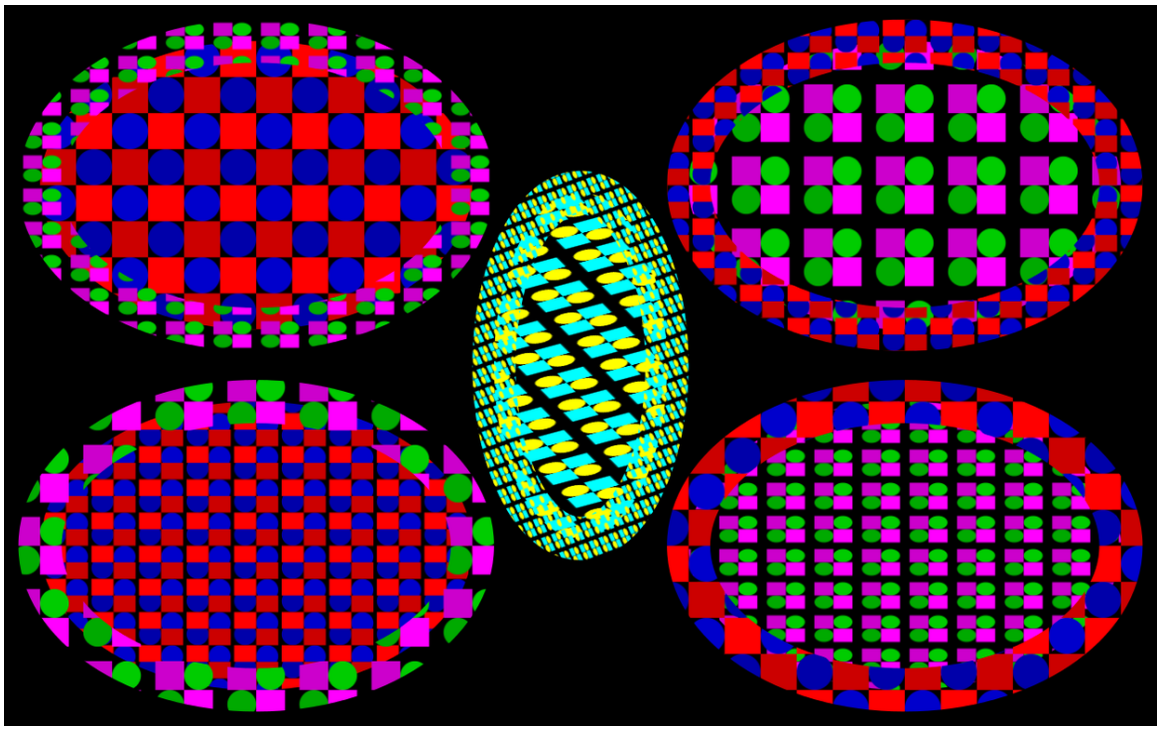


Abb. 178 einige Muster

Folgendes Beispiel zeigt einige weitere Kombinationsmöglichkeiten von Attributen und Attributwerten, um ein Muster zu definieren: einige Muster[21]

stroke und fill von Ellipsen sind jeweils mit einem Muster versehen, wobei zwischen verschiedenen Möglichkeiten gewechselt wird, Muster zu definieren.

21 `http://de.wikibooks.org/wiki/media%3ASVGpattern01.svg`

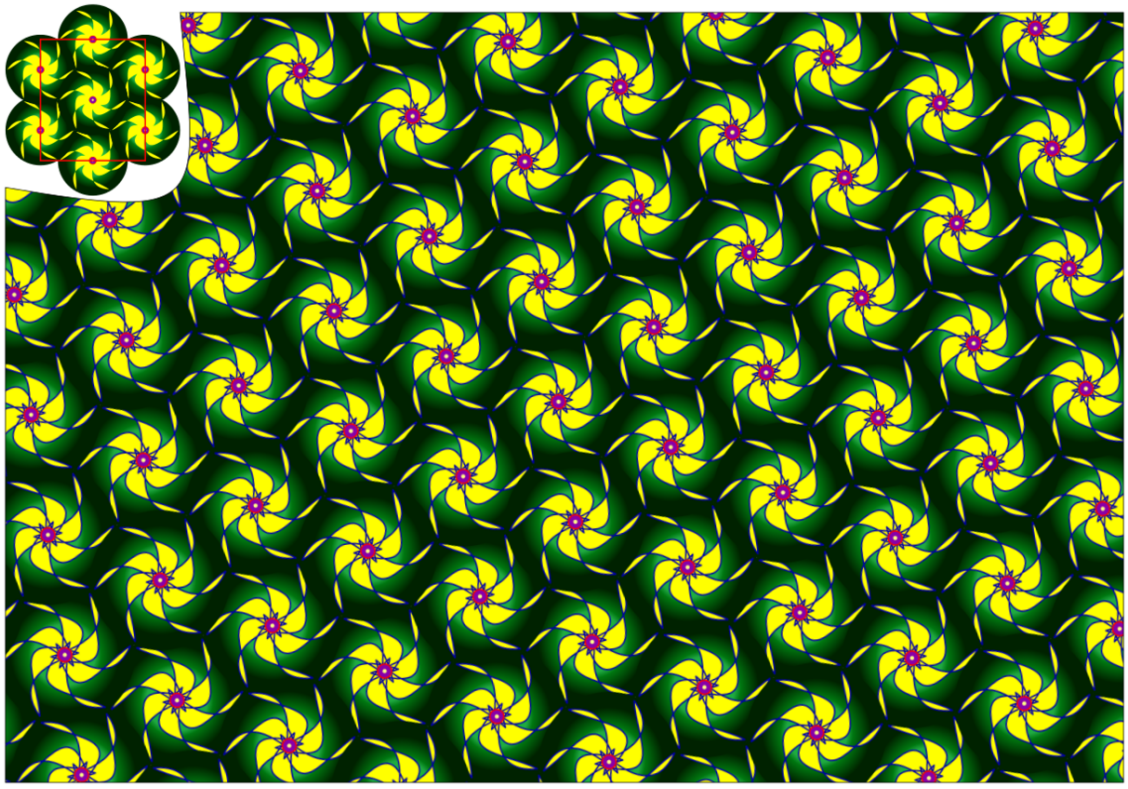


Abb. 179 p6 Blumenmuster, Strategie zur Mustererstellung

Wie eingangs bereits erwähnt, ist es nicht zwangsläufig trivial, Muster zu erstellen, insbesondere wenn diese eine andere Symmetrie haben als jene, die in SVG verwendet wird, um Muster zu erstellen. Folgendes Beispiel erläutert eine Strategie, um ein Muster zu erstellen anhand der ebenen Kristallographischen Gruppe p6: Blumenmuster[22]

Links oben ist das manuell zusammengestellte Muster zu sehen, in der großen Fläche die Anwendung des Musters.

Zunächst wird die Basisstruktur erstellt - hier Pfade, die die erwünschte Symmetrie aufweisen. Das beabsichtigte Grundmuster wird mit ausreichend manuell angeordneten Kopien erstellt. In diesem Grundmuster wird ein rechteckiger Auschnitt gesucht, beziehungsweise berechnet, der sich als periodische Fortsetzung eigne (rot). Dies Rechteck wird sodann als Muster verwendet (große Fläche). Mittels *patternTransform* kann zudem das Grundmuster noch transformiert werden. Das Grundmuster kann nach erfolgreicher Erstellung natürlich einfach ins Element ***defs*** oder ***pattern*** verschoben werden, damit es nicht direkt angezeigt wird.

Zur Gruppe p6:

Ein sechszähliges Rotationszentrum und zwei dreizählige Rotationszentren und drei zweizählige Rotationszentren. Spiegel- oder Gleitspiegelachsen gibt es nicht.

22 `http://de.wikibooks.org/wiki/media%3ASVGpattern02.svg`

Eine Übersicht zu den siebzehn möglichen Gruppen gibt es zum Beispiel bei wikipedia. Dies sind von der Herangehensweise der Symmetriegruppen alle Möglichkeiten, eine Fläche periodisch mit einem Muster zu füllen: Ebene Kristallographische Gruppen[23].

Zu den jeweils möglichen Gruppen gibt es natürlich beliebige Umsetzungsmöglichkeiten, nicht nur historisch besonders interessant sind dabei die Muster, die entstehen, wenn die Fläche periodisch mit regulären Polygonen zu füllen ist, was relativ einfach nur mit gleichseitigen Dreiecken oder Quadraten oder gleichseitigen Sechsecken ist. Es gibt aber auch sehr viele Muster, bei denen mehr als ein reguläres Polygon verwendet wird.

Parkettierungen mit regulären Polygonen[24] (wikipedia, englisch)

11.4 Nichtperiodische und Aperiodische Muster

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5
Konqueror (KSVG)	3.2
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Werden Muster dargestellt, so zwangsläufig nur über einen endlichen Bereich, somit ist die periodische Parkettierung der Ebene mehr eine konzeptionelle Idee oder im Fall von Element ***pattern*** eine formale Darstellungsvorschrift, welche auf beliebige Teilmengen der Ebene anwendbar ist.

23 `http://de.wikipedia.org/wiki/Ebene%20kristallografische%20Gruppe`
24 `http://en.wikipedia.org/wiki/Tiling_by_regular_polygons`

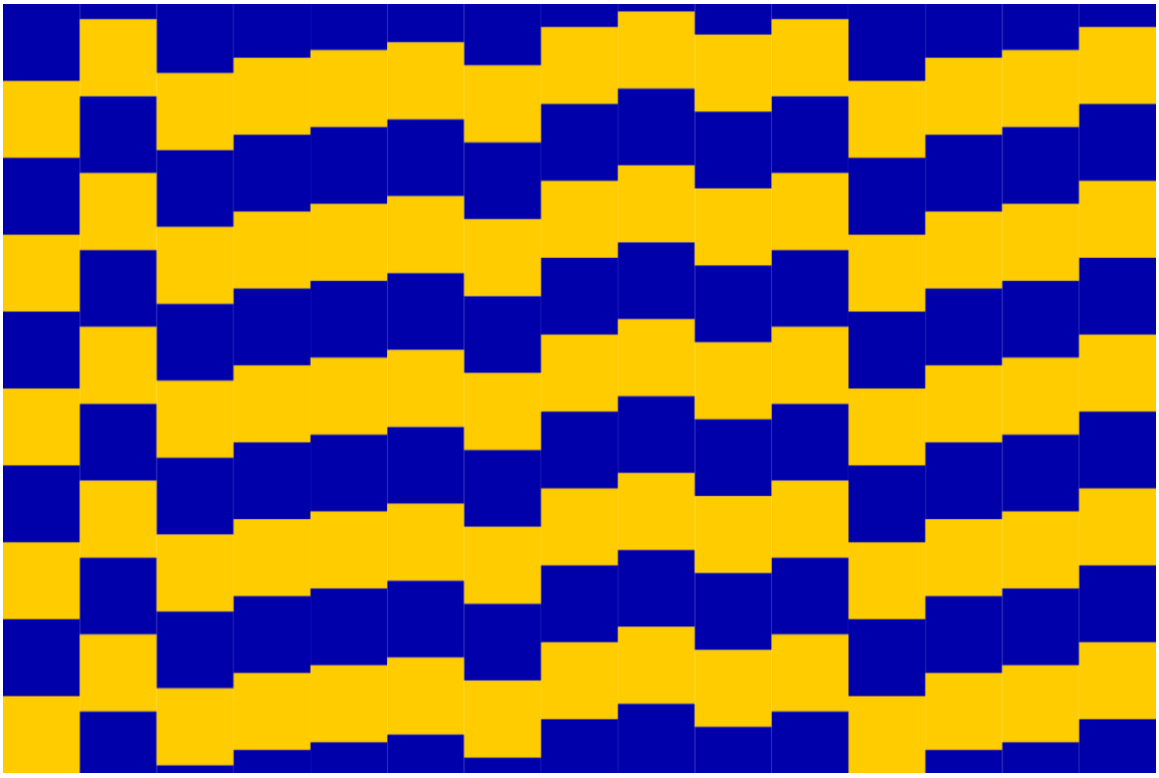


Abb. 180 Nichtperiodische Parkettierung durch zufällige Translation

Offenbar kann die Periodizität leicht gestört werden, indem immer weitere neue Grundformen hinzugefügt werden oder aber auch Teile der Muster zufällig verschoben werden. Solche nichtperiodischen Strukturen sind auch für sehr einfache Grundformen möglich, mit denen ohne die Verschiebung auch periodische Strukturen möglich sind:

Nichtperiodische Parkettierung durch zufällige Translation[25].

Hier werden Muster mit zwei Quadraten in jeder Spalte in y-Richtung zufällig verschoben, es gibt also nur eine Periodizität in y-Richtung. Das ist folglich keine periodische Parkettierung der Ebene. Gleichwohl kann aber das Element pattern in einem solchen Falle verwendet werden, um die Flächenfüllung in y-Richtung zu realisieren.

Ergibt sich hingegen aus den Grundstrukturen (gegebenenfalls mit speziellen Regeln zur Kombination derselben) nie eine periodische Parkettierung, aber dennoch eine Parkettierung der Ebene, so handelt es sich um eine aperiodische Parkettierung. Diskrete Rotations- oder Spiegelsymmetrien sind hingegen nicht ausgeschlossen.

25 `http://de.wikibooks.org/wiki/media%3ASVGpattern03.svg`

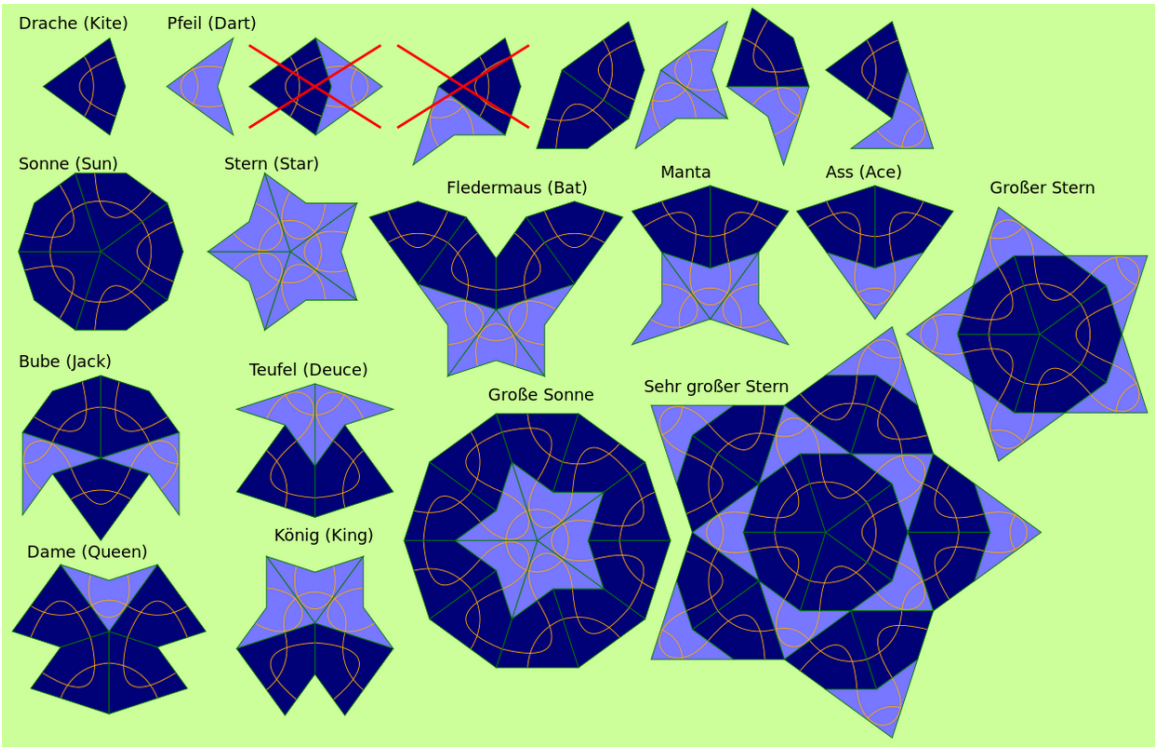


Abb. 181 Einige Grundformen zu Drachen und Pfeil

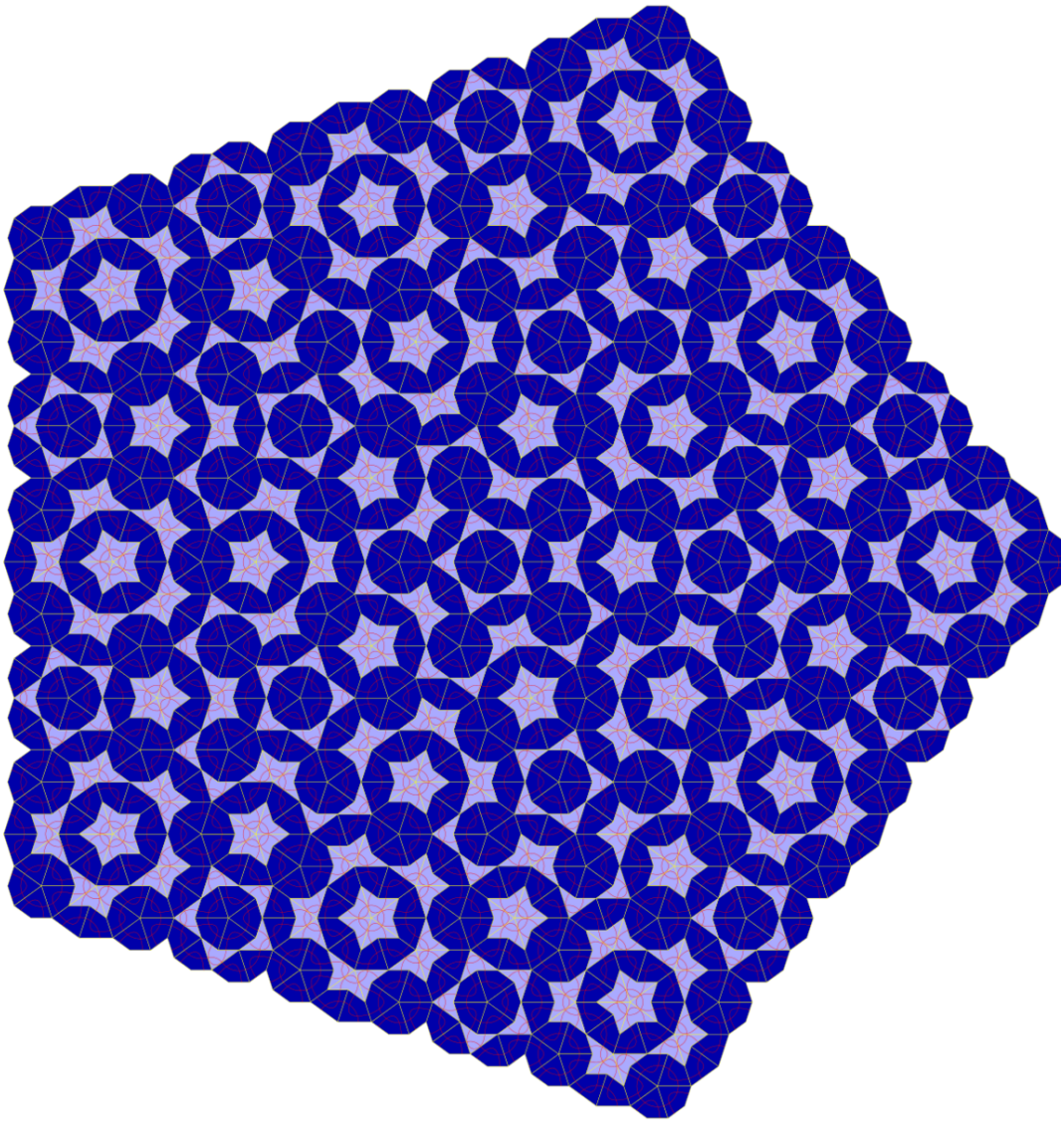


Abb. 182 Kleiner Ausschnitt einer Penrose-Parkettierung

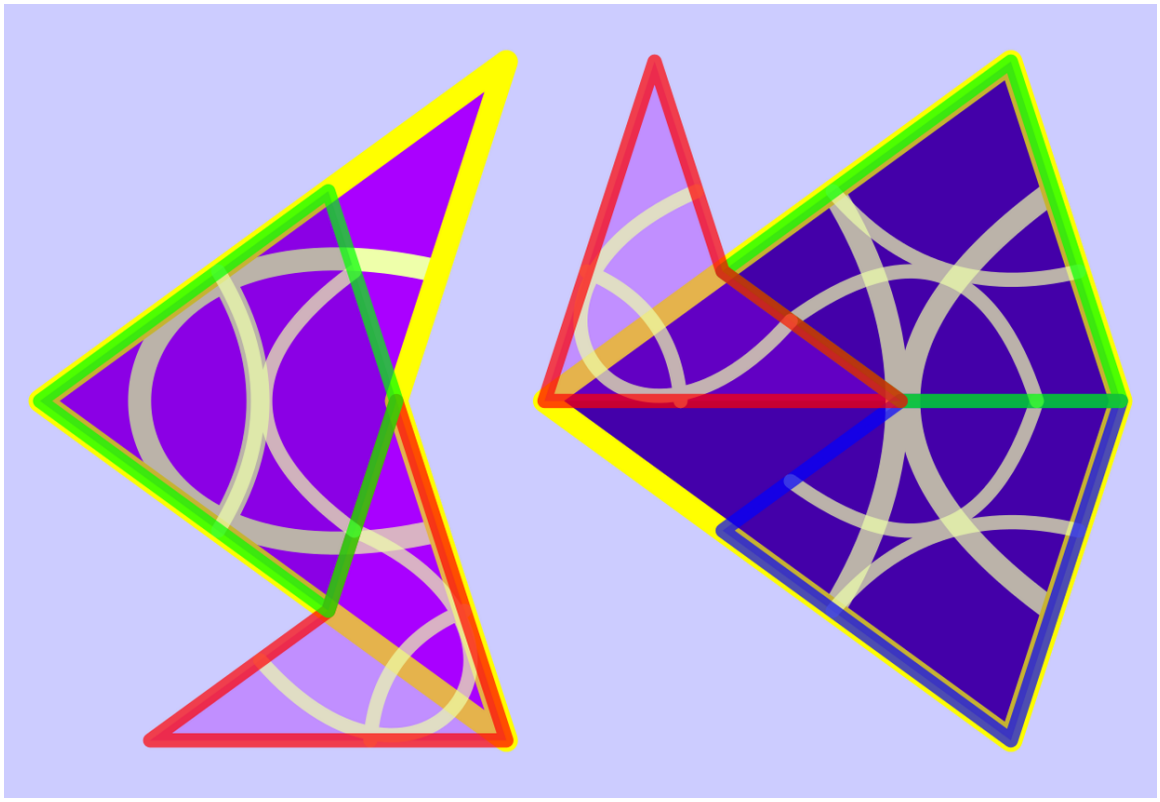


Abb. 183 Veranschaulichung der Deflationsmethode

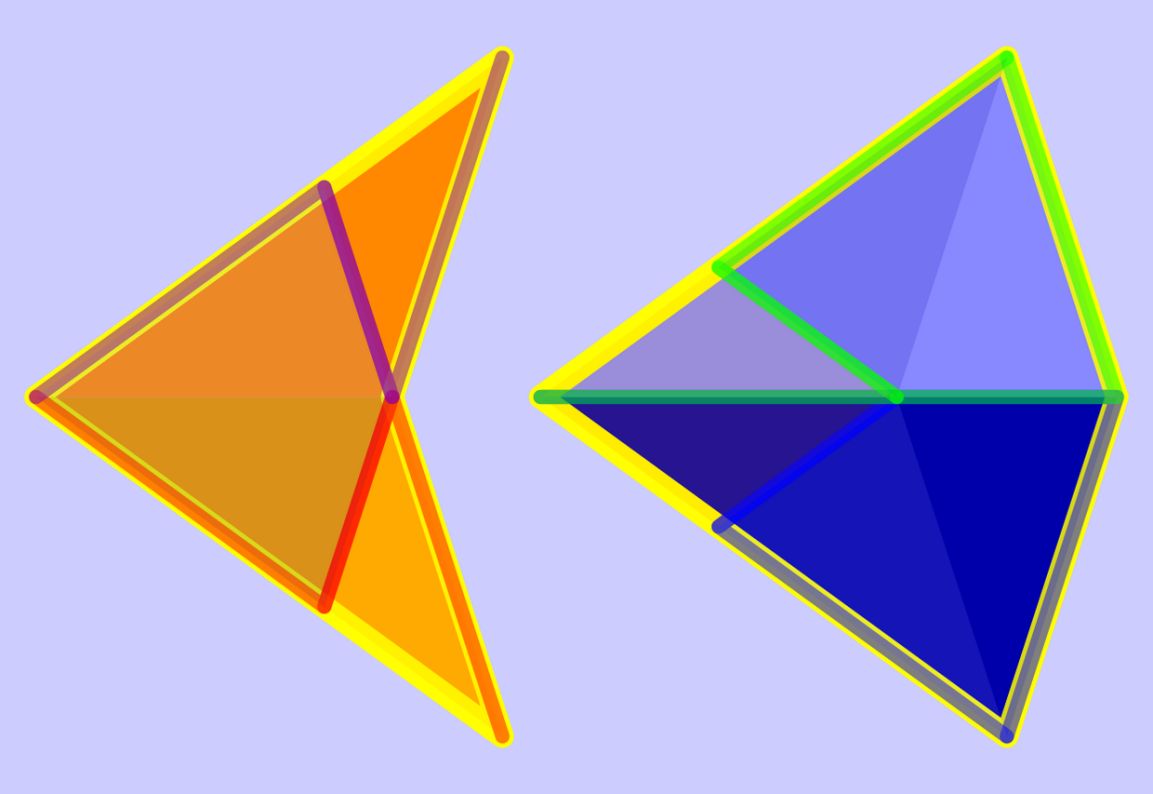


Abb. 184 Veranschaulichung der Deflationsmethode (mit Dreiecken)

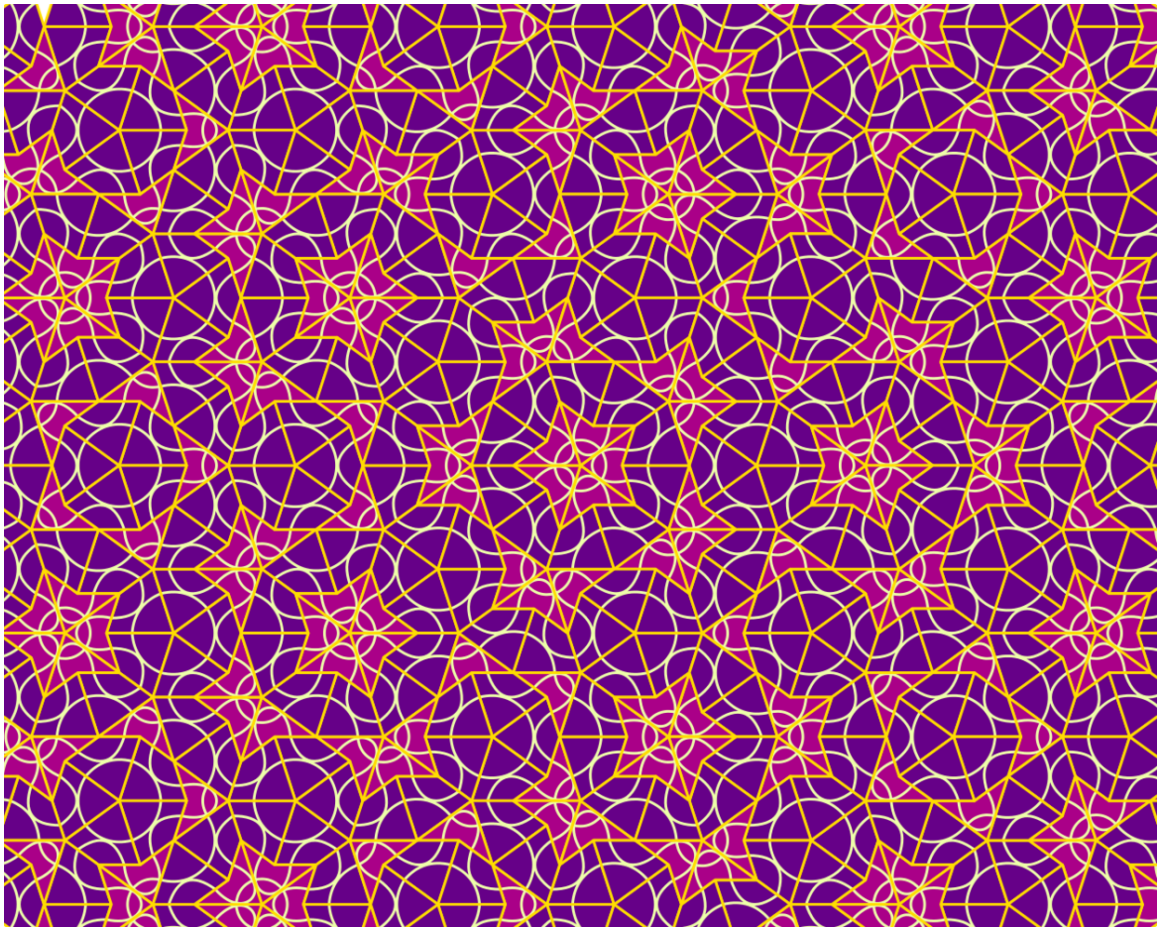


Abb. 185 Deflation Drachen und Pfeil

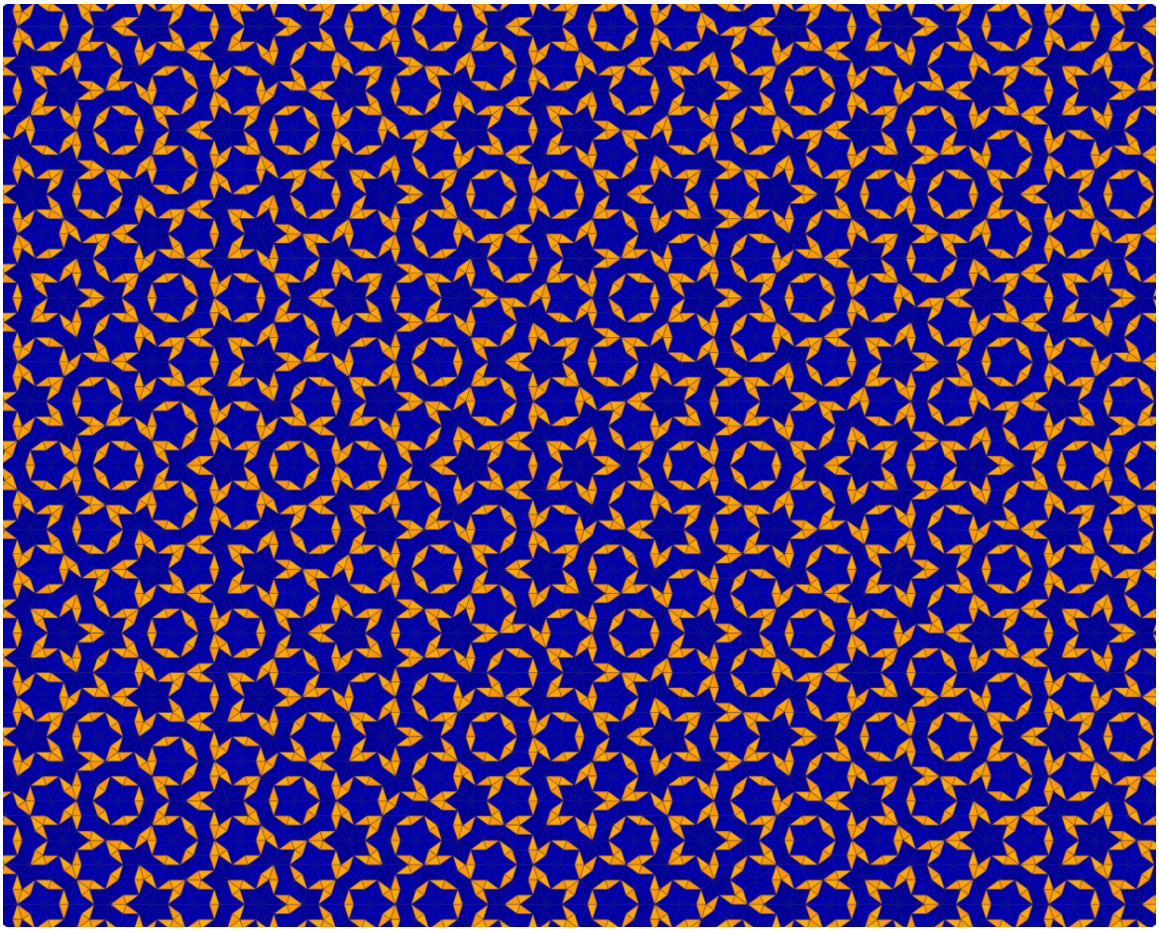


Abb. 186 Deflation mit Dreiecken zur Rautenvariante

Einfache Vorschläge für aperiodische Parkettierungen von Roger Penrose sind besonders bekannt geworden, dies sind Drachen und Pfeil oder dicke und dünne Raute, die zudem ineinander transformierbar sind. Die lokale Selbstähnlichkeit dieser Muster zeigt bereits die enge Verwandtschaft zu Fraktalen, die Parkettierungen haben allerdings noch eine Reihe weiterer interessanter Eigenschaften.

Anhand von Drachen und Pfeil wird anhand der folgenden Beispiele erläutert, wie solche Muster erstellt werden können, auch wenn dazu nicht das Element ***pattern*** verwendet werden kann, sondern lediglich ***use*** und das Attribut *transform* genau wie bei Fraktalen.

Bei Drachen und Pfeil handelt es sich um Vierecke mit exakt einer Spiegelachse und mit zwei Paaren von gleichlangen Seiten. Das Verhältnis von langer zu kurzer Seite entspricht dem Goldenen Schnitt mit der Goldenen Zahl $(1+5^{1/2})/2 = 1.618033988...$

Die Drachen genannte Figur ist eine konvexe Form, der Pfeil hingegen hat die kurzen Seiten nach Innen geklappt. Beim Drachen sind drei Innenwinkel 72 Grad und der zwischen den beiden kurzen Seiten 144 Grad. Beim Pfeil ist der Winkel zwischen den langen Seiten 72 Grad, der Außenwinkel zwischen den beiden kurzen Seiten 144 Grad und die beiden übrigen Innenwinkel 36 Grad.

Einige Grundformen zu Drachen und Pfeil[26].

Die Dekoration hilft, weil damit selbst bei den einfachen symmetrischen Formen die Regeln zur Zusammenlegung einfach eingehalten werden können. Die Regeln bestehen aus zwei Verboten, hier die Zusammenlegungen, die rot durchgestrichen sind.

Mit etwas Geduld kann so manuell ein Muster konstruiert werden:

Kleiner Ausschnitt einer Penrose-Parkettierung[27]

Weil die Goldene Zahl eng mit dem regulären Polygon Pentagramm verknüpft ist, ergibt sich oft eine fünfzählige Symmetrie, dies hängt aber von der Grundform in der Mitte ab.

Weil dieses Vorgehen des manuellen Zusammenlegens bei größere Flächen recht mühsam ist, wird für größere Flächen die Deflationsmethode verwendet, bei der aus einem einfachen Anfangsmuster wie der 'Sonne' durch Rekursion Flächen automatisch mit vielen Grundformen gefüllt werden.

Veranschaulichung der Deflationsmethode.[28]

Im Hintergrund in blau-violett sind die Grundformen Drachen und Pfeil dargestellt. Die Grundform wird in verkleinerte Duplikate zerteilt, hier wieder Drachen und Pfeil, blau, grün und rot gerandet. Diese ersetzen letztlich die Grundform und sind hier so angeordnet, dass sie erst korrekt zusammengesetzt eine Parkettierung ergeben.

Bei der Deflationsmethode werden die kleineren Duplikate abermals durch verkleinerte Duplikate ersetzt, und so fort, bis das Muster fein genug erscheint.

Prinzipiell ist es egal, auf welche Figur die Deflation letztlich angewendet wird. Um die endliche Fläche des Anzeigebereiches aber komplett und effektiv zu parkettieren, bietet es sich an, eine Figur zu wählen, die keine Teile hat, die allzuweit über den Anzeigebereich hinausragen, um Rechenleistung und Arbeitsspeicher des Darstellungsprogrammes nicht zu vergeuden.

Veranschaulichung der Deflationsmethode (mit Dreiecken).[29]

Im Hintergrund in blau und orange sind die Grundformen Drachen und Pfeil dargestellt. Die Grundform wird in verkleinerte Duplikate zerteilt, hier offene Dreiecke, Strich in blau, grün, rot und magenta. Diese ersetzen letztlich die Grundform und sind hier so angeordnet, dass sie korrekt zusammengesetzt eine Parkettierung ergeben. Je nachdem, wie die Dreiecke zusammengesetzt (oder eingefärbt) werden, ergibt sich eine Parkettierung mit Dreiecken, Rauten oder mit Drachen und Pfeil.

Beispiele:

26 http://de.wikibooks.org/wiki/media%3ASVG_Penrose02.svg
27 http://de.wikibooks.org/wiki/media%3ASVG_Penrose03.svg
28 http://de.wikibooks.org/wiki/media%3ASVG_Penrose04.svg
29 http://de.wikibooks.org/wiki/media%3ASVG_Penrose05.svg

Deflation Drachen und Pfeil[30]

Deflation mit Dreiecken zur Rautenvariante[31]

Die Grundformen Drachen und Pfeil können nun in einer bestimmten Weise deformiert werden, um komplett asymmetrische Grundformen zu erhalten, die auch nicht mehr falsch zusammenlegbar sind.

Eine Methode, um eine asymmetrische Form zu erhalten, besteht darin, im ersten Schritt die lange und die kurze Seite durch einen (nahezu beliebigen) Pfad zu ersetzen.

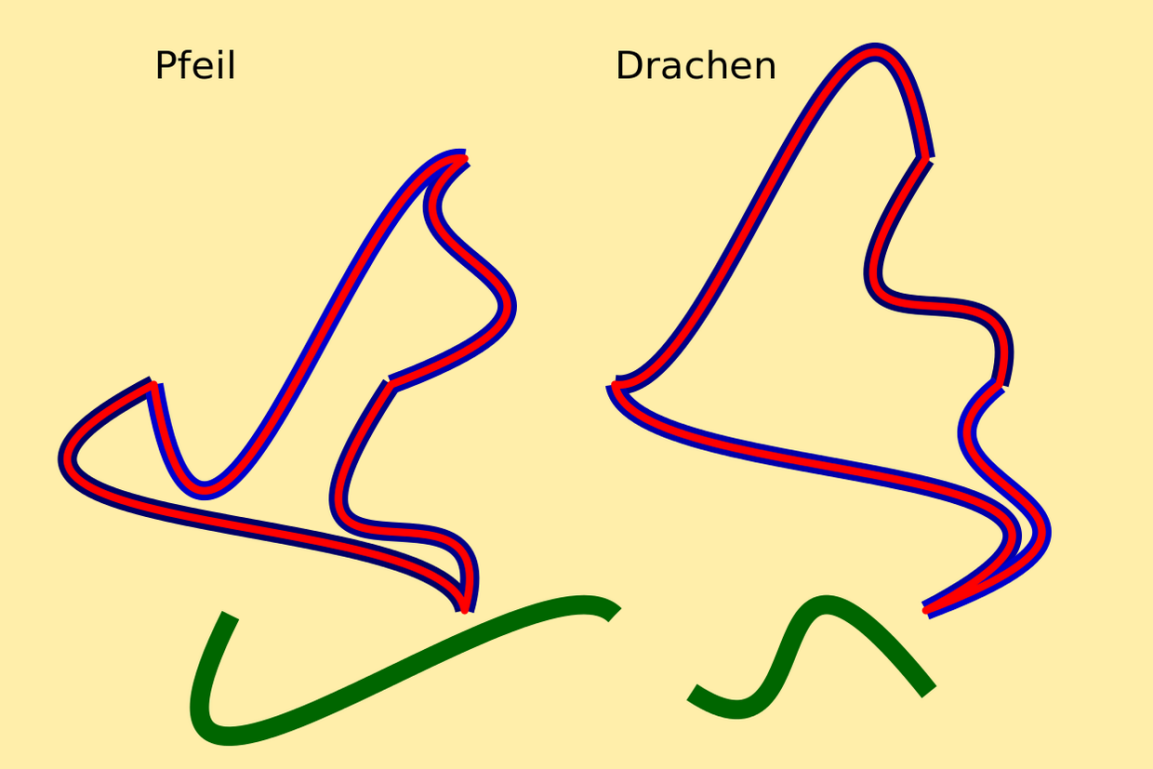


Abb. 187 Erzeugung asymmetrischer Grundformen

30 http://de.wikibooks.org/wiki/media%3ASVG_Penrose06.svg
31 http://de.wikibooks.org/wiki/media%3ASVG_Penrose07.svg

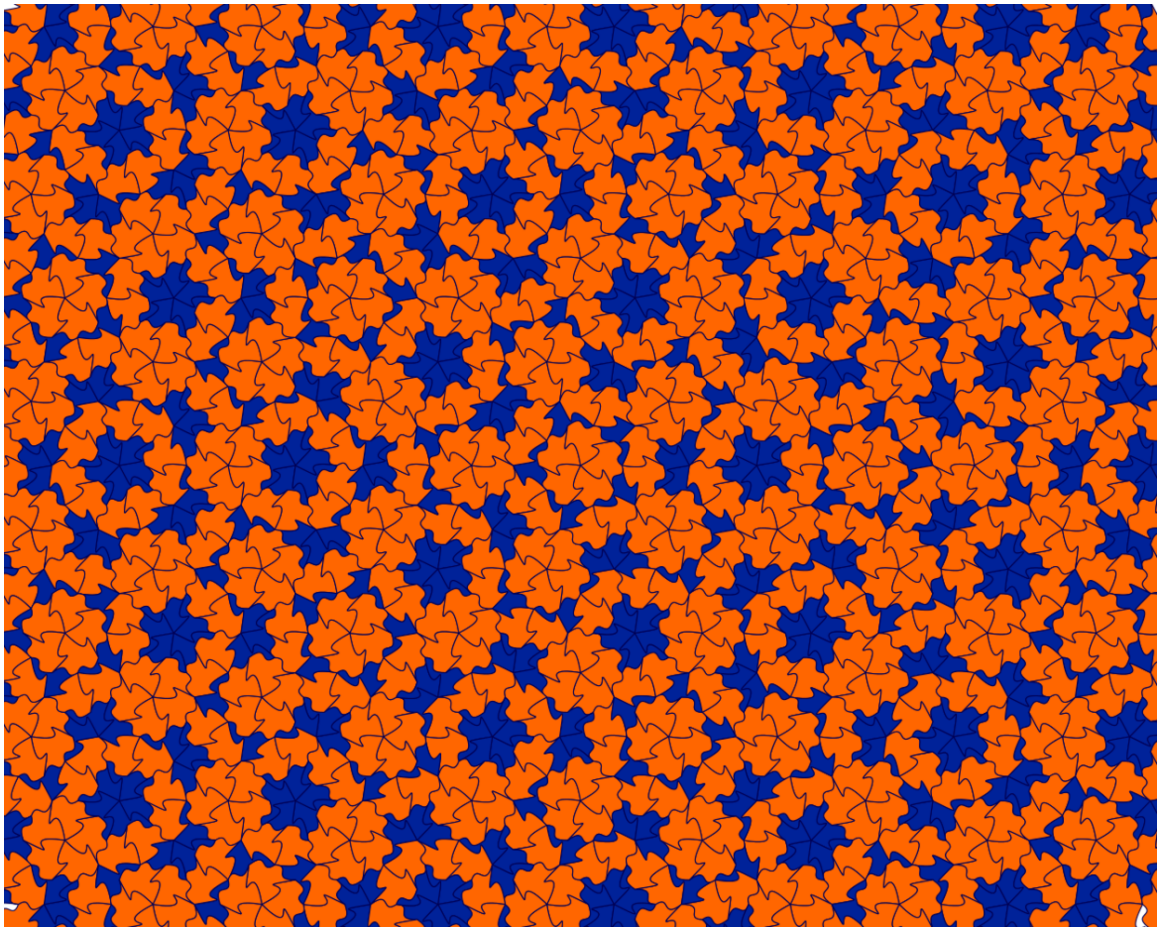


Abb. 188 Drachen und Pfeil mit asymmetrischen Grundformen

Entspricht zum Beispiel die lange Seite dem Pfad "M0 0 100 0", so können dazwischen Kontrollpunkte eingefügt werden, um eine kubische Kurve zu erhalten oder auch etwas komplizierteres. Eine kurze Seite ist dann wegen des festliegenden Seitenverhältnisses ein Pfad "M0 0 61.803398875 0", wie in folgendem Beispiel:

Erzeugung asymmetrischer Grundformen[32].

Durch entsprechende Drehungen und Verschiebungen können jeweils zwei lange und zwei kurze Pfade zu Drachen und Pfeil zusammengesetzt werden. Dies ist mit den blauen Einzelpfaden angedeutet.

Statt nun die Einzelpfade direkt malen zu lassen, sind die Transformationen mit einem Skript oder manuell auf die Punkte und Kontrollpunkte anzuwenden. Anschließend sind die so transformierten Pfade passend zu den Formen zusammenzusetzen. Zu beachten ist, dass sich die Reihenfolge der Notation bei der Zusammensetzung ändert, wie in den Beschreibung entsprechend der vorliegenden Reihenfolge angegeben, die ebenfalls durch Änderung des Blaus von dunkel zu hell angedeutet ist. Das Endergebnis, hier die roten Pfade, muss dann jedenfalls mit den Einzelpfaden übereinstimmen.

32 `http://de.wikibooks.org/wiki/media%3ASVG_Penrose08.svg`

Einfache Selbstüberschneidungen können verwendet werden, wenn *fill-rule* auf 'evenodd' gesetzt ist. Komplexere Überschneidungen über die nächste Grundform hinaus können jedoch problematisch sein.

Auch die Ecken können verschoben werden, was jedoch Auswirkungen auf zwei Seiten gleichzeitig hat und daher etwas schwieriger zu berücksichtigen ist.

In Kombination mit der Deflationsmethode ergeben sich so weitere interessante Variationen von Drache und Pfeil:

Drachen und Pfeil mit asymmetrischen Grundformen[33].

33 http://de.wikibooks.org/wiki/media%3ASVG_Penrose01.svg

12 Markierungen

12.1 Markierungen

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Bei Markierungen handelt es sich um Symbole, welche am Anfang, am Ende und an jedem angegebenen Punkt eines Pfades angezeigt werden können. Genauer sind Markierungen anwendbar auf die Elemente ***path***, ***polygon***, ***polyline*** und ***line***.

Markierungen sind in der Vollversion von SVG 1.1 zulässig.

Markierungen eignen sich insbesondere dazu, die aktuelle Richtung des Pfades zu kennzeichnen.

Die Graphik für die Markierung wird im Element ***marker*** notiert. Dies kann auch animierter Inhalt sein, der dann bei allen dargestellten Markierungen verwendet und animiert wird.

Markierungen können referenziert werden mit speziellen Eigenschaften, *marker-start* für den Beginn eines Pfades, *marker-end* für das Ende und *marker-mid* für alle angegebenen Punkte dazwischen. Die drei Eigenschaften sind auch als Präsentationsattribute verfügbar. Mit einer vierten Eigenschaft *marker* können alle drei Eigenschaften zusammengefasst werden. Für diese Eigenschaft gibt es kein Präsentationsattribut. Als Präsentationsattribut sind die Eigenschaften also immer einzeln zu notieren. Unter angegebenem Punkt, auch Knoten genannt (englisch: vertex oder node), wird ein Anfangs- oder Endpunkt eines Pfadfragmentes verstanden, nicht etwa ein Kontrollpunkt.

Somit kann eine Markierung nicht an eine beliebige Position entlang eines Pfades gesetzt werden. Die gewünschten Positionen der Markierungen sind also explizit als Pfadpunkte zu notieren.

Eine Verteilung in Anteilen der gesamten Länge eines Pfades ist eher mit dem Element ***animateMotion*** möglich, zusammen mit dem Attribut *keyPoints* und einem geeigneten Zeitablauf, welcher zu einer zeitunabhängigen Positionierung führt.

12.2 Element *marker*

Das Element ***marker*** definiert durch den in ihm enthaltenen Inhalt die Graphik, welche als Markierung verwendet wird.

marker hat einen eigenen Anzeigebereich.

Eigenschaften werden vom Elternelement in den ***marker*** vererbt, nicht vom referenzierenden Element aus. Das macht es insbesondere schwierig, die Eigenschaften vom referenzierenden Element automatisch zu übernehmen. Dafür wären dann kompliziertere Tricks oder Anordnungen notwendig, insbesondere wenn dieselbe Markierung so für verschiedene Elemente verwendet werden soll (zum Beispiel kann teilweise ***use*** in ***marker*** verwendet werden).

marker werden wie Symbole niemals direkt angezeigt, sondern nur referenziert.

marker können selbst nicht Ziel von Ereignissen sein.

Die Eigenschaft *overflow* von ***marker*** ist auf 'hidden' voreingestellt. Für ein anderes Verhalten muss die Eigenschaft also notiert werden.

12.2.1 Attribut *markerUnits*

Mit dem Attribut *markerUnits* wird angegeben, in welchen Einheiten, in welchem Koordinatensystem die Markierung notiert ist.

Mögliche Werte sind 'strokeWidth' und 'userSpaceOnUse'.

Falls 'strokeWidth' angegeben ist, entspricht der aktuelle Wert von *stroke-width* des referenzierenden Elementes einer Einheit im Koordinatensystem der Markierung. Dies bezieht sich auf *markerWidth*, *markerHeight* und den Inhalt von ***marker***.

Falls 'userSpaceOnUse' angegeben ist, wird die Einheit des aktuellen Koordinatensystems des referenziernden Elementes verwendet.

Falls *markerUnits* nicht angegeben ist, wird 'strokeWidth' angenommen. Das Attribut ist animierbar.

12.2.2 Attribute *refX* und *refY*

Mit den Attributen wird die x- beziehungsweise die y-Koordinate im Koordinatensystem der Markierung angegeben, welcher Punkt als der Punkt verwendet werden soll, der mit den zu markierenden Punkten des referenziernden Elementes übereinstimmen soll.

Der Wert ist jeweils eine Koordinate. Ist eines der Attribute nicht angegeben, wird für dieses fehlende jeweils 0 angenommen.

Die Attribute sind animierbar.

12.2.3 Attribute *markerWidth* und *markerHeight*

markerWidth und *markerHeight* geben Breite und Höhe des Anzeigebereiches der Markierung an.

Der Wert ist jeweils eine nicht negative Länge. Ein negativer Wert ist ein Fehler, der zum Abbruch der Darstellung des Dokumentes führt. Ist mindestens einer der beiden Werte 0, so unterbindet dies die Darstellung der Markierung. Für ein nicht angegebenes Attribut wird 3 angenommen.

Die Attribute sind animierbar.

12.2.4 Attribut *viewBox*

Das Attribut *viewBox* gibt eine rechteckige Region an, in welche die Markierung fällt. Das Attribut ist sinngemäß bereits für das Element ***svg*** beschrieben.

Das Attribut ist animierbar.

12.2.5 Attribut *preserveAspectRatio*

Wenn das Seitenverhältnis der *viewBox* nicht zu *width* und *height* passt, kann mit *preserveAspectRatio* angegeben werden, wie es in den Anzeigebereich einzupassen ist.

Die möglichen Werte sind bereits entsprechend für das Element ***svg*** beschrieben.

Das Attribut ist animierbar.

12.2.6 Attribut *orient*

Mit dem Attribut *orient* wird angegeben, wie die Markierung bei der Darstellung orientiert wird.

Mögliche Werte sind entweder 'auto' oder ein Winkel. Ein Winkel ist eine Zahl, optional direkt gefolgt von einer Einheit. Mögliche Einheiten sind: 'deg' für Grad (eine Umdrehung 360deg), 'grad' für Neugrad (eine Umdrehung 400grad) oder 'rad' (eine Umdrehung 2π). Ist keine Einheit angegeben, so wird 'deg' angenommen.

Der Wert 0 wird angenommen, wenn das Attribut nicht angegeben ist.

Das Attribut ist animierbar, aber nicht additiv.

Ist ein Winkel angegeben, so bestimmt dieser die Ausrichtung im lokalen Koordinatensystem des referenzierenden Elementes. 0 bedeutet zum Beispiel ein Ausrichtung der x-Achse der Markierung in Richtung der x-Achse des lokalen Koordinatensystems. Bei 90 wird die x-Achse der Markierung in Richtung der y-Achse des lokalen Koordinatensystems ausgerichtet.

Beim Wert 'auto' wird die x-Achse der Markierung entlang der Richtung des Pfades ausgerichtet. Die Richtung bei Ecken bestimmt sich als Mittelwert der Richtungen vor und nach der Ecke. An offenen Enden wird jeweils die Richtung des verfügbaren Pfadsegmentes gewählt. Bei geschlossenen Unterpfaden wird jeweils ein Mittelwert von der ersten und der letzten Richtung des Unterpfades genommen.

Nun gibt es auch Situationen, wo ein Pfadsegment keine Richtung hat. Beim Startpunkt wird zunächst versucht, die Richtung aus dem vorherigen Segment desselben Unterpfades zu ermitteln.

Beim Endpunkt wird zunächst versucht, die Richtung aus dem folgenden Segment desselben Unterpfades zu ermitteln.

Sofern für Anfangs- beziehungsweise für den Endpunkt keine Richtung gefunden wurde, wird die Richtung des End- beziehungsweise Anfangspunktes genommen, sofern dieser eine hat. Ist so immer noch keine Richtung auffindbar, so wird die lokale x-Achse als Richtung angenommen.

Für Z-Kommandos gibt es Sonderregeln. Folgt auf ein Kommando Z kein M oder m, so ist die in den Knoten eingehende Richtung jene, die zum Z-Pfadsegment gehört. Die ausgehende Richtung ist die des ersten Pfadsegmentes des Unterpfades.

Sonst gehört zwar auch die in den Knoten eingehende Richtung zum Z-Pfadsegment, die ausgehende Richtung gehört aber zum nächsten Pfadsegment des Unterpfades.

12.3 Markierungseigenschaften

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

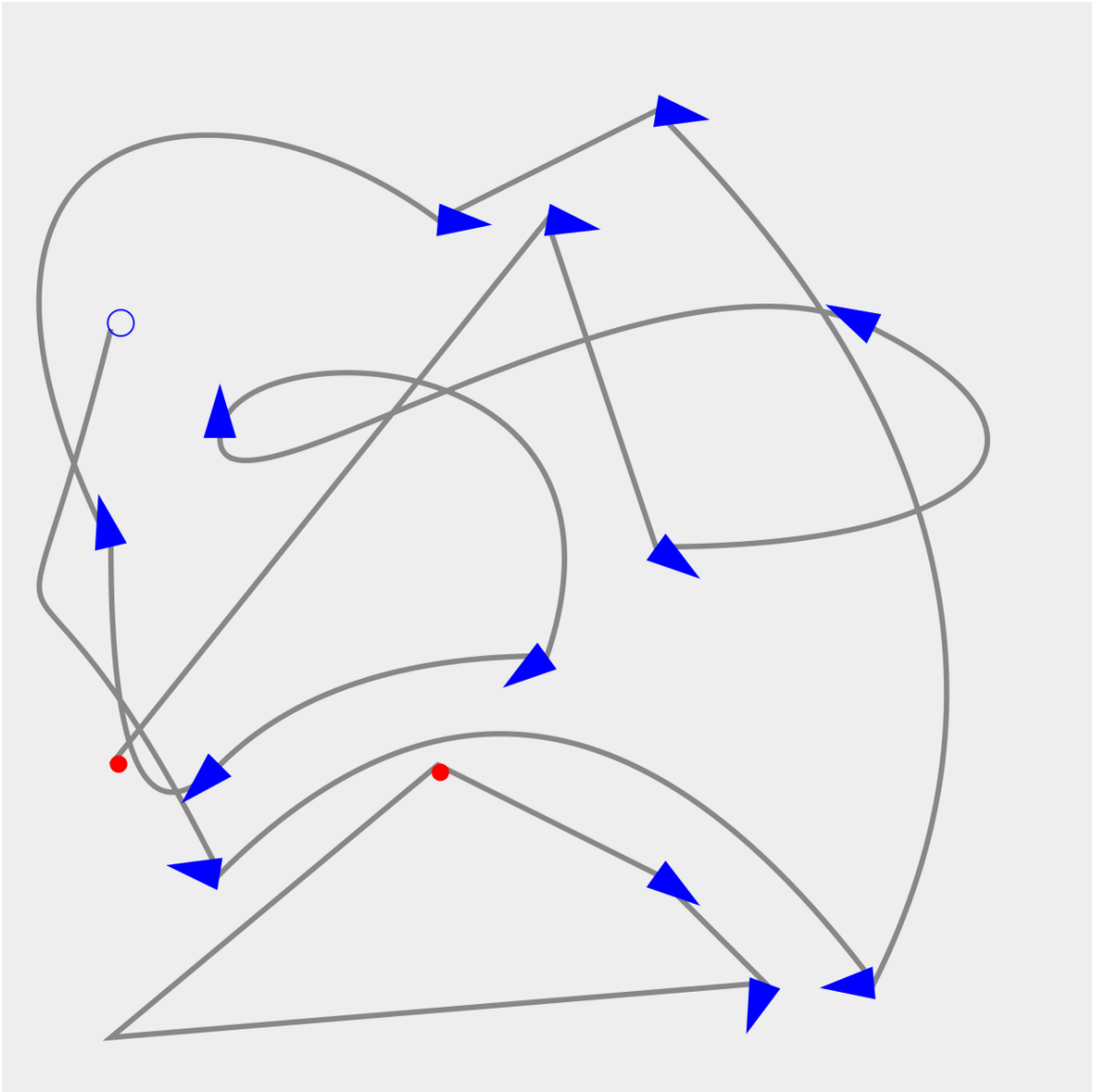


Abb. 189 Markierungen bei einfachen Pfaden

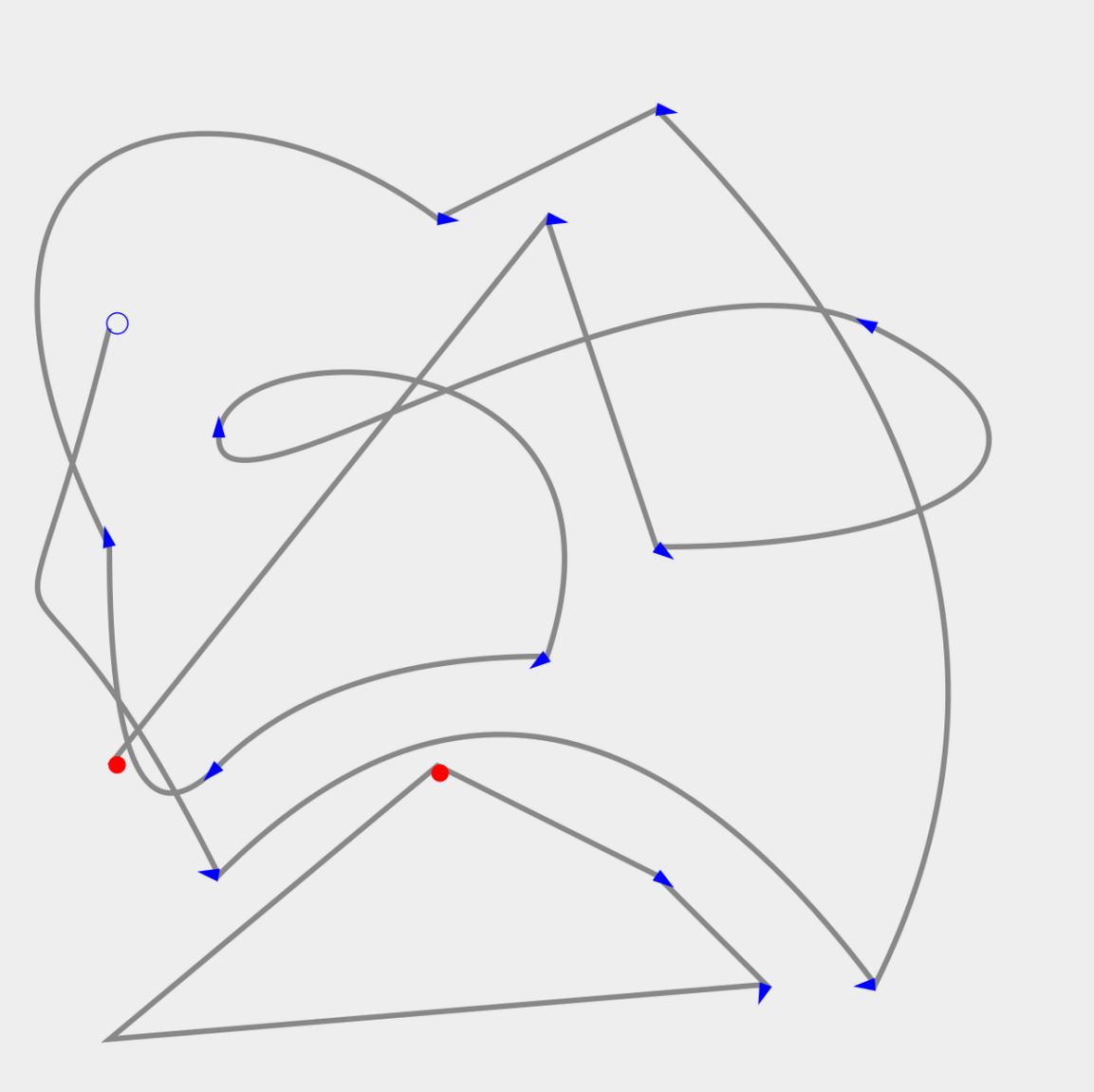


Abb. 190 Markierungen bei einfachen Pfaden (2)

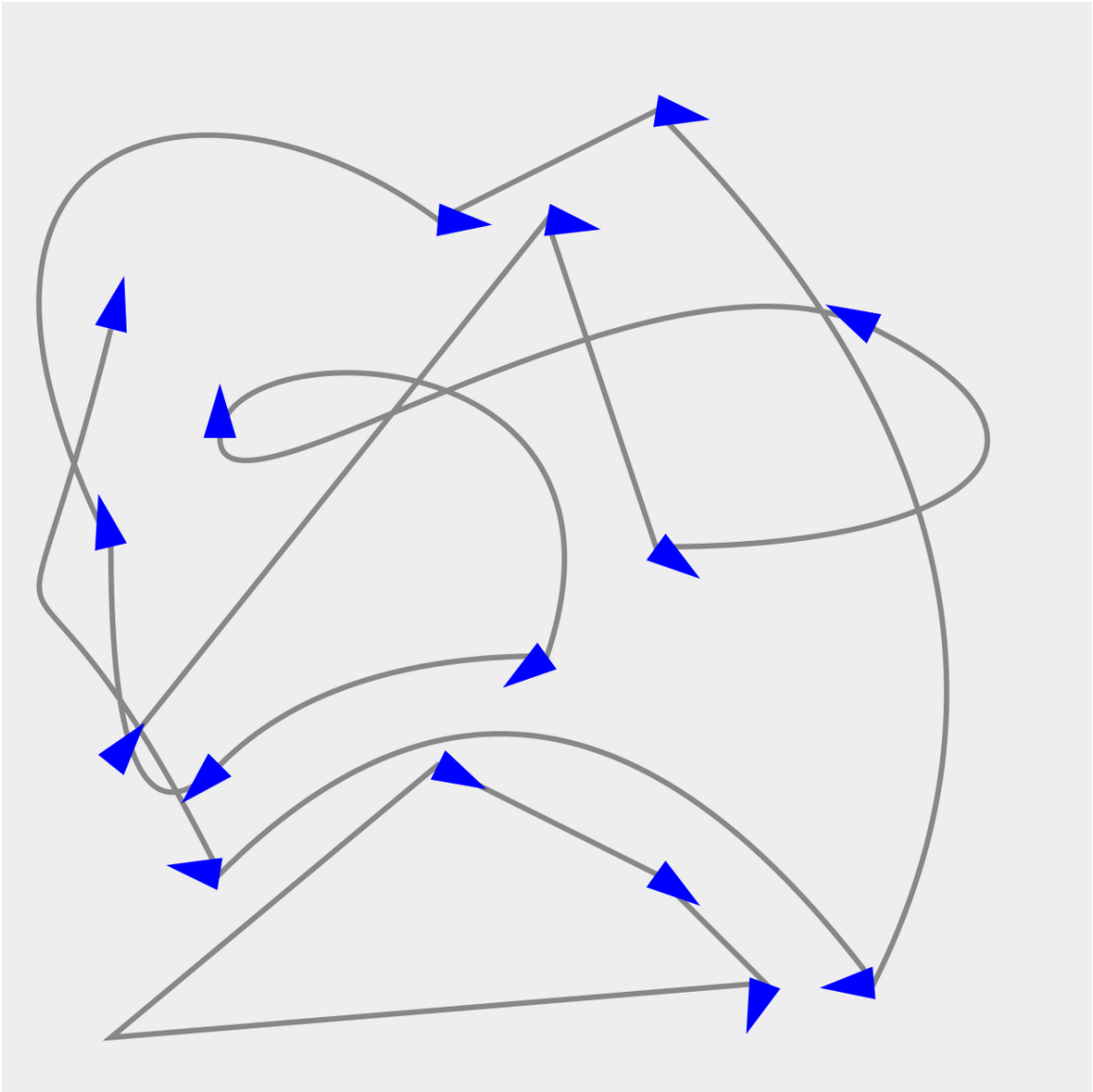


Abb. 191 Markierungen, Attribut *orient*

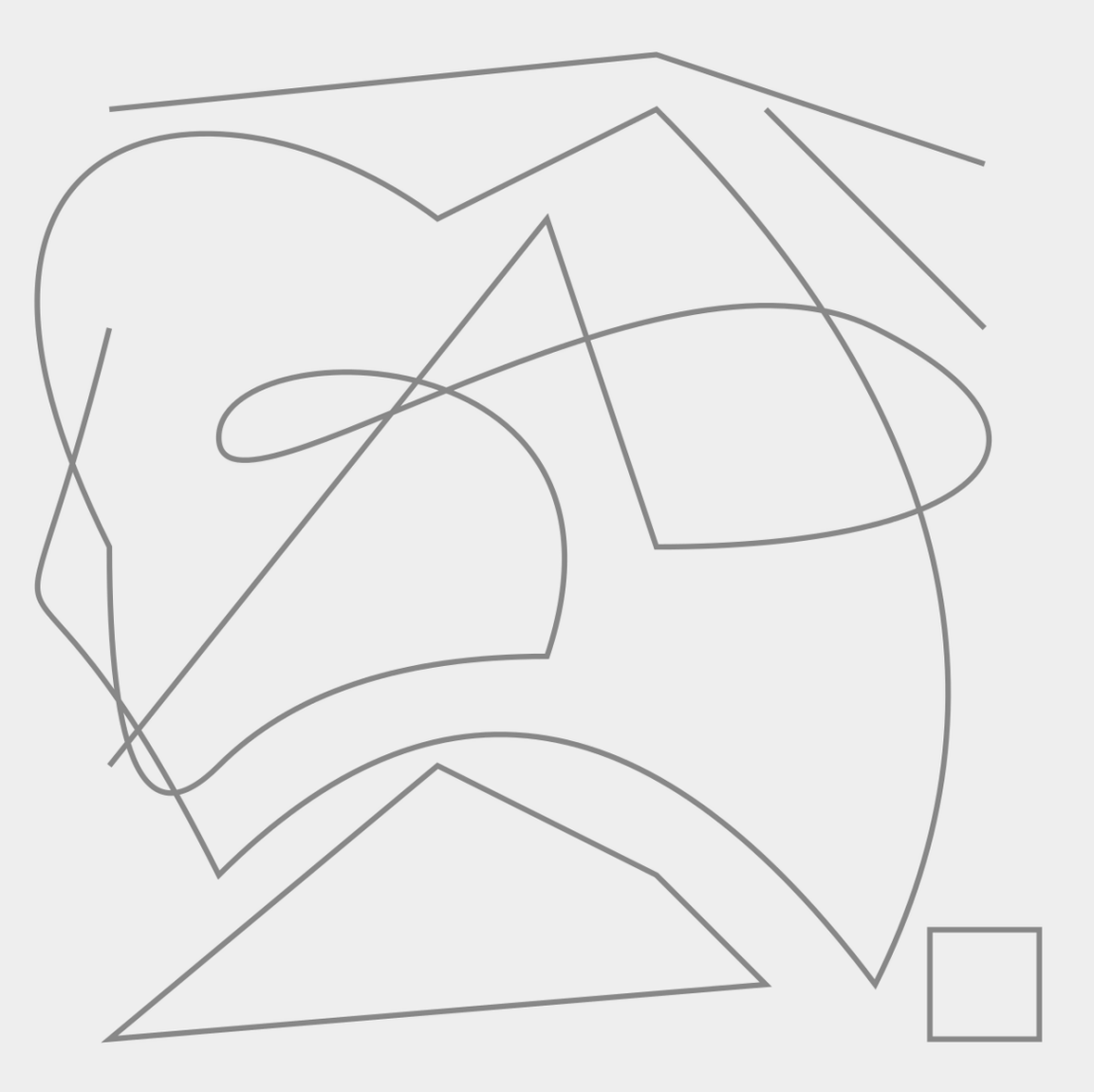


Abb. 192 Markierungen, Eigenschaft *marker*

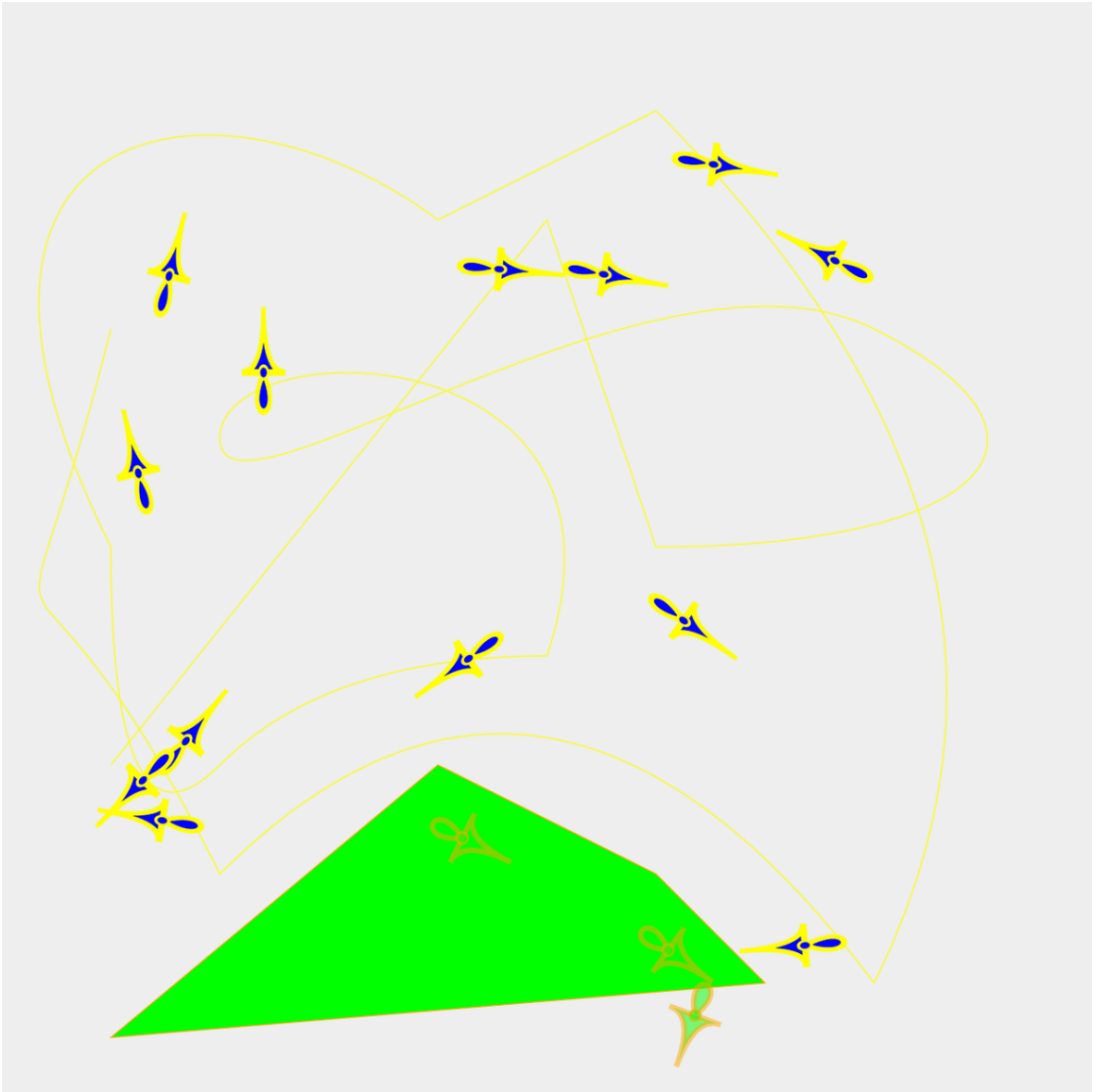


Abb. 193 Markierung und Vererbung

Bei ***path***, ***line***, ***polyline*** und ***polygon*** können Markierungen mit den Eigenschaften *marker-start*, *marker-end*, *marker-mid* und *marker* gesetzt werden. *marker* faßt die anderen drei Eigenschaften zusammen und ist nicht als Präsentationsattribut verfügbar.

Mögliche Werte sind:

inherit

geerbt

none

keine Markierung (Voreinstellung)

Funktionale URI

Fragmentidentifizierer eines Elementes ***marker***

Die Eigenschaften sind animierbar und werden vererbt.

Gibt es den angegebenen Fragmentidentifizierer nicht oder gehört dieser nicht zu einer Markierung, so wird keine Markierung verwendet.

Die Darstellung erfolgt so, dass erst der Pfad dargestellt wird und darauf die Markierung.

12.4 Beispiele für Markierungen

Markierungen bei einfachen Pfaden[1]

Beispielpfade ***path***, ***line***, ***polyline*** und ***polygon*** mit Markierungen, ***rect*** ohne.

Die Markierungseinheit richtet sich nach der Strichbreite.

Markierungen bei einfachen Pfaden (2)[2]

Beispielpfade ***path***, ***line***, ***polyline*** und ***polygon*** mit Markierungen, ***rect*** ohne.

Die Markierungseinheit richtet sich nach dem lokalen Koordinatensystem.

Markierungen, Attribut *orient*[3]

Beispielpfade ***path***, ***line***, ***polyline*** und ***polygon*** mit Markierungen, ***rect*** ohne.

5 Sekunden nach Dokumentbeginn wird das Attribut *orient* animiert, springt begonnen mit 0 Grad jeweils um 30 Grad weiter. Anfangs ist *orient* 'auto'.

Markierungen, Eigenschaft *marker*[4]

Beispielpfade ***path***, ***line***, ***polyline*** und ***polygon*** mit Markierungen, ***rect*** ohne.

Die Markierungen werden mit der Eigenschaft *marker* angewendet.

Markierung und Vererbung[5]

Vererbt werden Eigenschaften in das Element ***marker*** vom Elternelement, nicht vom referenzierenden Element. Ist etwas ähnliches wie das letztere erwünscht, sollten die Eigenschaften in einem gemeinsamen Elternelement von beiden notiert werden. Das ist eine Situation, bei der es auch mal inhaltlich sinnvoll sein kann, das Element ***marker*** mitten im Quelltext und nicht im Element ***defs*** des Hauptelementes ***svg*** zu notieren.

1 `http://de.wikibooks.org/wiki/media%3ASVGmarker01.svg`
2 `http://de.wikibooks.org/wiki/media%3ASVGmarker02.svg`
3 `http://de.wikibooks.org/wiki/media%3ASVGmarker03.svg`
4 `http://de.wikibooks.org/wiki/media%3ASVGmarker04.svg`
5 `http://de.wikibooks.org/wiki/media%3ASVGmarker05.svg`

Kompliziertere Inhalte von Markierungen lassen sich dann auch mit ***use***-Elementen wiederverwenden.

Wie an der hellen Fläche zu sehen ist, muss der Strich selbst auch nicht unbedingt gemalt sein, damit die Markierungen gesetzt werden können.

12.5 Markierungen und Richtungen

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	9 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (teilweise)
Konqueror (KSVG)	3.2 (teilweise)
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

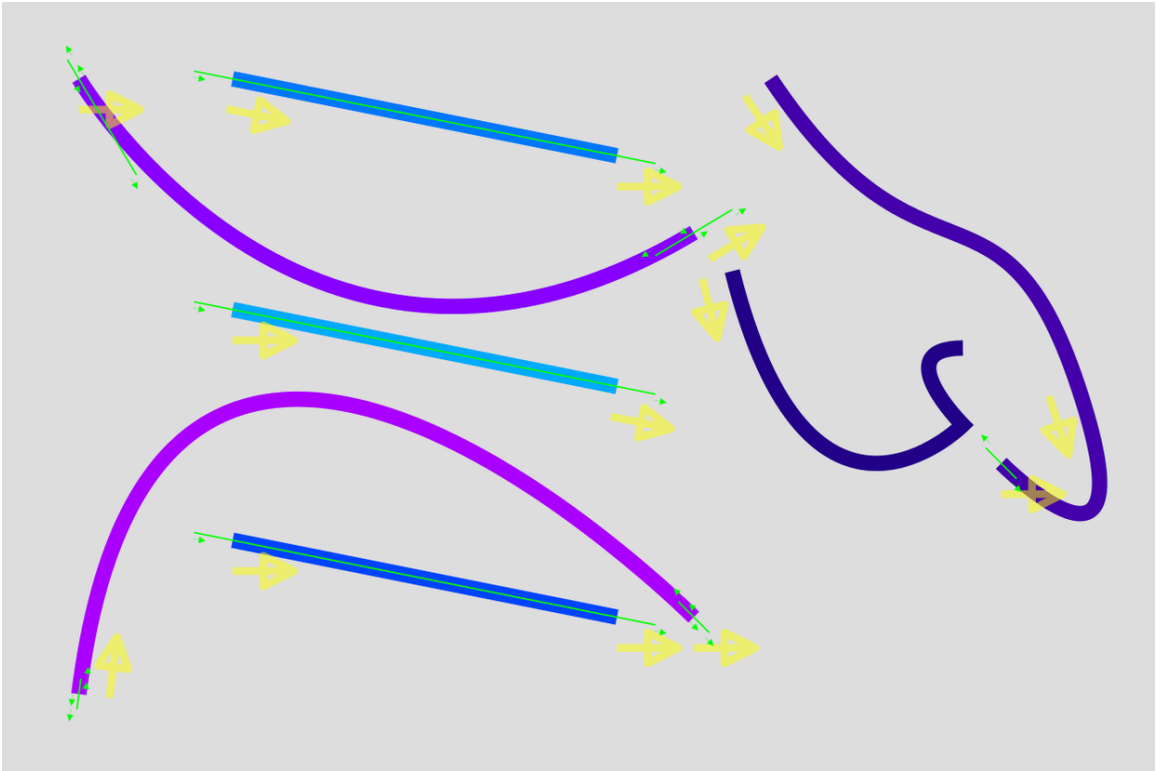


Abb. 194 Richtungen, wenn Punkte und Kontrollpunkte zusammenfallen

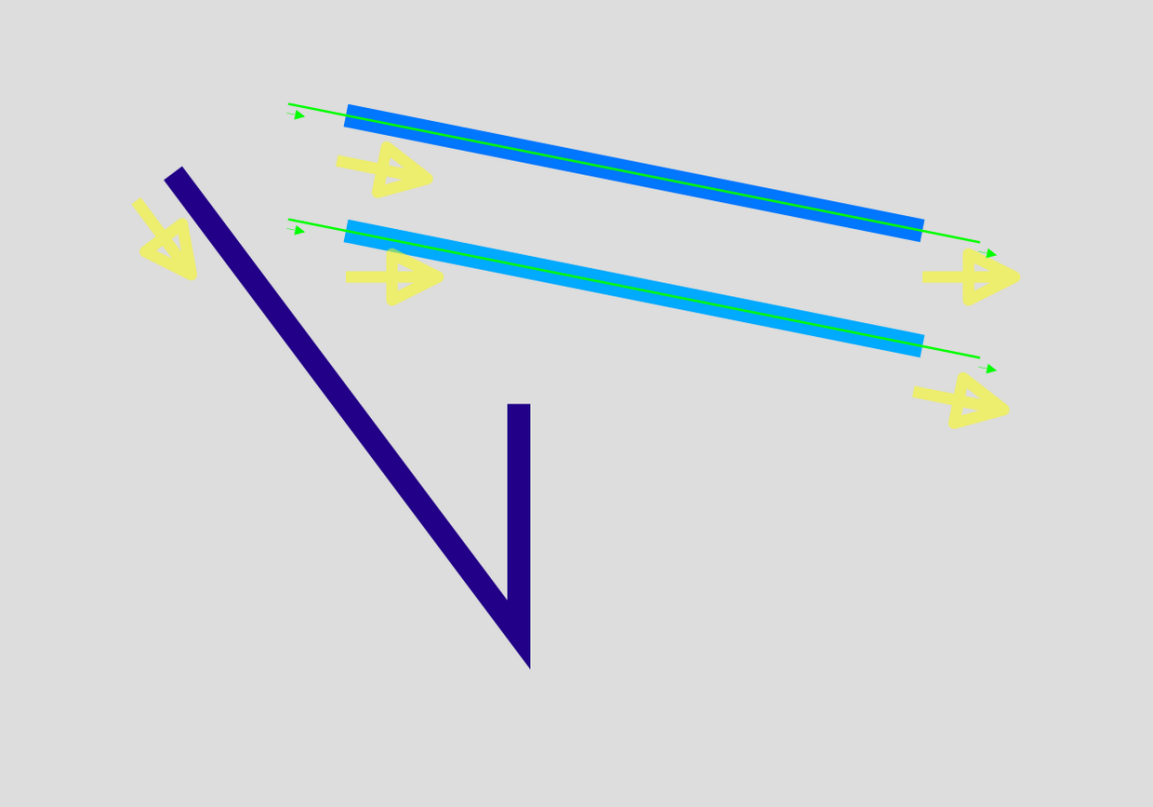


Abb. 195 Richtungen, wenn Punkte und Kontrollpunkt zusammenfallen

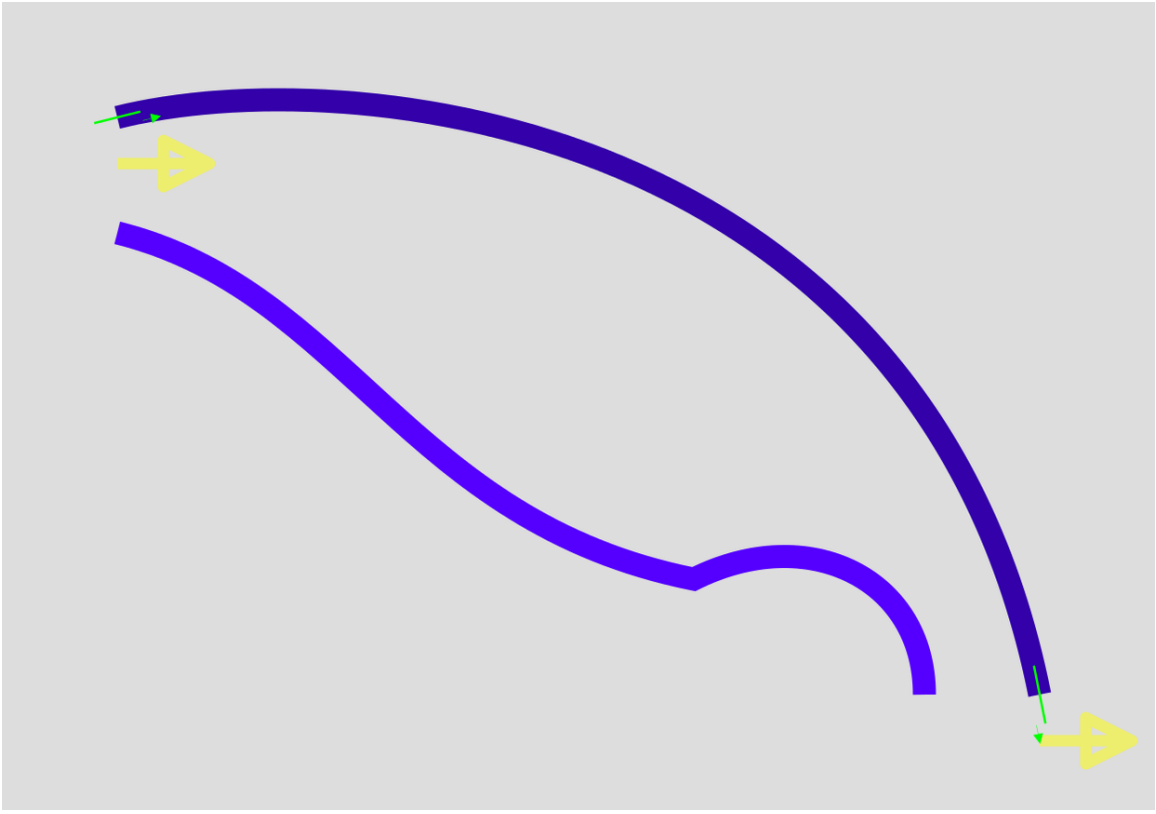


Abb. 196 Richtungen und Pfadlänge 0

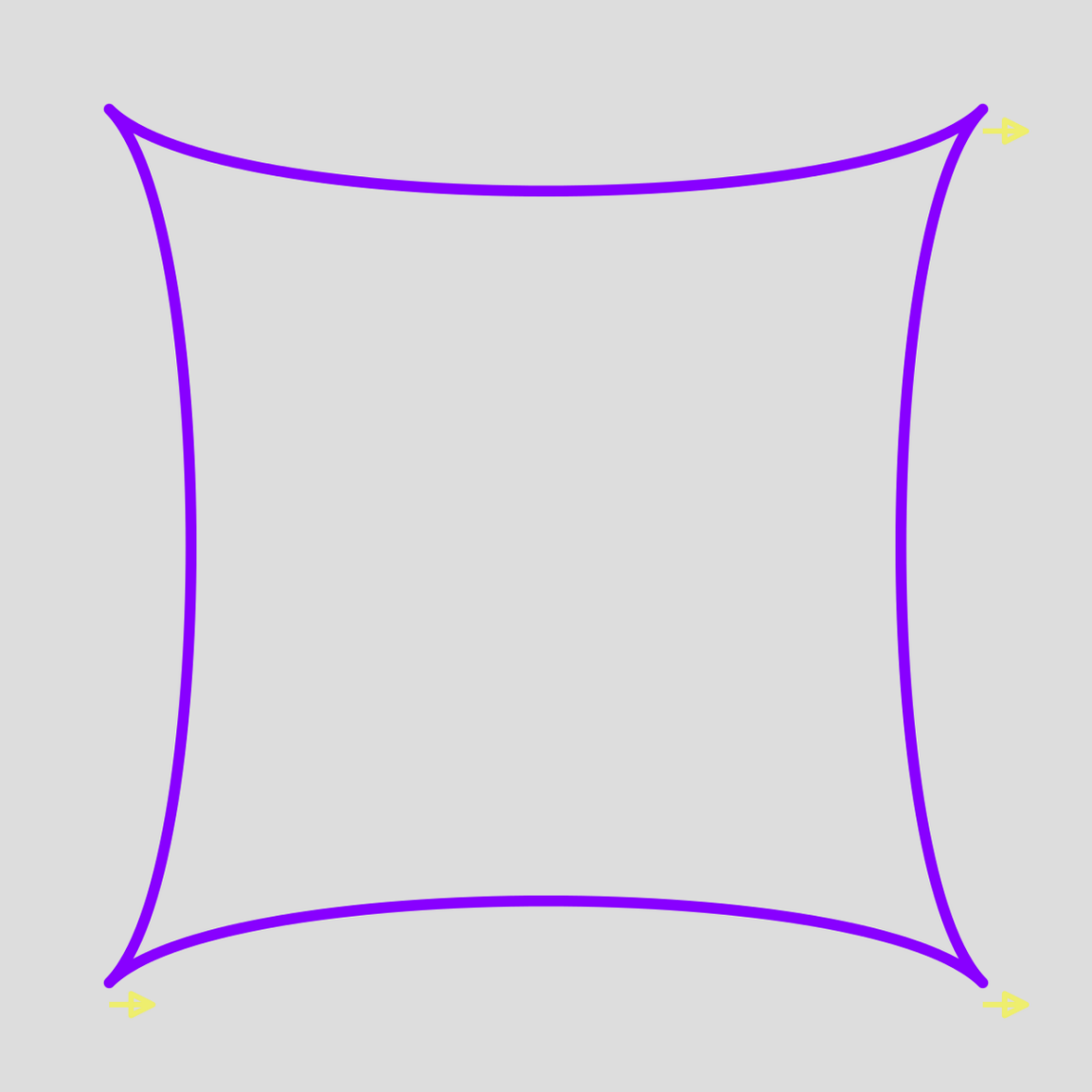


Abb. 197 Richtungen und Mittelwert 0

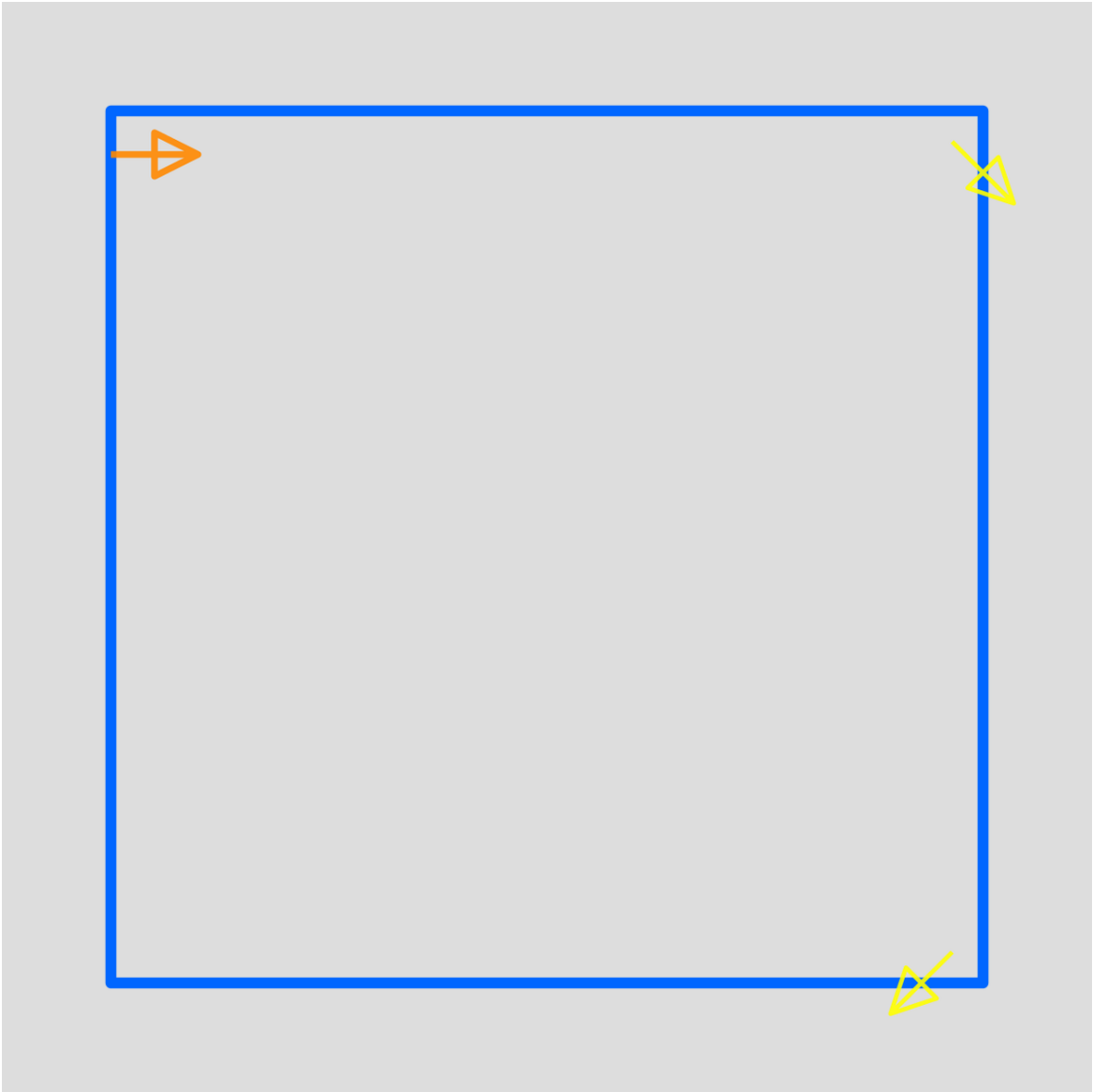


Abb. 198 Richtungen und Z-Kommando

Abb. 199 Richtungen und Z-Kommando (2)

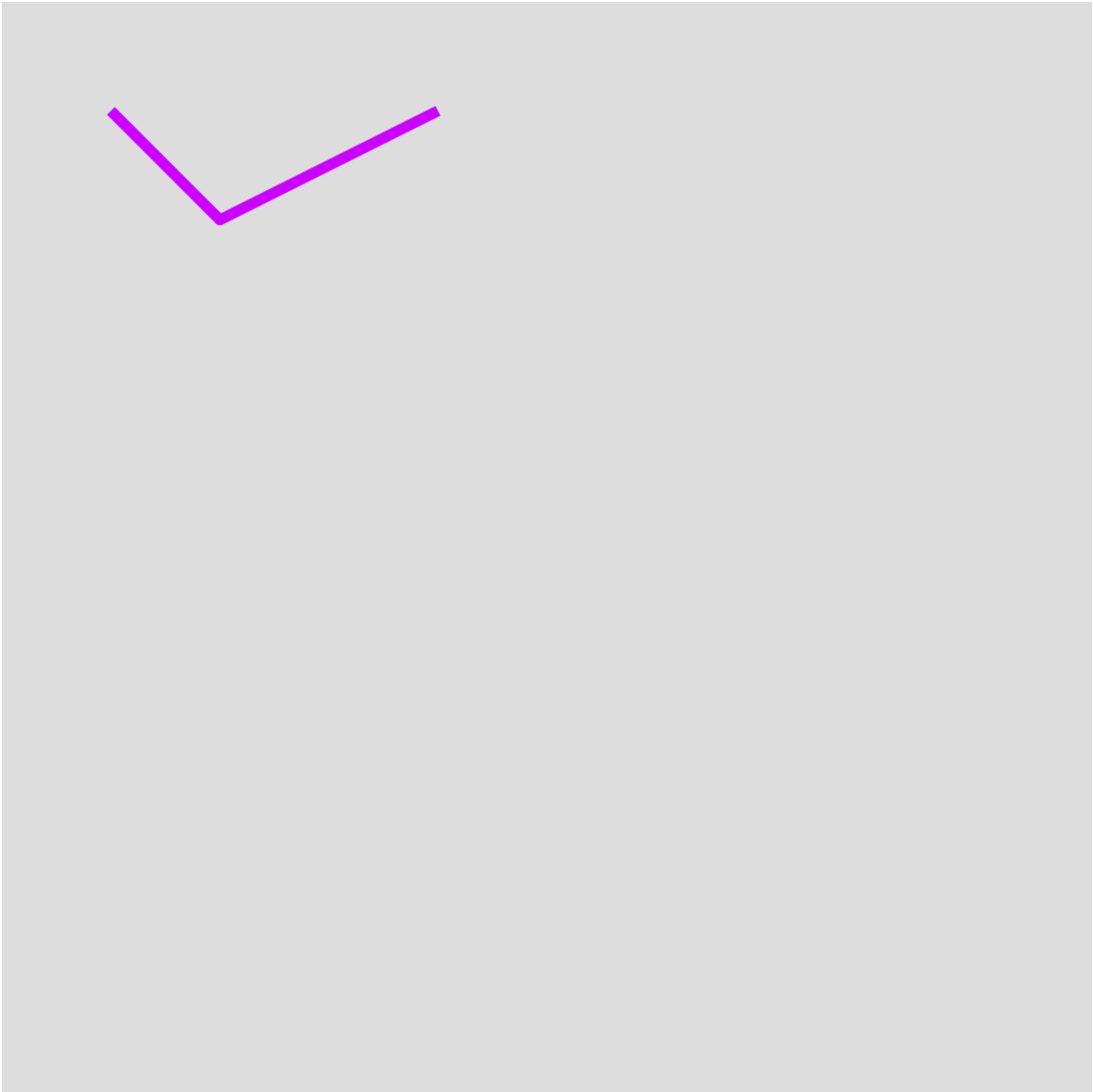


Abb. 200 Richtungen und ungeöhnliche Kommandokombinationen in Pfaden

Bereits im Kapitel über Pfade wurden Formeln angegeben, wie die Richtung einer Kurve an einer beliebigen Stelle bestimmt werden kann.

Neben den Fällen, die in der kanonischen Parametrisierung relativ einfach zu berechnen sind, gibt es auch jene, die etwas mehr Probleme bereiten können, insbesondere weil ein Kontrollpunkt mit einem zugehörigen Endpunkt des Kurvensegmentes zusammenfällt - oder auch beide Kontrollpunkte. Eigentlich sollten Darstellungsprogramme diese noch relativ harmlosen Fälle problemlos darstellen können, indes Kontrolle ist besser:

Richtungen, wenn Punkte und Kontrollpunkte zusammenfallen[6]

Richtung von kubischen Pfaden - Kurven, bei denen Kontroll- und Endpunkte übereinstimmen. Die Markierungen in Form von gelben Pfeilen sollen jeweils in Richtung der grünen Striche zeigen, sofern vorhanden, welche die korrekte Richtung des Pfades an den jeweils interessanten Stellen andeuten.

Richtungen, wenn Punkte und Kontrollpunkt zusammenfallen[7]

Richtung von quadratischen Pfaden - Kurven, bei der Kontroll- mit einem Endpunkt übereinstimmt. Sonst wie beim vorherigen Beispiel.

Problematisch können auch Pfadsegmente der Länge 0 sein. In solchen Fällen muss das Darstellungsprogramm wie beschrieben erst im vorherigen Pfadsegment desselben Unterpfades nach einer Richtung suchen, dann im folgenden.

Es auch möglich, dass gar keine Richtung ermittelt werden kann, weil die Länge des Unterpfades 0 ist, dann wird die Richtung der lokalen x-Achse angenommen.

Auch hier ist Kontrolle besser:

Richtungen und Pfadlänge 0[8]

Richtungsbestimmung bei Pfadsegmenten der Länge 0. Sonst wie beim vorherigen Beispiel.

Allerdings kann auch der Mittelwert zweier Richtungen gerade 0 werden:

Richtungen und Mittelwert 0[9]

Die Richtungen an den äußeren Ecken haben den Mittelwert 0.

Genaugenommen ist das Verhalten in den betroffenen Ecken jeweils außen undefiniert. Plausibel wäre eine Ausrichtung senkrecht zum Pfad. In vielen Fällen könnte eine Analyse der höheren Ableitungen eine Richtungsinformation bringen, aber nicht in allen, wie bei dem Kreuz in der Mitte. Vermutlich wird ein Autor daher am ehesten mit einer Ausrichtung in Richtung der lokalen x-Achse rechnen müssen.

Auch das Verhalten für Z-Kommandos kann ein Problem für Darstellungsprogramme sein:

6 http://de.wikibooks.org/wiki/media%3ASVG_Richtungen01.svg
7 http://de.wikibooks.org/wiki/media%3ASVG_Richtungen02.svg
8 http://de.wikibooks.org/wiki/media%3ASVG_Richtungen03.svg
9 http://de.wikibooks.org/wiki/media%3ASVG_Richtungen04.svg

Richtungen und Z-Kommando[10]

Richtungen und Z-Kommando (2)[11]

Bei der zweiten Datei bleibt es dem Leser zur Übung überlassen, die richtigen Richtungen zu ergänzen...

Spannend ist auch die Frage, ob richtig implementiert worden ist, wenn in einer unüblichen Weise Pfadkommandos aufeinandertreffen, also schon eine erlaubte Kombination darstellen, dies aber wohlmöglich nicht ausgiebig getestet wurde:

Richtungen und ungeöhnliche Kommandokombinationen in Pfaden[12]

10 http://de.wikibooks.org/wiki/media%3ASVG_Richtungen05.svg
11 http://de.wikibooks.org/wiki/media%3ASVG_Richtungen06.svg
12 http://de.wikibooks.org/wiki/media%3ASVG_Richtungen07.svg

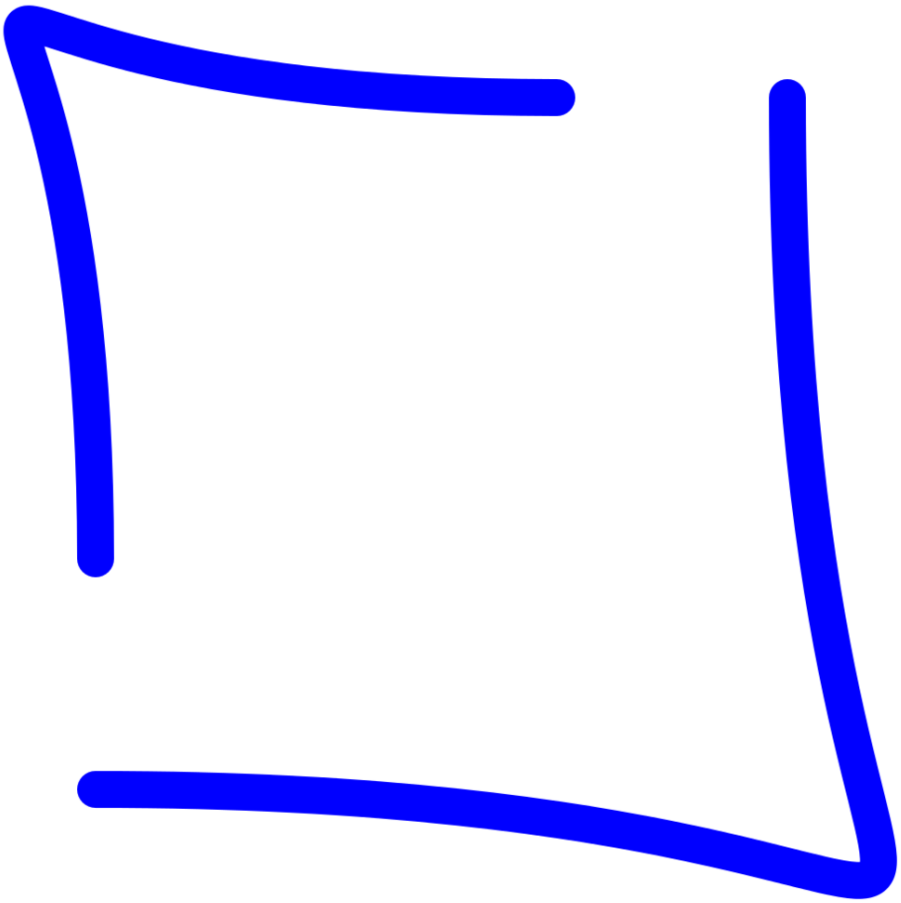


Abb. 201 *keyPoints* zur Positionierung von Markierungen

Eine weitere Möglichkeit, Markierungen an beliebigen Stellen entlang des Pfades anzubringen, wird auch im Kapitel über Animation diskutiert. Dies kann mit dem Element ***animateMotion*** und *keyPoints* geschehen: *keyPoints* zur Positionierung von Markierungen[13]

Die hellgrünen Pfeile werden mit ***animateMotion*** und *keyPoints* in gleichen Abständen entlang des Pfades positioniert. Anders als beim Einsatz von ***marker*** können mit ***animateMotion*** und *keyPoints* beliebige Positionen entlang des Pfades als Positionen für Markierungen festgelegt werden.

13 `http://de.wikibooks.org/wiki/media%3ASVGanimateMotion_keyPoints01.svg`

12.6 Weiterführende Literatur

- W3C-Referenz ***marker***[14]
- W3C - Implementierungsanmerkungen zu Pfaden und Richtungen[15]

14 `http://www.w3.org/TR/SVG11/painting.html#MarkerElement`
15 `http://www.w3.org/TR/SVG11/implnote.html#PathElementImplementationNotes`

13 Ausschnitt, Maskierung und Komposition

13.1 Ausschnitt, Maskierung und Komposition

In SVG (nicht in den tiny-Versionen) gibt es einige Möglichkeiten, den darzustellenden Ausschnitt festzulegen oder auch zwischen Darstellungen mittels einer teiltransparenten Maske überzublenden. Es entsteht also eine Komposition aus dem maskierten Element und den zuvor dargestellten Elementen.

Der Hauptunterschied zwischen einem Ausschnitt und einer Maske besteht darin, dass der Ausschnitt immer einen harten Übergang zwischen dem Bereich hat, der im Ausschnitt dargestellt wird und den Elementen, die dahinterliegen. Bei Maskierungen ist ein weicherer, teiltransparenter Übergang möglich.

Verfügbar ist in SVG eine sogenannte einfache alpha-Komposition wie folgt:

Seien

Er, Eg, Eb - Elementfarbwerte

Ea - Elementopazitätswert (alpha-Wert)

Cr, Cg, Cb - Farbwert des Hintergrundes (vor der Überblendung)

Ca - Opazitätswert des Hintergrundes (vor der Überblendung)

Cr', Cg', Cb' - Farbwert nach der Überblendung

Ca' - Opazitätswert nach der Überblendung

Dann gilt:

Ca' = 1 - (1 - Ea) * (1 - Ca)

Cr' = (1 - Ea) * Cr + Er

Cg' = (1 - Ea) * Cg + Eg

Cb' = (1 - Ea) * Cb + Eb

Die Eigenschaften *color-interpolation* und *color-rendering* bestimmen, in welchem Farbraum die Komposition stattfindet.

Zuständig für Ausschnitte ist zum einen die Eigenschaft *clip*, zum anderen kann aber mit dem Element ***clipPath*** auch ein beliebiger Pfad definiert werden, welcher mit der Eigenschaft *clip-path* referenziert wird. Analog dazu gibt es das Element ***mask*** zur Definition einer Maske und die Eigenschaft *mask* zur Referenzierung derselben.

13.2 Einfacher Ausschnitt, *clip*

Squiggle (Batik)	falsch
Opera (Presto)	9.0-9.2, ab 9.5 falsch
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

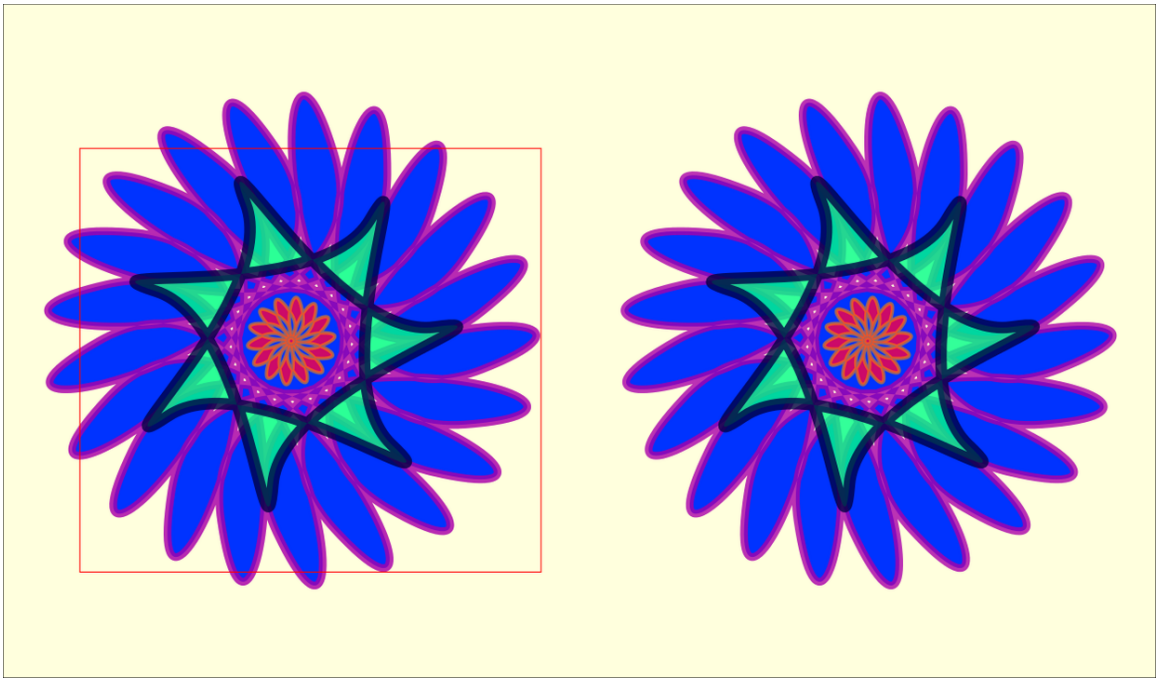


Abb. 202 Beispiel clip

Jedes Element, welches einen eigenen Anzeigebereich definiert, definiert damit auch einen Ausschnitt. Ob überstehende Inhalte angezeigt werden oder nicht, hängt dann von der Eigenschaft *overflow* ab.

Zusätzlich kann dann mit der Eigenschaft *clip* der darzustellende Bereich weiter eingeschränkt werden. Die Eigenschaft wird in CSS2.0 definiert (so wie sie für SVG 1.1 relevant ist, CSS2.1 hat leider eine dazu inkompatible Definition. Da es für CSS-Dokumente ferner keine Möglichkeit gibt anzugeben, welche Version gerade Verwendung findet, ist die CSS-Eigenschaft außerhalb von

SVG faktisch unbrauchbar, bezogen auf SVG muss aber auch mit fehlerhaften Implementierungen gerechnet werden).

Mögliche Werte sind:

inherit

geerbt

auto

automatisch festgelegt, stimmt mit dem Anzeigebereich überein

Form

Der Wert sieht so aus: rect ('oben' 'rechts' 'unten' 'links')

'oben', 'rechts', 'unten' und 'links' sind entweder Längen oder jeweils das Schlüsselwort 'auto'. Ausgehend vom Anzeigebereich geben die Werte jeweils den Versatz von der jeweiligen Seite aus an. Der darzustellende Bereich ist also oben um die Länge 'oben' verkleinert, dann rechts um 'rechts', unten um 'unten' und links um 'links'. In den Beispielen in CSS2.0 ist auch eine Notation mit Kommata angegeben, ebenso wie in CSS2.1 (wo die Werte aber eine andere Bedeutung haben; die Komma-Notation zusammen mit der Bedeutung von CSS2.0 funktioniert in Batik 1.7).

Der Anfangswert ist 'auto', die Eigenschaft wird nicht vererbt und ist animierbar.

Aufgrund der CSS-Definitionsprobleme ist eher zu empfehlen, statt dieser Eigenschaft ***clipPath*** zu verwenden.

Beispiel clip[1]

clip wird auf ein Symbol angewendet. Selbiges wird rechts ohne ***clip*** dargestellt, links mit. Das komplette Symbol ist zudem links blass hinterlegt. Der korrekte Ausschnitt ist rot gerandet dargestellt.

13.3 Ausschnitt, *clipPath* und *clip-path* und *clip-rule*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Die Idee bei der Definition eines Ausschnittes ist, dass die Kindelemente von ***clipPath*** eine 1-Bit-Maske definieren. Die Pfade selbst ohne Berücksichtigung von *fill*, *stroke* und *stroke-width* definieren die Silhouette, welche zur Maskierung dient. Diese Maske wird mittels *clip-path* von dem Element referenziert, auf welches sie angewendet werden soll.

clipPath erbt keine Ausschnittspfade von Elternelementen. ***clipPath*** und die Kindelemente können selbst wiederum Ausschnitte mittels *clip-path* verwenden. Dabei wird der Ausschnitt zunächst auf

1 `http://de.wikibooks.org/wiki/media%3ASVGclip01.svg`

das jeweilige Element angewendet. Das Ergebnis davon trägt dann zum übergeordneten Ausschnitt bei. Bei der Anwendung von *clip-path* auf ***clipPath*** wird die Schnittfläche von beidem als Ausschnitt verwendet.

13.3.1 Element *clipPath*

clipPath ist also ein Gruppierungselement, mit dem Elemente zu einer 1-Bit-Maske zusammengefasst werden. ***clipPath*** wird nicht direkt dargestellt, sollte also in der Regel innerhalb vom Element ***defs*** notiert werden.

Eigenschaften vererben sich von den Elternelementen in das Element ***clipPath***, nicht vom referenzierenden Element.

Attribut *clipPathUnits*

Mit dem Attribut *clipPathUnits* wird festgelegt, welches Koordinatensystem für die 1-Bit-Maske gilt. Mögliche Werte sind 'userSpaceOnUse' und 'objectBoundingBox', ersteres wird angenommen, wenn das Attribut nicht angegeben ist. Das Attribut ist animierbar.

Bei 'userSpaceOnUse' gilt das lokale Koordinatensystem des referenzierenden Elementes, bei 'objectBoundingBox' jenes Koordinatensystem, welches aus der das referenzierende Element umgebenden Box resultiert.

13.3.2 Eigenschaft *clip-path*

Mit der Eigenschaft *clip-path* wird ein Element ***clipPath*** referenziert, um einen Ausschnitt festzulegen.

Mögliche Werte sind:

inherit

geerbt

none

Kein Ausschnit

Funktionale URI

Referenzierung eines Elementes ***clipPath*** im gleichen Dokument in der Form url(#Ausschnitt), wobei 'Ausschnitt der Wert des Attributes *id* des zu referenzierden Elements ***clipPath*** ist. Wird etwas anderes referenziert, ist dies ein Fehler und führt zum Abbruch der Darstellung.

Der Initialwert ist 'none', die Eigenschaft wird nicht vererbt und ist animierbar.

13.3.3 Eigenschaft *clip-rule*

Analog zur Eigenschaft *fill-rule* wird mit *clip-rule* festgelegt, wie Innen und Außen für den Ausschnitt festgelegt wird, was also dargestellt wird (Innen) und was weggelassen wird (Außen).

Mögliche Werte sind die gleichen mit gleicher Bedeutung wie für *fill-rule*:

inherit

geerbt

nonzero

Festlegung nach der Regel 'nonzero'

evenodd

Festlegung nach der Regel 'evenodd'

Initialwert ist 'nonzero', die Eigenschaft wird vererbt und ist animierbar.

13.3.4 Beispiele - Ausschnitt

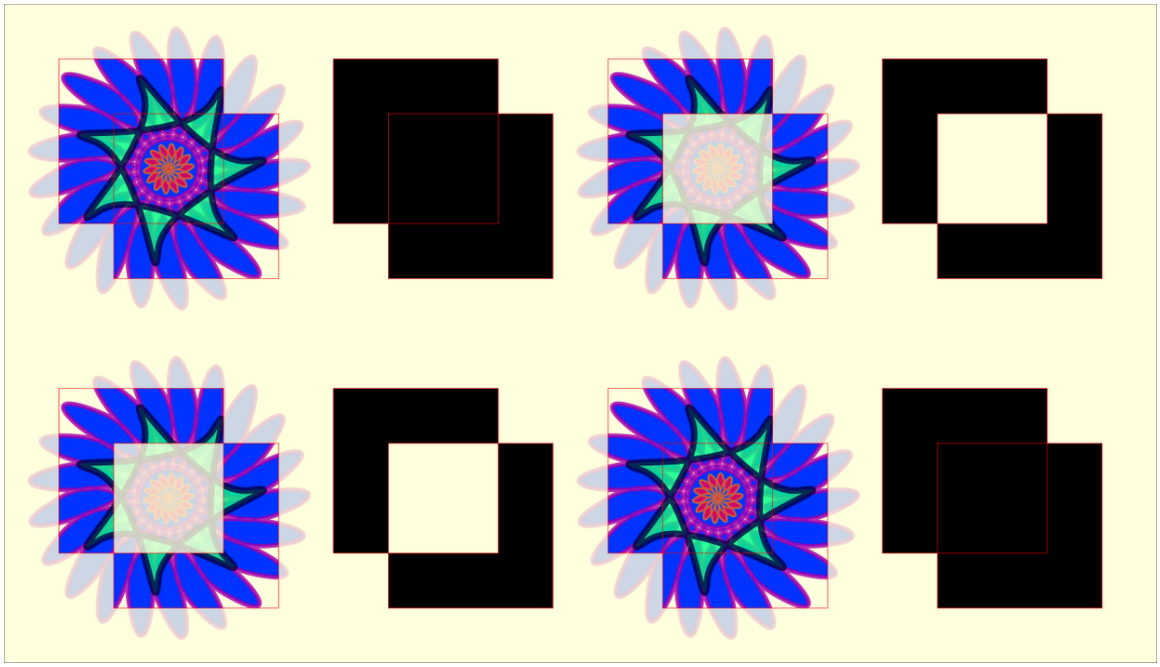


Abb. 203 ***clipPath*** mit *clipPathUnits* 'userSpaceOnUse'

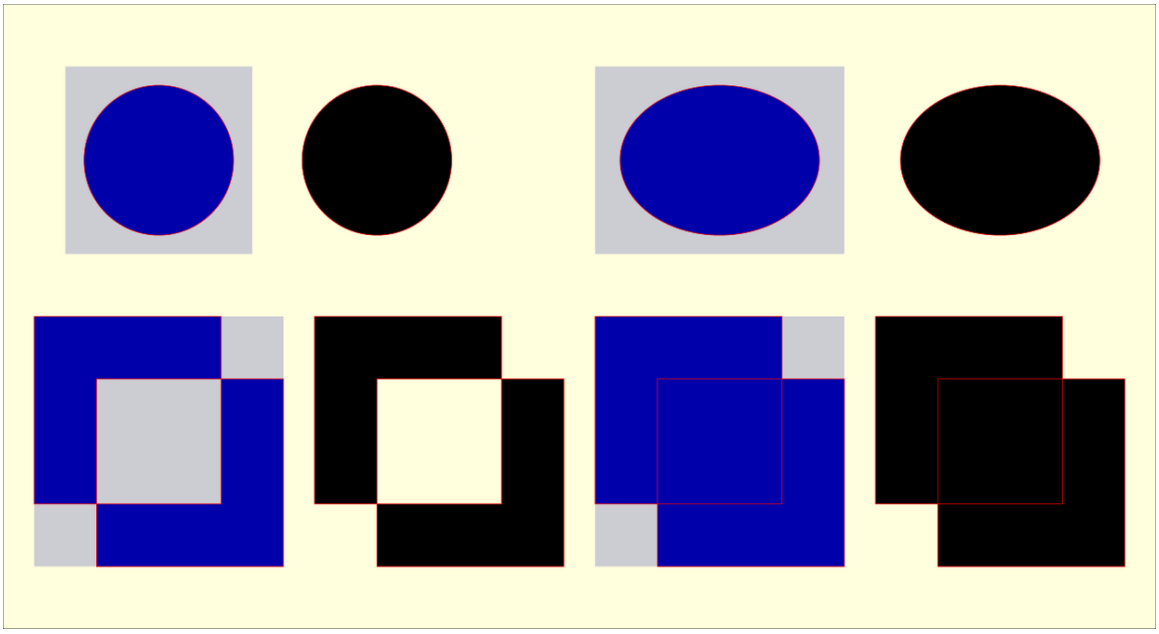


Abb. 204 ***clipPath*** mit *clipPathUnits* 'objectBoundingBox'

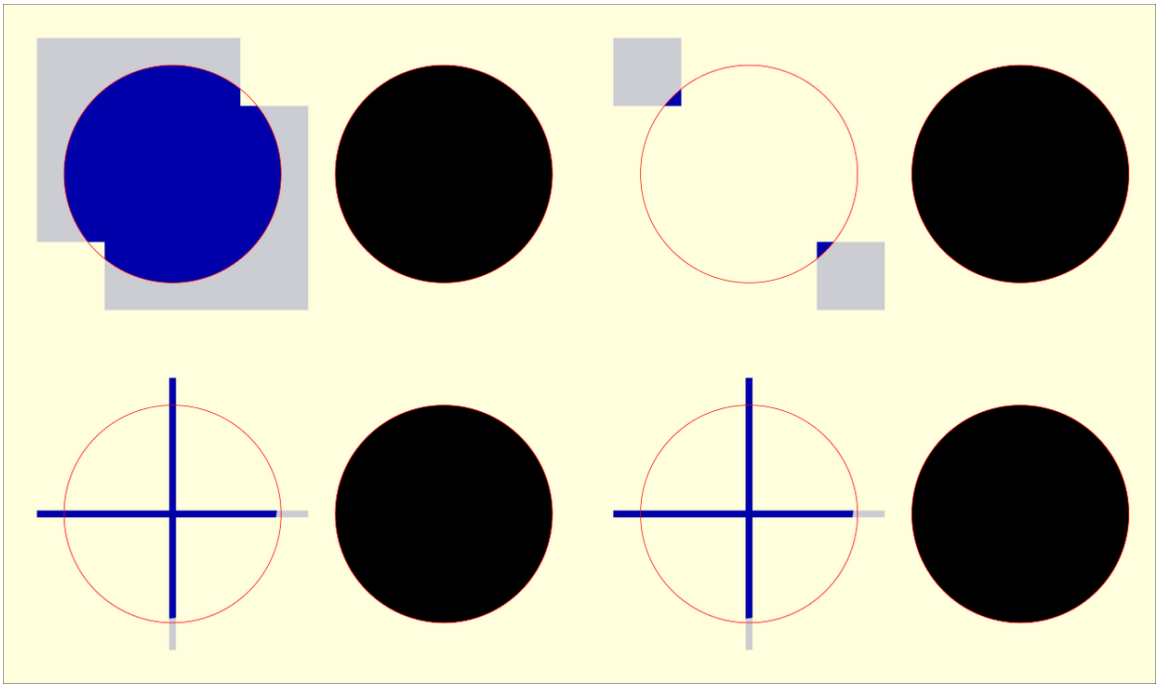


Abb. 205 Gruppen, ***clipPath*** mit *clipPathUnits* 'objectBoundingBox'

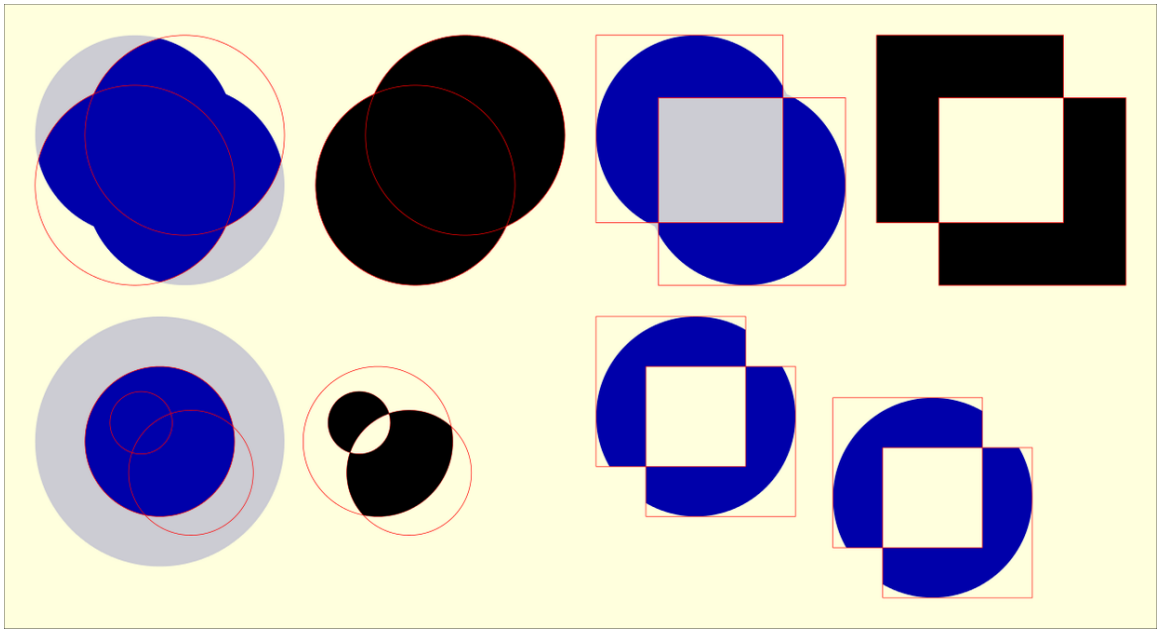


Abb. 206 Verschiedene kompliziertere Variationen zu clipPath

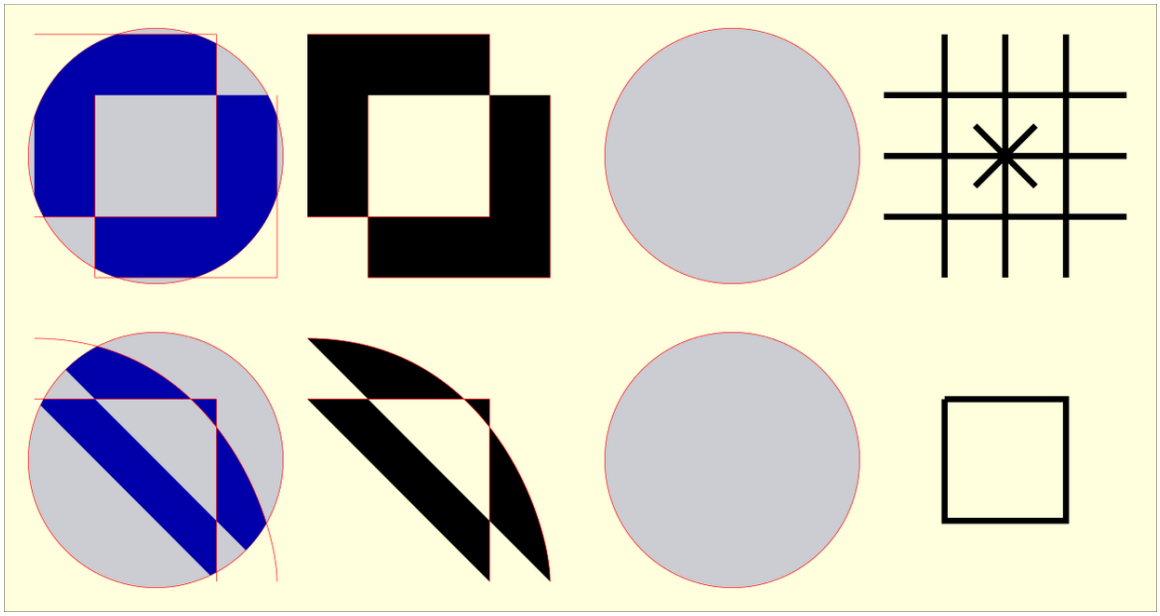


Abb. 207 Verschiedene offene Pfade und clipPath.

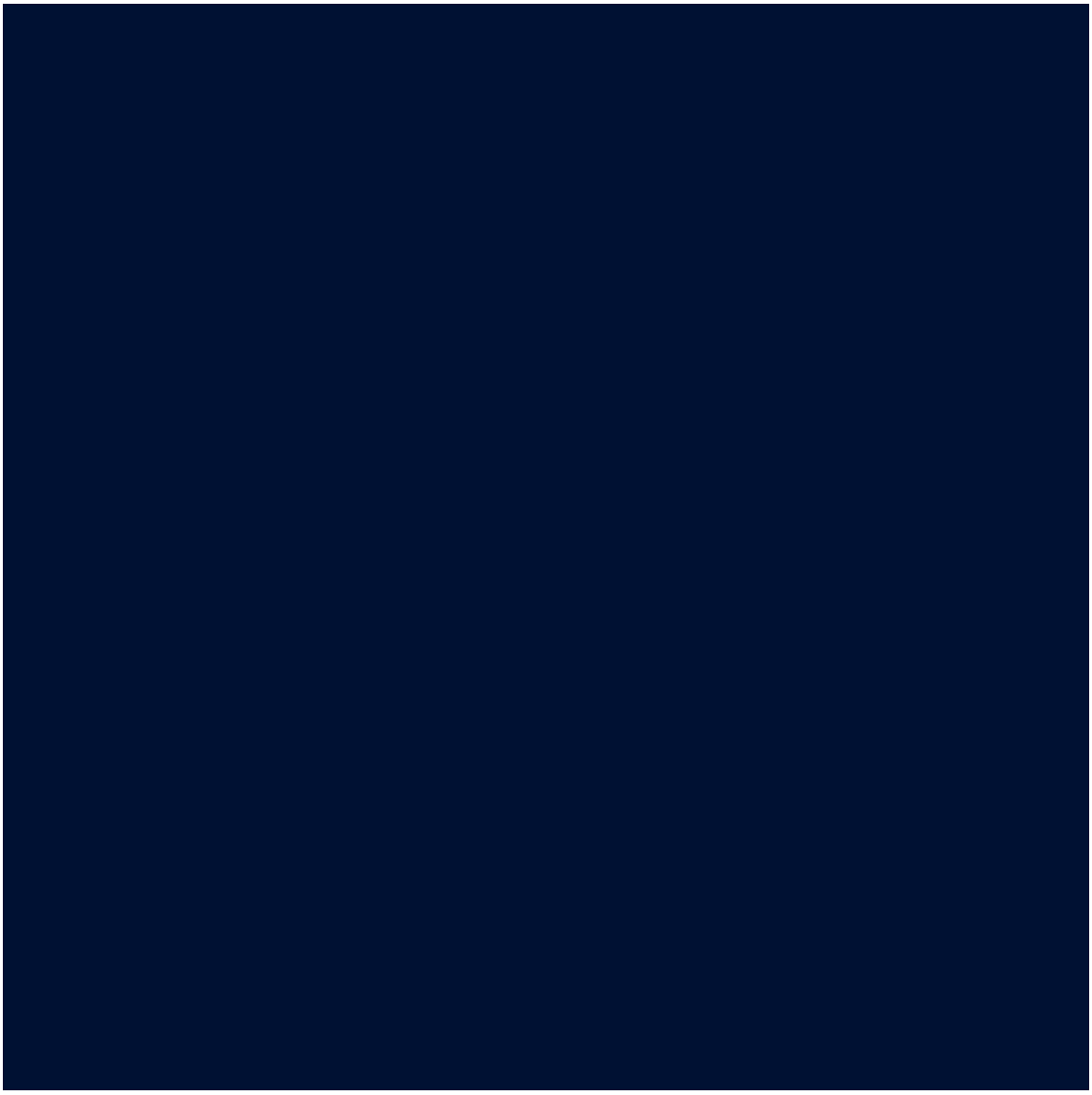


Abb. 208 Animierter Ausschnitt

Einige einfache Beispiele zum Ausschnitt:

Sinngemäß wird der gewünschte Ausschnittspfad im Element ***clipPath*** notiert und dies wird mit der Eigenschaft *clip-path* beim gewünschen Element notiert:

```
<defs>
<clipPath id="clip0">
<circle cx="300" cy="300" r="150" />
</clipPath>
</defs>
<rect x="100" y="100" width="300" height="300"
      fill="#00a" clip-path="url(#clip0)" />
```

clipPath mit *clipPathUnits* 'userSpaceOnUse'[2]

Verschiedene Variationen, ***clipPath*** mit implizierten *clipPathUnits* 'userSpaceOnUse' zu verwenden.

Das komplette Symbol ist zudem blass hinterlegt. Der korrekte Ausschnitt ist rot gerandet dargestellt und rechts daneben als schwarze Silhouette.

clipPath mit *clipPathUnits* 'objectBoundingBox'[3]

Verschiedene Variationen, ***clipPath*** mit *clipPathUnits* 'objectBoundingBox'.

Das komplette Symbol ist zudem blass hinterlegt. Der korrekte Ausschnitt ist rot gerandet dargestellt und rechts daneben als schwarze Silhouette.

Gruppen, ***clipPath*** mit *clipPathUnits* 'objectBoundingBox'[4]

Sofern der Ausschnitt auf ein Gruppierungselement wie ***g*** angewendet wird, wird aus den Kindelementen eine objectBoundingBox berechnet, so dass diese alle Kindelemente umschließt. Offenbar ist dies auch möglich, wenn die Kindelemente selbst gar keine eigene objectBoundingBox haben oder die Gruppe per ***use*** referenziert wird.

Verschiedene kompliziertere Variationen zu clipPath.[5]

Links ist der Inhalt von ***clipPath*** ein ***use***-Element, welches eine Gruppe mit zwei Kreisen referenziert.

Rechts wird *clip-path*="inherit" getestet, unten drunter sind die geerbten ausgeschnittenen Inhalte nebeneinander zum Vergleich angegeben.

Links unten wird innerhalb von ***clipPath*** ein weiterer ***clipPath*** referenziert, die Schnittfläche wird also als Ausschnitt verwendet.

Verschiedene offene Pfade und clipPath.[6]

Nur wenn ein Pfad eine Fläche aufspannt, kann diese auch einen sichtbaren Ausschnitt definieren. Links sind zwei Beispiele zu sehen, die Flächen aufspannen, rechts zwei, bei denen keine Fläche aufgespannt wird. Folglich wird rechts nach der Maskierung nichts angezeigt (nur die teiltransparent angedeutete nicht maskierte Form.

Animierter Ausschnitt[7]

Ein animierter clipPath in Form eines Kreises, dessen Position und Radius sich ändert, wird verwendet, um einen Ausschnitt auf eine Gruppe mit siebenzähliger Drehsymmetrie, aber ohne Spiegelachse anzuwenden.

2 http://de.wikibooks.org/wiki/media%3ASVGclipPath01.svg
3 http://de.wikibooks.org/wiki/media%3ASVGclipPath02.svg
4 http://de.wikibooks.org/wiki/media%3ASVGclipPath03.svg
5 http://de.wikibooks.org/wiki/media%3ASVGclipPath04.svg
6 http://de.wikibooks.org/wiki/media%3ASVGclipPath05.svg
7 http://de.wikibooks.org/wiki/media%3ASVGclipPath06.svg

Ein rot gerandeter Kreis gibt die Position des Ausschnittes an.

Abb. 209 Flagge, auf die der Auschneidepfad angewendet wird

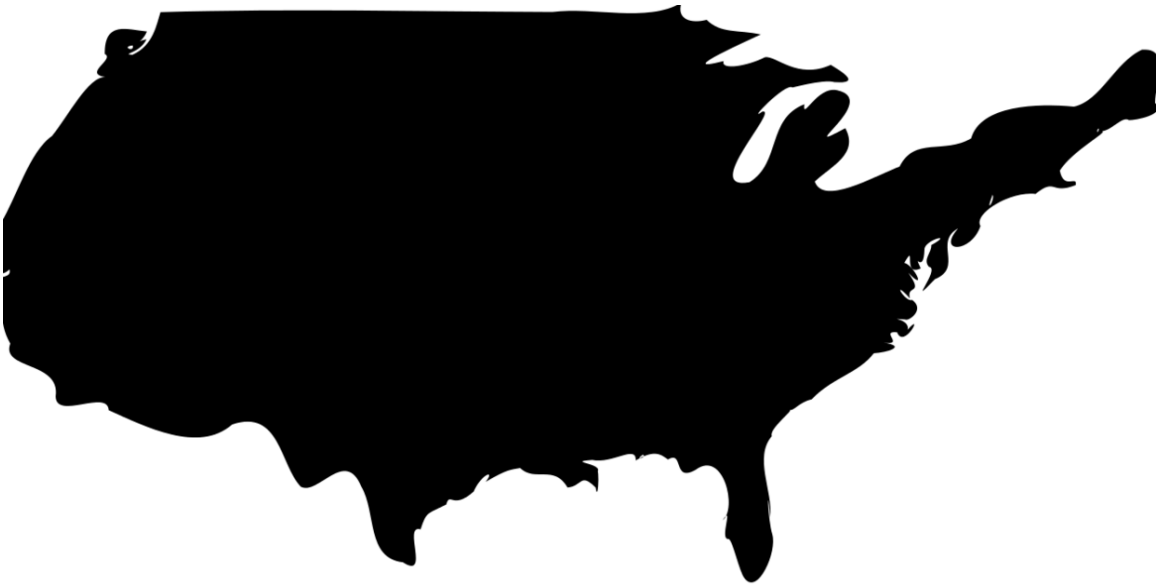


Abb. 210 Pfad

Abb. 211 Ergebnis

Als etwas umfangreicheres Beispiel ist rechts die Erstellung einer Vorlage für einen amerikanisch-patriotischen Aufkleber zu sehen:

Die Flagge der Vereinigten Staaten von Amerika wird als Vorlage verwendet, von der ein Ausschnitt darzustellen ist...

Die geographischen Umrisse der USA werden als Ausschnittspfad verwendet....

Beides zusammengenommen, innerhalb eines Elementes ***clipPath*** mit der Hilfe des Elementes ***use*** den Umriß referenziert und den Ausschneidepfad mit der Hilfe der Eigenschaft *clip-path* für die Gruppierung mit der Flagge angewendet, ergibt eine Vorlage für einen Aufkleber, zum Beispiel für ein amerikanisch-patriotisches Motorrad.

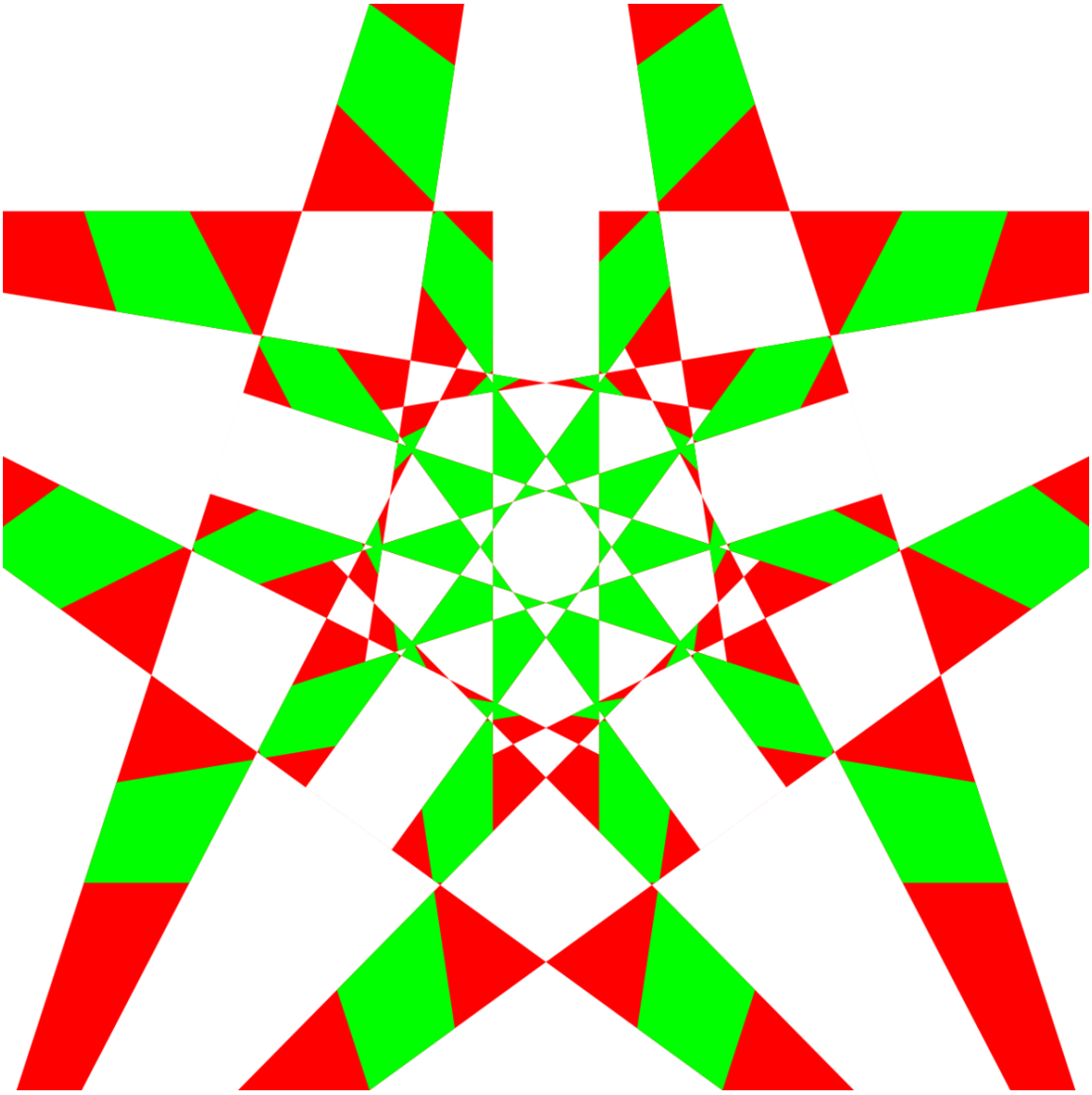


Abb. 212 ***clipPath*** - komplexeres Beispiel

Mit ***clipPath*** lassen sich auch komplexe Überlagerungen erzeugen:

```
<?xml version="1.0" standalone="no"?>
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     version="1.1" width="1000" height="1000"
     viewBox="0,0 1,1">
<defs>
<g id="pentagram">
<g id="cone">
  <polyline id="triangle" points="0,0 0,1 .5,1"
            transform="translate(0,-1) rotate(18)"/>
  <use xlink:href="#triangle" transform="scale(-1,1)"/>
</g>
<use xlink:href="#cone" transform="rotate(72)"/>
<use xlink:href="#cone" transform="rotate(-72)"/>
<use xlink:href="#cone" transform="rotate(144)"/>
```

```
<use xlink:href="#cone" transform="rotate(-144)"/>
</g>
<clipPath id="starfoo">
  <use xlink:href="#pentagram" />
</clipPath>
</defs>

<g transform="translate(0.5,0.5)" clip-path="url(#starfoo)" >
<circle r="1" fill="red"/>
<circle r="1" clip-path="url(#starfoo)" fill="green"
        transform="rotate(36)"/>
</g>
</svg>
```

W3C clipPath Referenz[8]

13.4 Maskierung, *mask* und *mask*

Squiggle (Batik)	1.7 (falscher Farbraum)
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3.5 (falscher Farbraum)
Konqueror (KSVG)	3.2 (falscher Farbraum)
Safari (Webkit)	stark variierende Fehler in verschiedenen Versionen
Chrome (Webkit)	7
Microsoft Internet Explorer (Trident)	
librsvg	

Der resultierende alpha-Kanal (die Opazität) der Kindelemente des Elementes ***mask*** definiert eine Maske, welche angewendet wird auf jenes Element, welches die Maske mit der Eigenschaft *mask* referenziert.

13.4.1 Element *mask*

Der Inhalt von ***mask*** sind Gruppierungselemente oder graphische Elemente. Die Maskierung funktioniert so, dass diese Element in einem gedachten, nicht direkt dargestellten Bereich mit schwarztransparentem Hintergrund gemalt werden. Die so gemalten Elemente stellen also die Bereiche dar, die ganz oder teilweise maskiert werden. Der RGBA-Wert wird nach einer Formel für den Filter ***feColorMatrix*** für 'luminance-to-alpha' in einen Grauwert konvertiert. Der Grauwert weiß entspricht komplett undurchsichtig und schwarz komplett durchsichtig.

mask wird nicht direkt dargestellt, sollte also in der Regel innerhalb vom Element ***defs*** notiert werden.

Eigenschaften vererben sich von den Elternelementen in das Element ***mask***, nicht vom referenzierenden Element.

8 http://www.w3.org/TR/SVG/masking.html#ClippingPaths

Attribut *maskContentUnits*

Mit dem Attribut *maskContentUnits* wird festgelegt, welches Koordinatensystem für den Maskeninhalt gilt. Mögliche Werte sind 'userSpaceOnUse' und 'objectBoundingBox', ersteres wird angenommen, wenn das Attribut nicht angegeben ist. Das Attribut ist animierbar.

Bei 'userSpaceOnUse' gilt das lokale Koordinatensystem des referenzierenden Elementes, bei 'objectBoundingBox' jenes Koordinatensystem, welches aus der das referenzierende Element umgebenden Box resultiert.

Attribut *maskUnits*

Mittels *maskUnits* wird festgelegt, welches Koordinatensystem für die folgenden Attribute *x*, *y*, *width*, *height* gilt. Mögliche Werte sind 'userSpaceOnUse' und 'objectBoundingBox', letzteres wird angenommen, wenn das Attribut nicht angegeben ist. Das Attribut ist animierbar.

Bei 'userSpaceOnUse' gilt das lokale Koordinatensystem des referenzierenden Elementes, bei 'objectBoundingBox' jenes Koordinatensystem, welches aus der das referenzierende Element umgebenden Box resultiert.

Attribute *x* und *y* und den Längen *width* und *height*

Mit den Attributen *x* und *y* und den Längen *width* und *height* werden Position (links oben) und Größe (Höhe und Breite) des Puffers der Maske angegeben. Alle Attribute sind animierbar. Nur was innerhalb des Puffers liegt, beziehungsweise in der Schniffläche mit einem möglichen Ausschnitt durch *clip-path*, dient auch als Maske.

Ist ein Attribut nicht angegeben, wird für das fehlende eine Voreinstellung angnommen, für *x* und *y* '-10%' und für *width* und *height* '120%'. Ein negativer Wert für *width* oder *height* ist ein Fehler und führt zum Abbruch der Darstellung.

13.4.2 Eigenschaft *mask*

Mit der Eigenschaft *mask* wird ein Element ***mask*** referenziert, um einen Ausschnitt festzulegen.

Mögliche Werte sind:

inherit

geerbt

none

Kein Ausschnit

Funktionale URI

Referenzierung eines Elementes ***mask*** in der Form url(#Ausschnitt), wobei 'Ausschnitt der Wert des Attributes *id* des zu referenzierden Elements ***mask*** ist. Wird etwas anderes referenziert, ist dies ein Fehler und führt zum Abbruch der Darstellung.

Der Initialwert ist 'none', die Eigenschaft wird nicht vererbt und ist animierbar.

13.4.3 Maskierung - Beispiele

Abb. 213 Vorlage mit patriotischem Aufkleber mit Maske

Rechts ist ein Beispiel einer Vorlage zum patriotischen Aufkleber entsprechend dem Beispiel zu ***clipPath*** zu sehen. Es wird ein Farbverlauf als Maske verwendet, letztlich mit dem Ergebnis einer teilweisen Abschattung der Kombination von Karte und Flagge.

Nach Umwandlung der graphischen Objekte in Grauwerte einer Maske, gilt grob, je heller der Grauwert, desto mehr ist vom Objekt zu sehen, je dunkler, desto mehr vom Hintergrund.

Veranschaulichung der Maskierung[9]

Hintergrund, Objekt, Maske und das Ergebnis der Maskierung werden übereinander getrennt dargestellt, um den Effekt der Maskierung verständlicher zu machen. Zum Vergleich ist ferner ein für dieses einfache Beispiel äquivalenter Farbverlauf angegeben.

9 `http://de.wikibooks.org/wiki/media%3ASVGmask01.svg`

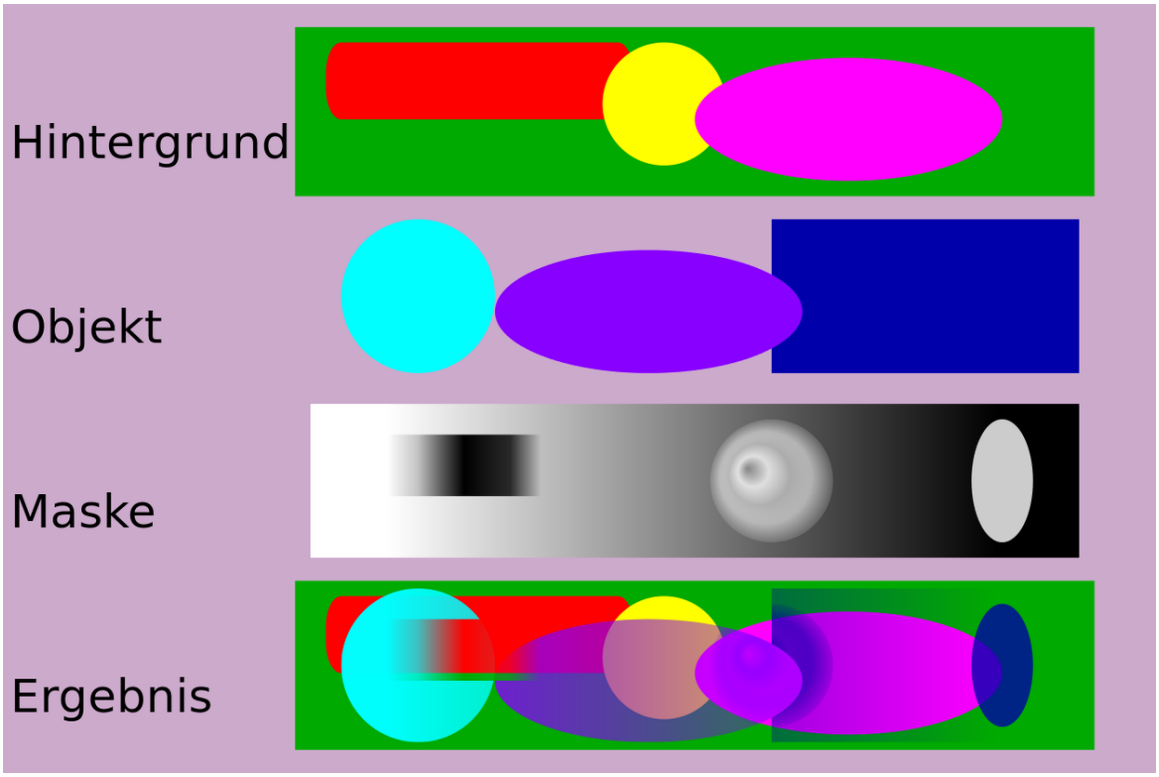


Abb. 214 Veranschaulichung der Maskierung (2)

Veranschaulichung der Maskierung (2)[10]

Gut zu sehen ist auch eine Möglichkeit, mit einer geeigneten Maske ein Objekt aus dem Hintergrund teilweise oder auch ganz in den Vordergrund zu holen. Dies kombiniert mit Animationen ist eine Möglichkeit, scheinbar die Zeichenreihenfolge zu ändern.

10 http://de.wikibooks.org/wiki/media%3ASVGmask02.svg

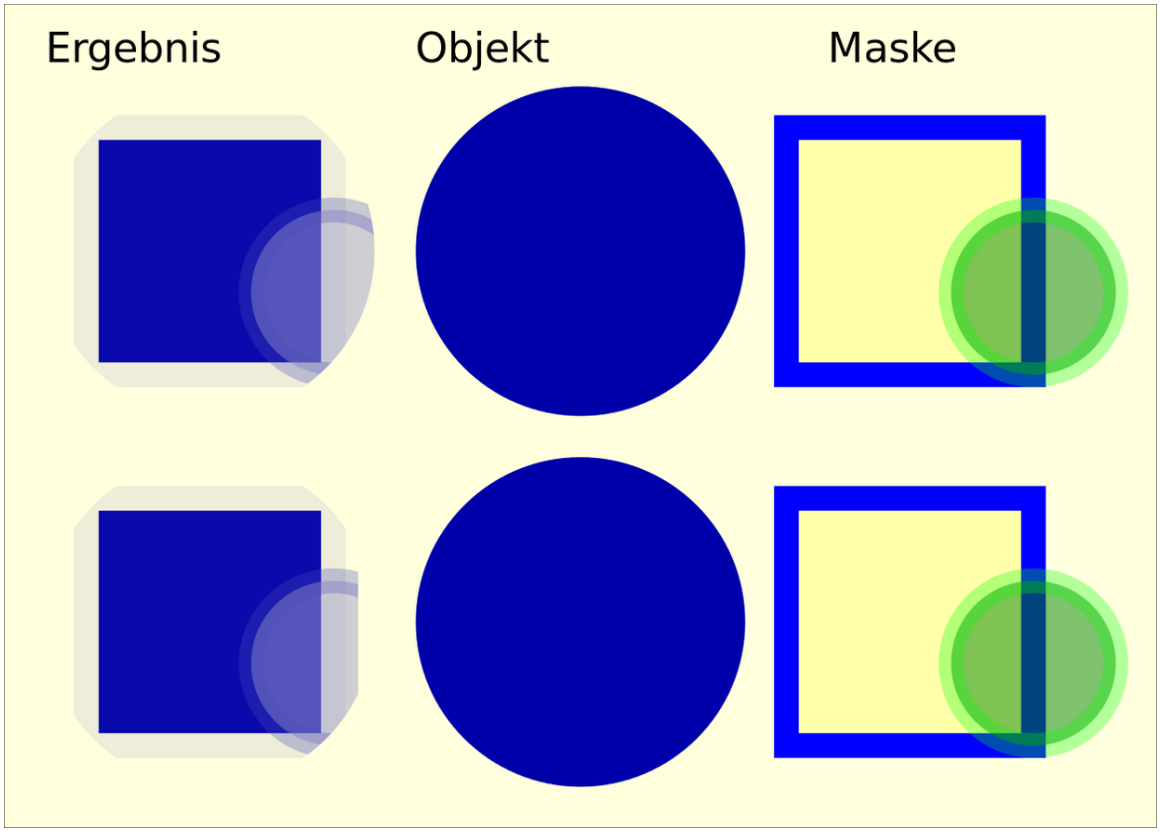


Abb. 215 Verwendung von *maskUnits* und *maskContentUnits*

Verwendung von *maskUnits* und *maskContentUnits*[11]

Beispiele zu Verwendung von *maskUnits* 'userSpaceOnUse' (oben) und 'objectBoundingBox' (unten).

In beiden Fällen werden *maskContentUnits* 'userSpaceOnUse' verwendet und die Masken sind so gewählt, dass jeweils das gleiche Ergebnis erzielt wird.

Die Maske ist rechts dargestellt, das Objekt, auf welche sie angewendet wird, in der Mitte und das Ergebnis links.

11 `http://de.wikibooks.org/wiki/media%3ASVGmask03.svg`

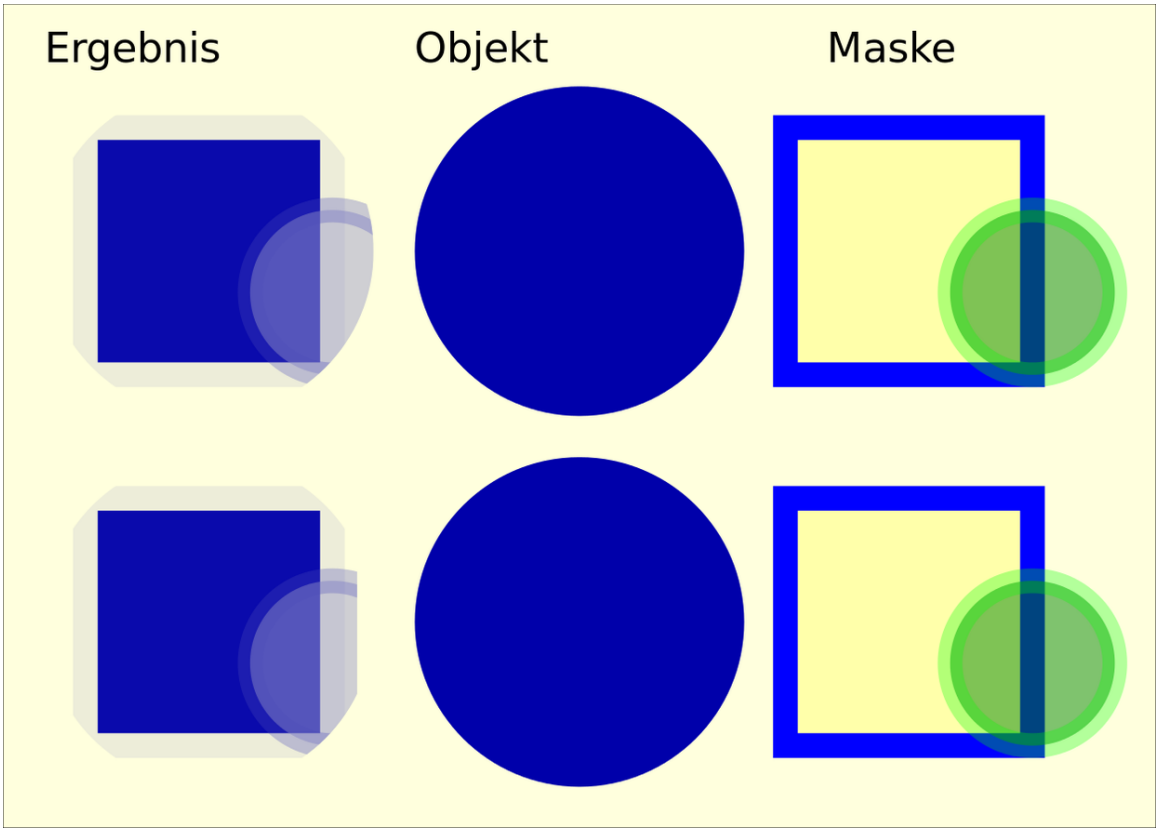


Abb. 216 Verwendung von maskUnits und maskContentUnits (2)

Verwendung von maskUnits und maskContentUnits (2)[12]

Beispiele zu Verwendung von *maskUnits* 'userSpaceOnUse' (oben) und 'objectBoundingBox' (unten).

In beiden Fällen werden *maskContentUnits* 'objectBoundingBox' verwendet und die Masken sind so gewählt, dass jeweils das gleiche Ergebnis erzielt wird.

Die Maske ist rechts dargestellt, das Objekt, auf welche sie angewendet wird, in der Mitte und das Ergebnis links.

12 `http://de.wikibooks.org/wiki/media%3ASVGmask04.svg`

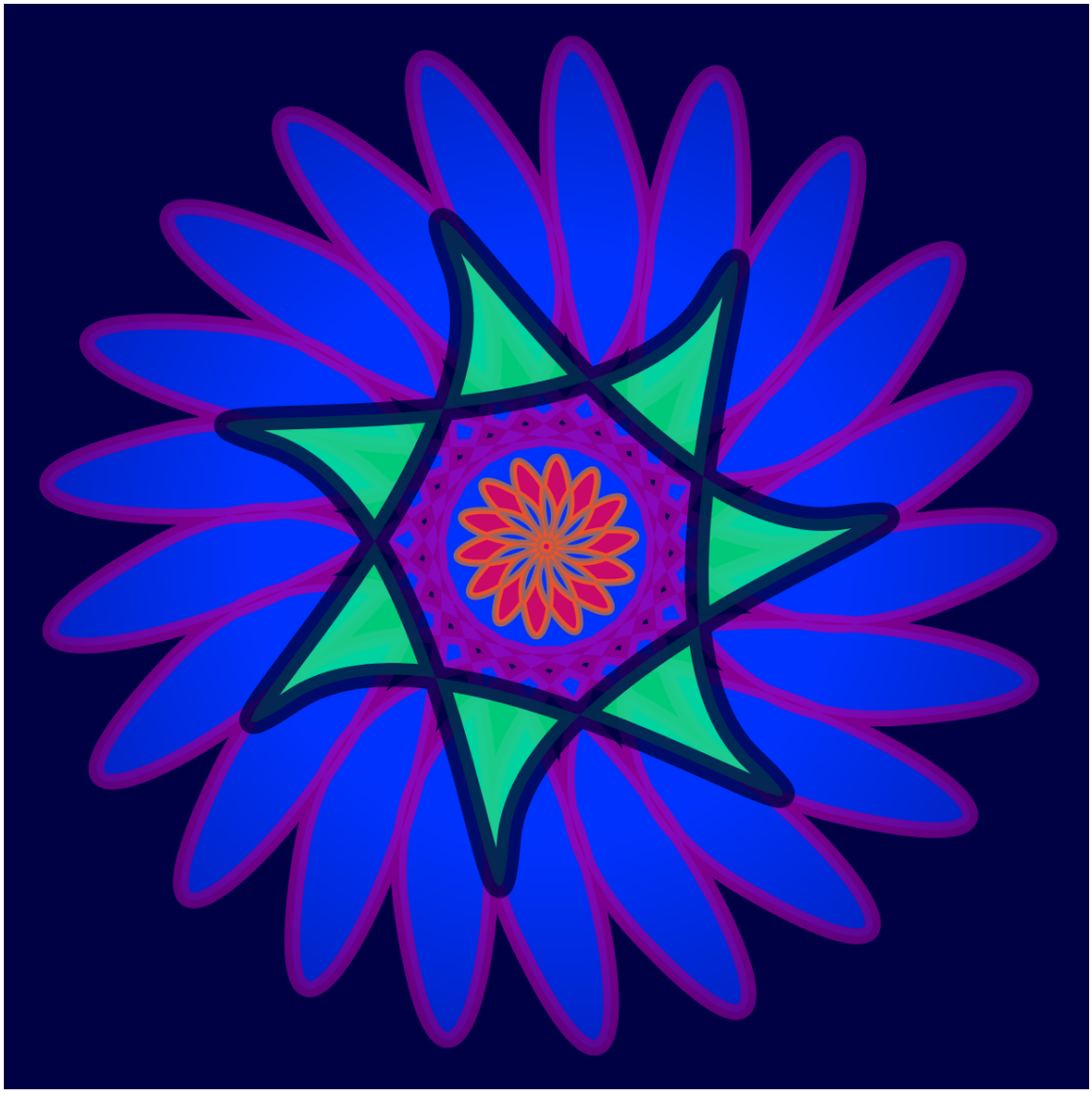


Abb. 217 Animierte Maske

Animierte Maske[13]

Eine animierte Maske mit radialen Farbverlauf, dessen Position und Radius sich ändert, wird verwendet, um einen Ausschnitt auf eine Gruppe mit siebenzähliger Drehsymmetrie, aber ohne Spiegelachse anzuwenden.

W3C Mask Referenz[14]

13 http://de.wikibooks.org/wiki/media%3ASVGmask05.svg
14 http://www.w3.org/TR/SVG/masking.html#Masking

14 Animation

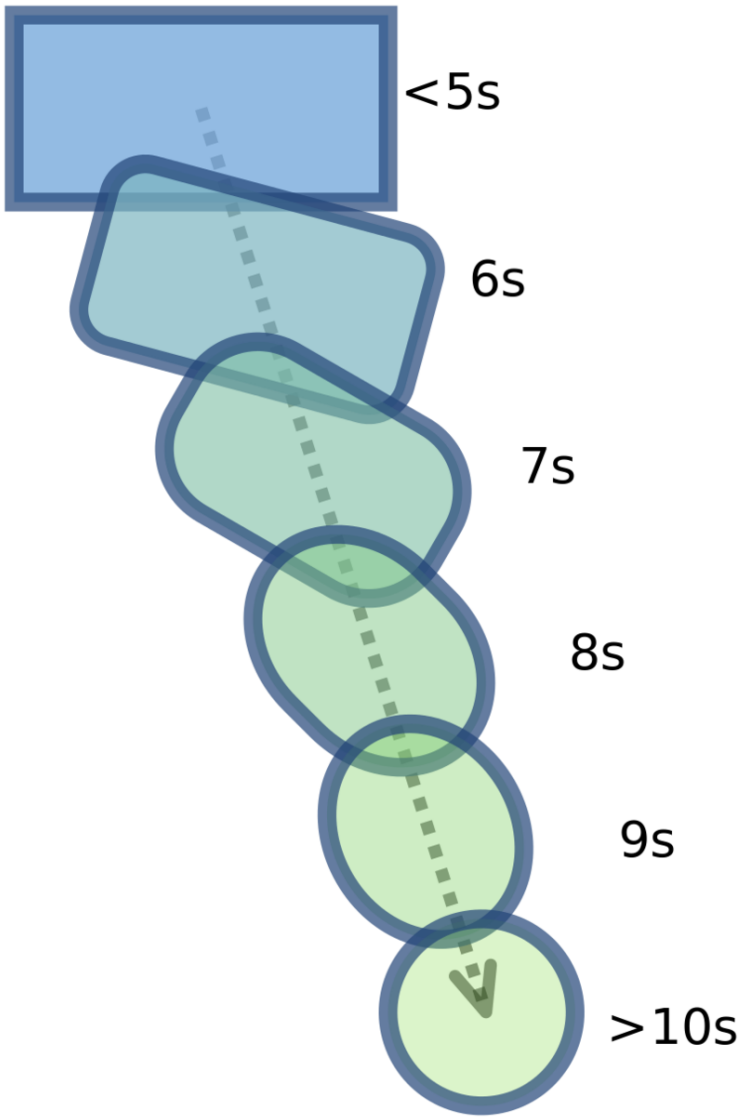


Abb. 218 Beispiel und schematische Darstellung einer Animation in SVG[a]: Gleichzeitige Animation von Breite, Radius der abgerundeten Ecken, Position, Füllfarbe und Füllopazität eines Rechteckes über 5 Sekunden.

a `http://de.wikibooks.org/wiki/media%3ASVG-Animation%20Beispiel01.svg`

14.1 Animation, Grundlagen

Als Animation versteht sich in SVG die zeitliche Änderung eines Attributes oder einer Eigenschaft eines Elementes oder die Änderung der Position von Elementen im aktuellen Koordinatensystem. In der graphischen Präsentation entspricht dies einer Änderung dieser Präsentation. Eigenschaften und Attribute können unabhängig voneinander animiert werden. Jede dieser Animationen hat einen eigenen Zeitablauf.

Die in SVG verwendete Animationsform nennt sich auch deklarative Animation und basiert auf SMIL, einer anderen XML-Sprache. Art und Notation werden von SMIL übernommen. In einigen speziell für SVG relevanten Teilen wird SMIL durch SVG erweitert. Bei der deklarativen Animation wird der Zeitablauf formal im Dokument beschrieben, es ist Aufgabe des Darstellungsprogrammes, aus dieser Beschreibung eine zeitabhängige Darstellung zu generieren.

Dies unterscheidet die deklarative Animation von der sogenannten rahmenbasierten Animation (englisch: frame-based), bei der der Autor das gewünschte Ergebnis erreicht, indem er für die gesamte Szenerie Einzelbilder erstellt, sogenannte Rahmen, die dann in der Darstellung mit definierten Anzeigedauern aufeinander folgen. Ein Unterschied ist auch gegeben zur Skriptanimation, bei der jeweils jede Änderung eines Attributes oder einer Eigenschaft im Skript einzeln und explizit angegeben wird und nach einem gewissen Zeitabstand eine weitere Änderung durch das Skript erfolgt.

Deklarative Animation ermöglicht diesen Methoden gegenüber eine Angabe der Interpolation über den gesamten Zeitablauf, neben der diskreten Änderung auch verschiedene Interpolationsmethoden für kontinuierliche Änderungen.

Die Referenz des W3C zu SMIL-Animation wie für SVG1.1 anwendbar ist in einer speziellen Empfehlung zu finden: SMIL-Animation[1].

In SVG tiny 1.2 wird verwiesen auf SMIL 2.1[2].

Die aktuelle Version (mit einigen Feinkorrekturen) ist SMIL 3.0[3].

In SVG selbst gibt es nur einige kurze Erklärungen der verwendeten Elemente und Attribute und der für SVG spezifischen Erweiterungen: Animation in SVG 1.1[4], Animation in SVG tiny 1.2[5].

Darüberhinaus verweist SVG jeweils auf SMIL. Die Kapitel in den SVG-Empfehlungen reichen allein nicht aus, um deklarative Animationen im Detail zu verstehen, dazu ist SMIL notwendig.

Weil aktuell (2010) verfügbare Darstellungsprogramme den Bereich Animation - wie andere Bereiche von SVG auch - nur lückenhaft interpretieren, kann es sich für Autoren als recht nützlich erweisen, zu recherchieren, ob eine bestimmte Funktionalität für die eigenen Zwecke ausreichend gut implementiert ist. Die SVG-Arbeitsgruppe bietet für alle Bereiche von SVG Tests an: Überblick,

1 http://www.w3.org/TR/smil-animation/
2 http://www.w3.org/TR/2005/REC-SMIL2-20051213
3 http://www.w3.org/TR/SMIL3/
4 http://www.w3.org/TR/SVG11/animate.html
5 http://www.w3.org/TR/SVGMobile12/animate.html

W3C-Testsuites[6] (englisch).

Ein Autor dieses Buches bietet auch eine systematische Sammlung von Beispielen und Tests zur Animation mit SVG[7] an (deutsch und englisch).

Diese und andere Quellen von Beispielen und Anwendungen können zum Lernen sehr nützlich sein, auch weil das Thema Animation sehr umfangreich ist und nicht immer ganz einfach zu verstehen - weder für Autoren von Dokumenten, noch für Programmierer der Darstellungsprogramme.

Für umfangreichere Anwendungen bietet es sich auch an, nachdem Kenntnisse in deklarativer Animation vorhanden sind, mit eigenen Skripten oder Programmen die Werte für den Ablauf einer Animation berechnen zu lassen. Dafür eignet sich jede Sprache mit einer Textausgabe.

14.1.1 Elemente

SVG übernimmt von der SMIL-Animation die Elemente ***set***, ***animate***, ***animateColor*** und ***animateMotion***. Speziell für Transformationen wird in SVG ***animateTransform*** eingeführt. Zusätzlich definiert SVG auch noch erweiterte Funktionalitäten für ***animateMotion***, mit welchem Elemente im aktuellen Koordinatensystem bewegt werden können. Anders als die anderen Animationselemente animiert ***animateMotion*** kein spezifisches Attribut oder eine Eigenschaft, sondern wird als unabhängige Möglichkeit einer Translation definiert. ***animateColor*** dient speziell der Animation von Farben. Mittels ***set*** steht eine einfache Methode zur Verfügung, ohne weitere Interpolation einen neuen Wert für ein Attribut oder eine Eigenschaft zu setzen, auch Farben. Ausgenommen sind davon in den aktuellen Versionen allerdings Transformationsattribute wie *transform*, für welche nur ***animateTransform*** verfügbar ist. ***animate*** ist ein allgemeines Element zur Animation, wie ***set*** mit der Ausnahme von Transformationsattributen.

Die genannten Elemente werden im Weiteren zusammengefasst zu Animationselementen. Animationselemente gehören zur allgemeineren Klasse der zeitabhängigen Elemente, von denen es in SMIL und in SVG tiny 1.2 noch einige weitere gibt.

Attribute, welche für Animationselemente spezifisch sind und somit also der näheren Beschreibung einer Animation dienen, seien im weiteren Animationsattribute genannt. Animationsattribute sind selbst allesamt nicht animierbar.

14.1.2 Methoden und Prinzip

Wie ein Darstellungsprogramm eine zeitlich abhängige Präsentation erreicht, ist in SVG nicht festgelegt. Es gelten nur wie für jede Darstellung die allgemeinen Genauigkeitsanforderungen. Damit für einen Menschen der Eindruck einer kontinuierlichen Änderung von visuell dargestellten Objekten erreicht wird, ist allerdings nur mäßiger Aufwand notwendig. Das menschliche Auge ist nicht in der Lage, eine schnelle Abfolge von Einzelbildern von einer kontinuierlichen Änderung zu unterscheiden. Die Grenze liegt bei etwa 20 bis 30 Einzelbildern pro Sekunde, in dem Bereich liegt auch die Technik von Fernsehen und Video.

6 `http://www.w3.org/Graphics/SVG/WG/wiki/Test_Suite_Overview`
7 `http://hoffmann.bplaced.net/svgtest/`

Für eine präzise Animation ist es notwendig, dass das Format einen definierten Zeitablauf festgelegt. Vor allem ist der Zeitbeginn des Dokumentes festzulegen, also der Zeitpunkt 0 für das Dokument und darin befindliche Animationen. Dies ist bei SVG 1.1 der Dokumentbeginn, wenn also das Hauptelement ***svg*** samt Inhalt komplett interpretiert ist und die Darstellung unmittelbar bevorsteht. Dies entspricht in SVG tiny 1.2 der Voreinstellung des Attributes *timelineBegin* des Elementes ***svg***, was dessen Wert 'onLoad' entspricht. Der andere mögliche Wert ist 'onStart', das ist im Regelfalle ein etwas früherer Zeitpunkt, wenn die Startmarkierung des Elementes ***svg*** komplett interpretiert und verarbeitet worden ist.

Bei der Art der Angabe von Werten, zwischen denen bei der Animation interpoliert werden soll, gibt es in SVG einige grundverschiedene Methoden. Die Basismethode ist die sogenannte *values*-Animation, bei der werden die Werte, zwischen denen interpoliert werden soll, in einem Attribut *values* jeweils mit einem Semikolon mit optionalen Leerzeichen getrennt. Eine weitere Methode liegt speziell für ***animateMotion*** neben der *values*-Animation vor - alternativ kann ein beliebiger Pfad notiert werden, entlang dessen die Bewegung erfolgen soll.

Die dritte Methode ist die sogenannte *to*-Animation, bei der wird angegeben, zu welchem Wert hin interpoliert werden soll, während der Anfangswert der aktuelle ist. SMIL gibt für diese Methode eine spezielle Interpolationsformel an.

Weitere Varianten, die *from-to*-Animation, die *from-by*-Animation und die *by*-Animation sind in SMIL definiert als Äquivalente zu speziellen *values*-Animationen, stellen also keine eigenen Methoden dar, sondern nur alternative Notationen.

Ein einfaches Beispiel zeigt bereits das wesentliche Prinzip:

```
<circle r="15">
<animateMotion dur="12s" values="0,300;400,0;400,300;0,300" />
</circle>
```

Mittels ***animateMotion*** wird ein Kreis entlang des Pfades, einem rechtwinkligen Dreieck, bewegt. Der Pfad wird hier im Attribut *values* notiert mit einer Liste von Punkten, die jeweils voneinander mit einem Semikolon separiert sind. Der Kreis bewegt sich zwischen den angegebenen Punkten, in der Reihenfolge, wie sie notiert sind. Mit dem Attribut *dur* wird die einfache Dauer der Animation angegeben, hier 12s. Weitere Angaben sind hier vom Autor nicht erforderlich. Die Angaben genügen, damit das Darstellungsprogramm die komplette Animation erzeugt.

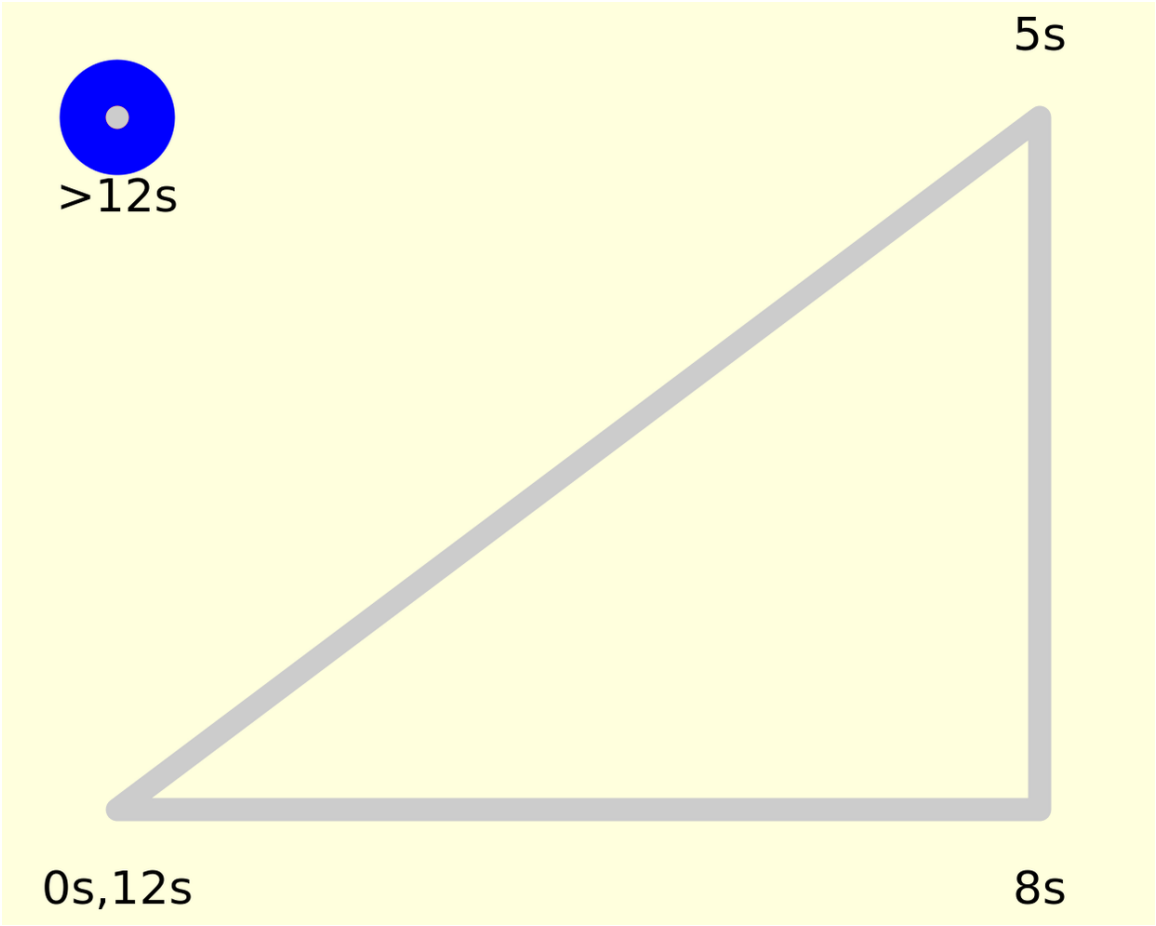


Abb. 219 Einfaches Beispiel ***animateMotion***

Etwas ausgeschmückt sieht das Beispiel dann so aus:

Einfaches Beispiel ***animateMotion***[8]:

Mittels ***animateMotion*** wird ein blau gerandeter Kreis entlang des grauen Pfades, einem rechtwinkligen Dreieck, bewegt. Wird die rote Füllung des Kreises sichtbar, liegt eine Fehlinterpretation vor. An den Ecken des Dreiecks ist jeweils angeben, zu welchen Zeitpunkten der Kreis an den Ecken ist. Nach 12s ist die Animation beendet und der Kreis springt zurück zum Ursprung, welcher ebenfalls grau gekennzeichnet ist.

14.1.3 Anordnung im Quelltext

Animationselemente können in allen graphischen und Gruppierungselementen notiert werden, jeweils hinter den gegebenenfalls vorhandenen Elementen ***title***, ***desc*** und ***metadata***.

8 `http://de.wikibooks.org/wiki/media%3ASVGanimateMotion01.svg`

Es ist üblich, Animationselemente als nicht direkt dargestellte Elemente entweder innerhalb des Elementes ***defs*** zu notieren oder wie im Beispiel in dem Element, auf welches die Animation wirken soll. Bei Elementen, die als Empfänger für Ereignisse dienen, zum Beispiel um so eine Animation zu starten, wird manchmal das Animationselement in diesem Element notiert, auch wenn die Animation auf ein anderes Element wirkt. Durch eine gute Strukturierung wird so das Erstellen, Verstehen und Überarbeiten von Dokumenten erleichtert.

14.1.4 Animationsfunktionen

Die Animation selbst ist über eine zeitabhängige Funktion definiert, welche die Interpolation zwischen zwei aufeinanderfolgenden Werten des Attributes *values* bestimmt. Die Animation bezieht sich immer exakt auf ein Zielelement und dort exakt auf ein Zielattribut, beziehungsweise eine Zieleigenschaft (der Einfachheit halber sei dies im Folgenden immer eingeschlossen; ferner ist bei ***animateMotion*** das Zielattribut impliziert und wird nicht explizit angegeben). Die Animation bestimmt eine Zeitabhängigkeit des Zielattributes. Die Zeitabhängigkeit basiert auf der sogenannten einfachen Animationsfunktion f(t), mit t der Zeit (englisch: simple animation function). Die einfache Animationsfunktion definiert für jeden Zeitpunkt der einfachen Animationsdauer (englisch: simple duration) einen Wert für das Zielattribut.

Sofern nicht anders angegeben, ist das Elternelement des Animationselementes das Zielelement der Animation. Der Basiswert eines Zielattributes ist der Wert des Attributes ohne Animationseffekt, was also als Attributwert notiert oder impliziert wird. Der Präsentationswert des Zielattributes zur Zeit t ist der Wert, der sich ergibt, wenn die Animation zum Zeitpunkt t angewendet wird. Die Animationseffekt kann auch von anderen Parametern abhängen, nicht nur vom Basiswert und der einachen Animationsfunktion. Das allgemeinere Konzept zum Basiswert ist der darunterliegende Wert (englisch: underlying value), welcher auch Animationsfunktionen mit niedrigerer Priorität einschließt. Die einfache Animationsfunktion wird dann notiert als f(t,u), wobei u die anderen Parameter darstellen, insbesondere den darunterliegenden Wert.

Animationen, die zum darunterliegenden Wert addiert werden, heißen additiv, solche, die den darunterliegenden Wert überschreiben, heißen nicht-additiv. Die einfache Animationsfunktion kann auch wiederholt werden, Wiederholungen können kumulieren. Der Endwert der Animation kann eingefroren werden oder der Wert springt nach Abschluss der Animation zurück auf den darunterliegenden Wert. Die sogenannte Animationseffektfunktion F(t,u) schließt anders als die einfache Animationsfunktion den Einfluss des darunterliegenden Wertes, von Wiederholungen und des Einfrierens mit ein. Während die einfache Animationsfunktion nur über die einfache Animationsdauer definiert ist, ist die Animationseffektfunktion über die gesamte aktive Animationsdauer definiert (englisch: active duration).

14.1.5 Modell des Zeitablaufs

Animationsdauern, der Beginn und das Ende einer Animation und die Unterteilung in verschiedene Interpolationsabschnitte durch die Liste der Werte im Attribut *values* legen Zeitintervalle fest. Für die deklarative Animation ist ein einheitliches Modell für die Intervalle und den Zeitablauf festgelegt. Dabei gehört der Anfang eines Intervalles zum Intervall und das Ende ist ausgeschlossen. So ist sichergestellt, dass, wie für die Animationsfunktionen notwendig, immer exakt ein Wert zu jedem Zeitpunkt gegeben ist. Lediglich für einen möglichen eingefrorenen Endwert nach der aktiven Dauer

der Animation gibt es einige Sonderregeln, die im Wesentlichen darauf hinauslaufen, dass sich dieser Endwert als Grenzwert des Verlaufes der Werte innerhalb des Intervalles gegen das Ende ergibt. Bei einem stetigen Verlauf deckt sich dies immer mit dem Wert, der sich ergäbe, wenn das Ende des Intervalles eingeschlossen wäre. Lediglich wenn zum Endzeitpunkt der Verlauf nicht stetig ist, ist der Grenzwert innerhalb des aktuellen Intervalles von praktischem Belang. (Diesbezüglich gibt es in SMIL 2 und 3 relativ zur SMIL Animationsempfehlung, welche für SVG 1.1 relevant ist, leider einige Inkonsistenzen. SVG tiny 1.2 bezieht sich zwar auf SMIL 2.1, eine Sonderregel für SVG gewährleistet aber das gleiche Verhalten in SVG tiny 1.2 wie in SVG 1.1).

14.1.6 Das Sandwich-Modell

Eine Animation ändert nicht direkt den Wert eines Attributes im Dokument-Objekt-Modell. Stattdessen wird ein Präsentationswert animiert, unabhängig vom Dokument-Objekt-Modell, von CSS oder einem anderen möglichen Modell. Der Präsentationswert wird angezeigt und verändert nicht den Wert im Dokument-Objekt-Modell oder im CSS.

Dieses Vorgehen erfordert, die Präsentationswerte verschiedener Animationen zu verwalten und in einer definierten Reihenfolge, der Priorität anzuwenden. Animationen mit höherer Priorität können die Effekte von Animationen mit niedrigerer Priorität überschreiben. Oder die Animationen mit niedrigerer Priorität dienen als der darunterliegende Wert der Animation mit höherer Priorität. So sind ganz verschiedene Lagen oder Schichten von Animationen zu verwalten, ähnlich wie es in einem Sandwich verschiedene Schichten gibt, die aufeinanderliegen.

Im sogenannten Sandwich-Modell wird in SMIL festgelegt, mit welcher Priorität Animationen behandelt werden, welche auf dasselbe Zielattribut desselben Zielelementes wirken. Das Modell legt auch fest, wie die Priorität der Animation einer Eigenschaft relativ zur Animation des korrespondierenden Präsentationsattributes ist.

Beziehen sich mehrere Animationen auf dasselbe Attribut desselben Zielelementes, so hat jene höhere Priorität, welche später beginnt. Da Animationen mehrfach beginnen können, bestimmt sich so die Priorität mit jedem Beginn erneut. Beginnen Animationen zum gleichen Zeitpunkt, so entscheidet die Reihenfolge im Quelltext über die Priorität. Die weiter hinten im Quelltext stehende Animation hat die höhere Priorität. Es gibt auch die Möglichkeit, dass bei zwei gleichzeitig beginnenden Animationen eine Animation von der anderen abhängt, zum Beispiel, dass der Beginn der einen vom Beginn der anderen abhängt. In solch einem Falle wird angenommen, dass die abhängige Animation der synchronisierenden mit höherer Priorität folgt, auch wenn die Reihenfolge im Quelltext anders angeordnet ist.

Das Sandwich-Modell ersetzt nicht die ansonsten definierten Regeln, wie Attribute oder Eigenschaften verarbeitet werden. Wird speziell ein Präsentationsattribut animiert und zur gleichen Zeit die korrespondierende CSS-Eigenschaft, so gilt nach wie vor die in CSS definierte Kaskade, bei der das Präsentationsattribut eine sehr niedrige Spezifität hat, die von einer CSS-Eigenschaft mit höherer Spezifität überschrieben wird. Auch wenn das Präsentationsattribut animiert wird, hat eine CSS-Angabe in einer CSS-Stilvorlage eine höhere Spezifität als das animierte Präsentationsattribut, welches so mit der Eigenschaft überschrieben wird. Es gibt keinen sichtbaren Animationseffekt. Wird auch die CSS-Eigenschaft animiert, so wird das Resultat der Animation des Präsentationsattributes und der CSS-Stilvorlage als darunterliegender Wert der Animation der CSS-Eigenschaft verwendet. Aufgrund der hohen Spezifität der !important-Regel in einer Stilvorlage des Benutzers überschreibt diese allerdings wieder einen Animationseffekt für dieselbe CSS-Eigenschaft.

Bei additiven Animationen können Zwischenergebnisse außerhalb des erlaubten Bereiches liegen, etwa bei Farben oder Opazitäten. In solchen Fällen soll vom Darstellungsprogramm nur der letztlich angezeigte Wert auf den erlaubten Bereich eingschränkt werden, nicht mögliche Zwischenergebnisse, weil sich durch die Addition aller Animationen wieder ein korrekter Wert ergeben kann, was bei einer Korrektur von Zwischenergebnissen zu einem anderen, falschen Resultat führen könnte.

Anfragen oder Manipulationen mit Skriptsprachen wie ecma-script beziehen sich auf den Wert im Dokument-Objekt-Modell und nicht auf den durch die Animationen resultierenden Präsentationswert. Der per Skript manipulierte Wert im Dokument-Objekt-Modell hat also im Sandwich-Modell die gleiche niedrige Priorität wie der nicht manipulierte Wert.

14.2 Animationsziel identifizieren

Ein Darstellungsprogramm muss immer identifizieren können, auf welches Element, welches Atttribut oder welche Eigenschaft sich eine Animation bezieht. Zu dem Zweck sind teils spezielle Attribute verfügbar, teils wird diese Information impliziert.

14.2.1 Zielelement

Squiggle (Batik)	1.7 (teilweise; Problem bei exotischen Fällen)
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	-
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Das Zielelement einer Animation wird angegeben mit dem Attribut *href* von XLink. Mit diesem Attribut stehen dann auch die anderen Attribute von XLink bei Animationselementen zur Verfügung.

Beim Zielelement ist also mit dem Attribut *id* (oder bei SVG tiny 1.2 alternativ auch *xml:id*) ein Fragmentidentifizierer anzugeben. Für das Animationselement kann dann das Attribut *href* von XLink notiert werden mit dem Fragmentidentifizierer des Zielelementes als Wert, einschließlich des vorangestellten '#'. Das Zielelement liegt immer im selben Dokument wie das Zielelement. Gibt es das angegebene Zielelement nicht im selben Dokument, so liegt in SVG 1.1 ein Fehler vor und die Darstellung des Dokumentes wird abgebrochen. In SVG tiny 1.2 hingegen wird die Animation in solch einem Falle zwar durchgeführt, hat im Sinne von SMIL aber keinen Effekt. Dies bedeutet, die Animation kann zur Synchronisation anderer Animationen verwendet werden, hat selbst aber keinen direkten Einfluss auf die Präsentation.

Wird alternativ *href* von XLink nicht notiert, so ist das Elternelement des Animationselementes das Zielelement der Animation.

Beispiel zur Verwendung von *href* von XLink:

```
<defs xmlns:xlink="http://www.w3.org/1999/xlink">
<animateMotion xlink:href="#c" dur="12s" values="0,300;400,0;400,300;0,300" />
```

```
</defs>
<circle id="c" r="15" />
```

Das mittels *href* von XLink angegebene Zielelement ist also der Kreis, welcher mittels *id*="c" identifiziert ist. Sofern bereits bei einem anderen Elternelement passiert, muss der Namensraum für XLink natürlich nicht erneut angegeben werden.

Bei folgendem Beispiel ist das Elternelement ***circle*** des Animationselementes ***animateMotion*** Zielelement der Animation, weil *href* von XLink nicht notiert worden ist:

```
<circle r="15">
<animateMotion dur="12s" values="0,300;400,0;400,300;0,300" />
</circle>
```

Abb. 220 Beispiele zum identifizieren des Animationszielelementes

Beispiele zum identifizieren des Animationszielelementes[9]

Verschiedene Methoden, das Animationszielelement zu identifizieren, einschließlich eines exotischen Falles unüblicher Notation für den Namensraum von XLink, die Autoren nicht zu empfehlen ist.

9 http://de.wikibooks.org/wiki/media%3ASVG_Animationsziel01.svg

Alle blauen Rechtecke vergrößern ihre Höhe, so dass sie immer exakt in den dunkelblauen Rahmen passen, der von diesen immer komplett ausgefüllt wird.

14.2.2 Zielattribut

Squiggle (Batik)	1.7 (teilweise, Problem mit Namensräumen)
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	-
Safari (Webkit)	4 (teilweise, Problem mit Namensräumen)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Das Zielattribut einer Animation wird angegeben mit dem Animationsattribut *attributeName*. Dies entfällt bei ***animateMotion***, bei welchem dies als spezielle Translationstransformation impliziert wird. Für die anderen Animationselemente ist die Angabe des Attributes in SVG 1.1 erforderlich. Fehlt das Attribut oder ist der Wert nicht anwendbar, liegt ein Fehler vor, der zum Abbruch der Darstellung des Dokumentes führt. In SVG tiny 1.2 ist die Angabe zwar optional, die Animation hat aber im Sinne von SMIL keinen Effekt (kann also allenfalls zur Synchronisation anderer Animationen verwendet werden), wenn der Wert nicht anwendbar ist oder das Attribut nicht notiert ist.

Sofern das Attribut zu einem im Dokument definierten Namensraum gehört, ist das entsprechende Präfix ebenfalls zu notieren, sofern das Attribut nicht zu einem Namensraum gehört, der so festgelegt ist, dass kein Präfix notwendig ist (typisch der für SVG in einem SVG-Dokument). Das Präfix wird dann so interpretiert, wie es für das Animationselement definiert ist, was bei einer entprechenden anderen Angabe für das Zielelement anders sein kann. Entscheidend ist im Zweifelsfalle also immer der Namensraum, der im Animationselement für das Präfix definiert ist. Es ist also insbesondere nicht relevant, welches Präfix beim Zielelement notiert ist. Ärger kann sich ein Autor ersparen, wenn für einen Namensraum pro Dokument exakt ein Präfix im Hauptelement ***svg*** notiert wird.

Beispiele:

```
<circle r="20">
<animate attributeName="r" dur="12s" values="0;100;50;400;10;500;100;300;60" />
</circle>
```

Als Zielattribut wird mit *attributeName* das Attribut *r* festgelegt. Das Elternelement des Animationslementes ***animate*** ist hier ein Element ***circle***, folglich wird der Radius dieses Kreises animiert.

```
<image xmlns:xlink="http://www.w3.org/1999/xlink"
       width="30" height="20" xlink:href="P0.png">
  <animate attributeName="xlink:href" dur="120s"
    values="P1.png;P2.jpeg;P3.PNG;P4.JPG;P5.jpg;P6.png" />
</image>
```

Das Attribut *href* von XLink wird für das Element ***image*** animiert, was zur Folge hat, dass ein paar Bilder nacheinander angezeigt werden, also eine Art Diashow. Das Präfix für den Namensraum von XLink ist hier direkt in der Anfangsmarkierung des Elementes ***image*** definiert, ist oft auch bei einem anderen Elternelement definiert und ist hier für das Element ***animate*** der gleiche wie für ***image***, die Zuordnung ist also unproblematisch.

Abb. 221 Beispiele Animationsattribut *attributeName*

Beispiele Animationsattribut *attributeName*[10]

Verschiedene Methoden, das Animationszielattribut zu identifizieren, einschließlich einiger exotischer Fälle unüblicher Notationen für den Namensraum von SVG oder XLink, die Autoren nicht zu empfehlen sind. Andererseits kann es in Dokumenten mit mehreren Formaten durchaus vorkommen, dass ein Namensraum für SVG explizit als Präfix angegeben wird.

Alle blauen Rechtecke vergrößern ihre Höhe, so dass sie immer exakt in den dunkelblauen Rahmen passen, der von diesen immer komplett ausgefüllt wird.

10 `http://de.wikibooks.org/wiki/media%3ASVGattributeName01.svg`

14.2.3 Zielattributstyp

Squiggle (Batik)	-
Opera (Presto)	9 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4 (teilweise)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Nun kommt es vor, dass sowohl ein Präsentationsattribut als auch eine Eigenschaft mit gleichem Namen existiert. Folglich kann auch festgelegt werden, welches von beiden animiert werden soll. Dieser Angabe dient das Animationsattribut *attributeType*.

Bei ***animateMotion*** ist auch das Attribut *attributeType* nicht anwendbar. Bei ***animateMotion*** ist das Zielattribut bereits eindeutig impliziert.

Werden nur Präsentationsattribute verwendet und keine Stilvorlagen, so ist die Angabe dieses Attributes weder notwendig, noch sinnvoll. Insbesondere ist die Angabe innerhalb eines Dokumentes des Profils SVG tiny 1.1 nicht sinnvoll, weil dort gar keine Stilvorlagen interpretiert werden. Auch wenn Stilvorlagen interpretiert werden, ist zumindest in SVG die Angabe des Attributes in der Regel nicht notwendig, weil Attribut und Eigenschaft die gleiche Bedeutung haben und das notierte Präsentationsattribut mit niedriger Spezifität in die Kaskade der Stilvorlage integriert ist.

Ferner ist die Funktionalität so definiert, dass in SVG nicht anwendbare Eigenschaften auch nicht Ziel der Animation sein können. Zum Beispiel tritt das Problem theoretisch auf bei CSS-Eigenschaften wie *width* und *height*, weil in SVG Elemente wie ***rect*** Attribute gleichen namens haben. Weil aber die CSS-Eigenschaften nicht anwendbar sind, werden ohne Angabe von *attributeType* die Attribute animiert und nicht die gleichnamigen Eigenschaften, die keinen Effekt hätten.

Mögliche Werte von *attributeType* sind:

XML

Es wird das Attribut (aus dem Namensraum des Zielelementes) animiert, sofern es dieses gibt.

CSS

Es wird die Eigenschaft animiert, sofern es eine solche gibt.

auto

Sofern es eine anwendbare Eigenschaft gibt, wird diese animiert, sonst das Attribut (aus dem Namensraum des Zielelementes, wenn kein Präfix angegeben ist), sofern es dieses gibt.

'auto' ist die Voreinstellung, wenn das Attribut nicht notiert ist. Wenn es das Zielattribut, beziehungsweise die Zieleigenschaft nicht gibt, hat die Animation im Sinne von SMIL keinen Effekt (kann also allenfalls zur Synchronisation anderer Animationen verwendet werden). Das Attribut oder die Eigenschaft müssen nirgends explizit notiert sein, sie müssen nur auf das Zielelement anwendbar sein, damit die Animation einen Effekt haben kann.

Beispiele:

```
<circle r="20">
<animate attributeName="fill"
```

```
        dur="9s" values="blue;green" />
</circle>
```

Weil *attributeType* nicht gesetzt ist, wird die Eigenschaft *fill* des Kreises von blau nach grün animiert.

Dies hat den gleichen Effekt wie folgendes Beispiel:

```
<circle r="20">
<animate attributeName="fill" attributeType="auto"
        dur="9s" values="blue;green" />
</circle>
```

Für Autoren meist wohl unerwünschte Ergebnisse werden bei folgendem Beispiel erzielt:

```
<circle r="20" style="fill:blue">
<animate attributeName="fill" attributeType="XML"
        dur="9s" values="#a00;#f60" />
</circle>
```

Weil *attributeType* auf 'XML' gesetzt ist, wird das Präsentationsattribut *fill* im roten Bereich animiert. Das (animierte) Präsentationsattribut hat in der CSS-Kaskade allerdings eine niedrigere Spezifität als die ebenfalls angegeben Eigenschaft *fill*. Die Animation wird also von der Eigenschaft überschrieben und hat keinen sichtbaren Effekt.

```
<circle r="20" fill="red">
<animate attributeName="fill" attributeType="CSS"
        dur="9s" values="#00a;#0cf" additive="sum" />
</circle>
```

Weil *attributeType* auf 'CSS' gesetzt ist, wird die Eigenschaft *fill* animiert. Das Präsentationsattribut *fill* ist mit niedriger Spezifität in der CSS-Kaskade und dessen Wert dient daher als darunterliegender Wert der Animation. Da diese additiv ist (was später noch erläutert wird), addiert sich die Animation zum darunterliegenden Wert und das Resulat ist eine Animation im violetten Bereich.

Dies in einem Dokument vom Profil SVG tiny 1.1 hätte hingegen einen anderen Effekt. Weil Stilvorlagen nicht interpretiert werden, wäre das Ergebnis einfach eine rote Füllung.

```
<circle r="20" fill="red">
<animate attributeName="fill" attributeType="CSS"
        dur="9s" values="#00a;#00f" additive="sum" />
<animate attributeName="fill" attributeType="XML"
        dur="9s" values="#060;#0f0" additive="sum" />
</circle>
```

Hier ergibt die additive Animation des Präsentationsattributes *fill* zunächst eine Animation im orangen Bereich. Der zeitabhängige Wert ist mit niedriger Spezifität in der CSS-Kaskade und der Wert dient wiederum als darunterliegender Wert der additiven Animation vom *attributeType* 'CSS'. Letztlich ist das Ergebnis eine Animation von #f6a nach #fff, also von rosa-violett nach weiß. Kürzer könnte dies also geschrieben werden als:

```
<circle r="20">
<animate attributeName="fill"
        dur="9s" values="#f6a;#fff" />
</circle>
```

Für SVG tiny 1.1 wäre wieder die Animation der Eigenschaft zu ignorieren und das Ergebnis wäre die Animation im orangen Bereich.

Folgendes ist ein Beispiel, bei dem der Autor vermutlich nicht verstanden hat, dass es für SVG-Elemente keine andwendbare Eigenschaft *width* gibt:

```
<rect x="100" y="100" width="20" height="100">
<animate attributeName="width" attributeType="CSS"
         dur="9s" values="60;200;30" />
<animate attributeName="height" attributeType="auto"
         dur="9s" values="60;200;30" />
</rect>
```

Weil die Eigenschaft *width* auf das Element ***rect*** nicht anwendbar ist, hat die erste Animation vom *attributeType* 'CSS' keinen Effekt. Es gibt auch keine anwendbare Eigenschaft *height*, aber das Element ***rect*** hat ein Attribut *height*, weswegen die zweite Animation vom *attributeType* 'auto' einen Effekt hat.

Weitere Beispiele:

Abb. 222 Animationsattribut *attributeType*

Animationsattribut *attributeType*[11]

Weil CSS-Eigenschaften eine höhere Spezifität haben als XML-Attribute, hat die CSS-Animation die XML-Animation zu überschreiben. Die Farbe der Kreise ändert sich von gelb nach blau und nicht von orange nach rot.

11 http://de.wikibooks.org/wiki/media%3ASVGattributeType01.svg

Falls das Darstellungsprogramm CSS nicht unterstützt, wie bei SVG tiny 1.1, sollte die Farbe sich von orange nach rot und nicht von gelb nach blau ändern. Ein Gemisch ist immer ein Fehler.

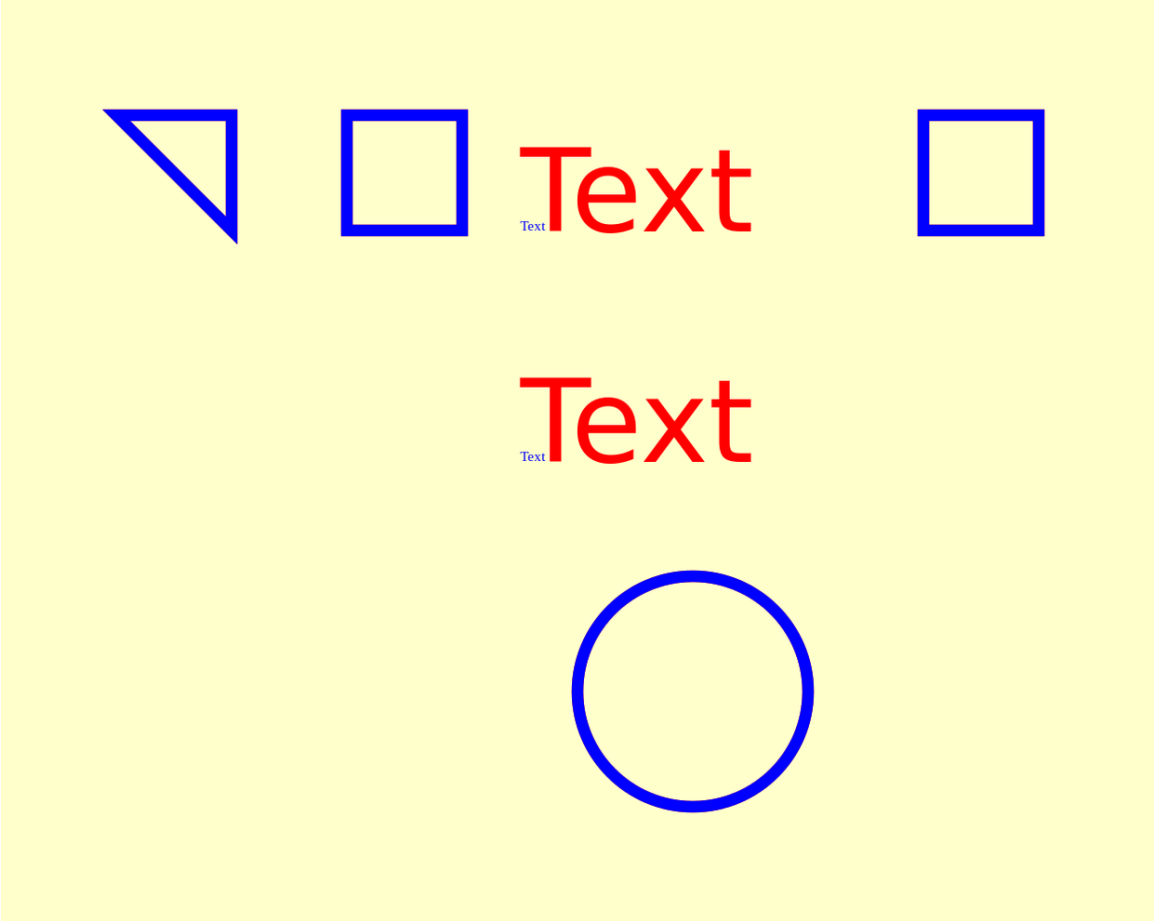


Abb. 223 Animationsattribut *attributeType* (2)

Animationsattribut *attributeType* (2)[12]

Verschiedene Tests werden angeboten: Falls *attributeName* ein Attribut identifiziert und es gibt keine Eigenschaft mit dem gleichen Namen und der *attributeType* ist CSS, kann es keine Animation geben.

Falls eine CSS-Eigenschaft existiert, aber nicht anwendbar ist, aber ein XML-Attribut mit dem gleichen Namen existiert, kann es keine Animation mit *attributeType* CSS geben.

Aber zum Beispiel *font* ist eine Eigenschaft ohne entsprechendes XML-Attribut, daher wird dies mit *attributeType* CSS animiert, aber nicht mit XML.

12 `http://de.wikibooks.org/wiki/media%3ASVGattributeType02.svg`

Falls der *attributeType* 'auto' ist, aber der *attributeName* nicht zum animierten Element passt, gibt es keine Animation.

Falls *attributeType* CSS ist, die animierte Eigenschaft existiert, aber nicht zum animierten Element passt, gibt es keine Animation.

Falls etwas Rotes sichtbar wird, ist ein Fehler aufgetreten.

Viele aktuelle Darstellungsprogramme (Stand 2012) haben Probleme mit der korrekten Implementierung einerseits des Sandwich-Modells und andererseits auch der CSS-Kaskade, weswegen es leicht zu fehlerhaften Darstellungen kommen kann. Der Autor kann dies vermeiden, indem er auf die Verwendung des Attributes *style* verzichtet, gar keine Stilvorlagen verwendet für Eigenschaften, die animiert werden sollen und eine explizite Angabe des Attributes *attributeType* vermeidet.

14.3 Die einfache Animationsdauer

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	3.2 (teilweise)
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Animationsattribut *dur* wird die einfache Animationsdauer festgelegt, also die Zeit für einen einfachen Durchlauf der Animation.

Mögliche Werte sind:

'indefinite'

die einfache Animationsdauer ist unendlich

'media'

Die intrinsische Dauer eines referenzierten Mediums legt die einfache Animationsdauer fest.

Zeitwert

Angabe eines Zeitwertes größer als 0 wie im Folgenden genauer festgelegt,

im einfachsten Falle eine Zahl für die Dauer in Sekunden. Ist der angegebene Wert von *dur* fehlerhaft, so wird das Attribut ignoriert, als sei es nicht angegeben worden. Ist das Attribut nicht angegeben, wird 'indefinite' angenommen.

14.3.1 'indefinite'

Ist 'indefinite' angegeben, wird zu Beginn der Animation der erste Wert angenommen. Es findet keine weitere Interpolation zu gegebenenfalls vorhandenen weiteren Werten statt. Dies kann sinnvoll sein, wenn nur ein Wert im Attribut *values* notiert ist oder ***set*** verwendet wird und keine endliche Dauer vorgesehen ist.

Die Animationsfunktion ist folglich f(t) = f(0).

14.3.2 'media'

Für Animationselemente wird das Attribut *dur* ignoriert, falls 'media' angegeben wird, weil diese Elemente keine intrinsische Dauer festlegen. In SVG tiny 1.2 gibt es allerdings zeitabhängige Elemente wie ***audio*** oder ***video***, bei denen dieser Wert sinnvoll ist, es wird dann die vom referenzierten Dokument festgelegte Dauer als einfache Animationsdauer angenommen.

14.3.3 Zeitwert

Ein Zeitwert besteht aus mindestens einer positiven Zahl, optional mit direkt anschließender Einheit.

In SVG tiny sind die Einheiten 's' und 'ms' zugelassen. 's' steht für Sekunden, 'ms' für Millisekunden. In der Vollversion von SVG 1.1 gibt es weitere mögliche Einheiten, dies sind 'h' für Stunden und 'min' für Minuten.

Genaugenommen ist die Einschränkung zumindest in SVG tiny 1.1 nicht eindeutig formuliert. Die Einschränkung kann auch so interpretiert werden, dass immer eine Einheit angegeben werden muss (dann also 's', 'ms', 'min' oder 'h') oder aber auch zusätzlich im Falle von 's' die Einheit weggelassen werden kann, also nur im weiteren diskutierte gemischte Notationen ausgeschlossen werden. In SVG tiny 1.2 ist die Einschränkung auf 'ms', 's' oder keine Einheit eindeutig.

Um bei SVG tiny 1.1 Probleme zu vermeiden, kann es also sinnvoll sein, immer entweder 's' oder 'ms' als Einheit anzugeben.

Beispiele:

4 entspricht 4 Sekunden

9.342 entspricht 9.342 Sekunden

00016.2 entspricht 16.2 Sekunden

12s entspricht 12 Sekunden

138.9432s entspricht 138.9432 Sekunden

3058.63ms entspricht 3058.63 Millisekunden oder 3.05863 Sekunden

1.2min entspricht 1.2 Minuten oder 72 Sekunden

0.4h entspricht 0.4 Stunden oder 24 Minuten oder 1440 Sekunden

Alternativ dazu gibt es in der Vollversion von SVG 1.1 noch zwei weitere mögliche Notationen.

Die eine besteht darin, exakt zwei Ziffern für Minuten zu notieren (Bereich 00 bis 59), gefolgt von einem Doppelpunkt, gefolgt von exakt weiteren zwei Ziffern für Sekunden (Bereich 00 bis 59). Darauf kann optional ein Punkt folgen, gefolgt von mindestens einer weiteren Ziffer für Bruchteile eine Sekunde.

Beispiele:

01:00 entspricht 1 Minute oder 60 Sekunden

02:38 entspricht 2 Minuten und 38 Sekunden oder 158 Sekunden

01:00.45 entspricht 1 Minute und 0.45 Sekunden oder 60.45 Sekunden oder 60450 Millisekunden

Die andere Möglichkeit besteht darin, eine beliebige positive Anzahl von Ziffern für Stunden zu notieren, direkt gefolgt von einem Doppelpunkt, direkt gefolgt von exakt zwei Ziffern (Bereich 00 bis 59) für Minuten, gefolgt von einem Doppelpunkt, gefolgt von exakt weiteren zwei Ziffern (Bereich 00 bis 59) für Sekunden. Darauf kann optional ein Punkt folgen, gefolgt von mindestens einer weiteren Ziffer für Bruchteile eine Sekunde.

Beispiele:

00:02:38 entspricht 2 Minuten und 38 Sekunden oder 158 Sekunden

00:01:00.45 entspricht 1 Minute und 0.45 Sekunden oder 60.45 Sekunden oder 60450 Millisekunden

01:01:00 entspricht 1 Stunde und 1 Minute oder 61 Minuten oder 3660 Sekunden

02:00:32.14 entspricht 2 Stunden und 32.14 Sekunden oder 7232.14 Sekunden

1234:07:00.2 entspricht 1234 Stunden, 7 Minuten und 200 Millisekunden oder 4442820.2 Sekunden

Insgesamt ist es erlaubt, vor und hinter dem Wert Leerzeichen zu notieren, aber nicht innerhalb des Wertes.

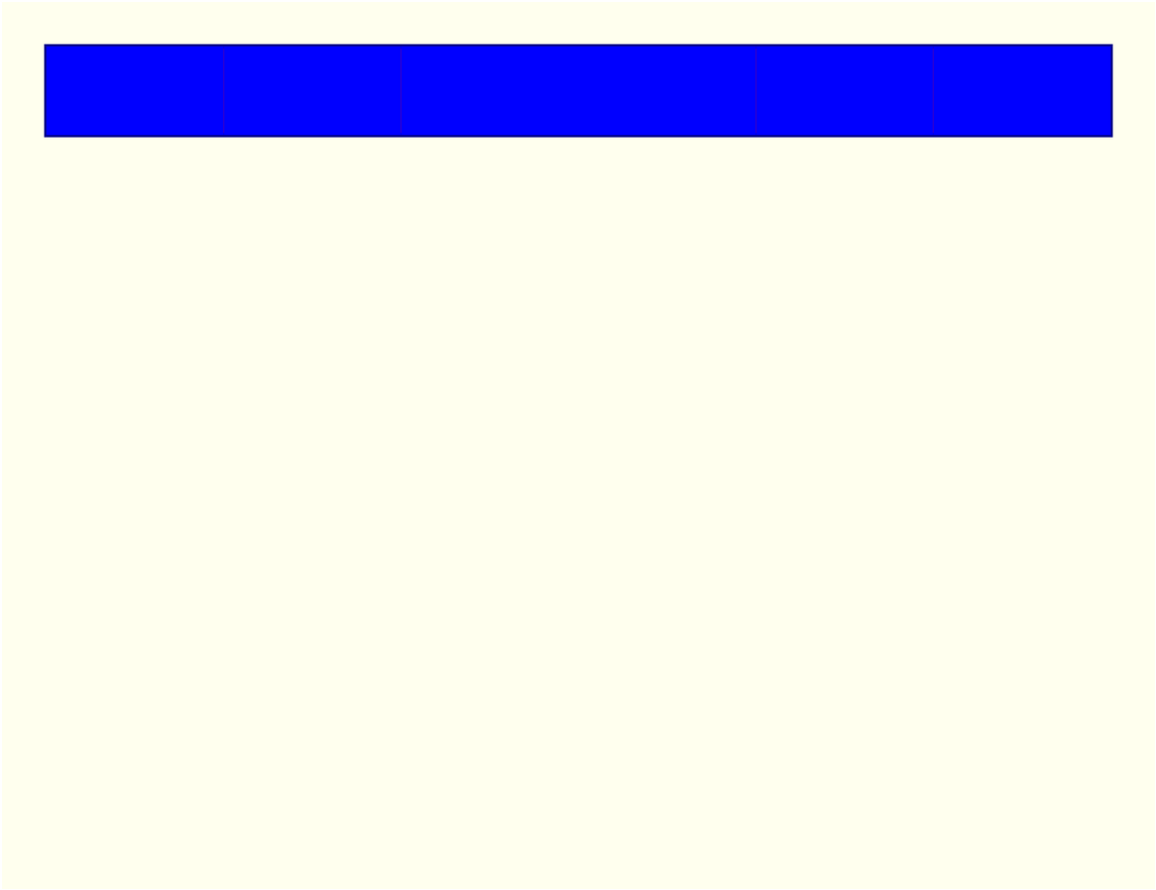


Abb. 224 Einfache Angabe der Animationsdauer

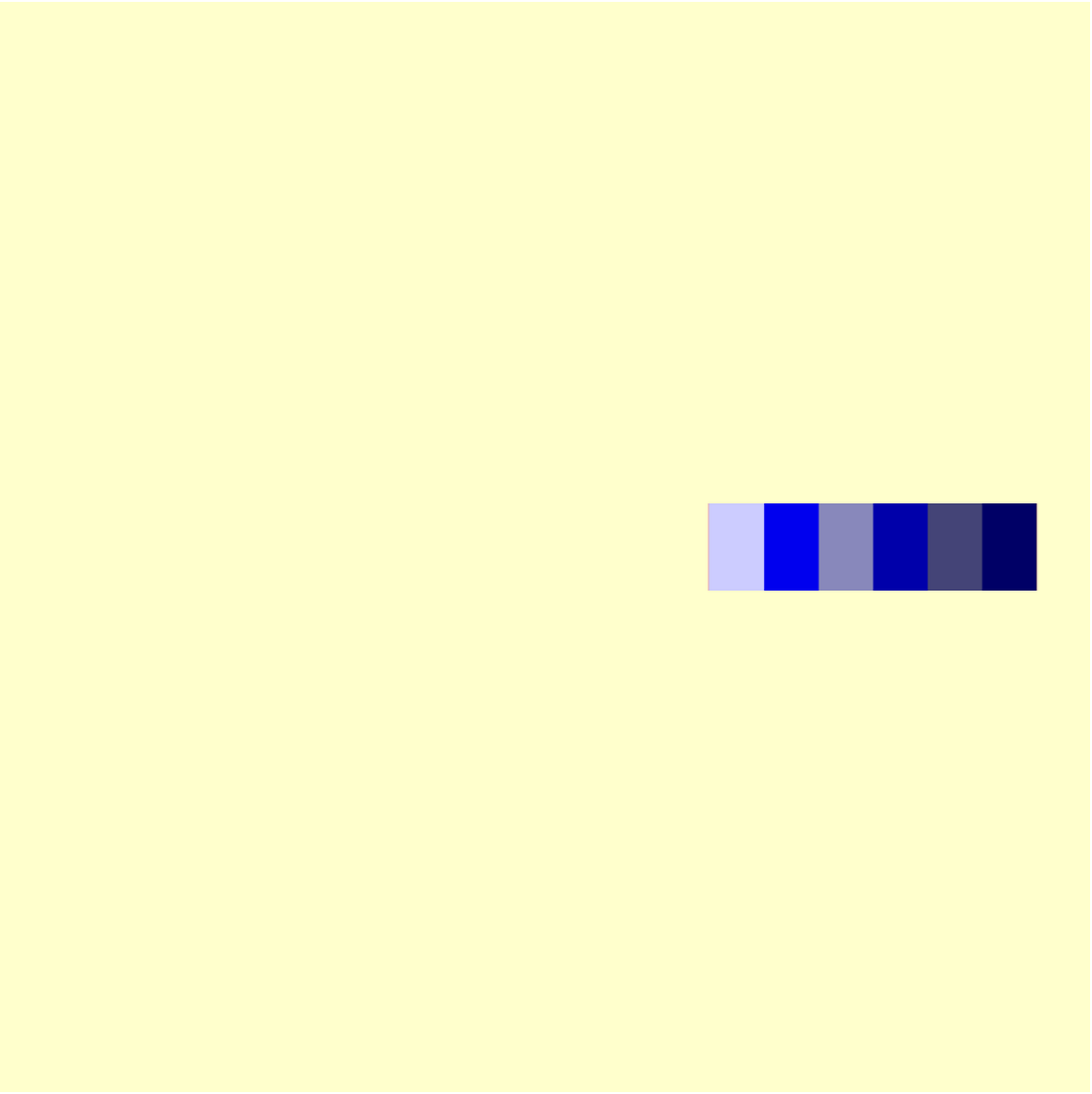


Abb. 225 Gemischte Angabe der Animationsdauer

Einfache Angabe der Animationsdauer[13]

Verschiedene Methoden, die einfache Animationsdauer mit Einheitsangabe zu definieren.

Alle blauen Rechtecke vergrößern ihre Höhe, so dass sie immer exakt in den dunkelblauen Rahmen passen, der von diesen immer komplett ausgefüllt wird.

Gemischte Angabe der Animationsdauer[14]

13 http://de.wikibooks.org/wiki/media%3ASVGdur01.svg
14 http://de.wikibooks.org/wiki/media%3ASVGdur02.svg

Verschiedene Methoden, die einfache Animationsdauer mit zu definieren. Verwendet werden vor allem Notationen mit kombinierten Angaben von Sekunden, Minuten und Stunden.

Die blauen Rechtecke drehen sich mit der gleichen Winkelgeschwindigkeit. Wird etwas Rotes sichtbar, ist ein Fehler aufgetreten.

14.4 Listen für den Beginn und das Ende einer Animation

Mit den Attributen *begin* und *end* können Zeitpunkte für Beginn und Ende einer Animation angegeben werden. Für beides gibt es diverse Möglichkeiten, einen Wert zu notieren oder eine Abhängigkeit von Beginn und Ende anzugeben. Ferner ist genau festgelegt, wie diese Listen interpretiert werden. Aus den Listen, die der Autor angibt, erzeugt das Darstellungsprogramm eigene Listen, in die die jeweiligen Zeiten explizit in Dokumentzeit eingetragen werden. Einige der Zeitpunkte werden erst im Verlauf der Dokumentzeit aufgelöst, diese werden auch dann erst in die Listen eingetragen. Einige Einträge werden auch wieder gelöscht, andere verbleiben dauerhaft in den Listen.

Der Wert ist eine Liste von Werten, welche jeweils mit einem Semikolon voneinander getrennt werden. Optionale Leerzeichen vor und nach dem Semikolon sind ebenfalls erlaubt.

Ist *end* notiert, so wird zum aktuellen Wert von *begin* der kleinste spätere Wert in der der Liste für *end* gesucht, um ein Zeitintervall für die zu beginnende Animation festzulegen. Gibt es nur frühere Werte in der Liste für *end*, so gibt es ein solches Intervall nicht und die Animation kann nicht beginnen. Diese Einschränkung gibt es nicht, wenn kein *end* notiert ist.

Mögliche Werte der Listen sind:

Zeitversatz

Ein Zeitwert, wie für *dur* definiert, mit optionalem '+' oder '-' vorangestellt; bezeichnet einen Zeitversatz relativ zum Beginn des Dokumentes

Synchronisationswert

Identifiziert eine andere Animation oder ein zeitabhängiges Element zum Zwecke der Synchronisation

Ereigniswert

Identifiziert ein Ereignis als Synchronisation

Wiederholungsereignis

Ein Ereignis der Wiederholung einer Animation dient der Synchronisation

Tastaturereignis

Das Betätigen einer Taste der Tastatur dient als Synchronisation

Uhrzeit

Eine Uhrzeit notiert im internationalen Format für Datum und Uhrzeit dient als Synchronisation. Dieser Wert ist in SVG tiny 1.2 nicht verfügbar.

'indefinite'

Kein endlicher Wert ist festgelegt. Die Sychronisation kann durch ein externes Ereignis ausgelöst werden (DOM oder Verweis).

Die Werte werden in folgenden Abschnitten genauer beschrieben.

14.4.1 Beginn der Animation

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	8 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4 (teilweise)
Konqueror (KSVG)	3.2 (teilweise)
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Attribut *begin* wird eine Liste von Zeitpunkten angegeben, zu denen die Animation aktiv werden soll oder beginnen soll. Ist das Attribut nicht angegeben, entspricht dies dem Wert '0', also dem Dokumentbeginn.

Einfaches Beispiel:

```
<circle r="15">
<animateMotion dur="12s" begin="6"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Die Animation beginnt 6 Sekunden nach Dokumentbeginn.

Liegt der Beginn einer Animation in der Vergangenheit, so beginnt die Präsentation der Animation unmittelbar sofort, als habe sie zum angegebenen Zeitpunkt in der Vergangenheit bereits begonnen, es sei denn, das Ende liegt auch bereits in der Vergangenheit. Die Animation wird also nur begonnen, wenn der aktuelle Zeitpunkt innerhalb des Intervalles der aktiven Animationsdauer liegt.

Abb. 226 Verschiedene Möglichkeiten, Animationen zu beginnen

Verschiedene Möglichkeiten, Animationen zu beginnen[15]

Blaue Quadrate werden entlang eines quadratischen Pfades bewegt. Neben einem Start nach 5s kann die Animation des größten Quadrates durch die Knöpfe rechts aktiviert werden, deren title-Elemente das entsprechende Ereignis angeben. Zudem kann mit der Taste b gestartet werden.

Die anderen Quadrate werden mit Synchronisationswerten oder einem Wiederholungswert gestartet.

15 `http://de.wikibooks.org/wiki/media%3ASVGbegin01.svg`

14.4.2 Ende der Animation

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	8 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4 (teilweise)
Konqueror (KSVG)	3.2 (teilweise)
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Attribut *end* wird eine Liste von Zeitpunkten angegeben, zu denen die Animation beendet werden soll. Dies erlaubt es, die aktive Dauer einer Animation einzuschränken. Ist das Attribut nicht angegeben, gibt es keine Einschränkung der aktiven Dauer durch dies Attribut. Dies Verhalten entspricht keinen angebbaren Attributwert.

Einfaches Beispiel:

```
<circle r="15">
<animateMotion dur="12s" begin="6" end="16"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Die Animation beginnt 6 Sekunden nach Dokumentbeginn und endet bereits 10s nach Dokumentbeginn, die aktive Animationsdauer beträgt also nur 10s. Die Animation erreicht also nicht den Endpunkt 0,300, sondern springt nach 16s zurück zum darunterliegenden Wert.

Abb. 227 Verschiedene Möglichkeiten, Animationen zu beenden

Verschiedene Möglichkeiten, Animationen zu beenden[16]

Blaue Quadrate werden entlang eines quadratischen Pfades bewegt. Neben einem Start nach 5s kann die Animation des größten Quadrates durch aktivieren, anklickern oder Taste b gestartet werden.

Neben einem Ende nach 10s kann die Animation des größten Quadrates durch die Knöpfe rechts aktiviert werden, deren title-Elemente das entsprechende Ereignis angeben. Zudem kann sie mit der Taste e beendet werden.

Die anderen Quadrate werden mit einem Synchronisationswert gestartet und mit Synchronisationswerten oder einem Wiederholungswert beendet.

16 `http://de.wikibooks.org/wiki/media%3ASVGend01.svg`

14.4.3 Zeitversatz

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	9.5 (ab 8 teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4 (teilweise)
Konqueror (KSVG)	3.2 (teilweise)
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Ein Zeitversatz ist ein Zeitwert wie für *dur* definiert, optional mit vorangestelltem Vorzeichen '+' oder '-', optional mit Leerzeichen vor oder nach dem Vorzeichen. Der Zeitwert beschreibt einen Versatz relativ zum zeitlichen Beginn des Dokumentes. Kein Vorzeichen oder das '+' steht für einen positiven Versatz, die Animation beginnt dann entsprechend später als der Dokumentbeginn. '-' entspricht einem negativen Zeitversatz und beschreibt einen Zeitpunkt vor Dokumentbeginn.

Wenn sowohl der Beginn als auch das Ende der aktiven Dauer einer Animation vor dem Dokumentbeginn liegen, so startet die Animation nicht und kann auch nicht zur Sychronisation verwendet werden. Ansonsten ist nur der Teil der Animation sichtbar, welcher nach dem Dokumentbeginn liegt.

Einfache Beispiele:

```
<circle r="15">
<animateMotion dur="12s" begin="-6s"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Die Animation beginnt formal -6 Sekunden zum Dokumentbeginn, also davor. Das bedeutet, die sichtbare Präsentation beginnt kurz hinter der oberen rechten Ecke des Dreiecks und endet 6s nach Dokumentbeginn, weil *dur* als 12s angegeben ist.

```
<circle r="15">
<animateMotion dur="12s" begin="6s; 60s"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Die Animation beginnt nach 6s, endet nach der einfachen Animationsdauer, also nach 18s und startet erneut nach 60s und endet wieder nach 72s.

```
<circle r="15">
<animateMotion dur="12s" begin="-6s" end="-2.1s"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Weil sowohl *begin* als auch *end* vor Dokumentbeginn liegen, startet die Animation nicht.

```
<circle r="15">
<animateMotion dur="12s" begin="5s" end="30s"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Die einfache Animationsdauer ist kürzer als die Differenz zwischen *begin* und *end*, daher wird die aktive Animationsdauer nicht eingeschränkt.

```
<circle r="15">
<animateMotion dur="12s" begin="5s" end="6s"
```

```
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Die einfache Animationsdauer ist länger als die Differenz zwischen *begin* und *end*, daher wird die aktive Animationsdauer auf die Differenz von einer Sekunde eingeschränkt.

```
<circle r="15">
<animateMotion dur="12s" begin="10s" end="5s"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Das einzige *end* liegt vor dem einzigen *begin*, die Animation wird nicht gestartet.

```
<circle r="15">
<animateMotion dur="12s" begin="2s;6s" end="10s;12s"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Die Animation beginnt 2 Sekunden nach Dokumentbeginn und erneut 6s nach Dokumentbeginn und endet 10 Sekunden nach Dokumentbeginn. Der zweite Endwert von 12s hat keine Auswirkungen, weil zu dem Zeitpunkt die Animation nicht mehr aktiv ist.

```
<circle r="15">
<animateMotion dur="12s" begin="2s;20s" end="10s;25s"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Die Animation beginnt 2s nach Dokumentbeginn und endet 10s nach Dokumentbeginn, beginnt erneut 20s nach Dokumentbeginn und endet 5s später.

Abb. 228 Einfacher Zeitversatz

Einfacher Zeitversatz[17]

Verschiedene Methoden, Anfang und Ende einer Animation mit einem Zeitversatz relativ zum Dokumentbeginn anzugeben.

Alle blauen Rechtecke vergrößern ihre Höhe, so dass sie immer exakt in den dunkelblauen Rahmen passen, der von diesen immer komplett ausgefüllt wird.

Wird etwas Rotes sichtbar oder passen die blauen Rechtecke nicht in den Rahmen, ist ein Fehler aufgetreten.

17 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_offset01.svg

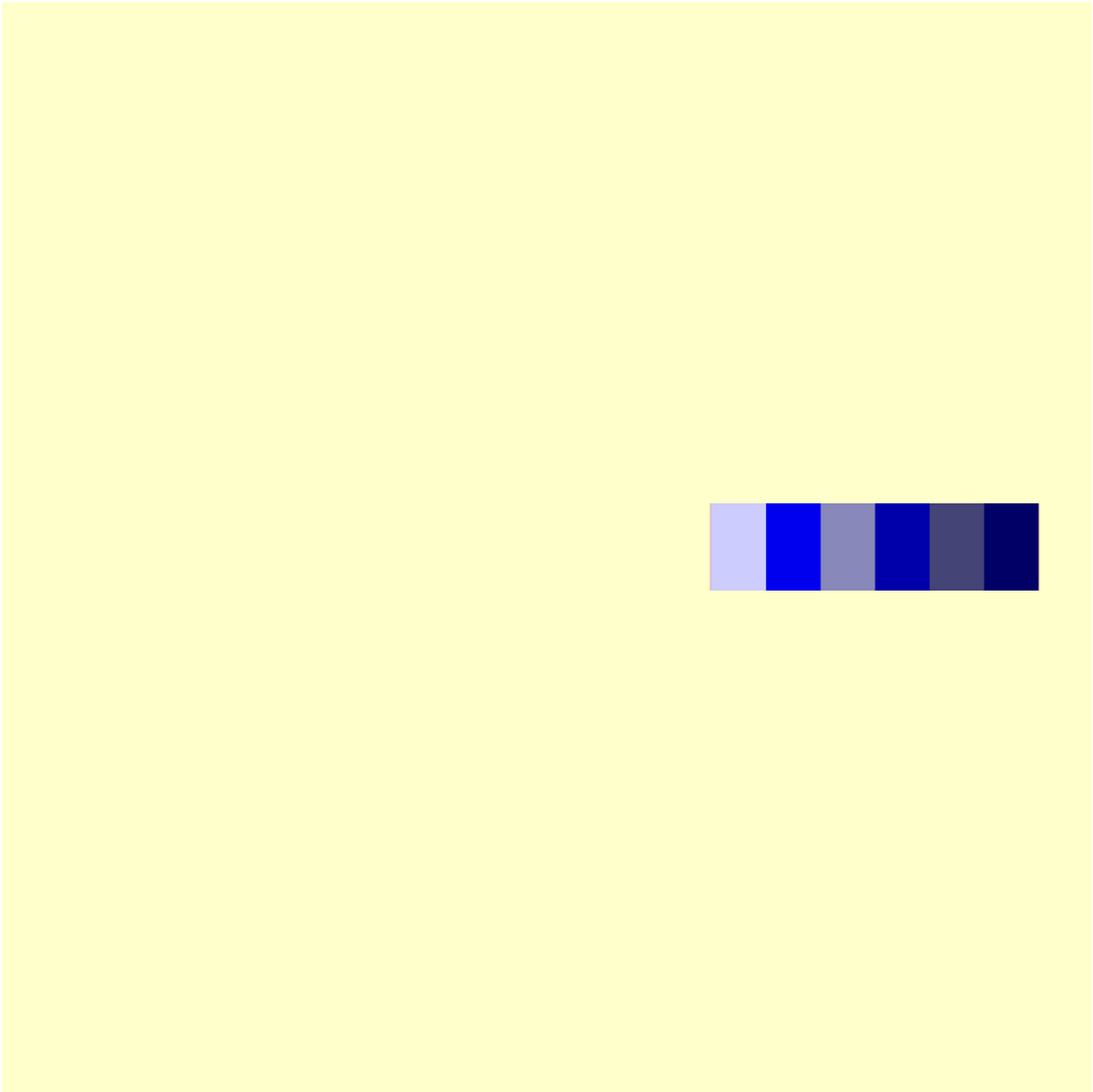


Abb. 229 Zeitversatz, kombinierte Zeitangaben

Zeitversatz, kombinierte Zeitangaben[18]

Verschiedene Methoden, Anfang und Ende einer Animation mit einem Zeitversatz relativ zum Dokumentbeginn anzugeben.

Verwendet werden vor allem Notationen mit kombinierten Angaben von Sekunden, Minuten und Stunden.

Die Animationen beginnen 74.0s Sekunden nach Dokumentbeginn und enden 740.4 Sekunden (12.34 Minuten) nach Dokumentbeginn.

18 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_offset02.svg

Die blauen Rechtecke drehen sich mit der gleichen Winkelgeschwindigkeit. Wird etwas Rotes sichtbar, ist ein Fehler aufgetreten.

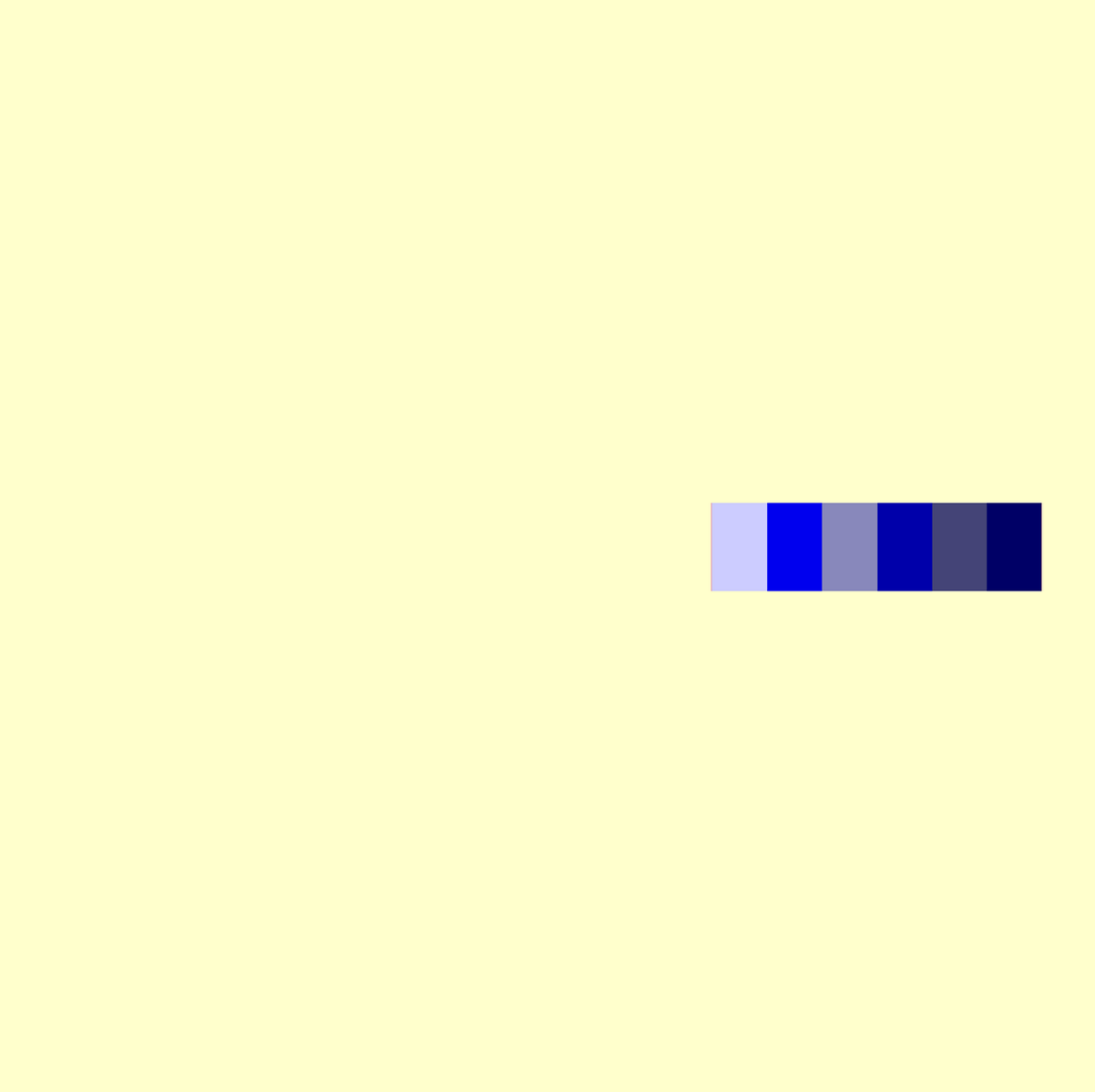


Abb. 230 Zeitversatz, kombinierte Zeitangaben

Zeitversatz, kombinierte Zeitangaben[19]

Verschiedene Methoden, Anfang und Ende einer Animation mit einem Zeitversatz relativ zum Dokumentbeginn anzugeben.

Verwendet werden vor allem Notationen mit kombinierten Angaben von Sekunden, Minuten und Stunden.

19 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_offset03.svg

Die Animationen beginnen 74.0s Sekunden vor Dokumentbeginn und enden 740.4 Sekunden (12.34 Minuten) nach Dokumentbeginn. Negative Zeiten für end sind auch angegeben, da diese allerdings allesamt vor dem Beginn der Animation liegen und ein weiterer Wert dahinter liegt, sind die negativen Endwerte nicht anwendbar.

Die blauen Rechtecke drehen sich mit der gleichen Winkelgeschwindigkeit. Wird etwas Rotes sichtbar, ist ein Fehler aufgetreten.

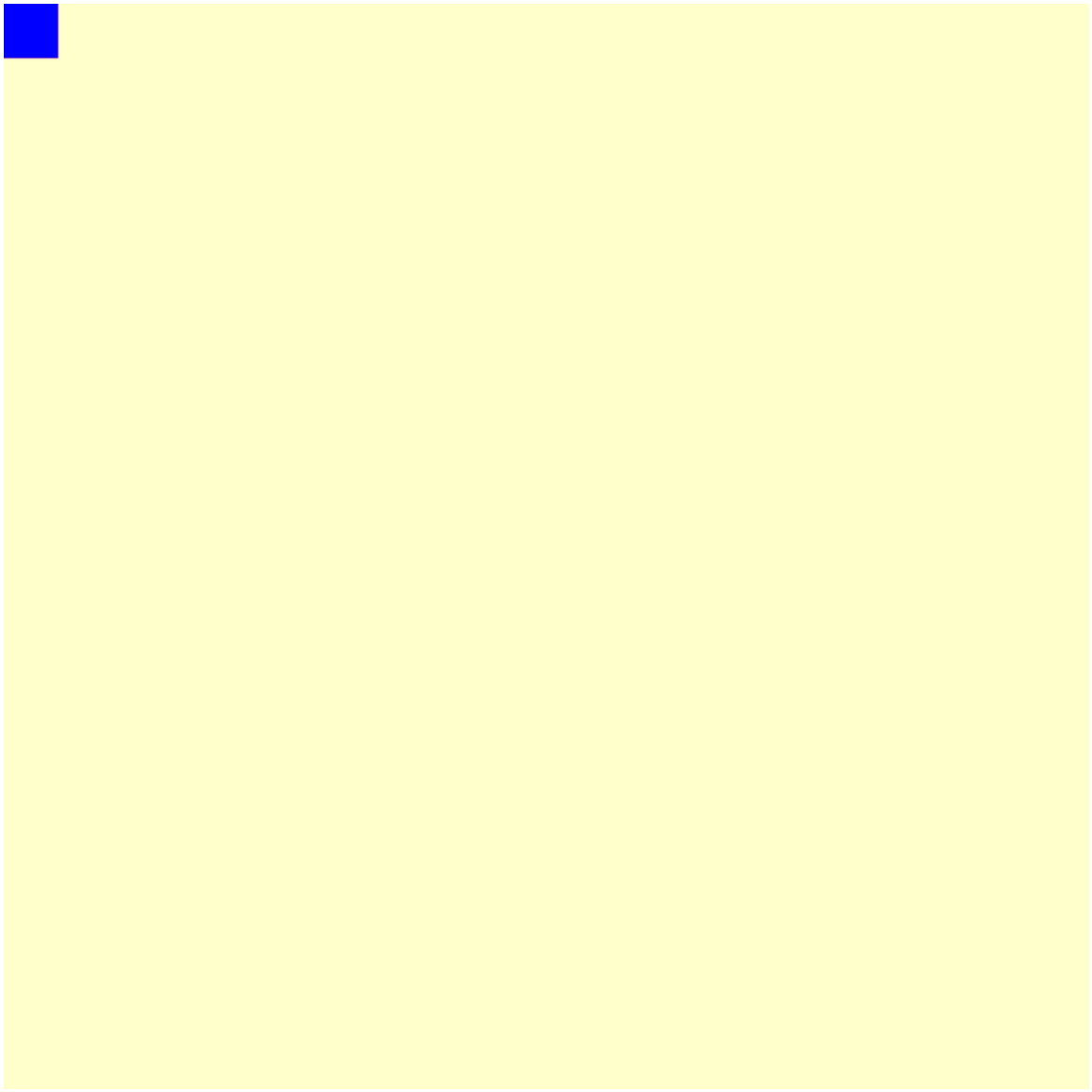


Abb. 231 Zeitversatz, Listen

Zeitversatz, Listen[20]

20 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_offset04.svg

Ein blaues Quadrat wird entlang eines quadratischen Pfades bewegt. Listen für *begin* und *end* werden notiert.

Die Animationen werden verglichen mit Einzelanimationen eines roten Quadrates, welches sich immer direkt hinter dem blauen befindet.

Wird etwas Rotes sichtbar, ist ein Fehler aufgetreten.

14.4.4 Synchronisationswert

Squiggle (Batik)	1.7 (für *end* nur teilweise)
Opera (Presto)	9 (nur teilweise für *end* und ab 8 teilweise für *begin*)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4 (teilweise)
Konqueror (KSVG)	3.2 (teilweise)
Safari (Webkit)	4 (teilweise, gar nicht für *end*)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit einem Synchronisationswert wird ein anderes Animationselement zur Synchronisation verwendet. Beim zur Synchronisation zu verwendendenden Animationselement ist ein Attribut *id* (oder auch *xml:id in SVG tiny 1.2)* mit einem Fragmentidentifizierer zu notieren.

Als Synchronisationswert ist dann der Fragmentidentifizierer zu notieren, dann ein Punkt '.' und dann eines der Schlüsselwörter 'begin' oder 'end' nach optionalen Leerzeichen kann dann optional ein Zeitversatz notiert werden, dessen genaue Syntax bereits beschrieben wurde.

Einfache Beispiele:

```
<circle r="15">
<animate id="bsp" attributeName="stroke" values="red" begin="20s;120s"
 end="25s;130s" />
<animateMotion dur="12s" begin="bsp.begin+2s;bsp.end-1s"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Mit ***animate*** wird der Strich des Kreises 20s nach Dokumentbeginn auf rot gesetzt. Dies endet mit der Endzeit von 25s. Ein weiterer Beginn liegt bei 120s mit einer Endzeit bei 130s.

begin und *end* werden zur Synchronisation des Beginns von ***animateMotion*** verwendet. Der Beginn ist festgelegt auf zwei Sekunden nach jedem Beginn der Strich-Animation und eine Skunde vor dem Ende der Strichanimation. Die Bewegung beginnt also in Dokumentzeit bei 22s, 24s, 122s, 129s.

```
<circle r="15">
<animate id="bsp" attributeName="stroke" values="red" begin="20s;120s"
 end="25s;130s" />
<animateMotion dur="12s" begin="bsp.end+2s" end="bsp.begin+15s"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Bei diesem Beispiel dient wieder die Strich-Animation zur Synchronisation. Die Angabe zu *begin* ist äquivalent zu begin="27s;132s", die Angabe zu *end* ist äquivalent zu end="35s;135s".

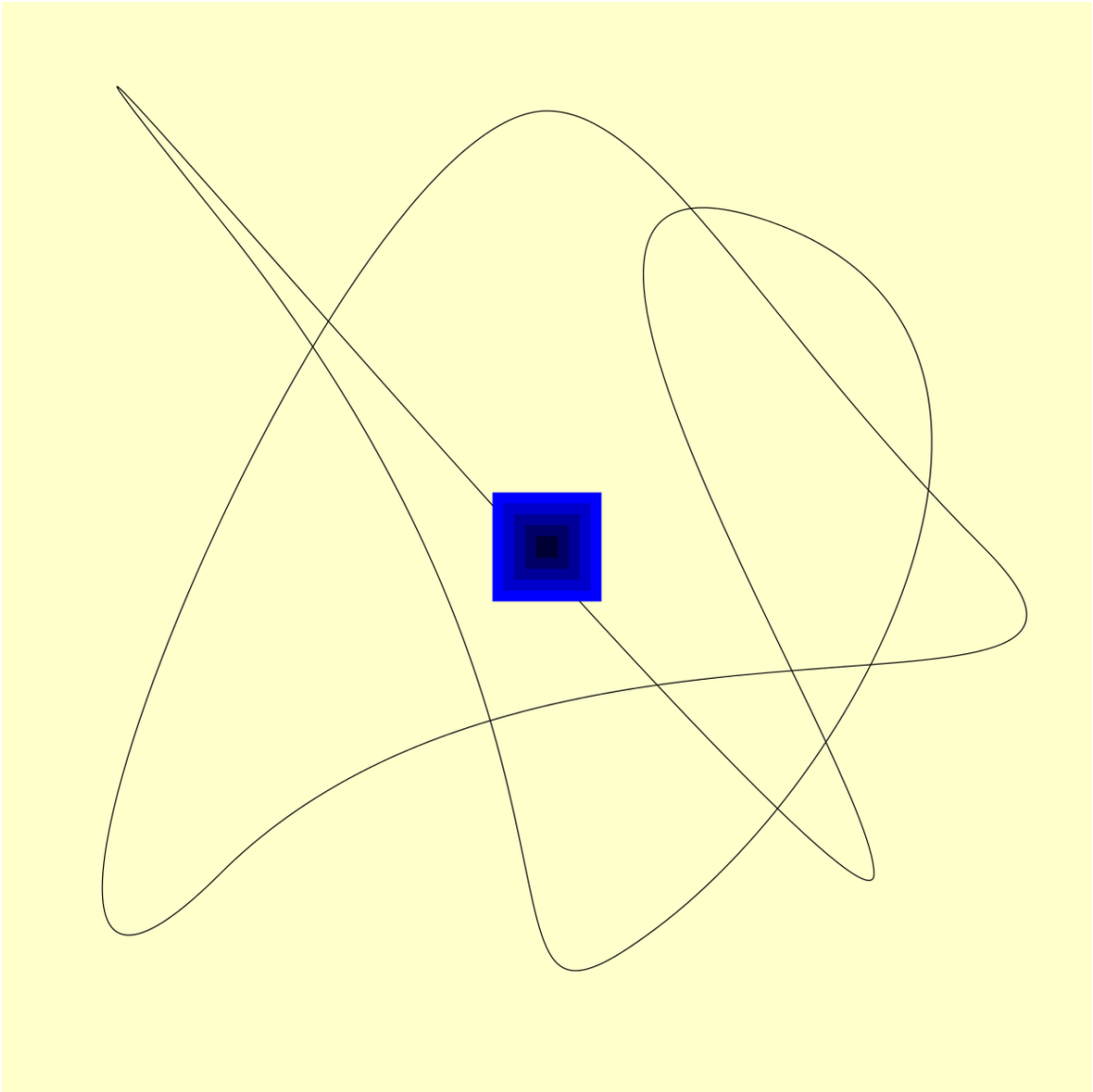


Abb. 232 Schleife durch gekoppelte Animationen

Schleife durch gekoppelte Animationen[21]

Blaue Quadrate werden entlang eines Pfades bewegt.

Für Beginn und Ende werden zum großen Teil Synchronisationswerte mit Zeitversatz verwendet. Durch eine geschlossene Schleife mit zwei Animationen wird gewährleistet, dass die Animationen immer wieder gestartet werden.

21 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_syncbase01.svg

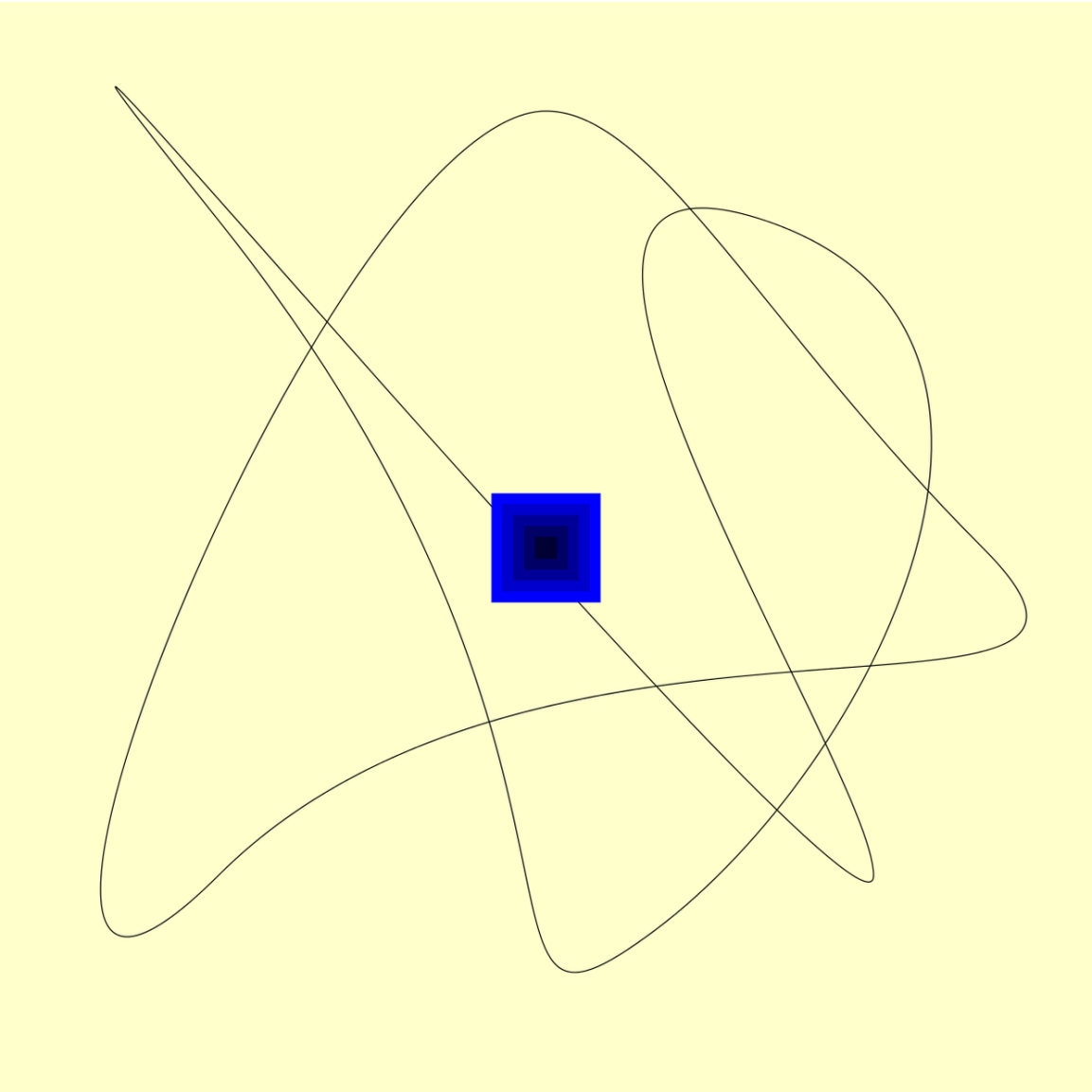


Abb. 233 Schleifen und gekoppelte Animationen

Schleifen und gekoppelte Animationen[22]

Blaue Quadrate werden entlang eines Pfades bewegt.

Für Beginn und Ende werden zum großen Teil Synchronisationswerte mit Zeitversatz verwendet. Durch zwei geschlossene Schleifen mit jeweils zwei Animationen wird gewährleistet, dass die Animationen immer wieder gestartet werden. Die fünfte Animation bezieht Anfang und Ende von beiden Schleifen, hat daher keinen kurzzeitig gleichmäßigen wiederholten Ablauf.

22 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_syncbase02.svg

14.4.5 Ereigniswert

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	9 (teilweise, in geringem Umfange ab 8)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4-8 (teilweise) ab 9 nicht mehr
Konqueror (KSVG)	-
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	7 (teilweise)
Microsoft Internet Explorer (Trident)	
librsvg	

Animationen können auch mit Ereignissen begonnen oder beendet werden. Dazu werden Ereigniswerte notiert.

Es kann ebenfalls notiert werden, welches Element der Empfänger des Ereignisses sein soll. Die Angabe ist optional. Erfolgt sie nicht, ist das Zielelement der Animation auch der Empfänger des Ereignisses (nicht zwangsläufig das Elternelement).

Beim Empfänger des Ereignisses ist ansonsten ein Attribut *id* (oder auch *xml:id* in SVG tiny 1.2) mit einem Fragmentidentifizierer zu notieren. Als Empfänger des Ereignisses ist dann optional der Fragmentidentifizierer zu notieren, dann ein Punkt '.' und dann eines der Schlüsselwörter für ein Ereignis. Nach optionalen Leerzeichen kann dann optional ein Zeitversatz notiert werden, dessen genaue Syntax bereits beschrieben wurde.

Autoren sollten immer bedenken, dass die späteren Nutzer eines Dokumentes nicht von allein erkennen können, wie eine Animation gestartet wird, es wird daher in der Regel sinnvoll sein, gewisse Konventionen zu vermitteln oder Beschreibungen beizufügen, wie ein interaktives Dokument genutzt werden kann. Der spätere Nutzer wird die Gedankengänge des Autors nur selten erahnen können. Was der Autor für selbstverständlich oder effektiv hält, muss für den Nutzer noch lange nicht klar sein. Eine für den Autor eingängige Symbolik mag für einige Nutzer komplett unverständlich sein. Der Autor sollte nicht davon ausgehen, dass seine Vorstellungen objektivierbar sind und für den Nutzer nachvollziehbar sind. Auch an sich geniale und ergonomische Lösungen bedürfen oft erst einmal Hilfen, um den Nutzer an die prinzipielle Idee heranzuführen, um dann auch wirklich von der Lösung profitieren zu können.

Für Ereignisse gibt es folgende Schlüsselwörter, einschließlich einer Erklärung, wann sie eintreten:

activate

Aktivierung eines Elementes, zum Beispiel mit einem Mausklick oder mit der Tastatur. Dies ist ein geräteunabhängiges Ereignis und ist daher hinsichtlich der Zugänglichkeit eine bessere Wahl als ein geräteabhängiges Ereignis.

mousedown

Knopf eines Zeigergerätes wird über dem Element gedrückt.

mouseup

Knopf eines Zeigergerätes wird über dem Element losgelassen.

click

Klickereignis bei einem Zeigergerät über dem Element, also über dem selben Element erst onmousedown dann onmouseup. In der Reihenfolge erfolgt onclick dann unmittelbar nach dem onmouseup.

mouseover

Ein Zeigergerät wird auf das Element bewegt, so dass es zuvor nicht über dem Element ist und dann über dem Element. Nicht gemeint ist, dass das Element sich unter das Zeigergerät bewegt.

mousemove

Ein Zeigergerät wird bewegt, während es sich über dem Element befindet.

mouseout

Ein Zeigergerät wird von einem Element wegbewegt, so dass es zuvor über dem Element ist und dann nicht mehr darüber. Nicht gemeint ist, dass sich das Element unter dem Zeigergerät wegbewegt.

focusin

Ein Element bekommt den Fokus, zum Beispiel wenn ein Text ausgewählt wird. In SVG 1.1 können alle Elemente ausgewählt werden. In SVG tiny 1.2 kann auch explizit angegeben werden, dass Elemente nicht auswählbar sind.

focusout

Ein Element verliert den Fokus, zum Beispiel wenn ein Text ausgewählt war und dann nicht mehr ausgwählt ist (weil zum Beispiel ein anderes ausgewählt wird).

SVGLoad (in tiny 1.2

load):Der Dokumentbeginn.

SVGUnload

Das Ende des Dokumentes, wenn dieses aus dem Anzeigebereich entfernt wird (nicht in tiny 1.2).

SVGAbort

Abbruch des Ladens des Dokumentes bevor dies komplett ist (nicht in tiny 1.2).

SVGError

Fehler beim Laden oder einer Skriptausführung (nicht in tiny 1.2).

SVGResize (in tiny 1.2

resize):Die Größe des Anzeigebereiches wird verändert.

SVGScroll (in tiny 1.2

scroll):Der Anzeigebereich wird gerollt, die Darstellung wird also relativ zum Anzeigebereich durch eine Nutzerinteraktion mit dem Darstellungsprogramm verschoben.

SVGZoom (in tiny 1.2

zoom):Die Darstellung des Dokumentes wird vergrößert oder verkleinert.

beginEvent

Beginn eines Animationselementes.

endEvent

Ende eines Animationselementes.

repeatEvent

Wiederholung einer Animation, jedesmal, wenn sich die Animation wiederholt, nach dem ersten Durchlauf.

rotate (nur tiny 1.2)

Das Dokument wird gedreht.

Beispiele:

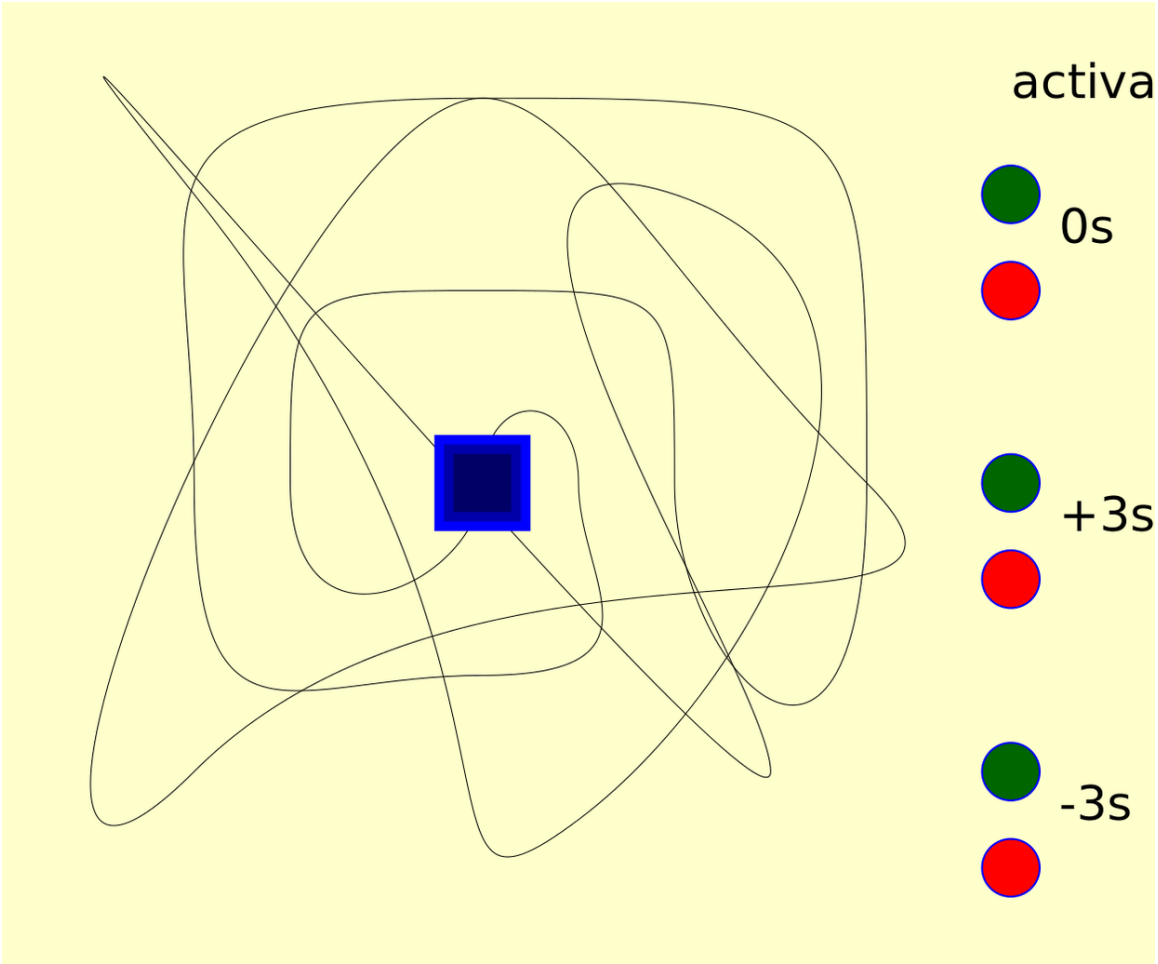


Abb. 234 *begin* und *end* mit 'activate'

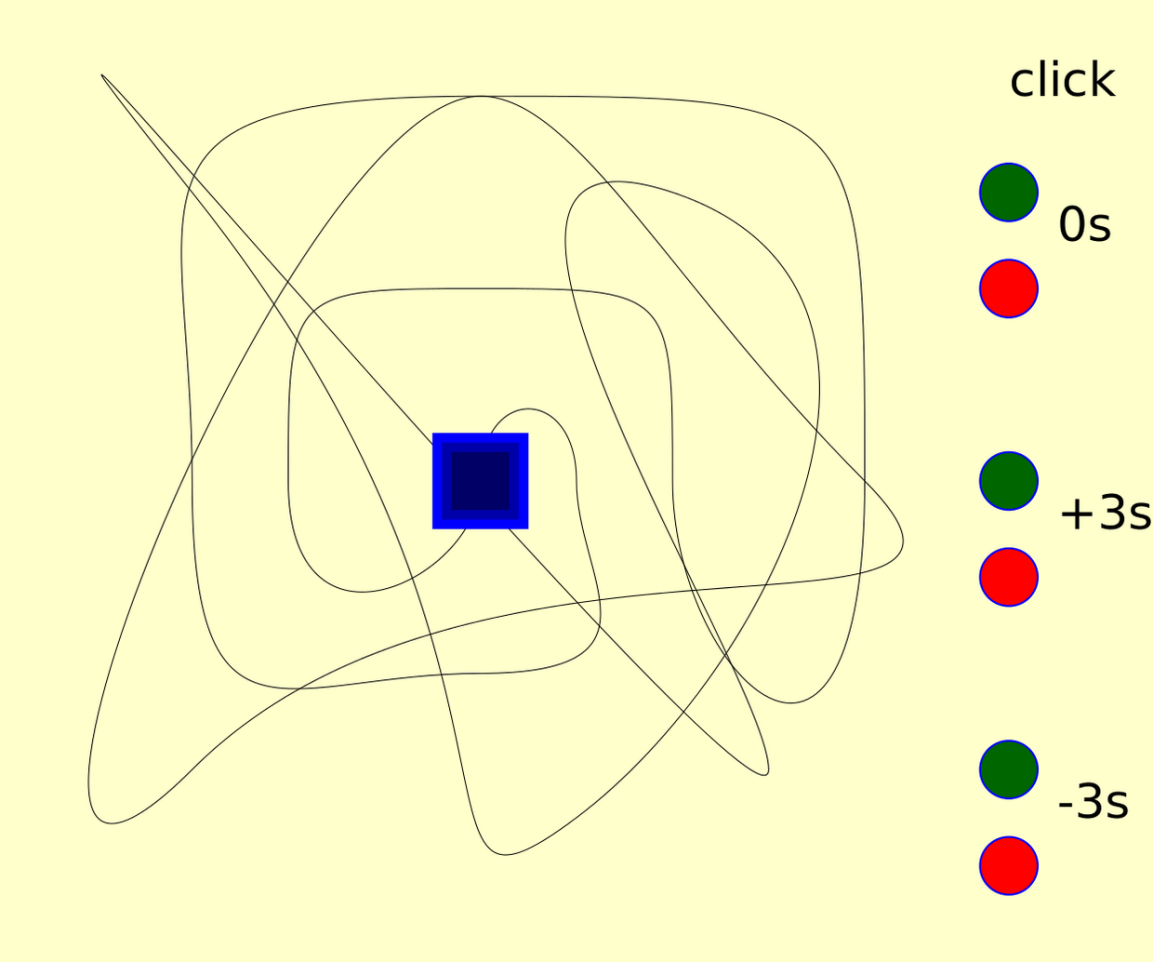


Abb. 235 *begin* und *end* mit 'click'

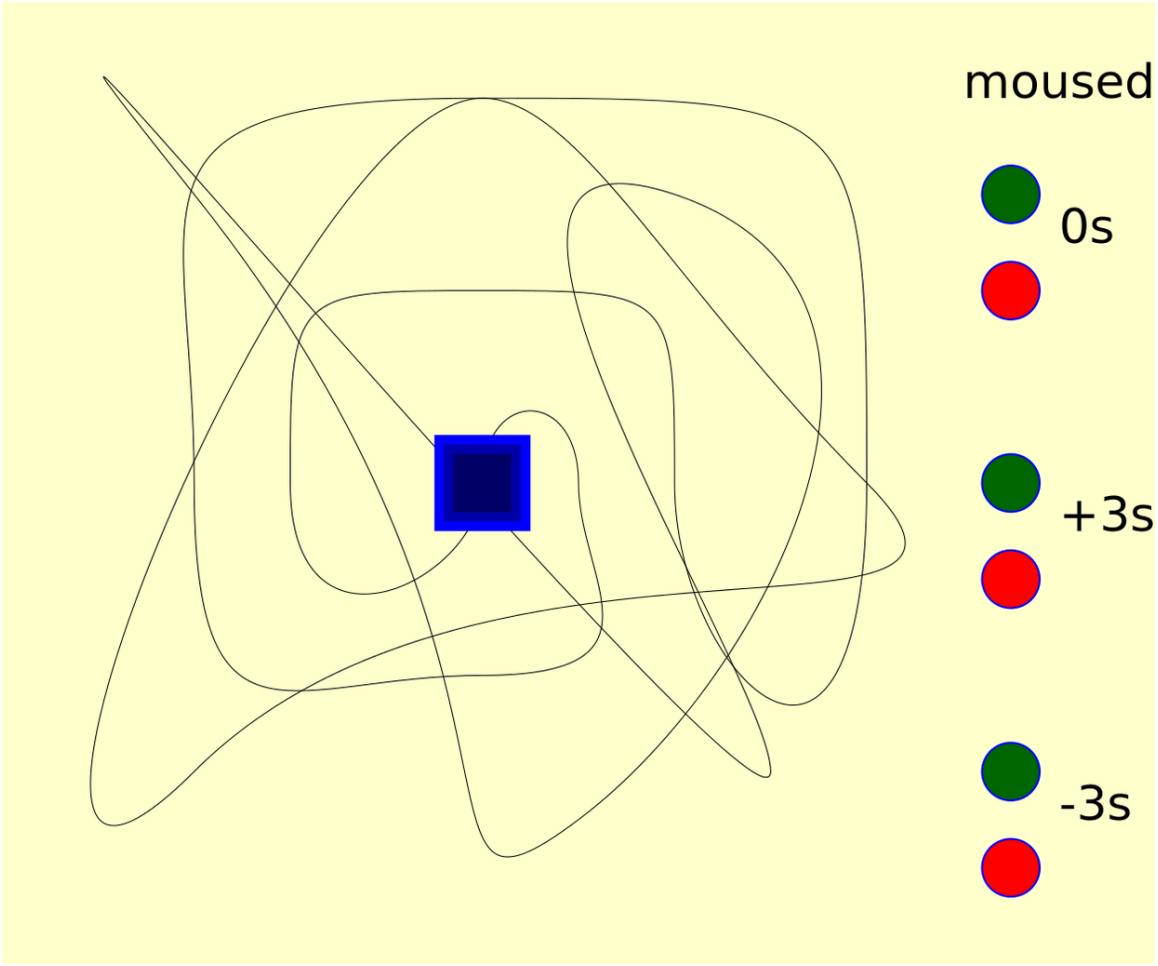


Abb. 236 *begin* und *end* mit 'mousedown'

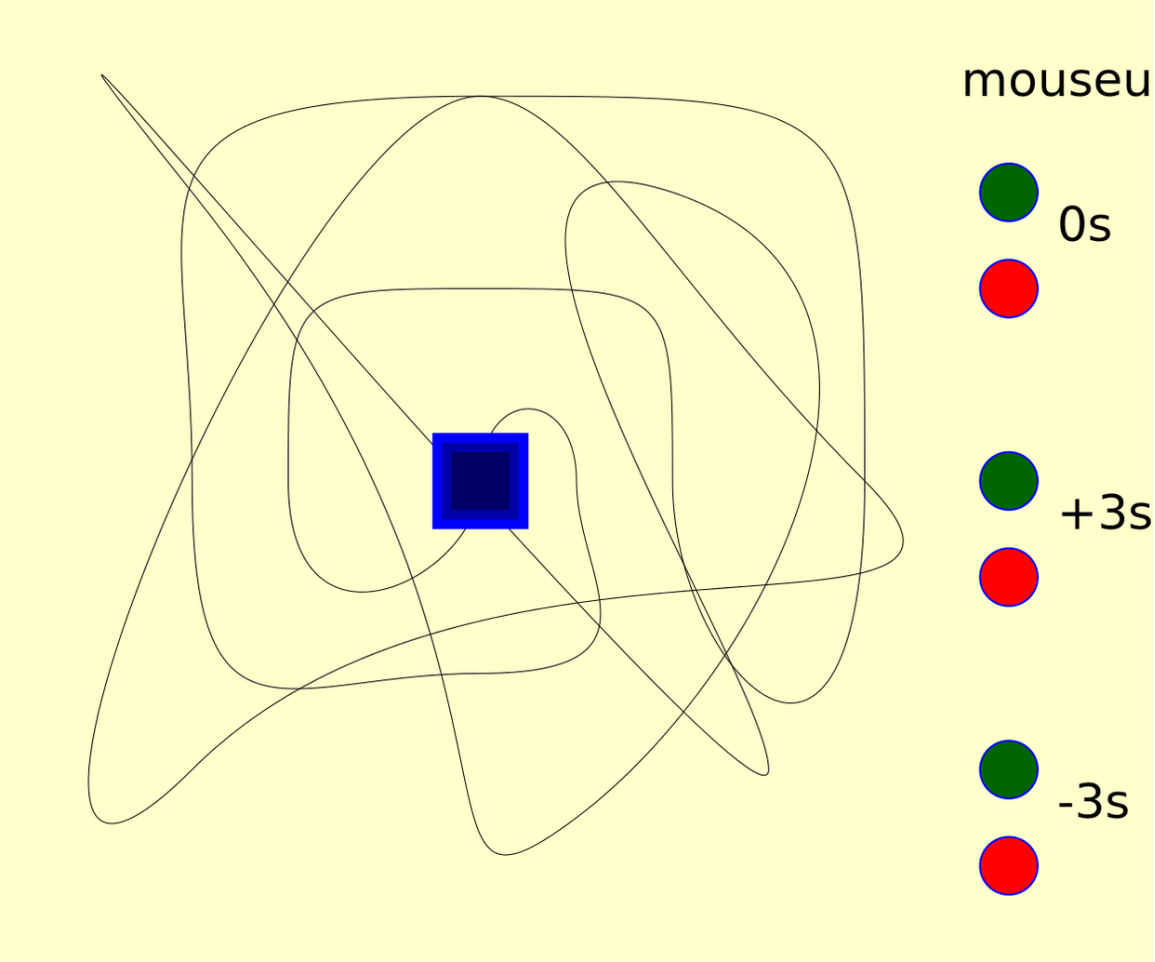


Abb. 237 *begin* und *end* mit 'mouseup'

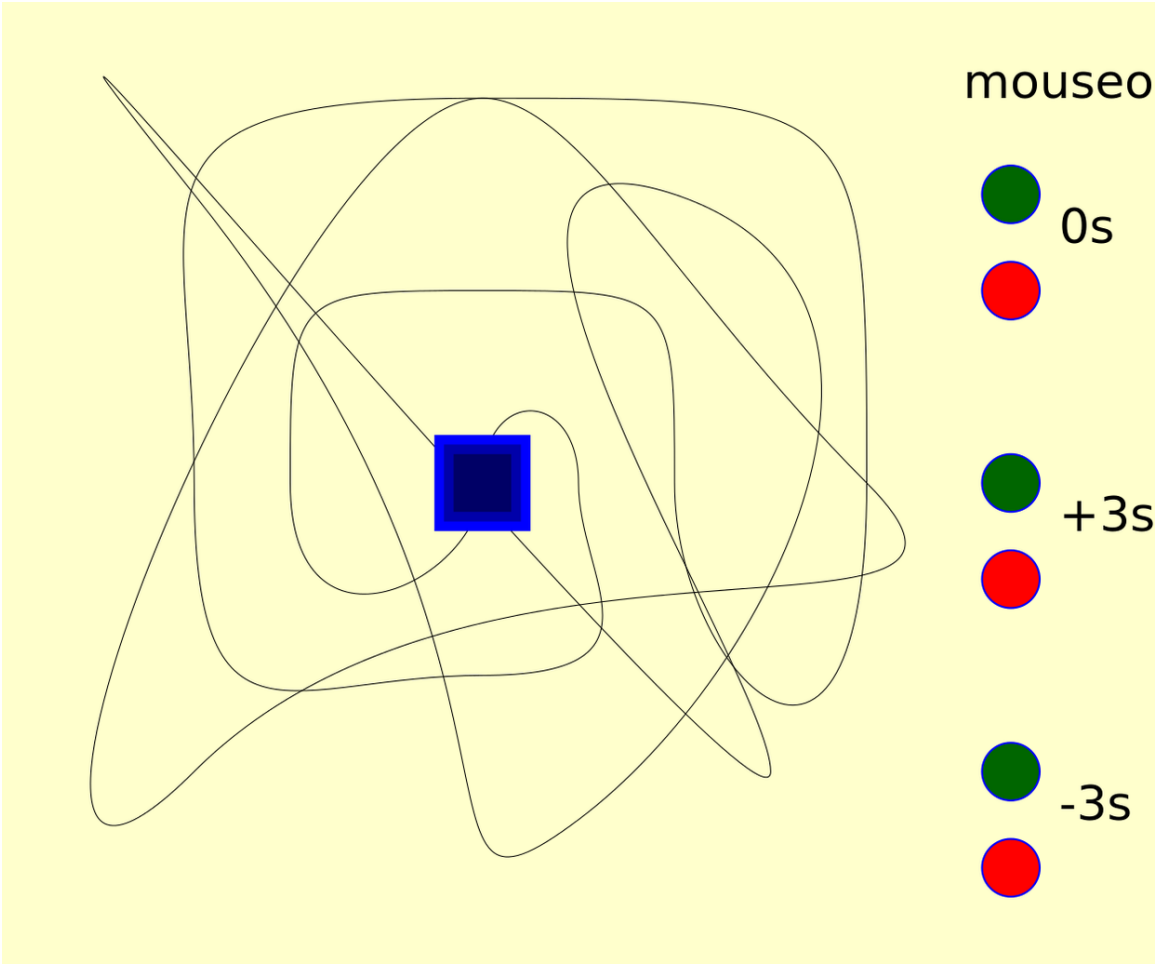


Abb. 238 *begin* und *end* mit 'mouseover'

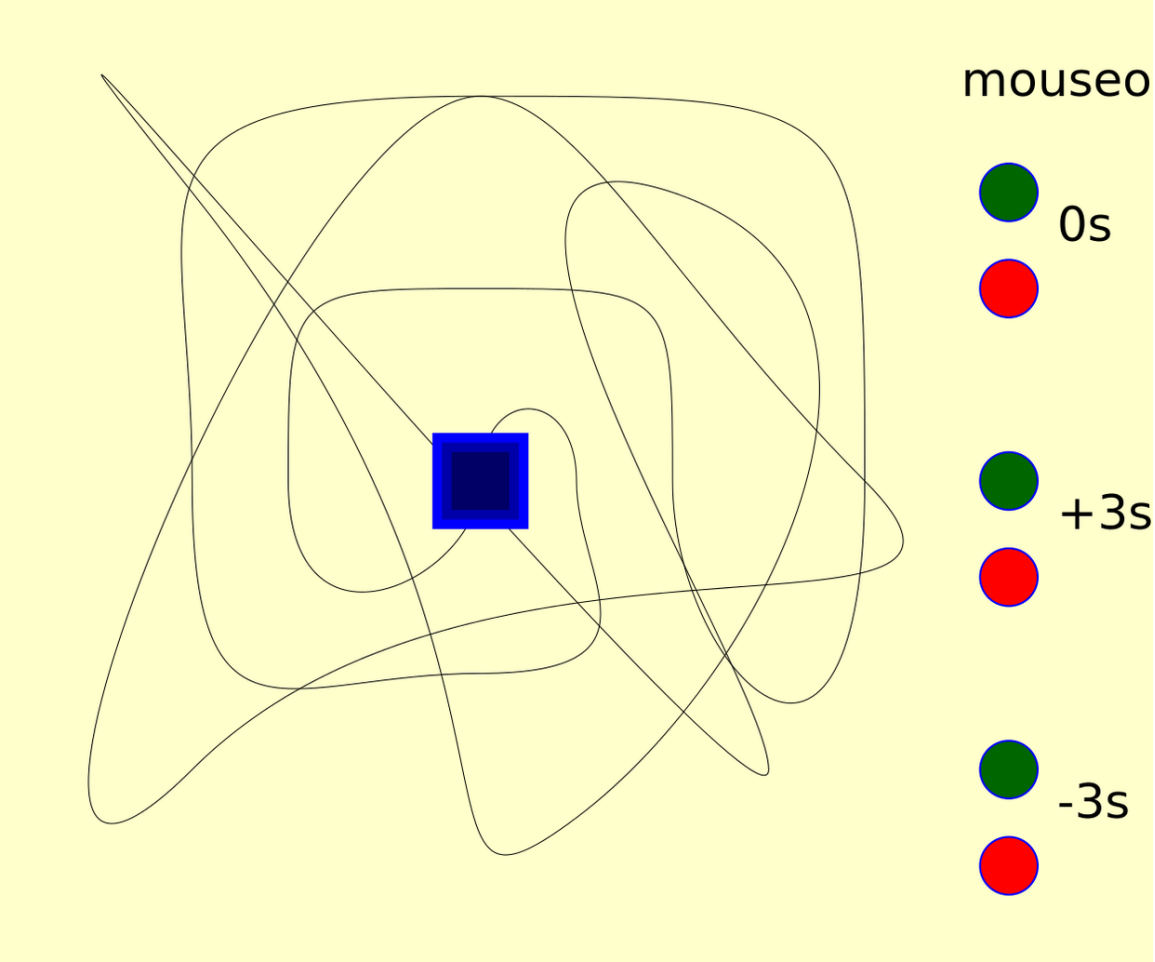


Abb. 239 *begin* und *end* mit 'mouseout'

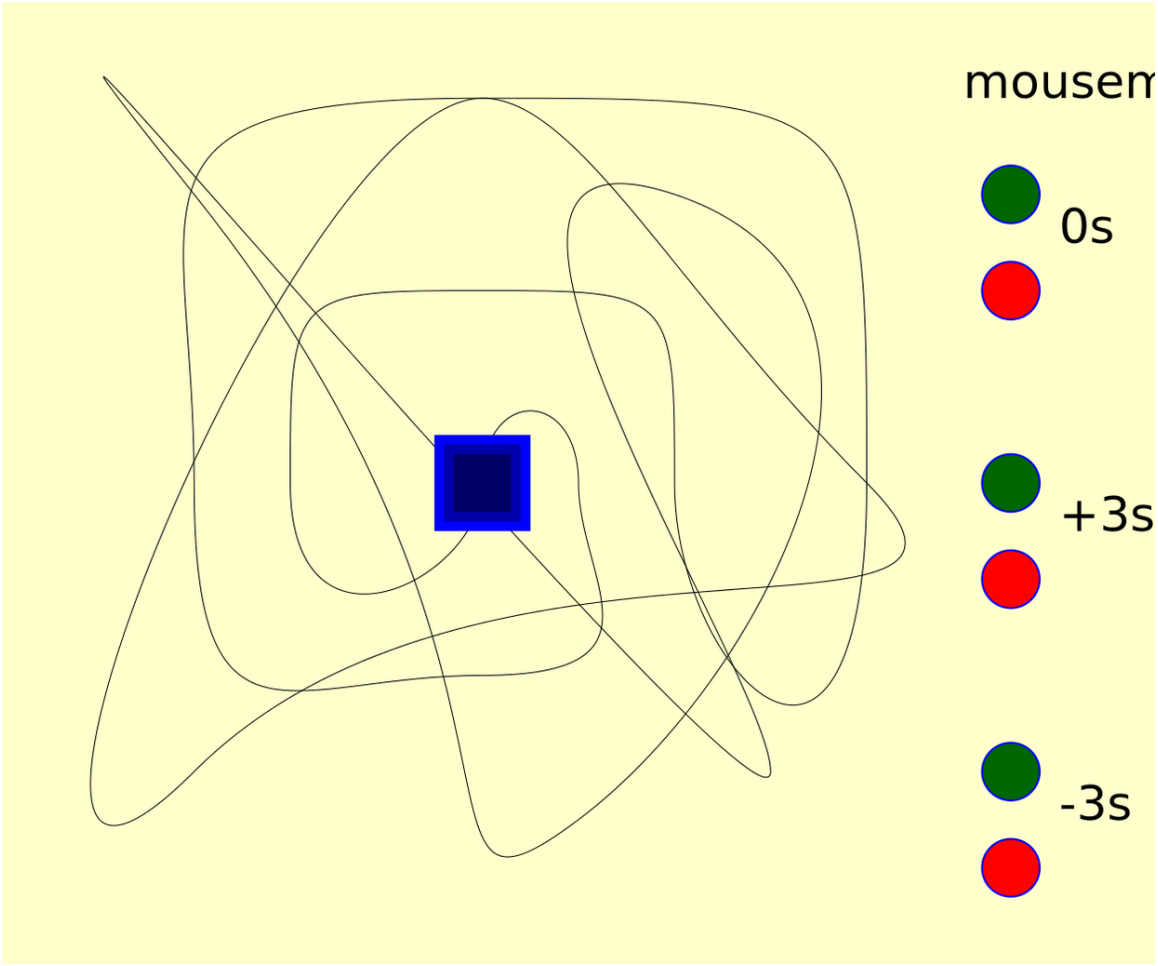


Abb. 240 *begin* und *end* mit 'mousemove'

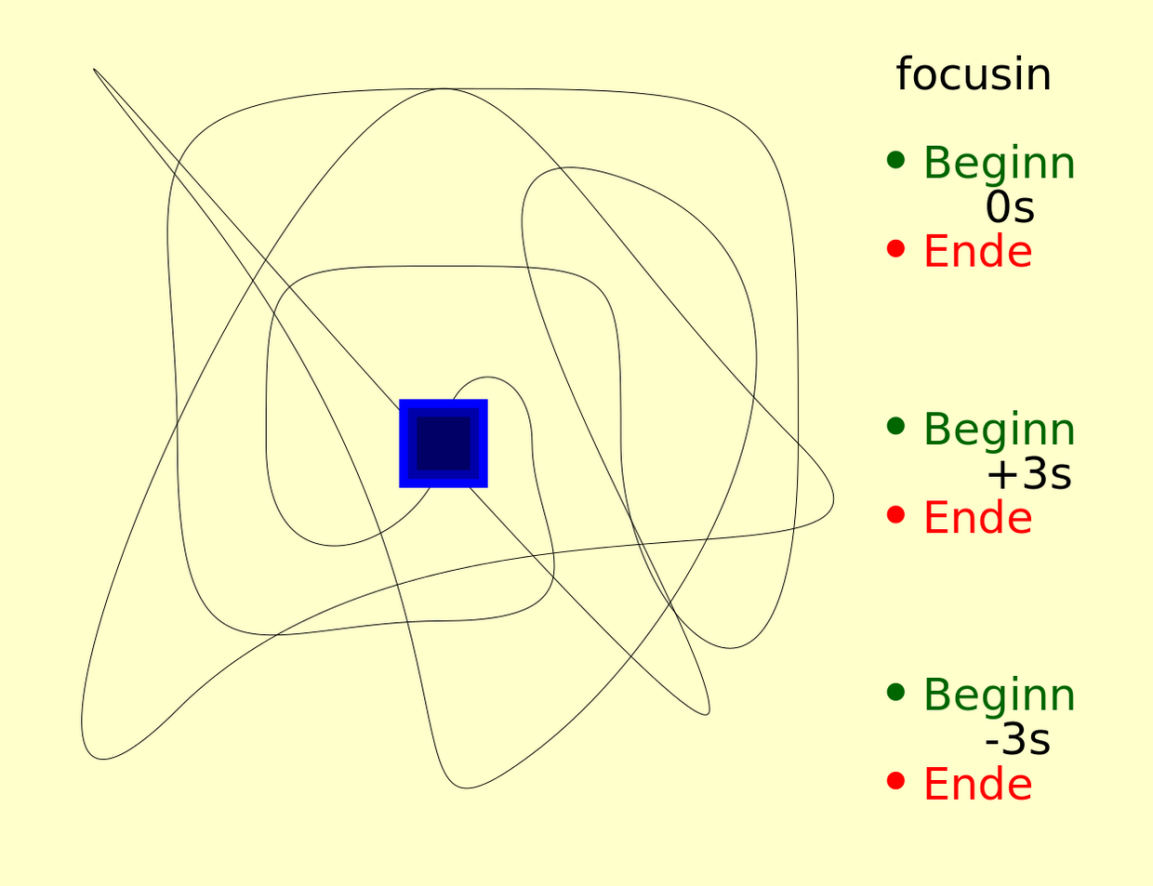


Abb. 241 *begin* und *end* mit 'focusin'

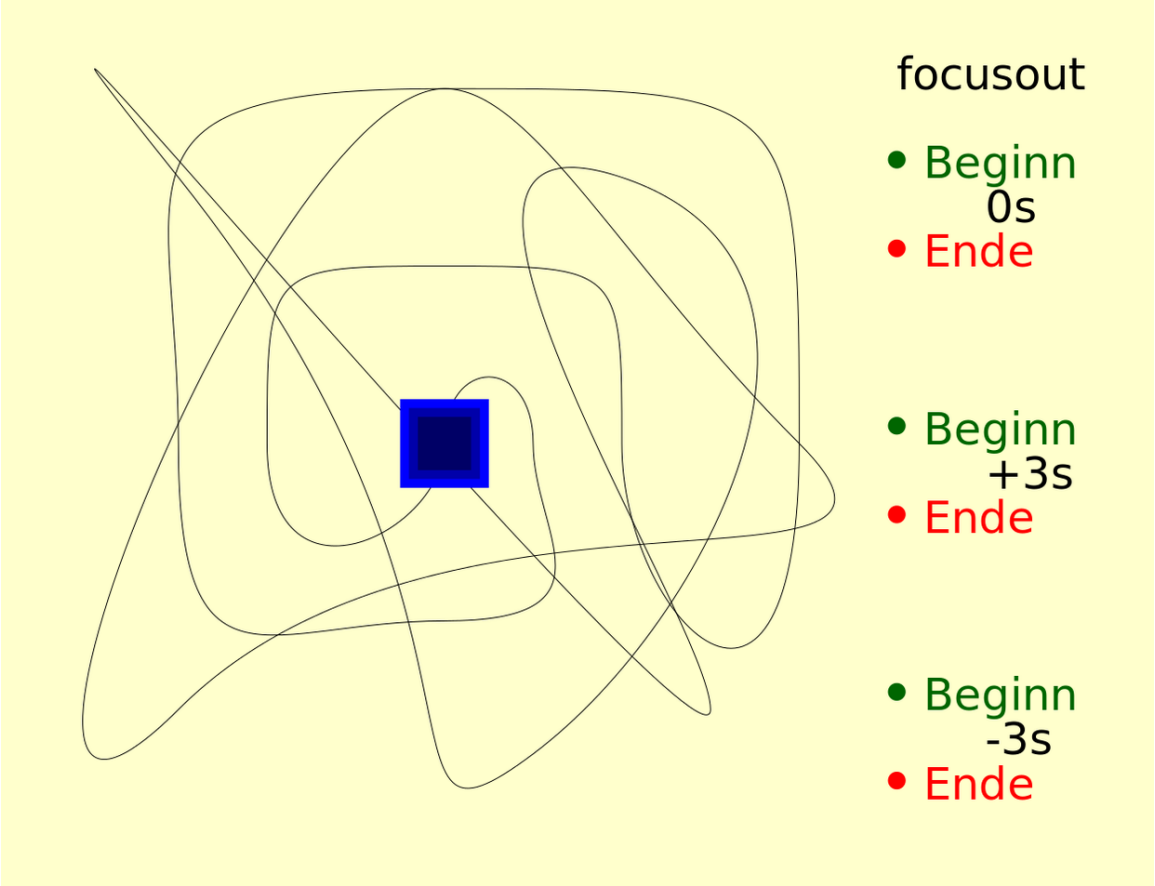


Abb. 242 *begin* und *end* mit 'focusout'

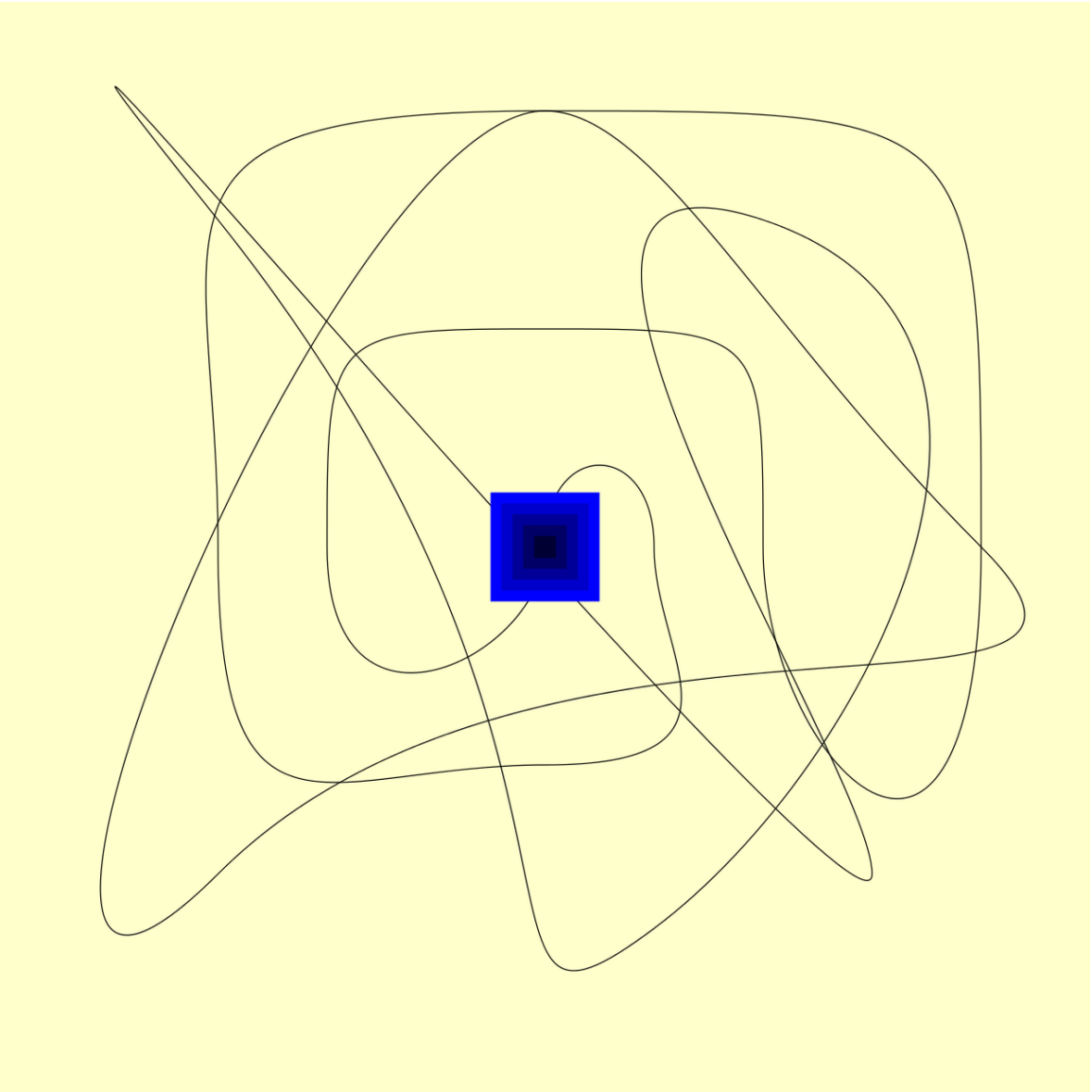


Abb. 243 *begin* und *end*, weitere Ereignisse (tiny 1.2-Variante)

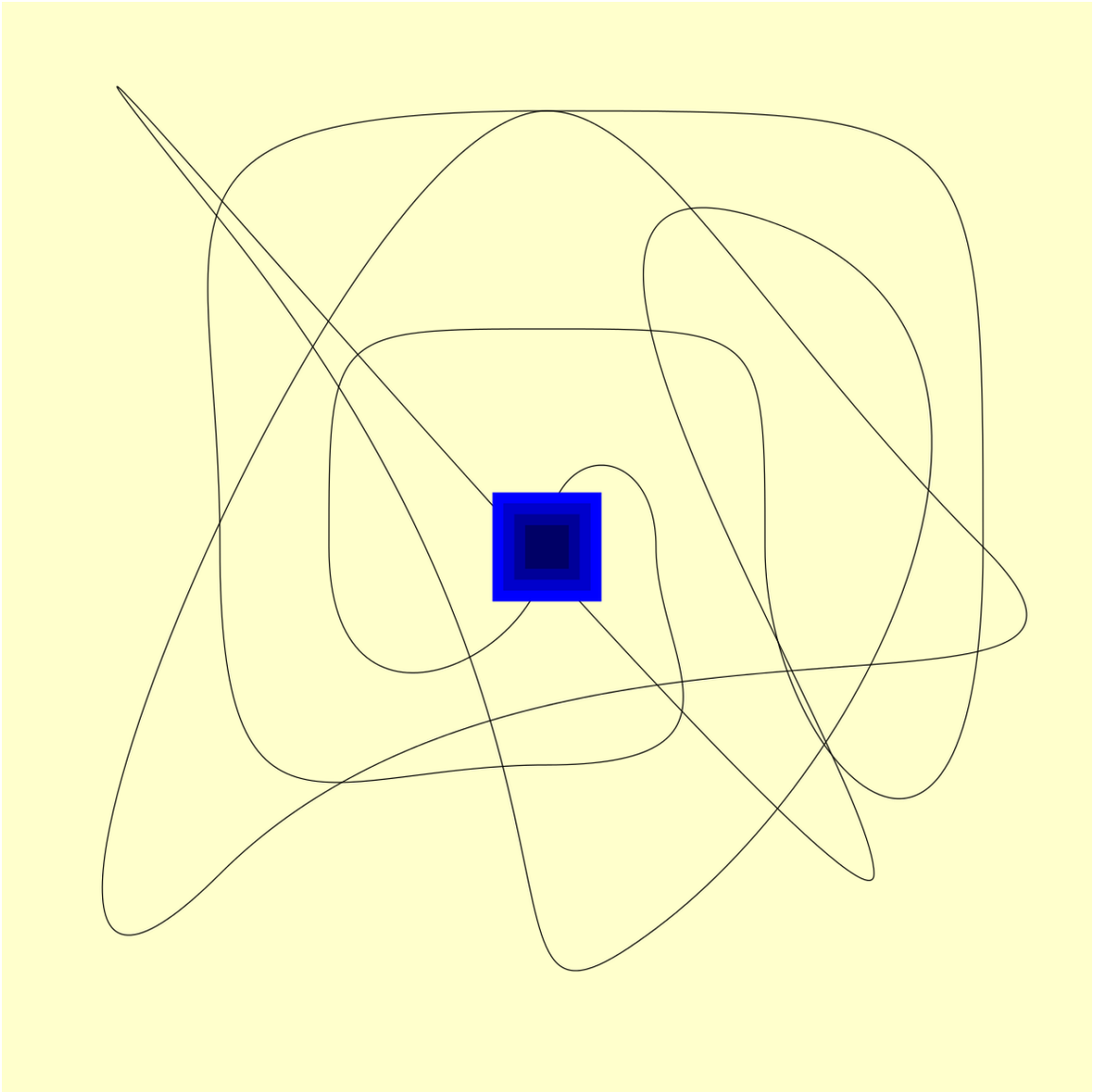


Abb. 244 *begin* und *end*, weitere Ereignisse (2)

```
<circle r="15">
<animateMotion dur="12s" begin="activate;click"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Durch aktivieren des Kreises oder klickern mit dem Zeigergerät auf den Kreis wird die Animation begonnen. Der Vorteil von 'activate' ist, dass die Animation auch begonnen werden kann, wenn kein Zeigergerät verfügbar ist. Der Vorteil, zusätzlich 'click' hinzuzufügen, besteht darin, dass so die Animation auch Nutzern zugänglich ist, die ein Darstellungsprogramm verwenden, welches aufgrund eines Fehlers 'activate' nicht verfügbar haben, dies ist zum Beispiel der Fall bei Opera bis mindestens Version 10.

```
<circle cx="50" r="15" fill="blue"
        xmlns:xlink="http://www.w3.org/1999/xlink">
<animateMotion xlink:href="#c" dur="12s" begin="activate;click"
```

```
     values="0,300;400,0;400,300;0,300" />
</circle>
<circle id="c" r="15" fill="green" />
```

Durch aktivieren/anklickern des grünen (!) Kreises wird die Animation des grünen Kreises aktiviert.

```
<circle id="b" cx="50" r="15" fill="blue"
        xmlns:xlink="http://www.w3.org/1999/xlink">
<animateMotion xlink:href="#c" dur="12s" begin="b.activate;b.click"
     values="0,300;400,0;400,300;0,300" />
</circle>
<circle id="c" r="15" fill="green" />
```

Durch aktivieren/anklickern des blauen (!) Kreises wird die Animation des grünen (!) Kreises aktiviert.

```
<circle id="b" r="15" fill="blue">
<animateMotion dur="12s" begin="c.activate;c.click"
     values="0,300;400,0;400,300;0,300" />
</circle>
<circle id="c" cx="50" r="15" fill="green" />
```

Durch aktivieren/anklickern des grünen (!) Kreises wird die Animation des blauen (!) Kreises aktiviert.

```
<circle r="15" fill="blue">
<animateMotion dur="12s" begin="c.mousdown" end="c.mouseup"
     values="0,300;400,0;400,300;0,300" />
</circle>
<circle id="c" cx="50" r="15" fill="green" />
```

Drücken des Zeigergerätes über dem grünen Kreis startet die Animation des blauen Kreises. Ein Loslasses des Zeigergerätes über dem grünen Kreis beendet die Animation vorzeitig, sofern diese noch aktiv ist.

```
<circle r="15" fill="blue">
<animateMotion dur="12s" begin="c.mousdown + 2s" end="c.mouseup+2s"
     values="0,300;400,0;400,300;0,300" />
</circle>
<circle id="c" cx="50" r="15" fill="green" />
```

Das Gleiche wie zuvor, nur jeweils mit einer Verzögerung von zwei Sekunden.

```
<circle r="15" fill="blue">
<animateMotion dur="12s" begin="c.mousdown - 2s" end="c.mouseup-2s"
     values="0,300;400,0;400,300;0,300" />
</circle>
<circle id="c" cx="50" r="15" fill="green" />
```

Das Gleiche wie zuvor, nur ist das Vorzeichen des Zeitversatzes diesmal negativ. Da das Ereignis selbst nicht vorhersagbar ist, bleiben die ersten zwei Sekunden der Animation unsichtbar und das Ende erfolgt gegebenenfalls verspätet beim Eintritt des Ereignisses.

```
<circle r="15" fill="blue">
<animateMotion dur="12s" begin="c.activate;c.click;60s"
     values="0,300;400,0;400,300;0,300" />
</circle>
<circle id="c" cx="50" r="15" fill="green" />
```

Die Animation startet durch Aktivierung des grünen Kreises, anklickern und nach 60s (nicht alternativ sondern in allen Fällen).

Bei folgenden Beispielen kann jeweils ein Ereignis ausprobiert werden, welches im Dokumenttitel und im Dokument selbst angegeben ist. Es werden jeweils blaue Quadrate entlang eines Pfades bewegt. Grüner Knopf für Beginn und roter Knopf für Ende sowohl ohne als auch mit Zeitversatz:

begin und *end* mit 'activate'[23]

begin und *end* mit 'click'[24]

begin und *end* mit 'mousedown'[25]

begin und *end* mit 'mouseup'[26]

begin und *end* mit 'mouseover'[27]

begin und *end* mit 'mouseout'[28]

begin und *end* mit 'mousemove'[29]

Grüner Knopf oder Text für Beginn und roter Knopf oder Text für Ende:

begin und *end* mit 'focusin'[30]

begin und *end* mit 'focusout'[31]

begin und *end*, weitere Ereignisse (tiny 1.2-Variante)[32]

Für Beginn und Ende werden die Ereignisse load, resize, scroll, zoom und rotate verwendet. Ende jeweils 15s nach Beginn. Um resize und scroll auslösen zu können, muss das Anzeigefenster kleiner als das Dokument gewählt werden. resize bezieht sich auf eine Änderung der Fenstergröße, scroll auf ein Rollen mit einem entsprechenden Balken.

23 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_activate01.svg
24 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_click01.svg
25 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_mousedown01.svg
26 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_mouseup01.svg
27 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_mouseover01.svg
28 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_mouseout01.svg
29 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_mousemove01.svg
30 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_focusin01.svg
31 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_focusout01.svg
32 http://de.wikibooks.org/wiki/media%3ASVGbegin_end01.svg

begin und *end*, weitere Ereignisse (2)[33]

Für Beginn und Ende werden die Ereignisse beginEvent, endEvent und repeatEvent verwendet. Ende jeweils 15s nach Beginn. Die Ereignisse beziehen sich auf die Animation des größten Quadrates, welches mit einer einfachen Liste für begin mehrfach gestartet wird.

14.4.6 Wiederholungsereignis

Squiggle (Batik)	1.7 (nicht für *end*)
Opera (Presto)	9.5 (9 teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4 (teilweise)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Ein Ereignis der Wiederholung einer Animation dient der Synchronisation.

Es kann ebenfalls notiert werden, welches Animationselement zur Synchronisation dienen soll. Die Angabe ist optional.

Beim Animationselement, welches der Synchronisation dient, ist ansonsten ein Attribut *id* (oder auch *xml:id in SVG tiny 1.2) mit einem Fragmentidentifizierer zu notieren.* Der Wert beginnt dann optional mit dem Fragmentidentifizierer, dann folgt ein Punkt '.' und dann die Zeichenfolge 'repeat(', gefolgt von einer positiven ganzen Zahl und von ')'. Nach optionalen Leerzeichen kann dann optional ein Zeitversatz notiert werden, dessen genaue Syntax bereits beschrieben wurde.

Die ganze Zahl gibt die Anzahl der Wiederholungen an, nach denen die Synchronisation stattfinden soll. 1 bezeichnet also den Begin der ersten Wiederholung, 0 entspricht formal dem Beginn des Elementes, was aber kein Wiederholungsereignis erzeugt.

Wiederholungen beziehen sich nicht auf mehrere Einträge im Attribut *begin*, sondern auf Wiederholungen, die mit den noch zu besprechenden Attributen *repeatDur* oder *repeatCount* angegeben werden.

Beispiele:

```
<circle r="15">
<animate id="a" attributeName="stroke" values="black;white" dur="5s"
 repeatCount="10" />
<animateMotion dur="12s" begin="a.repeat(2); a.repeat(4)" end="a.repeat(5)"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Die Wiederholung der Animation des Striches des Kreises dient der Synchronisation der Bewegung des Kreises. Die Bewegung beginnt bei der zweiten Wiederholung 10s nach Dokumentbeginn und erneut bei der vierten Wiederholung 20s nach Dokumentbeginn und endet dann mit der fünften Wiederholung 25s nach Dokumentbeginn.

```
<circle r="15">
```

33 http://de.wikibooks.org/wiki/media%3ASVGbegin_end02.svg

```
<animate id="a" attributeName="stroke" values="black;white" dur="5s"
 repeatCount="10" />
<animateMotion dur="12s" begin="a.repeat(2)+2s; a.repeat(4)+ 2s" end="a.repeat(5)
 + 2s"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Die gleiche Animation, nur jeweils mit einem Zeitversatz von 2s.

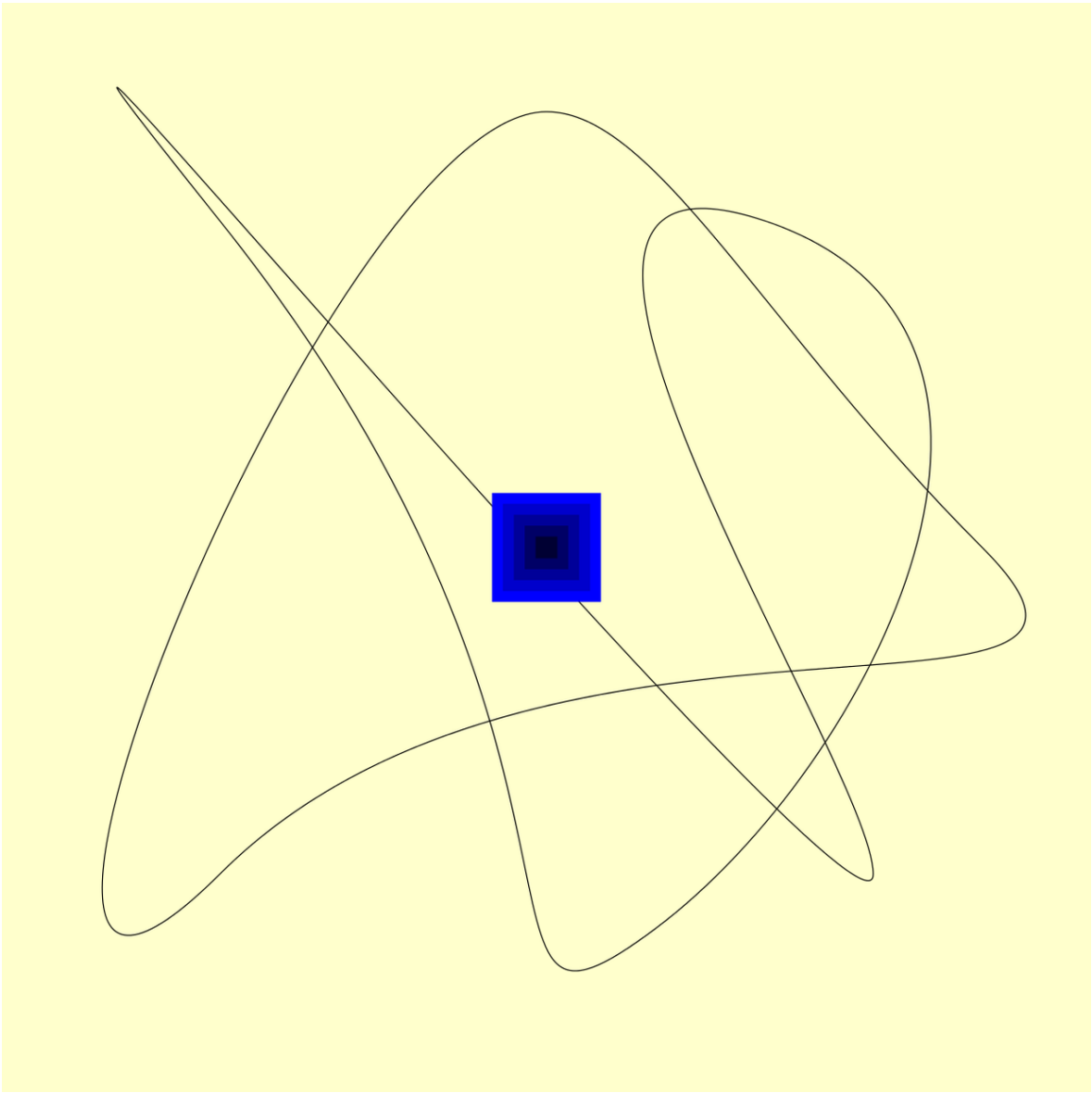


Abb. 245 Kopplung von Animationen über Wiederholungsereignisse

Kopplung von Animationen über Wiederholungsereignisse[34]

Blaue Quadrate werden entlang eines Pfades bewegt.

34 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_repeat01.svg

Für Beginn und Ende werden zum großen Teil Wiederholungswerte mit Zeitversatz verwendet. Durch eine geschlossene Schleife mit zwei Animationen wird gewährleistet, dass die Animationen immer wieder gestartet werden.

14.4.7 Tastaturereignis

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	- (ab 9 teilweise mit zusätzlichen Aktivierungstasten)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4-8 (ab Version 9 nicht mehr)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	-
Microsoft Internet Explorer (Trident)	
librsvg	

Das Betätigen einer Taste der Tastatur dient als Synchronisation. Der Wert beginnt mit der Zeichenfolge 'accessKey(', in SVG 1.1 gefolgt von exakt einem Zeichen, gefolgt von ')', optional gefolgt von einem Zeitversatz, dessen genaue Syntax bereits beschrieben wurde.

Das Betätigen der Taste (Ereignis im Sinne von DOM: keydown) löst das Ereignis aus. Es sind keine weiteren Tasten zu betätigen, um die Funktionalität zu aktivieren (In dem Sinne ist das Verhalte zum Beispiel von Opera bis mindestens Version 10 fehlerhaft, denn die Funktionalität muss zunächst aktiviert werden, indem die Tasten 'shift' und 'escape' gedrückt werden. Immerhin bietet Opera dann eine komplett Übersicht über alle verwendbaren Tasten, allerdings ohne Möglichkeit einer Beschreibung, was bei dem jeweiligen Tastaturereignis passiert).

Das notierte Zeichen korrespondiert mit einer Taste auf der Tastatur des Nutzers. Zum Beispiel bezeichnet 'accessKey(a)' oder 'accessKey(A)' das Betätigen der Taste mit der Aufschrift 'a' beziehungsweise 'A'.

Alternativ zu exakt einem Zeichen ist es in SVG tiny 1.2 erlaubt, ein Schlüsselwort wie 'Enter' für die Eingabetaste anzugeben oder den Unicode des Zeichens, zum Beispiel 'U+0041' für 'A'. In SVG tiny 1.2 ist eine Liste mit Schlüsselwörtern angegeben.

Beispiel:

```
<circle r="15">
<animateMotion dur="12s" begin="accessKey(b)" end="accessKey(e)+2s"
     values="0,300;400,0;400,300;0,300" />
</circle>
```

Taste 'b' startet die Animation und sofern sie noch aktiv ist, beendet die Taste 'e' die Animation zwei Sekunden, nachdem die Taste gedrückt ist.

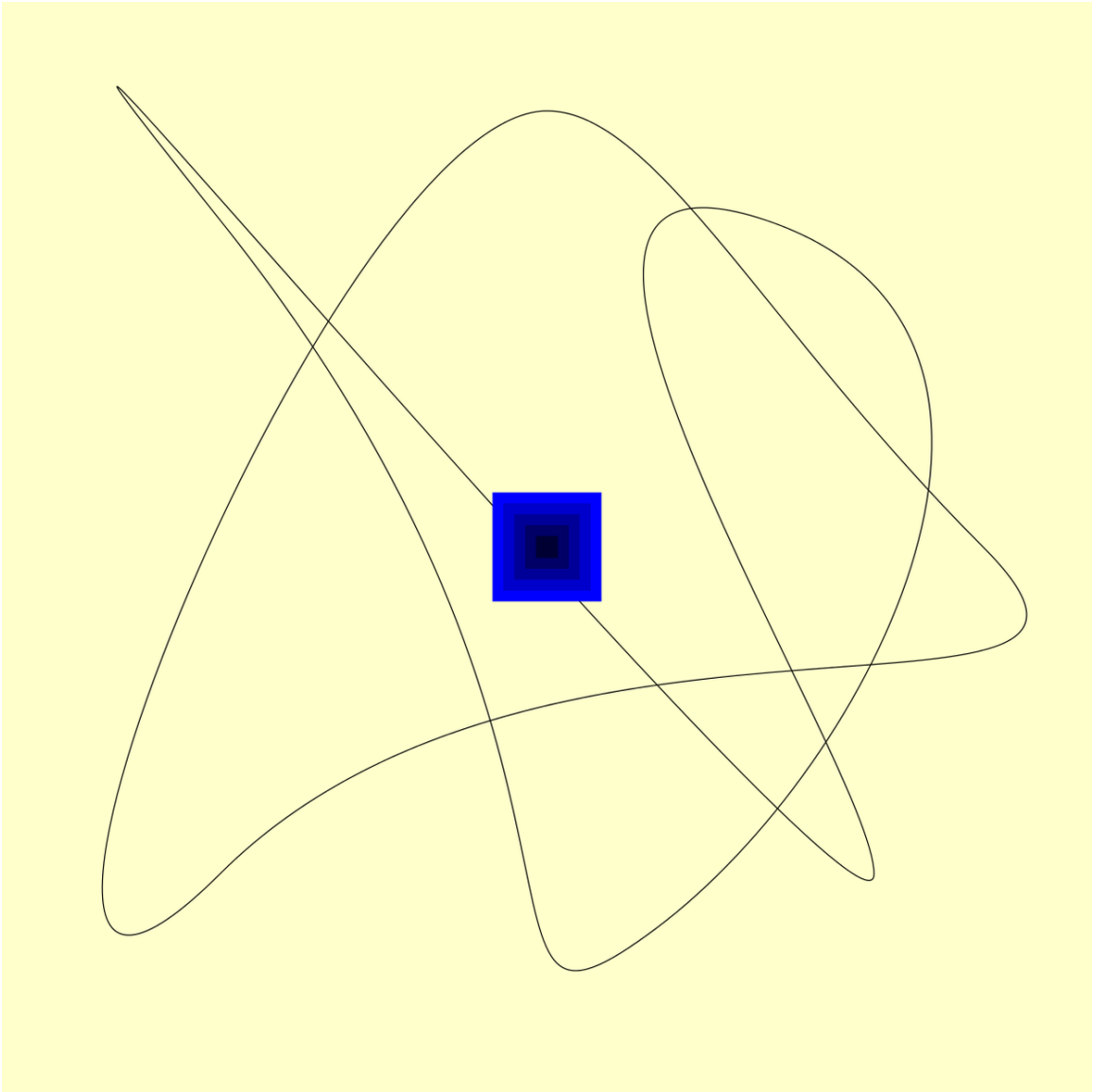


Abb. 246 Tastaturereignisse für Beginn und Ende

Tastaturereignisse für Beginn und Ende[35]

Blaue Quadrate werden entlang eines Pfades bewegt.

Für Beginn und Ende werden Tastaturereignisse verwendet. Die Tasten a bis e für den Beginn, 0 bis 4 entsprechend für das Ende.

35 http://de.wikibooks.org/wiki/media%3ASVGbegin_end_accessKey01.svg

Abb. 247 Animation mit Tastatur beginnen, Methode nach SVG tiny 1.2

Animation mit Tastatur beginnen, Methode nach SVG tiny 1.2[36]

Die neue Notation für accessKey in SVG tiny 1.2 kann mit allen verfügbaren speziellen Tasten ausprobiert werden. Alternativ kann auch die einfache Notation von SVG 1.1 verwendet werden, hier representiert durch das erste 'a'. accessKey sollte ohne weitere Tastenkombinationen verfügbar sein, wenigstens für alles Tasten der Tastatur.

Der Effekt einer Aktivierung besteht darin, dass sich die Farbe jenes Kreises ändert, welcher links neben der Bezeichnung einer Taste steht.

14.4.8 Uhrzeit

Squiggle (Batik)	1.7 (teilweise, gar nicht für *end*)
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	-
Microsoft Internet Explorer (Trident)	
librsvg	

36 `http://de.wikibooks.org/wiki/media%3ASVGbegin_accessKey01.svg`

Eine Uhrzeit notiert in einer einfachen Variante des internationalen Formates für Datum und Uhrzeit dient als Synchronisation. Die jeweilige Uhrzeit wird vom Darstellungsprogramm jeweils in eine Dokumentzeit umgewandelt und dann effektiv wie ein Zeitversatz behandelt. Dieser Wert ist in SVG tiny 1.2 nicht verfügbar.

Der Wert beginnt mit der Zeichenfolge 'wallclock(', gefolgt optionalen Leerzeichen, gefolgt von einer Uhrzeitangabe, gefolgt von optionalen Leerzeichen, gefolgt von ')'.

Die Uhrzeitangabe ist entweder vom Typ 'DateTime' oder vom Typ 'WallTime' oder vom Typ 'Date'. Wird 'WallTime' verwendet, bezieht sich die Angabe auf den aktuellen Tag zum Dokumentbeginn. Wird 'Date' verwendet, bezieht sich die Angabe auf 0 Uhr des angegebenen Tages in lokaler Zeit.

Der Typ 'DateTime' setzt sich zusammen aus 'Date', gefolgt von 'T'. gefolgt von 'WallTime'.

Der Type 'Date' besteht aus exakt 4 Ziffern (jeweils 0 bis 9) für das Jahr, gefolgt von '-' gefolgt von zwei Ziffern (Bereich 01 bis 12) für den Monat, gefolgt von '-', gefolgt von zwei Ziffern für den Tag (Bereich 01 bis 31, je nach Monat ein gültiger Tag für den Monat).

'Walltime' ist die Angabe einer Zeit. Diese beginnt mit zwei Ziffern (Bereich 00 bis 23) für Stunden, gefolgt von ':', gefolgt von zwei Ziffern für Minuten (Bereich 00 bis 59).

Optional folgt dann eine weitere Zeichenfolge für Sekunden, beginnend mit ':', gefolgt von zwei Ziffern für ganze Sekunden (Bereich 00 bis 59; das Format erlaubt auch den Wert 60 im Falle der selten eintretenden Schaltsekunden, allerdings kann das bei realen Programmen und Computern Probleme nach sich ziehen, weil die oft Schaltsekunden nicht korrekt berücksichtigen können, weil diese nur tabllarisch verfügbar sind und nicht langfristig für die Zukunft festgelegt sind, nur für die Vergangenheit). Darauf folgen optional Bruchteile von Sekunden, beginnend mit '.', gefolgt von mindestens einer Ziffer (0 bis 9).

Statt des Sekundenteils oder auf diesen folgend kann eine Angabe zur Zeitzone notiert werden. Diese Zeichenfolge ist entweder ein 'Z' für eine Angabe der UTC, also der globalen Zeit, oder die Angabe einer Zeitzone. Die Angabe der Zeitzone beginnt entweder mit '+' oder '-', gefolgt von zwei Ziffern (Bereich 00 bis 23) für Stunden, gefolgt von ':', gefolgt von zwei Ziffern (Bereich 00 bis 59) für Minuten.

Fehlt die Zeitzonenangabe, so ist die Zeitzone am Präsentationsort gemeint, die lokale Zeit des Nutzers also.

Beispiele:

Abb. 248 Digitale Uhr (lokale Zeit)

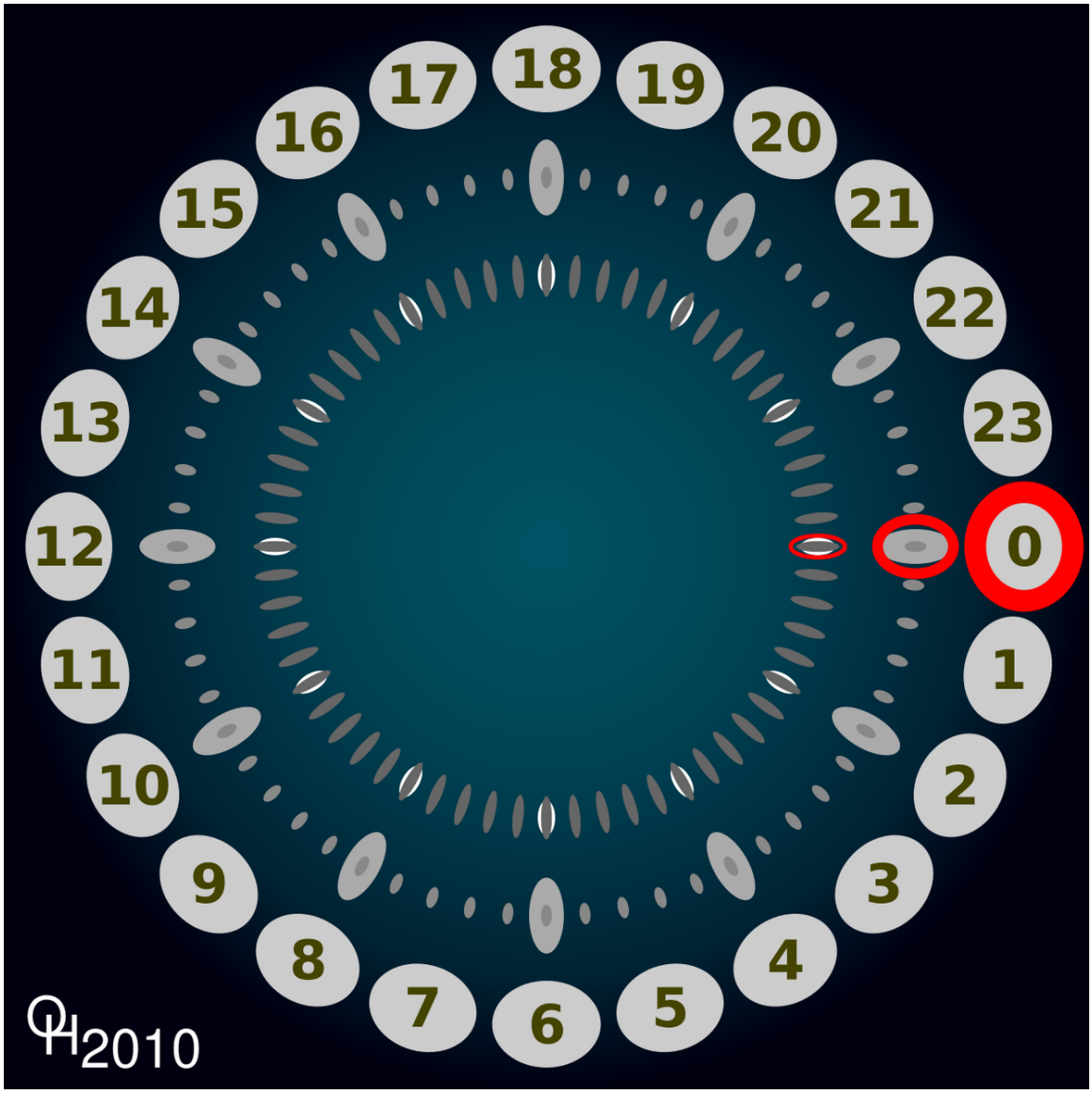


Abb. 249 Analoge 24-Stunden-Uhr (UTC)

UTC-Jahresbeginn 2010:

begin="wallclock(2010-01-01T00:00Z)"

begin="wallclock(2010-01-01T00:00:00Z)"

begin="wallclock(2010-01-01T00:00:00.0Z)"

begin="wallclock(2010-01-01T00:00:00.0Z)"

begin="wallclock(2010-01-01T00:00:00.0-00:00)"

Beginn des aktuellen Tages in lokaler Zeit, zum Beispiel brauchbar für eine lokale Uhr:

begin="wallclock(00:00)"

Beginn des aktuellen Tages in globaler Zeit UTC, zum Beispiel brauchbar für eine globale Uhr:

begin="wallclock(00:00Z)"

oder

begin="wallclock(00:00+00:00)

Ende der Animation mittags in lokaler Zeit, wenn die Animation nur vormittags laufen soll:

end="wallclock(12:00)"

Oder mit einer anderen Zeit (UTC):

end="wallclock(2042-03-29T17:43:11.1278Z)"

Anwendungen für ein komplettes Datum könnten etwa ein Herunter- oder Heraufzählen von oder zu einem bestimmten Ereignis sein, ansonsten sind die Anwendungsmöglichkeiten von Uhrzeitwerten vermutlich eher begrenzt. Leider ist für einen Uhrzeitwert auch keine zusätzliche Angabe eines Zeitversatzes vorgesehen, was dann wiederum einige solcher Anwendungen erschwert.

Auch um etwa bei Wettkämpfen oder Glücksspielen darüber die Anwendbarkeit von interaktiven Anwendungen zu begrenzen, statt über eine server-seitige Strategie, ist nicht besonders schlau, weil SVG-Dokumente leicht mit einem Texteditor modifiziert werden können, somit können durch den Anwender also auch leicht feste Beginn- und Endzeiten verändert werden.

Digitale Uhr (lokale Zeit)[37]

Analoge 24-Stunden-Uhr (UTC)[38]

14.4.9 'indefinite'

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	8 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4 (teilweise)
Konqueror (KSVG)	-
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

37 http://de.wikibooks.org/wiki/media%3ASVG_Digitaluhr01.svg
38 http://de.wikibooks.org/wiki/media%3ASVG_Analoguhr02.svg

Kein endlicher Wert ist festgelegt. Die Sychronisation kann durch ein externes Ereignis ausgelöst werden (DOM oder Verweis).

Ist 'indefinite' für *begin* angegeben und kein anderer Wert in der Liste, so beginnt die Animation nicht ohne weitere Interaktion. Der Beginnzeitpunkt der Animation ist (noch) nicht aufgelöst.

Um mit einem Verweis eine Animation beginnen zu können, unabhängig davon, ob 'indefinite' angegeben ist oder nicht, ist bei dem Animationselement ein Fragmentidentifizierer zu notieren. Wird ein Verweis auf das Animationselement betätigt, so startet die Animation. Ist für *begin* 'indefinite' notiert, wird der Animationsbeginn mit der Betätigung des Verweises aufgelöst und in die entsprechende Liste des Darstellungsprogrammes für Beginnzeiten eingetragen. Stehen allerdings in der Liste bereits Beginnzeiten, so wird zunächst von der aktuellen Zeit rückwärts nach einem gültigen Wert für *begin* gesucht. Wird einer gefunden, wird die gesamte Historie von dem Zeitpunkt an wiederholt. Gibt es keinen gültigen Wert in der Vergangenheit, so wird der erste in der Zukunft gewählt und der Zeitablauf von dort aus fortgesetzt. Gibt es keine Beginnzeitpunkt in der Liste, so wird die aktuelle Zeit als Beginnzeit eingetragen und die Animation beginnt.

end ist nicht direkt mit einem Verweis ansprechbar, allenfalls indirekt über eine Synchronisation mit einem *begin*, welches mit einem Verweis ausgelöst wird. Allerdings kann in der Werteliste von *end* ein 'indefinite' notiert werden, um sicherzustellen, dass auf jeden Fall ein Endwert vorhanden ist, der größer ist als jeder Beginnwert, so dass also für jedes Beginn ein gültiges Intervall für eine Animation gefunden werden kann. Das ist nicht notwendig, wenn gar kein *end* notiert ist.

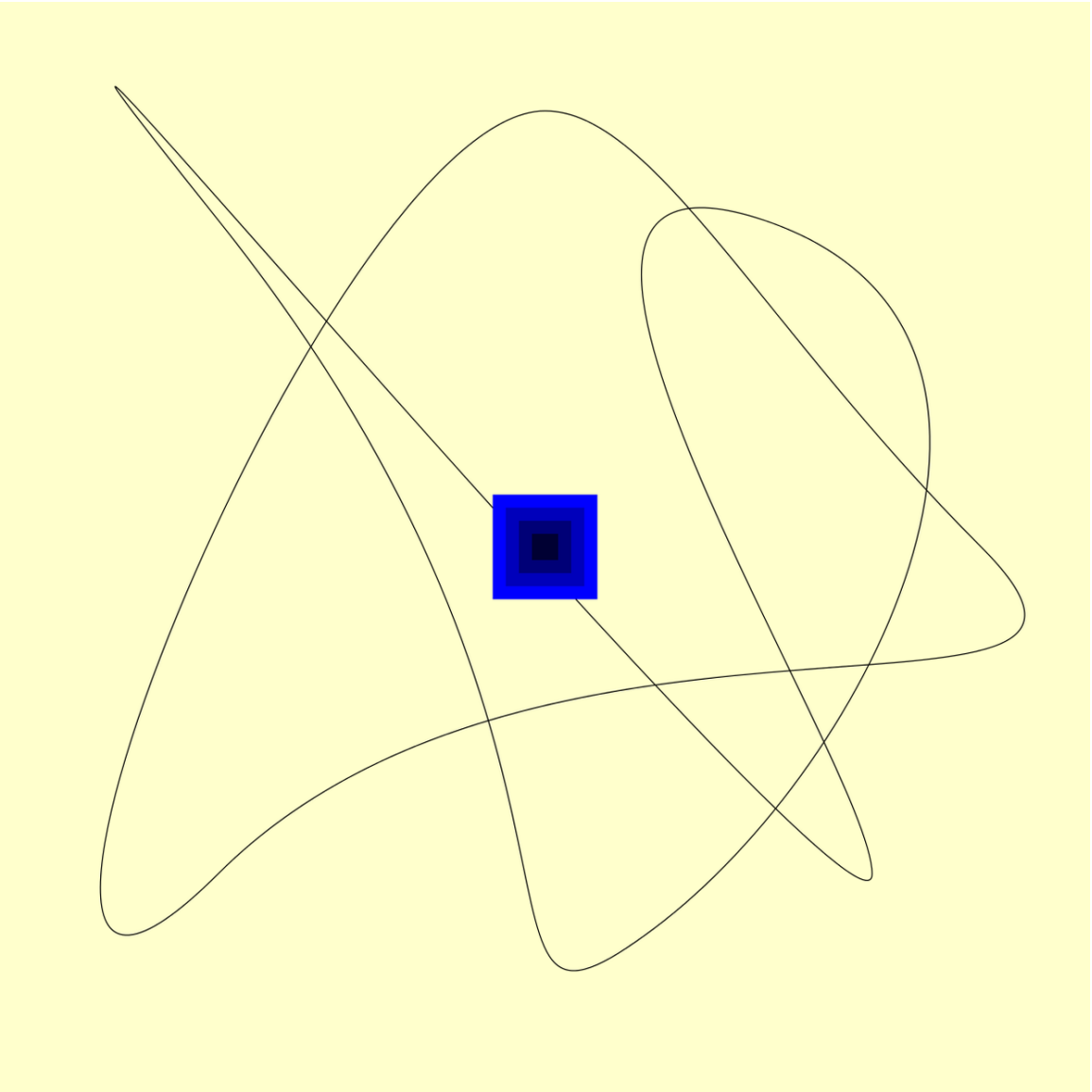


Abb. 250 Animation mit Verweis aktivieren

Animation mit Verweis aktivieren[39]

Blaue Quadrate werden entlang eines Pfades bewegt.

Für die vier Animationen wird begin indefinite notiert. Diese können durch einen Verweis gestartet werden, die Verweise umgeben jeweils die Quadrate.

Interessant bei der Aktivierung einer Animation durch einen Verweis ist, dass bei einer abermaligen Aktivierung der gesamte Zeitablauf seit der letzten Aktivierung wiederholt wird, weil durch die Aktivierung der Zeitpunkt für den Beginn aufgelöst wird und bei einer erneuten Aktivierung auf die letzte aufgelöste Beginnzeit zurückgegriffen wird.

39 `http://de.wikibooks.org/wiki/media%3ASVGbegin_indefinite01.svg`

14.4.10 Auswertung von Beginn- und Endzeitlisten

Das Modell der Animation basiert auf Zeitintervallen, in denen die Animation stattfindet. Ein Zeitintervall ist definiert durch eine Beginnzeit und eine Endzeit, gegeben durch Werte in den Listen der Attribute *begin* und *end*. Für dasselbe Animationsziel kann es im Laufe der Zeit mehrere Intervalle geben. Zu jedem Zeitpunkt gibt es exakt ein aktuelles Intervall bezogen auf ein Animationsziel. Die Intervalle werden erzeugt durch eine Auswertung von Listen für Beginnzeiten und Endzeiten, basierend auf den Listen von *begin* und *end*.

Die Zeiten, die verwendet werden, um die Intervalle zu erzeugen, werden Instanzzeiten genannt. Einige Bedingungen, die in *begin* und *end* notiert sein können, erzeugen exakt eine Instanzzeit, wie etwa ein einfacher Zeitversatz, andere wie Ereigniswerte oder Synchronisationswerte können mehrere erzeugen.

Einige Instanzzeiten wie Zeitversätze verbleiben immer in den Listen der Beginn- oder Endzeiten, andere wie Ereigniswerte werden nur hinzugefügt, wenn das Ereignis wirklich passiert und werden wieder entfernt, wenn die Animation neu startet, sofern die zugehörigen Instanzzeiten nicht mehr zum aktuellen Zeitintervall gehören, sondern in der Vergangenheit liegen, also vor dem aktuellen Zeitintervall. Synchronisationswerte werden neu ermittelt, wenn die Animation neu startet. Eine aufgelöste Zeit resultierend aus einem eingetretenen Sychronisationsereignis wird wieder aus den Listen gelöscht, wenn sowohl die sychronisierende als auch die abhängige Animation neu startet.

Zum Dokumentbeginn wird für jedes Animationsziel versucht, ein aktuelles Intervall zu ermitteln. Wenn der Beginn aufgelöst ist, das Ende aber nicht, kann die Animation auch erneut starten, während sie noch aktiv ist, dies beendet das aktuelle Intervall mit dem Beginn des neuen aktuellen Intervalls.

Weil Beginn- und Endzeiten auf anderen Zeiten basieren können, die sich ändern mögen, ist auch das aktuelle Intervall Gegenstand einer möglichen zeitabhängigen Änderung. Insbesondere kann die Instanzzeit für das Ende des aktuellen Intervalls sich ändern. Dies wird dynamisch angepasst. Einmal eingetretene Zeiten werden allerdings nicht mehr korrigiert. Insbesondere kann die Beginnzeit sich nicht mehr ändern, wenn das aktuelle Intervall erst einmal begonnen hat. Und wenn das Intervall zu Ende ist, kann sich auch die Endzeit nicht mehr ändern.

Synchronisationswerte erzeugen eine Abhängigkeit zwischen zwei Animationszielen. Nur Beginn- und Endzeiten des aktuellen Intervalles erzeugen zeitlich veränderliche Abhängigkeiten bei abhängigen Animationszielen.

Die verschiedenen Eintragsmöglichkeiten für *beginn* und *end* werden unterschiedlich in Instanzzeiten umgewandelt:

Am einfachsten passiert dies für Zeitversätze. Aus jeden Zeitversatz ergibt sich eine Instanzzeit, die immer in der Liste verbleibt.

Synchronsationen mit einer Uhrzeit sind ähnlich einfach. Die Uhrzeit wird zu einem Zeitversatz in Dokumentzeit umgewandelt und dieser Zeitversatz erzeugt wiederum jeweils eine immer in der Liste verbleibende Instanzzeit.

Ereigniswerte, Tastaturereignisse und Wiederholungsereignisse werden alle ähnlich behandelt. Aus diesen Bedingungen ergibt sich erst eine Instanzzeit, wenn das jeweilige Ereignis wirklich

eingetreten ist. Jedesmal, wenn das Ereignis eintritt, wird eine neue Instanzzeit erzeugt. Die Zeit des Ereignisses, gegebenenfalls mit dem notierten Zeitversatz, wird als Instanzzeit in der Liste notiert. Wenn das Animationsziel neu startet, wird die Zeit allerdings wieder gelöscht.

Auch Synchronisationswerte werden ähnlich behandelt. Es ergibt sich erst eine Instanzzeit, wenn die synchronisierende Animation ein aktuelles Intervall erzeugt. Daraus ergibt sich dann jeweils eine einzige Instanzzeit. Die Zeit, gegebenenfalls mit dem notierten Zeitversatz, wird als Instanzzeit in der Liste notiert. Allerdings wird die Zeit nicht notwendig gelöscht, wenn das Animationsziel neu startet. Stattdessen wird die Instanzzeit erst gelöscht, wenn die synchronisierende und die abhängige Animation neu starten. Wenn die synchronisierende Animation neu startet, wird die Instanzzeit allerdings neu berechnet.

'indefinite' erzeugt keine Instanzzeit in der Liste der Beginnzeiten. Es erzeugt allerdings eine Instanzzeit bei unendlich in der Liste der Endzeiten.

Sind bereits alle möglichen Intervalle abgearbietet, mag es kein weiteres Intervall mehr geben. Das letzte Intervall gilt solange als das aktuelle, bis ein neues beginnt. Der Sachverhalt kann sich mit jeder neu hinzugefügten Instanzzeit ändern, so dass jede Änderung eine erneute Verarbeitung der Listen vom aktuellen Zeitpunkt aus zur Folge hat, die ein neues Intervall erzeugen kann.

Die Verarbeitung einer Animation besteht im Wesentlichen aus folgenden Schritten:

1. Start - erstes Intervall finden
2. Warten, das Intervall zu beginnen
3. Aktive Zeit - Animation durchführen
4. Ende des Intervalls - das nächste berechnen und abhängige Animationen informieren
5. Nach der aktiven Zeit - gegebenenfalls eingefrorene Werte anwenden und auf das nächste Intervall warten

Die Punkte zwei bis fünf können sich wiederholen. Während des zweiten Punktes kann sich die Beginnzeit ändern, während des zweiten und dritten die Endzeit. Wenn dies passiert, wird dies an abhängige Animationen weitergereicht.

Das erste Intervall ist recht einfach zu ermitteln. Problematisch sind dabei nur Intervalle, welche komplett vor dem Dokumentbeginn liegen, wo also Beginn- und Endzeit vor dem Dokumentbeginn liegen. Diese Intervalle werden aussortiert und führen zu keiner Animation und keiner Synchronisation. Das erste akzeptable Intervall ist also jenes, welches die früheste Endzeit nach dem Dokumentbeginn aufweist.

Die Phase des Wartens, um das ein Intervall zu beginnen, tritt nur auf, wenn das aktuelle Intervall nicht sofort beginnt. Eine Änderung der Beginnzeit in dieser Phase wirkt sich auf die Bestimmung des aktuellen Intervalles aus, welches einschließlich aller Abhängigkeiten korrigiert wird.

Die aktive Zeit liegt zwischen dem Beginn und dem Ende des aktuellen Intervalles, sobald das Intervall begonnen hat. In dieser Zeit kann sich zwar die Endzeit weiterhin ändern, aber nicht mehr die Anfangszeit. Änderungen in der Liste dier Beginnzeiten ändern nicht das aktuelle Intervall, können aber zu einem erneuten Start der Animation führen, also ein neues Intervalle erzeugen und zuvor das aktuelle beenden.

Anders als beim ersten Intervall wird nach dem Ende eines Intervalles nur nach einem neuen Beginn nach dem Ende des Intervalles gesucht, nicht davor. Insbesondere kann dies bei einem zusätzlichen

negativen Zeitversatz zu einem Ereignis dazu führen, für das die Instanzzeit nicht akzeptabel ist, wenn sie vor dem Ende des beendeten, aber noch aktuellen Intervalles liegt. Ein Neustart kann zwar das aktuelle Intervall beenden, da sich aber die Beginnzeit eines bereits begonnenen Intervalles nicht mehr ändert, kann das neue Intervall nicht vor dem aktuellen beginnen und wäre somit nicht akzeptabel. Das schränkt die wirksamen negative Zeitversätze für Ereignisse auf jene ein, die zu Instanzeiten führen, die nach dem Beginn des aktiven Intervalls liegen. Ist das aktuelle Intervall gar beendet, so wird auch die Endzeit nicht mehr korrigiert und es ist nur noch eine Instanzzeit als Beginn eines neuen Intervalles akzeptabel, die hinter dem Ende des aktuellen Intervalles liegt.

Nach der aktiven Zeit wird gegebenenfalls ein eingefrorener Wert angewendet und auf das nächste Intervall gewartet. Zeiten für das beendete Intervall können sich nicht mehr ändern.

Abhängigkeiten und Synchronisationen können zu zyklischen Abhängigkeiten führen, die nicht immer auflösbar sein müssen. Auflösbare gegenseitige Abhängigkeiten können sehr nützlich sein und führen zu fortschreitenden Zyklen. Kann in einer zyklischen Abhängigkeit keine Beginnzeit aufgelöst werden, so kann der Zyklus nicht fortschreiten, es findet keine Animation statt, solange keine Zeit aufgelöst wird, etwa durch einen aktivierten Verweis auf eines der beteiligten Animationselemente.

14.5 Eingefrorene Animation

Squiggle (Batik)	1.7 (oft falsch)
Opera (Presto)	9 (8 teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	-
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Nach dem Ende der aktiven Dauer einer Animation springt der Wert per Voreinstellung zurück auf den darunterliegenden Wert. Mit dem Animationsattribut *fill*, welches auf Animationselemente anwendbar ist und nichts mit dem Präsentationsattribut *fill* für darstellbare Elemente zu tun hat, kann auf die Präsentation nach der aktiven Dauer Einfluss genommen werden.

Mögliche Werte des Animationsattributes *fill* sind:

'remove'

Nach der aktiven Dauer wird zur Voreinstellung zurückgesprungen. Die Animation beeinflusst nicht länger das Animationsziel.

'freeze'

Der Wert, welcher als letztes in der aktiven Dauer eingenommen wurde, wird beibehalten.

Wird die aktive Dauer einer Animation eingeschränkt, so endet die Animation vorzeitig mit einem Zwischenwert. Bei *fill* 'freeze' wird dieser Zwischenwert beibehalten.

Bei einem nicht stetigen Animationsverlauf kann es passieren, dass das Ende der aktiven Dauer gerade auf eine Unstetigkeitsstelle fällt. Weil Zeitintervalle so festgelegt sind, dass der Anfang eingeschlossen ist und das Ende nicht, ergibt sich der eingefrorene aus dem Grenzwert der Animation vor dem Ende, nicht zwangsläufig durch den Wert, der für den Endzeitpunkt notiert ist.

Eine Ausnahme ergibt sich, wenn das Ende der Animation gerade auf ein ganzzahliges Vielfaches der einfachen Animationsdauer fällt. In dem Falle wird der letzte Wert eingefroren, der in der *values*-Liste steht. Bei kumulativen Animationen wird entsprechend der kumulierte Endwert eingefroren.

Beispiele:

```
<circle r="15">
<animateMotion dur="12s" values="0,300;400,0;400,300;0,300" fill="freeze" />
</circle>
```

Der Kreis springt nicht am Ende der Animation an den Ursprung zurück, sondern bleibt bei 0,300.

```
<line x1="100" x2="400" y1="100" y2="400" stroke="black" stroke-width="10">
<animate attributeName="stroke-linecap" dur="6s" end="4s"
         values="square;round;butt" fill="freeze" />
</line>
```

Die Form der Linienenden wird animiert. Mittels *end* wird die Animationsdauer eingeschränkt und die Animation ist eingefroren. Das Ende der aktiven Dauer fällt auf einen Zeitpunkt, wo die Form des Linienendes gerade wechseln soll. Der eingefrorene Wert ist 'round' und nicht 'butt'.

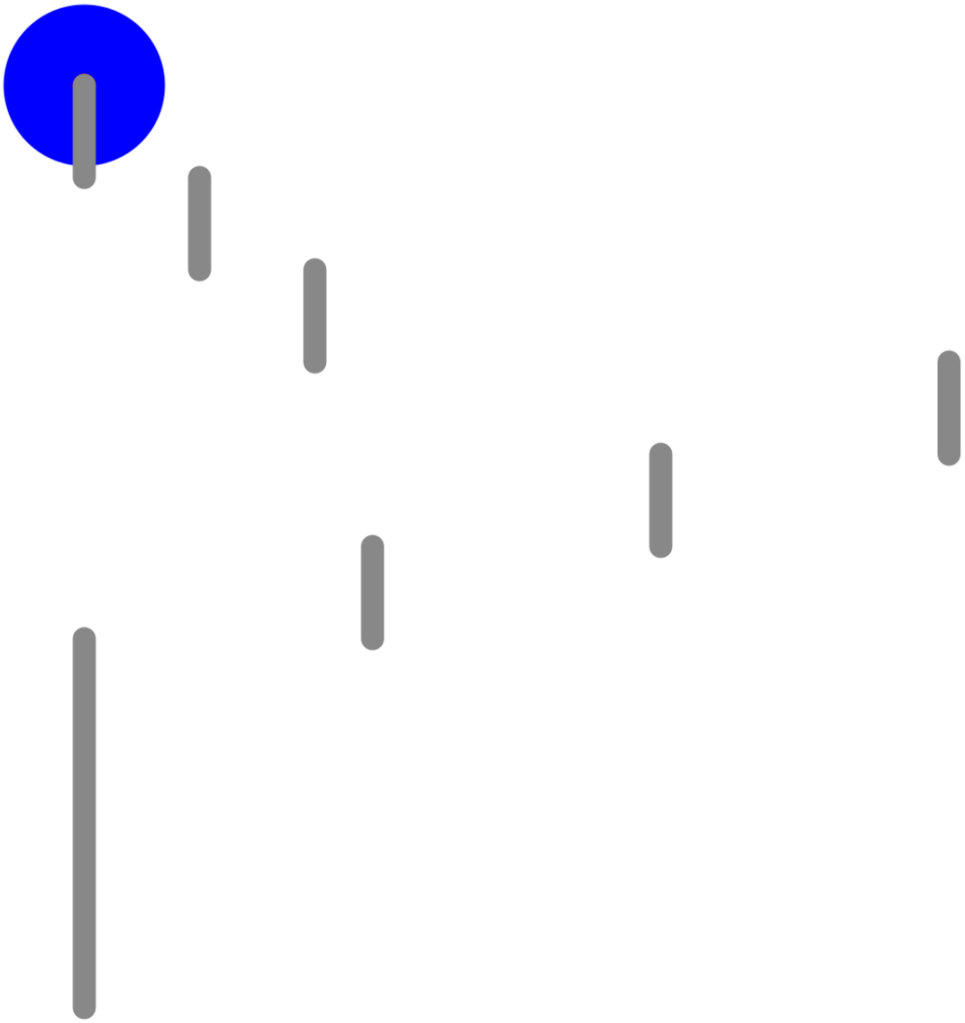


Abb. 251 Diskrete Animation und fill

Diskrete Animation und fill[40]

cx und cy eines Kreises werden diskret in 10s von klein nach groß animiert. Auf cx wird eine zweite Animation nach 3s für 4s von mittel zu klein mit dem fill-Wert freeze angewendet. Dies bedeutet für die Gesamtanimation ein konstant kleines cx für die letzten 3s. Ein grauer Pfad zeigt die korrekte Trajektorie des Kreises. Wenn das rote Zentrum des Kreises sichtbar wird, ist ein Fehler aufgetreten.

40 http://de.wikibooks.org/wiki/media%3ASVGanifill01.svg

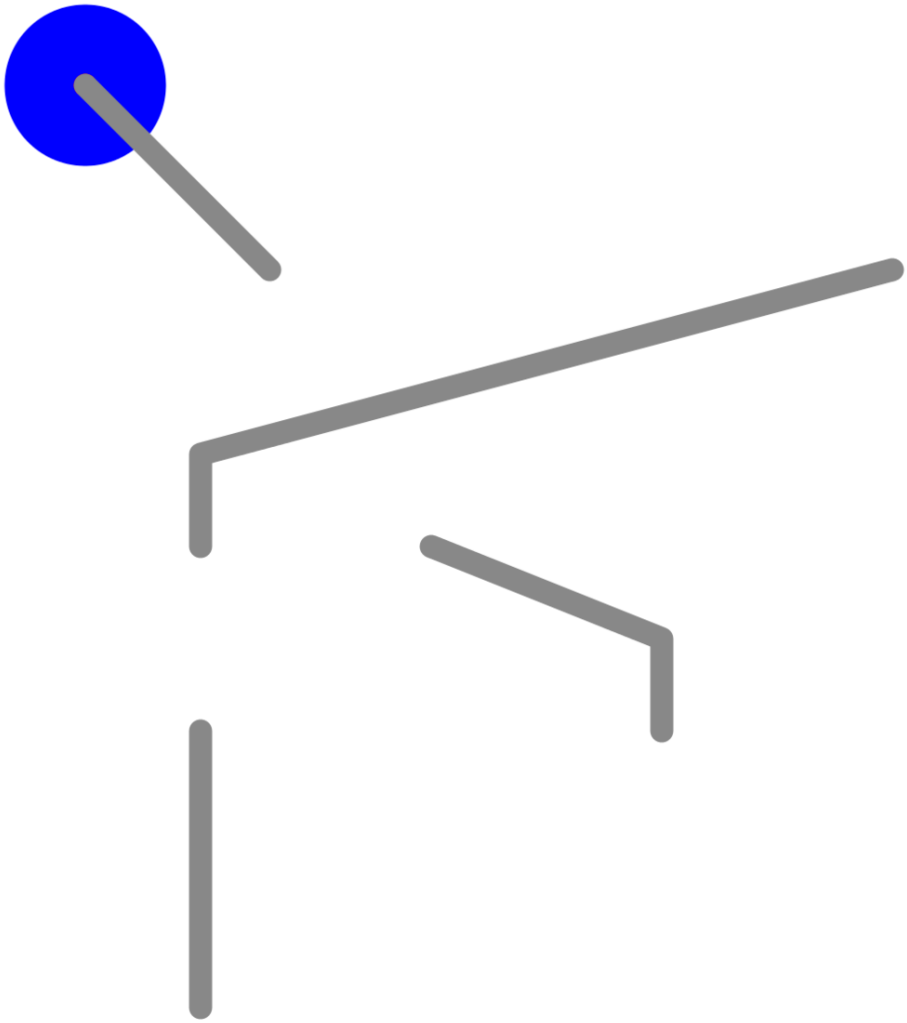


Abb. 252 Mehrere Animationen und fill

Mehrere Animationen und fill[41]

cx und cy eines Kreises werden 10s animiert, cx mit fill freeze und remove. Dies Beispiel testet speziell das Verhalten am Ende der einfachen Dauer bei einer noch immer aktiven Animation mit niedrigerer Priorität. Ein grauer Pfad zeigt die korrekte Trajektorie des Kreises für den ersten Umlauf. Wenn das rote Zentrum des Kreises sichtbar wird, ist ein Fehler aufgetreten.

41 http://de.wikibooks.org/wiki/media%3ASVGanifill02.svg

14.6 Sich wiederholende Animationen

Die einfache Animationsdauer einer Animation kann wiederholt werden, um den Animationseffekt mehrfach hintereinander darzustellen. Die kann mit den Attributen *repeatDur* oder *repeatCount* notiert werden. Es sollte höchstens eines der beiden Attribute angegeben werden. Werden beide angegeben, so wird das Minimum der sich daraus ergebenden aktiven Dauern angenommen.

14.6.1 *repeatDur*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	-
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mittels *repeatDur* wird die Dauer der wiederholten Animation angegeben. Der Wert ist entweder 'indefinite' oder ein Zeitwert (wie bereits für *dur* definiert). Bei 'indefinite' wiederholt sich die Animation permanent (bis das Dokument oder die Animation anderweitig beendet wird).

Beispiele:

```
<circle r="15">
<animateMotion dur="12s" repeatDur="6s"
   values="0,300;400,0;400,300;0,300" fill="freeze" />
</circle>
```

Die aktive Dauer der Animation wird durch *repeatDur* verkürzt, weil die dafür angegebene Dauer kürzer ist als die einfache Animationsdauer. Die Bewegung des Kreises wird in der Mitte der Animation eingefroren.

```
<circle r="15">
<animateMotion dur="12s" repeatDur="indefinite"
   values="0,300;400,0;400,300;0,300" />
</circle>
```

Weil *repeatDur* "indefinite" angegeben ist, wiederholt sich die Animation, bis die Anzeige des Dokumentes beendet wird.

```
<circle r="15">
<animateMotion dur="12s" repeatDur="16s"
   values="0,300;400,0;400,300;0,300" fill="freeze" />
</circle>
```

Die Animation läuft einmal komplett durch und wiederholt nochmal das erste Drittel und dann wird die Bewegung eingefroren.

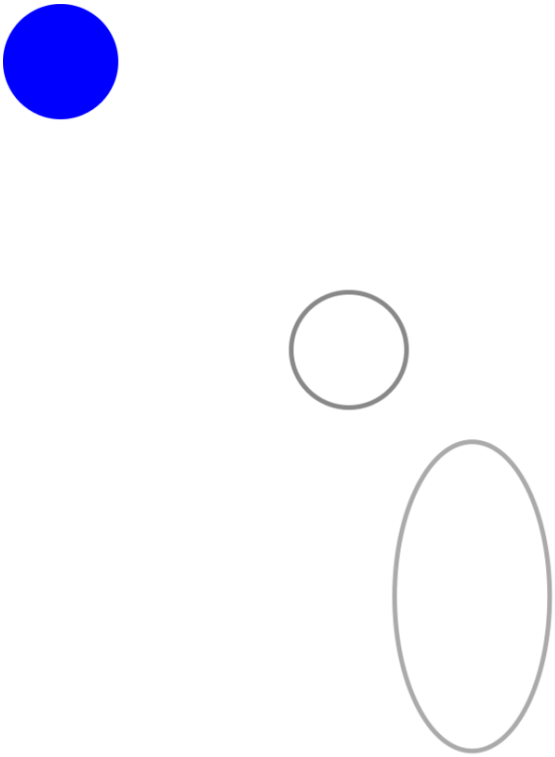


Abb. 253 Animationsattribut *repeatDur*

Animationsattribut *repeatDur*[42]

cx einer Ellipse wird animiert von 300 nach 700 mit einer Dauer von 20s und einem *repeatDur* von 25.3451s.

cy einer Ellipse wird animiert von 300 nach 700 mit einer Dauer von 10s und einem *repeatDur* von 25.3451s.

42 `http://de.wikibooks.org/wiki/media%3ASVGrepeatDur01.svg`

rx einer Ellipse wird animiert from 50 nach 300 mit einer Dauer von 5s und einem *repeatDur* von 25.3451s.

ry einer Ellipse wird animiert from 50 nach 300 mit einer Dauer von 4s und einem *repeatDur* von 25.3451s.

Dies bedeutet, alle Animationen enden exakt nach 25.3451s.

Anfangs- und Endform sind zum Vergleich in grau gegeben.

14.6.2 *repeatCount*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	-
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mittels *repeatCount* wird die Anzahl der gewünschten Durchläufe angegeben. Der Wert ist entweder 'indefinite' oder eine Zahl größer als 0. Bei 'indefinite' wiederholt sich die Animation permanent (bis das Dokument oder die Animation anderweitig beendet wird). 1 entspricht also der einfachen Animationsdauer, 0.5 der halben, die aktive Animationsdauer wird also auf die Hälfte der einfachen Animationsdauer eingeschränkt. Mit 10 werden also zehn Durchläufe angegeben, was der zehnfachen einfachen Animationsdauer entspricht.

Beispiele (entsprechend zu denen von *repeatCount*):

```
<circle r="15">
<animateMotion dur="12s" repeatCount="0.5"
   values="0,300;400,0;400,300;0,300" fill="freeze" />
</circle>
```

Die aktive Dauer der Animation wird durch *repeatCount* verkürzt, weil die sich daraus ergebende Dauer kürzer ist als die einfache Animationsdauer. Die Bewegung des Kreises wird in der Mitte der Animation eingefroren.

```
<circle r="15">
<animateMotion dur="12s" repeatCount="indefinite"
   values="0,300;400,0;400,300;0,300" />
</circle>
```

Weil *repeatCount* "indefinite" angegeben ist, wiederholt sich die Animation, bis die Anzeige des Dokumentes beendet wird.

```
<circle r="15">
<animateMotion dur="12s" repeatCount="1.3333"
   values="0,300;400,0;400,300;0,300" fill="freeze" />
</circle>
```

Die Animation läuft einmal komplett durch und wiederholt nochmal grob das erste Drittel und dann wird die Bewegung eingefroren.

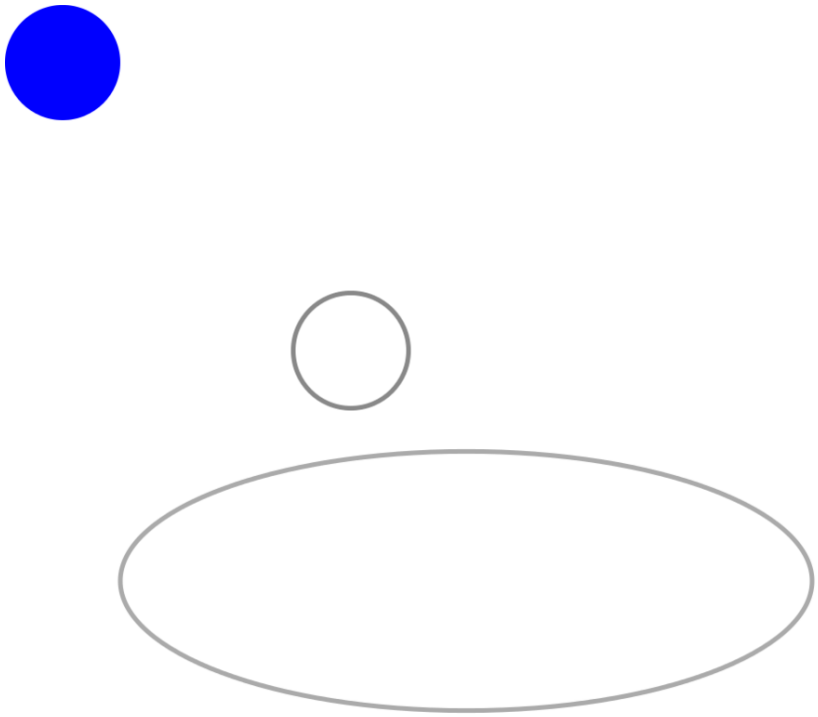


Abb. 254 Animationsattribut *repeatCount*

Animationsattribut *repeatCount*[43]

cx einer Ellipse wird animiert von 300 nach 700 mit einer Dauer von 20s und einem *repeatCount* von 1.25.

43 `http://de.wikibooks.org/wiki/media%3ASVGrepeatCount01.svg`

cy einer Ellipse wird animiert von 300 nach 700 mit einer Dauer von 10s und einem *repeatCount* von 2.5.

rx einer Ellipse wird animiert from 50 nach 300 mit einer Dauer von 5s und einem *repeatCount* von 5.

ry einer Ellipse wird animiert from 50 nach 300 mit einer Dauer von 4s und einem *repeatCount* von 6.25.

Dies bedeutet, alle Animationen enden exakt nach 25s.

Anfangs- und Endform sind zum Vergleich in grau gegeben.

14.7 Minimale und maximale aktive Dauer angeben

Mit den Attributen *min* und *max* kann die minimale, beziehungsweise maximale aktive Dauer genauer kontrolliert werden.

Die beiden Attribute geben eine untere, beziehungsweise obere Grenze der Animationsdauer an, welche effektiv andere Einschränkungen überschreiben. Sind die Attribute nicht angegeben, ergeben sich daraus jeweils keine weiteren Einschränkungen. Bei fehlerhaften Angaben wird das jeweilige Attribut mit der fehlerhaften Angabe ignoriert.

Sind beide Attribute angegeben, darf *max' nicht kleiner als* min*sein,* sonst werden beide Attribute ignoriert.

Jedesmal, wenn die aktive Dauer einer Animation neu bestimmt wird, wird dies zunächst ohne Berücksichtigung von *min* und *max* getan. Liegt das Ergebnis innerhalb der von *min* und *max* vorgegebenen Grenzen, wird das Ergebnis beibehalten.

Sonst gibt es zwei Möglichkeiten, die vorliegen können:

Ist die berechnete aktive Dauer größer als der Wert von *max*, so ist die aktive Dauer der Wert von *max*.

Ist die berechnete aktive Dauer kleiner als der Wert von *min*, so ist die aktive Dauer der Wert von *min*.

Falls die einfache Amimationsdauer oder die Dauer der Animation bedingt durch Wiederholungen größer als *min* ist, so erfolgt die Präsentation über die durch *min' eingeschränkte Dauer ganz normal.* Sonst erfolgt die Präsentation samt gegebenenfalls eingefrorenem Wert wie ohne Angabe von *min*. Das Ende der aktiven Dauer ist jedoch trotzdem durch *min* gegeben und wird dann gegebenenfalls als Synchronisationswert verwendet, nicht das sichtbare Ende der Präsentation.

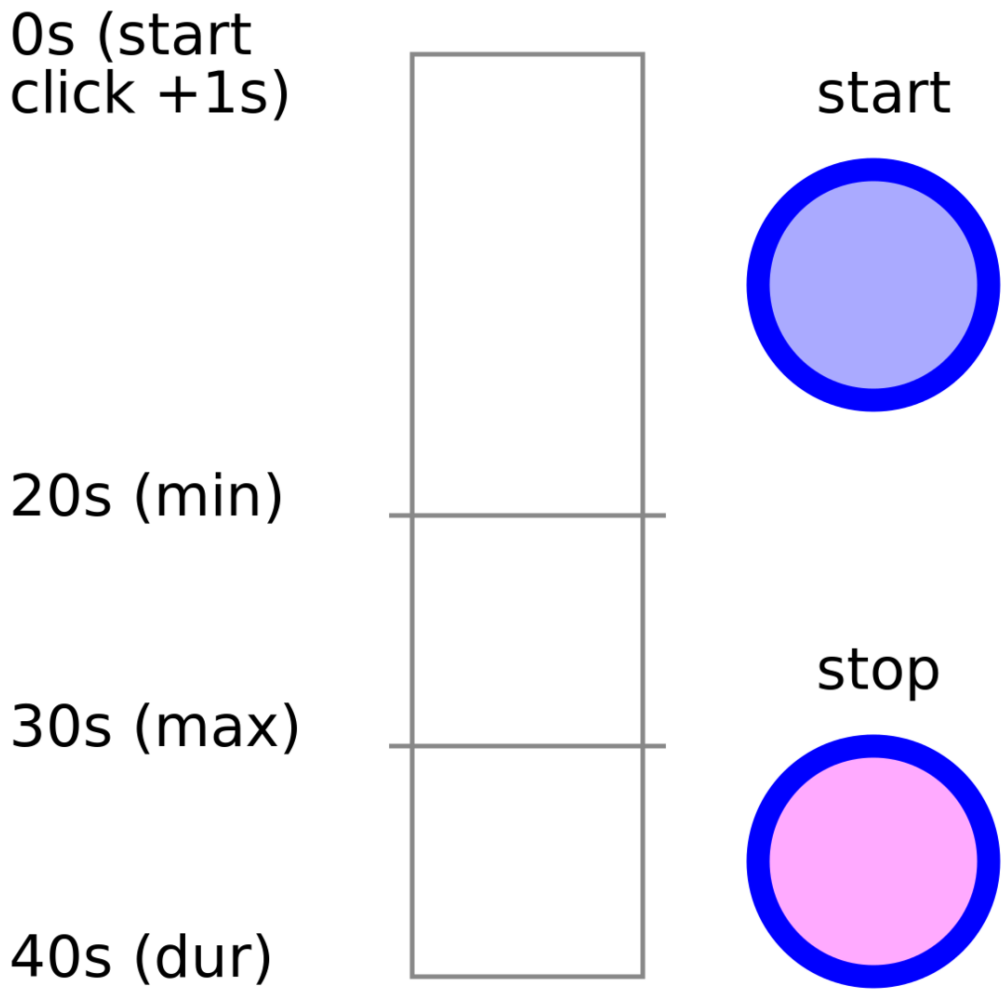


Abb. 255 Interaktives Beispiel zum Ausprobieren

Interaktives Beispiel zum Ausprobieren[44]

Die Höhe eines Rechtecks kann mit einem Klick auf den Startknopf vergrößert werden, was mit einem Klick auf den Stopknopf abgebrochen werden kann. Es startet 1s nach dem Klick auf den Startknopf. Die Dauer ist 40s, *min* ist 20s, *max* ist 30s, die mit diesen Zeiten verknüpften Größen sind grau markiert.

Dies bedeutet:

44 http://de.wikibooks.org/wiki/media%3ASVGmin_max01.svg

1. Die Animation kann mit einem Klick auf den Startknopf gestartet werden, auch wiederholt, weil dies vom Dokument nicht verhindert wird.

2. Falls die Animation gestartet wird und nicht vor ihrem Ende neue gestartet wird, so endet sie irgendwo zwischen den Werten für *min* und *max*.

3. Falls der Stopknopf nach Erreichen des Wertes für *max* oder nie betätigt wird, wurde die Animation bereits bei *max* angehalten und nichts weiter passiert.

4. Falls der Stopknopf zwischen *min* und *max* betätigt wird, hält die Animation unmittelbar an.

5. Falls der Stopknopf vor Erreichen des Wertes für *min* betätigt wird, dauert die Animation an und wird erst beim Wert für *min* angehalten.

14.7.1 *min*

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	9.5 (ab 9 teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit *min* wird die minimale aktive Dauer einer Animation angegeben. Der Wert ist entweder 'media' oder ein Zeitwert, wie bereits für *dur* definiert. 'media' ist nicht relevant für Animationselemente, nur für Elemente, die eine intrinsische Dauer haben (zum Beispiel ***audio*** und ***video*** in SVG tiny 1.2). Dann richtet sich die angegebene Dauer nach der intrinsischen Dauer des Mediums. Sofern 'media' für ein Animationselement angegeben ist, wird das Attribut ignoriert.

min kann zum Beispiel Einschränkungen durch *end* wieder aufheben. Oft ergibt sich aber nur ein wahrnehmbarer Effekt, wenn das Ende der Animation als Sychronisationswert verwendet wird.

Bei negativen Beginnzeiten erfolgt der Vergleich vom notierten Beginn, nicht vom beobachteten. Das kann dazu führen, dass eine Animation gar nicht beginnt, wenn das Intervall der korrigierten aktive Dauer vor dem Dokumentbeginn liegt.

Beispiele:

```
<circle r="15">
<animateMotion id="a" dur="12s" begin="4s" min="20s"
   values="0,300;400,0;400,300;0,300" fill="freeze" />
<animateMotion dur="12s" begin="a.end"
   values="400,300;0,300;400,0;400,300" fill="freeze" />
</circle>
```

Die aktive Dauer der ersten Animation wird von 12s auf 20s verlängert, nicht die einfache Animationsdauer. Die Bewegung wird nach der einfachen Animationsdauer eingefroren, die Animation bleibt aber aktiv. Das Ende bei 24s wird als Synchronisationswert für die zweite Animation verwendet.

```
<circle r="15">
<animateMotion
   dur="12s" repeatDur="indefinite"
   begin="4s" end="activate;click"
   min="20s"
   values="0,300;400,0;400,300;0,300" fill="freeze" />
</circle>
```

Die Animation kann durch Aktivierung oder durch Anklickern beendet werden. Da die aktive Dauer auf mindestens 20s festgelegt ist, verzögert sich das Ende allerdings entsprechend.

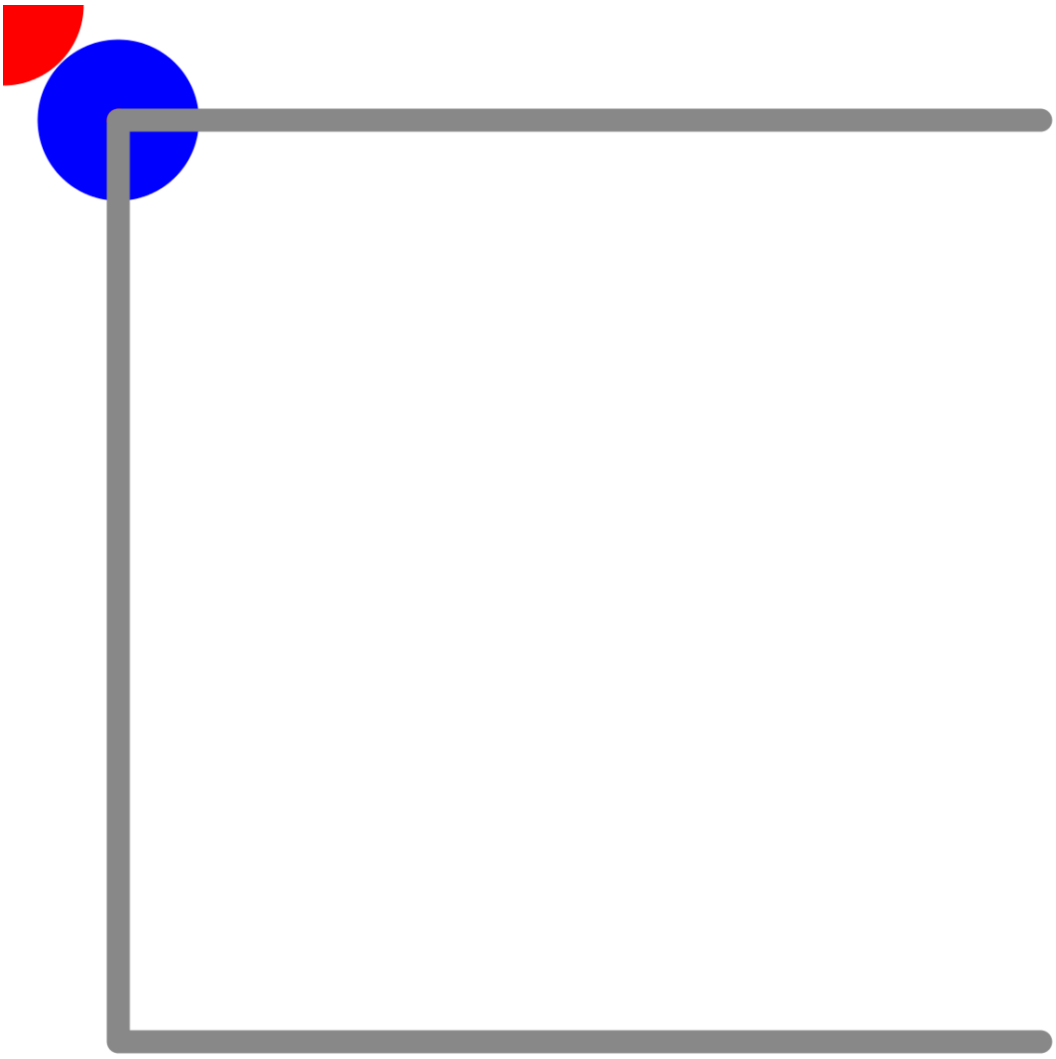


Abb. 256 Animationsattribut *min*

Animationsattribut *min*[45]

45 http://de.wikibooks.org/wiki/media%3ASVGmin01.svg

Einfache Animation für *cx* und *cy* eines Kreises mit *dur*-, *end*- und *min*-Attributen. Der graue Pfad gibt die richtige Trajektorie vor. Falls etwas Rotes sichtbar wird, ist ein Fehler aufgetreten.

14.7.2 *max*

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	9.5 (ab 9 teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4 (teilweise)
Konqueror (KSVG)	-
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit *max* wird die maximale aktive Dauer einer Animation angegeben. Der Wert ist entweder 'media' oder ein Zeitwert, wie bereits für *dur* definiert. 'media' ist nicht relevant für Animationselemente, nur für Elemente, die eine intrinsische Dauer haben (zum Beispiel ***audio*** und ***video*** in SVG tiny 1.2). Dann richtet sich die angegebene Dauer nach der intrinsischen Dauer des Mediums. Sofern 'media' für ein Animationselement angegeben ist, wird das Attribut ignoriert.

max schränkt die aktive Dauer einer Animation ein, sofern sie länger als die angegeben Dauer ist und überschreibt damit gegebenenfalls andere Angaben.

Beispiele:

```
<circle r="15">
<animateMotion id="a" dur="12s" begin="4s" end="14s" max="6s"
   values="0,300;400,0;400,300;0,300" fill="freeze" />
<animateMotion dur="12s" begin="a.end"
   values="400,300;0,300;400,0;400,300" fill="freeze" />
</circle>
```

Die aktive Dauer der ersten Animation wird von 12s auf 6s verkürzt, nicht die einfache Animationsdauer. Die Bewegung wird nach der verkürzten Dauer eingefroren. Das Ende bei 10s wird als Synchronisationswert für die zweite Animation verwendet.

```
<circle r="15">
<animateMotion
   dur="12s" repeatDur="indefinite"
   begin="4s" end="activate;click"
   max="3600s"
   values="0,300;400,0;400,300;0,300" fill="freeze" />
</circle>
```

Die Animation kann durch Aktivierung oder durch Anklickern beendet werden. Da die aktive Dauer auf maximal eine Stunde festgelegt ist, wird die Animation spätestens dann beendet und eingefroren.

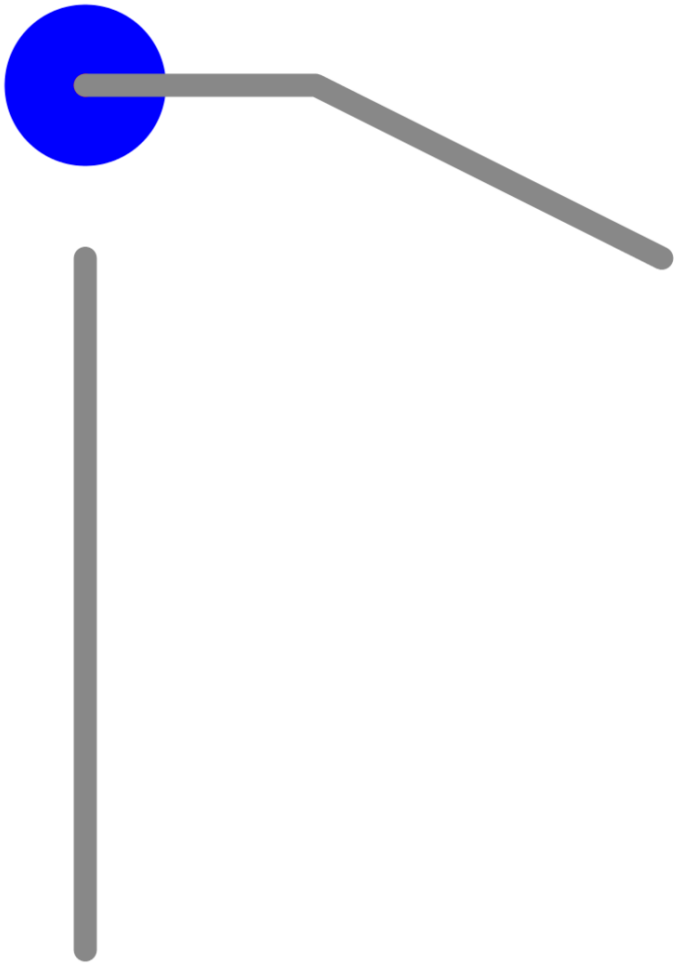


Abb. 257 Animationsattribut *max*

Animationsattribut *max*[46]

Einfache Animation mit *from* und *to* für *cx* und *cy* eines Kreises werden mit *max*-Attributen gestoppt (überschreibt *dur* und 'syncbase'-Werte für *end*). Der graue Pfad gibt die richtige Trajektorie vor. Falls das rote Zentrum des Kreis sichtbar wird, ist ein Fehler aufgetreten.

46 `http://de.wikibooks.org/wiki/media%3ASVGmax01.svg`

14.8 Bestimmung der aktiven Dauer der Animation

Die aktive Dauer der Animation definiert das gesamte Intervall, in der die jeweilige Animation einschließlich Wiederholungen und Korrekturen aktiv ist. Sowohl die durch *dur* festgelegte einfache Animationsdauer, als auch *end*, *repeatDur*, *repeatCount*, *min* und *max* haben Einfluss auf die aktive Dauer.

SMIL folgend sei im Weiteren B der Beginn der Animation, d die aktive Animationsdauer, PAD die vorläufige aktive Dauer der Animation, bevor *min* und *max* ausgewertet werden, IAD das Ergebnis einer Zwischenrechnung und AD die aktive Dauer der Animation.

Die Bestimmung der aktiven Dauer basiert auf den Informationen, die zu der Zeit verfügbar sind, wenn die Berechnung gemacht wird. Änderungen, Ereignisse, Synchronisationen können jederzeit eine erneute Berechnung erfordern.

Wenn *end* angegeben ist, aber weder *dur*, *repeatDur* und *repeatCount*, dann ist die einfache Animationsdauer unendlich, die aktive Dauer wird bestimmt durch den Endwert.

Wenn *end* ein aufgelöster (endlicher) Wert ist, dann ist PAD = *end* - B.

Sonst, falls *end* 'indefinite' (unendlich) ist, dann ist PAD = 'indefinite'.

Sonst, falls *end* nicht aufgelöst ist, ist auch PAD nicht aufgelöst und muss erneut berechnet werden, wenn mehr Informationen verfügbar sind.

Wenn *end* nicht angegeben ist oder 'indefinite' ist, dann wird IAD aus der Zwischenrechnung benötigt, dann ist

PAD = IAD

mit folgender Zwischenrechnung:

Für diese seien drei Größen eingeführt, um die zwischenzeitliche aktive Dauer IAD zu bestimmen, die in der vorherigen Betrachtung gebraucht wird.

p0 ist die einfache Animationsdauer.

Wenn *repeatCount* nicht angegeben ist, bekommt p1 den Wert 'indefinite'. Sonst ist p1 die kumulierte Summe der angegebenen Zahl von Wiederholungen der einfachen Dauer. p1 ist nicht aufgelöst, bis die einfache Dauer (einer jeden Wiederholung) aufgelöst ist. Teilweise Wiederholungen tragen entsprechend anteilig der einfachen Dauer zur Summe bei.

p2 ist der Wert von *repeatDur*. Ist dies nicht angegeben, so ist p2 'indefinite'.

Wenn p0 gleich 0 ist, so ist auch IAD gleich 0.

Falls sonst *repeatCount* und *repeatDur* nicht angegeben sind, gilt IAD = p0.

Sonst ist IAD das Minimum von p1, p2 und 'indefinite'.

Liegt hingegen ein anderer Wert für *end* als 'indefinite' vor mit wenigstens einer Angabe zu *dur*, *repeatDur* oder *repeatCount*, so ist PAD das Minimum von IAD und der Dauer *end* - B :

Schließlich ergibt sich die berechnete aktive Dauer AD, indem *min* und *max* angewendet werden auf PAD. Ein fehlender Wert für *min* kann als 0 angenommen werden, ein fehlender für *max* unendlich ('indefinite'):

AD ist das Minimum von *max* und dem Maximum von *min* und PAD.

14.9 Kontrolle des Neustarts von Animationen

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	9 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4 (teilweise)
Konqueror (KSVG)	-
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Es gibt mehrere Möglichkeiten, eine Animation erneut zu starten. Ob tatsächlich ein Neustart erfolgt, wird bestimmt durch das Attribut *restart*.

Eine Möglichkeit, einen Neustart zu notieren ist, mehrere Beginnzeiten in der Liste des Attribute *begin* anzugeben. Sind in *begin* Ereigniswerte angegeben oder Synchronisationswerte, kann dies ebenfalls mehrere Neustarts bewirken, wenn die Ereignisse oder Synchronisationen mehrmals zutreffen. Ferner kann mit einem Verweis (oder einem Zugriff über das Dokument-Objekt-Modell) ein Neustart veranlaßt werden.

Mögliche Werte von *restart* sind:

always

Die Animation kann jederzeit erneut gestartet werden

whenNotActive

Die Animation kann nur erneut gestartet werden, wenn sie nicht aktiv ist, zum Beispiel nach dem Ende der aktiven Dauer. Versuche, die Animation während der aktiven Dauer erneut zu starten, werden ignoriert.

never

Die Animation kann nicht innerhalb der Dokumentzeit erneut gestartet werden, wenn die Animation bereits einmal gestartet worden ist. Es müsste also das komplette Dokument erneut gestartet werden.

Sofern nicht notiert, wird 'always' angenommen.

Wenn eine Animation erneut startet, hängt das Verhalten nicht von früheren Starts ab, einschließlich des Verhaltens von *fill*.

Wird eine aktive Animation erneut gestartet, so wird zunächst die aktive Dauer beendet, dann erfolgt der Neustart. Das Ende erzeugt ein Endereignis, welches zur Synchronisation an abhängige Animationen weitergegeben wird. Weil ein erneuter Start allenfalls durchgeführt wird, nachdem die aktive Dauer berechnet wurde, hat dies keinen Einfluss auf die Anwendung des Attributes *min*.

Beispiele:

```
<circle r="15">
<animateMotion dur="12s" begin="activate;click"
  values="0,300;400,0;400,300;0,300"
  restart="whenNotActive" />
</circle>
```

Die Animation wird durch aktivieren oder klickern des Kreises gestartet. Sie kann nur erneut gestartet werden, wenn sie nicht mehr aktiv ist.

```
<circle r="15">
<animateMotion dur="12s" begin="activate;click;60s"
  values="0,300;400,0;400,300;0,300"
  restart="never" />
</circle>
```

Die Animation wird durch aktivieren oder klickern des Kreises gestartet oder automatisch nach 60s. Sie kann nicht erneut gestartet werden.

```
<circle r="15">
<animateMotion dur="12s" begin="activate;click"
  values="0,300;400,0;400,300;0,300"
  restart="always" />
</circle>
```

Die Animation wird durch aktivieren oder klickern des Kreises gestartet. Sie kann jederzeit erneut gestartet werden.

```
<circle r="15">
<animateMotion dur="12s" repeatDur="indefinite"
  begin="activate;click" end="activate;click"
  values="0,300;400,0;400,300;0,300"
  restart="whenNotActive" />
</circle>
```

Die sich wiederholende Animation kann durch aktivieren oder klickern des Kreises gestartet werden. Ist sie aktiv, kann sie durch erneutes aktivieren oder klickern beendet werden. Dies trifft zu, weil für *restart* 'whenNotActive' notiert ist und das Ende nicht vor dem Beginn eintreten kann. Wäre 'always' notiert, täte ein erneutes aktivieren oder klickern die Animation zwar auch beenden, gleichzeitig aber auch neu starten, was so nicht möglich ist. Wäre 'never' notiert, so könnte die Animation zwar genau einmal gestartet und einmal beendet werden, aber nicht mehrfach.

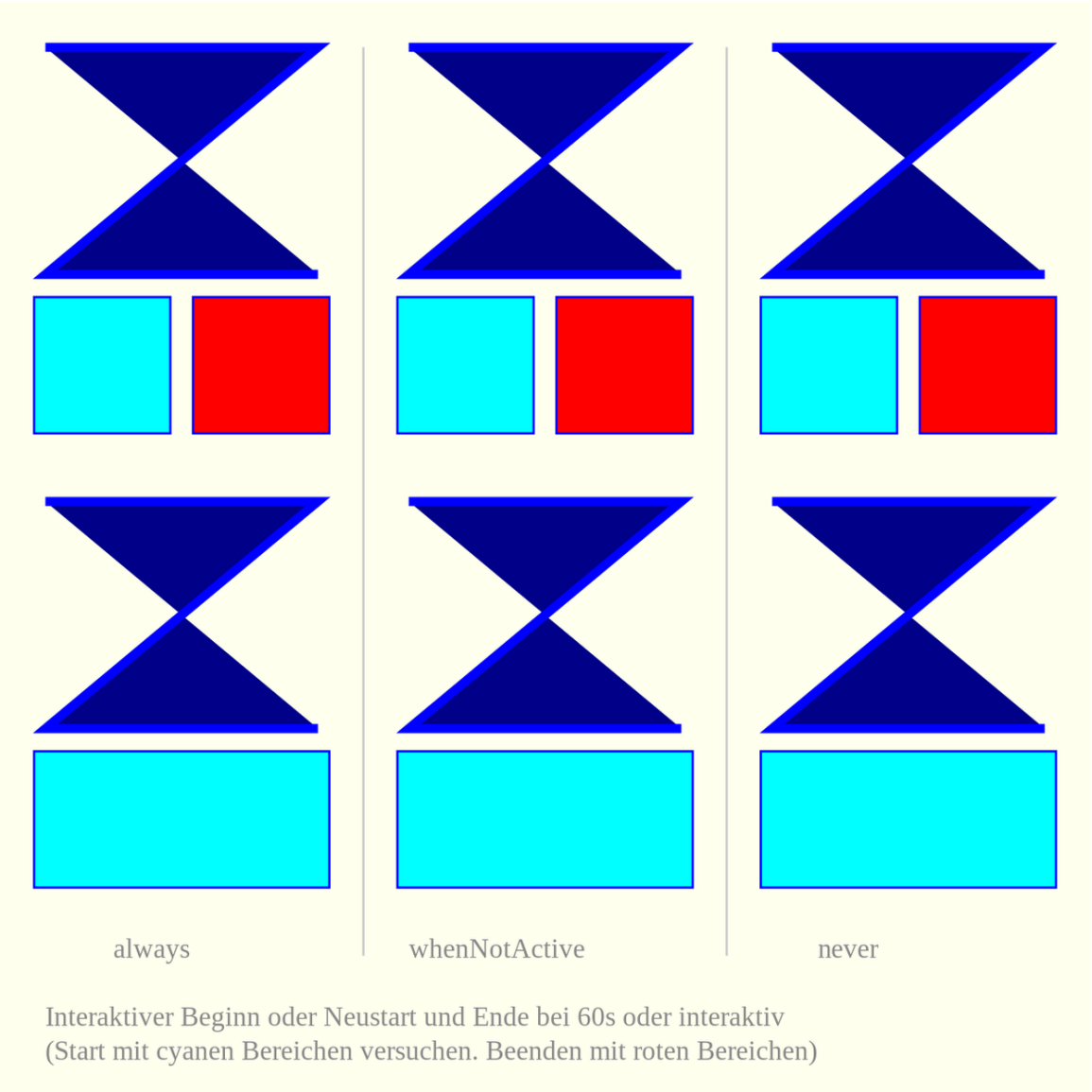


Abb. 258 Interaktiver *restart* und *end*

Interaktiver *restart* und *end*[47]

Das Dokument beinhaltet sechs Subtests mit verschiedenem Verhalten. Sie haben cyane Bereiche, um die Animation einer Polylinie zu versuchen zu aktivieren (activate oder mouseup). Die oberen Subtests haben auch rote Bereiche um die Animation zu versuchen zu stoppen (activate oder mouseup). Alle Animationen haben *end*-Attribute mit dem Wert 60s, daher kann nach 60s keine Animation mehr gestartet oder neu gestartet werden. Dies wird mit einer Änderung der Bereiche auf grau gekennzeichnet. Die *dur*-Werte für die Animationen sind 5s, *repeatDur* is 10s. Die Tests rechts können immer neu gestartet werden, in der Mitte whenNotActive, rechts nie.

47 `http://de.wikibooks.org/wiki/media%3ASVGrestart01.svg`

Zu testendes Verhalten:

Oben links: Innerhalb von 60s immer neustartbar, stop hat einen sichtbaren Effekt, wenn die Animation aktiv ist.

Unten links: Innerhalb von 60s immer neustartbar.

Oben mitte: Neustartbar whenNotActive und innerhalb von 60s, stop hat einen sichtbaren Effekt, wenn die Animation aktiv ist.

Unten mitte: Neustartbar whenNotActive und innerhalb von 60s.

Oben rechts: Nie neustartbar oder kein Start nach 60s, stop hat einen sichtbaren Effekt, wenn die Animation aktiv ist.

Unten rechts: Nie neustartbar oder kein Start nach 60s.

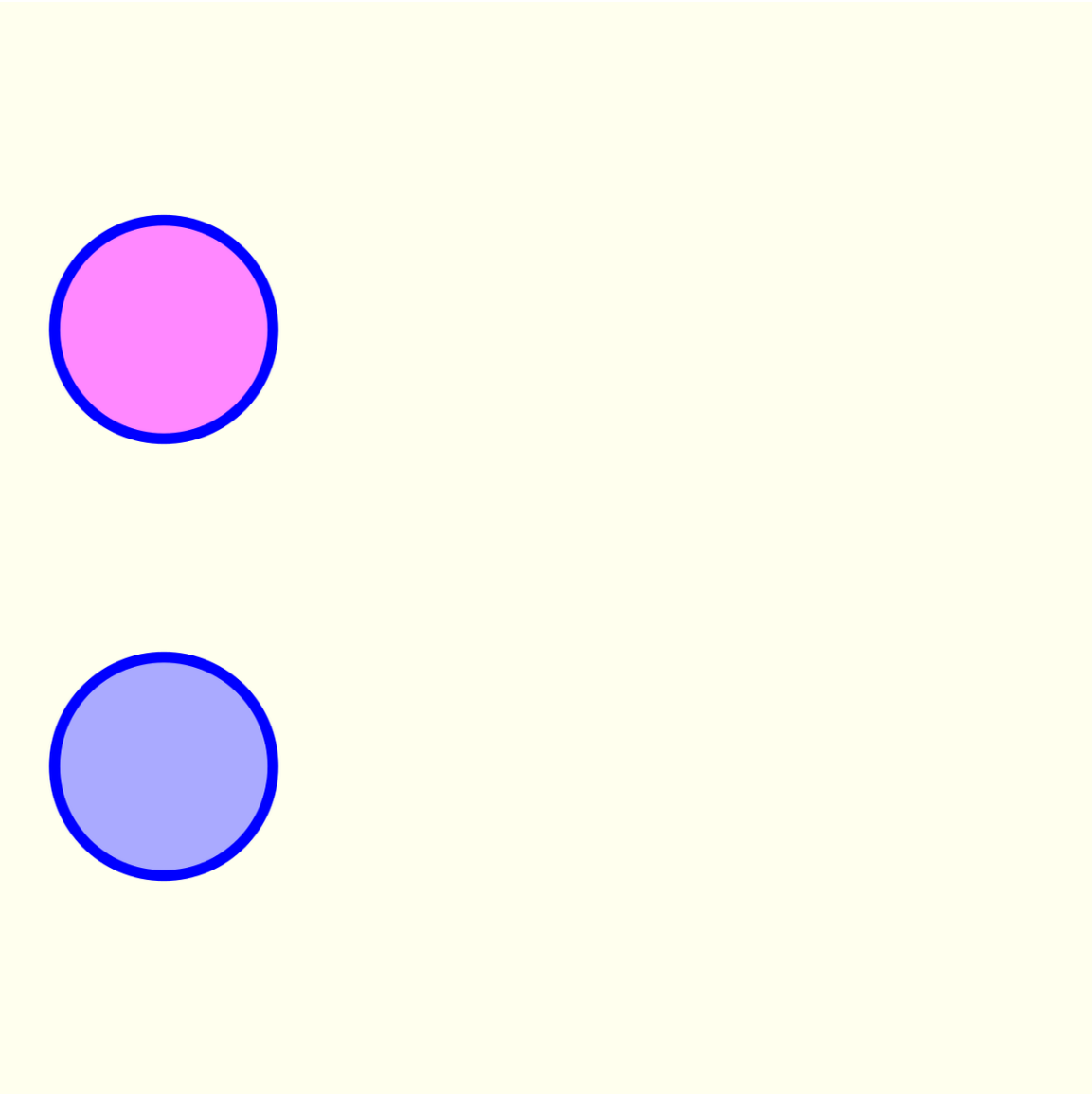


Abb. 259 begin und end mit einem Klick

begin und end mit einem Klick[48]

Mit einem Klick auf den Kreis wird dessen Animation gestartet. Mit einem weiteren Klick wird sie gestoppt. Dies funktioniert, weil restart auf never für den magenta Kreis gesetzt ist und whenNotActive für den hellblauen.

48 http://de.wikibooks.org/wiki/media%3ASVGrestart02.svg

14.10 Additive und kumulative Animation

Eine Animation kann zu einem darunterliegenden Wert addiert werden (sofern Addition für das Animationsziel eine definierte und erlaubte Operation ist). Die Reihenfolge des Vorgehens ist dabei so, dass die einfache Animationsfunktion unabhängig vom darunterliegenden Wert bestimmt wird. Anschließend werden beide Werte zur Animationseffektfunktion aufaddiert.

Sich wiederholende Animationen können ferner kumulativ sein (sofern Addition für das Animationsziel eine definierte und erlaubte Operation ist). In dem Falle werden die einzelnen Durchläufe der Wiederholung zu einem Animationseffekt aufaddiert. Die Wiederholung ist zu unterscheiden von einem erneuten Start der Animation, der niemals kumulativ ist, also nicht auf früheren Durchläufen basieren kann.

Kumulierende Animationen sind nicht zwangsläufig additive Animationen und umgedreht, daher gibt es zwei Attribute, die das Verhalten unabhängig voneinander bestimmen.

Jedes Attribut und jede Eigenschaft ist in SVG definiert als entweder additiv oder nicht-additiv. Gemeint ist damit nicht, was der Autor für die Animation angibt, sondern was in der Spezifikation steht, ob eine additive Operation prinzipiell definiert ist.

Bei nicht-additiven wird also bei einer Animation das Attribut *additive* ignoriert. Bei additiven kann das Attribut *additive* verwendet werden, um festzulegen, ob die Animation additiv sein soll oder nicht.

In SVG ist die Frage, ob eine Animation auch kumulativ sein kann, daran gekoppelt, ob das zu animierende Attribut oder die Eigenschaft additiv ist oder nicht. Wenn nicht, wird auch das Attribut *accumulate* entsprechend ignoriert.

Ob in SVG Attribute oder Eigenschaften additiv sind oder nicht, hängt im Wesentlichen davon ab, zu welchem Datentyp sie gehören.

Winkel, Farben, Koordinaten, einzelne Zahlen, einzelne Längen, Prozentwerte, Transformationen sind additiv. Nicht additiv sind Frequenzangaben, Listen von Werten, Zeitangaben, URIs und alle anderen Datentypen.

Weil in SVG tiny 1.2 einige Datentypen geändert wurden, kann das zu einigen wenigen Änderungen hinsichtlich der Additivität führen. Zum Beispiel sind in SVG 1.1 für ***text*** und Verwandte die Attribute *x*, *y* etc als gegebenenfalls mehrere Koordinaten aufgeführt (oder bei *rotate* mehrere Winkel). In SVG tiny 1.2 werden diese Attribute jedoch als Listen geführt. Daraus kann abgeleitet werden, dass diese Attribute in SVG 1.1 noch additiv sein können, in SVG tiny 1.2 nicht mehr. Weil eine korrekte Implementierung des additiven Verhaltens mit jeweils einer beliebigen Anzahl von Koordinaten oder Winkeln aber ohnehin nicht trivial ist, ist es für Autoren sicherer, bei solch komplexen Attributen von einer additiven Animation auch in SVG 1.1 abzusehen, wenn ein einheitliches Verhalten erzielt werden soll.

Problematisch sind die etwas überfrachteten Eigenschaften *fill* und *stroke*, weil diese auch Werte haben können, für die es keine additive Operation gibt, wie etwa Maldienste oder der Wert 'none'. Prinzipiell sind die Attribute additiv, solange die Werte und sofern relevant der darunterliegende Wert einen Farbwert repräsentieren. Für Farbwerte ist eine additive Operation definiert, für Maldienste oder 'none' nicht. Wegen dieser Komplikation müsste ein Darstellungsprogramm prinzipiell

untersuchen, ob eine additive Animation durchführbar ist oder nicht und dann entscheiden, ob die Angabe von *additive* oder *accumulate* ignoriert werden muss oder nicht. Die Antwort auf diese Frage kann bei komplexeren Animation mit Nutzerinteraktion zeitabhängig und schwierig zu finden sein. In der Praxis sind folglich die meisten Darstellungsprogramme mit einer Analyse und einer korrekten Schlußfolgerung vor Beginn der aktiven Dauer überfordert. Ist ein Autor daran interessiert, ein einheitliches Verhalten zu bewirken, so ist dringend zu empfehlen, Animationen von *fill* oder *stroke* nicht als additiv oder kumulativ zu kennzeichnen, wenn dies unter gewissen Umständen nicht über die ganze aktive Dauer der Animation durchführbar ist.

Wird dies nicht beachtet, kann ein undefiniertes, teils sicherlich auch fehlerhaftes Verhalten in den meisten Darstellungsprogrammen beobachtet werden, zum Beispiel könnten nicht additive Werte prinzipiell als schwarz interpretiert werden oder das Programm ignoriert *additive* oder *accumulate* erst und vielleicht auch nur so lange, wie die Werte nicht addierbar sind.

14.10.1 *accumulate*

Squiggle (Batik)	1.7 (meist falsch)
Opera (Presto)	10 (ab 8 teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	-
Safari (Webkit)	4 (meist falsch)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Attribut *accumulate* wird angegeben, ob eine Animation kumulativ sein soll oder nicht. Kumulativ heißt, dass Wiederholungen bedingt durch *repeatDur* oder *repeatCount* aufaddiert werden.

Mögliche Werte sind:

sum

Die Animation ist kumulativ. Wiederholungen werden aufeinander aufaddiert.

none

Die Animation ist nicht kumulativ und die einfache Animationsfunktion wird lediglich wiederholt.

'none' wird angenommen, wenn das Attribut nicht angegeben ist. Das Attribut wird ignoriert, falls das Animationsziel keine additive Animation erlaubt (in SVG; in SMIL, wenn es keine additive Operation erlaubt) oder die Animation sich nicht wiederholt.

Beim ersten Durchlauf ist die einfache Animationsfunktion $f_0(t)$ unbeeinflusst von kumultiven Effekten.

Also $f_0(t) = f(t)$.

Ist die Animation nicht kumulativ, so gilt für den i-ten Durchlauf (mit i einer positiven ganzen Zahl den Durchlauf angebend):

$f_i(t) = f(t - (i*d))$

Ist die Animation kumulativ, dann gilt:

$f_i(t) = (f(d) * i) + f(t - (i*d))$

Die kumlative Animationsfunktion ist dann

$f_c(t) = f_i(t)$, wobei $i = floor(t/d)$

(wobei floor eine Operation ist, die zur nächstkleineren ganzen Zahl abrundet).

Wird der erste Wert der *values*-Liste als 0 gewählt, so gibt es keine Sprünge bei der Wiederholung einer kumulativen Animation. Wird der letzte Wert 0 gewählt, entspricht dies wieder einer nicht kumulativen Animation. 0 ist in dem Sinne als das neutrale Element der Addition zu verstehen, wie es für den Wert des Animationszieles definiert ist. Bei Farben ist das zum Beispiel schwarz oder #000. Bei Skalaren (einfachen Zahlen, optional auch mit Einheiten) ist es die 0, bei Vektoren der 0-Vektor, bei einer Liste von Zahlen der Eintrag 0 für jeden Wert der Liste.

Transformationen, welche mit ***animateTransform*** animiert werden können, stellen sich zwar intern als Matrixmultiplikationen dar. Entscheidend ist hier aber, was als zu animierender Wert notiert ist und das sind Zahlen oder Listen von Zahlen, folglich ist die additive Operation die von Zahlen, nicht die Multiplikation von Matrizen. Die Matrixmultiplikation gibt hingegen nur an, wie letztlich die Animationseffektfunktion zur Präsentation gebracht wird, nicht wie sie berechnet wird.

Beispiele:

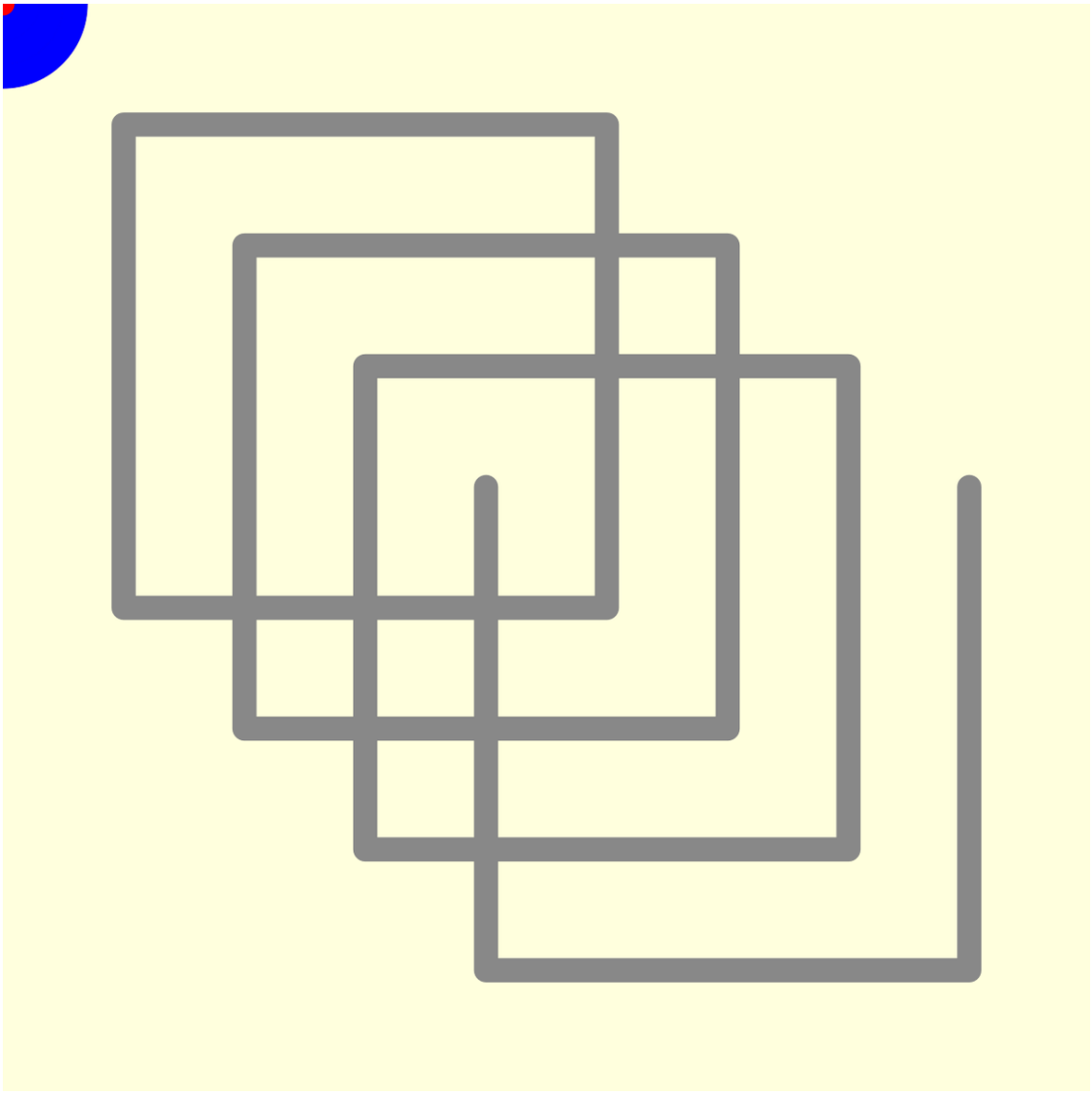


Abb. 260 kumulative ***animateMotion***

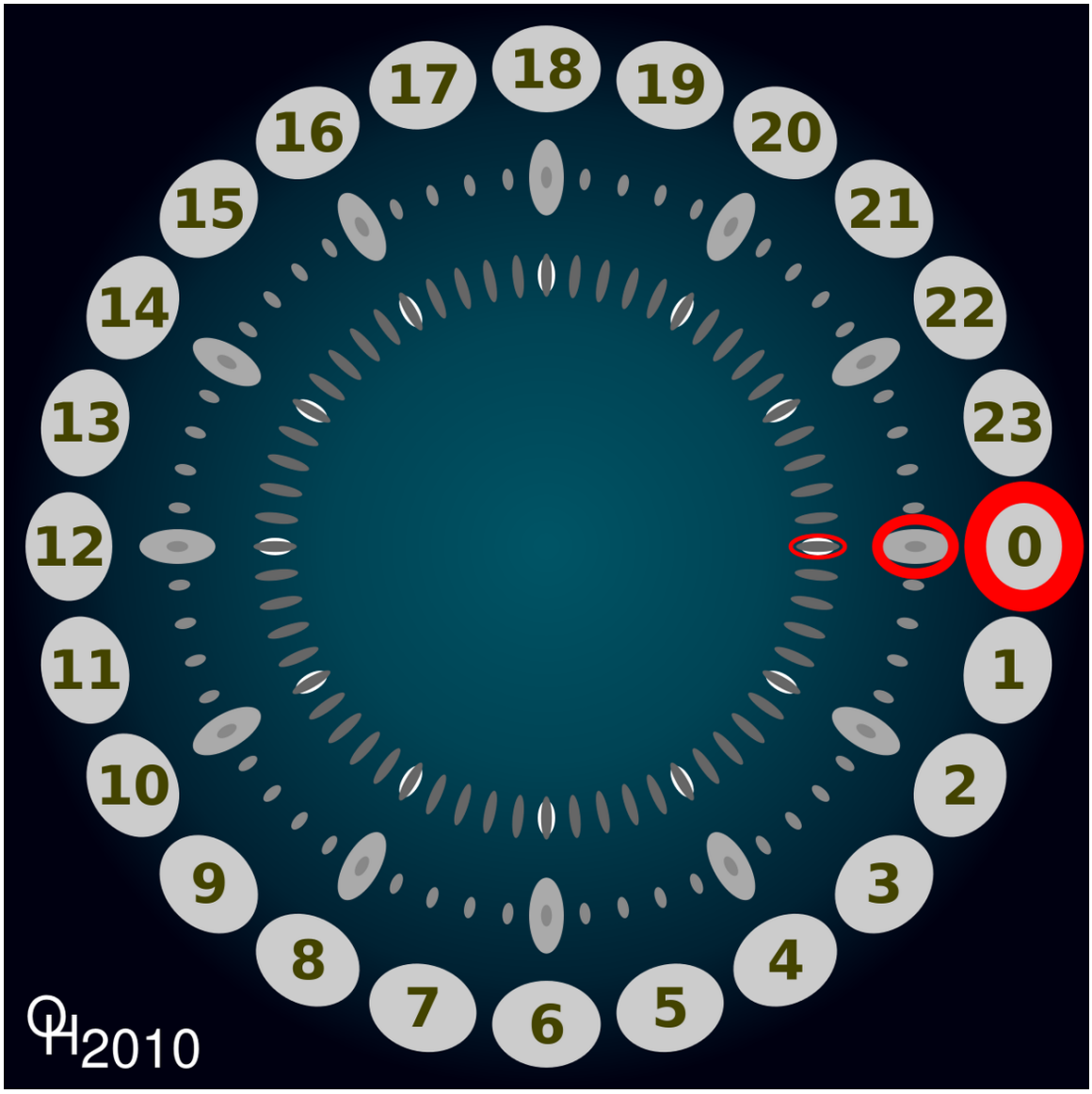


Abb. 261 Analoguhr

```
<circle r="15">
<animate attributeName="cx" dur="10s" repeatDur="40s"
accumulate="sum"
values="0;100;500;200;300;100" />
</circle>
```

Die einzelnen Durchläufe der Wiederholung kumulieren, bei 10s beginnt der neue Durchlauf also mit 100, nicht mit 0. Effektiv sind die Werte dann 100;200;600;300;400;200. Beim nächsten Durchlauf entsprechend 200;300;700;400;500;300.

```
<circle r="15">
<animate attributeName="cx" dur="10s" repeatDur="40s"
accumulate="none"
values="0;100;500;200;300;100" />
</circle>
```

Die einzelnen Durchläufe der Wiederholung kumulieren nicht, bei 10s beginnt der neue Durchlauf also wieder mit 0.

kumulative ***animateMotion***[49]

Kreise werden entlang von Quadraten gegeben mit *values*, *repeatDur* und *accumulate* mit einer Dauer von 30s und Wiederholung bewegt. Auch sind verschiedene Werte für das Attribut *rotate* gegeben, was keinen sichtbare Effekt hat, weil zentrierte Kreise bewegt werden. Falls etwas Rotes sichtbar wird, ist ein Fehler aufgetreten.

Auch folgende Uhr profitiert vom kumulativen Verhalten, weil so nur jeweils für den Sprung einer Sekunde, einer Minute und einer Stunde die entsprechenden Werte von *values*, *keyTimes* und *keySplines* angegeben werden müsssen und nicht für ganze Listen von Sekunden, Minuten und Stunden: Analoguhr[50]

14.10.2 *additive*

Squiggle (Batik)	1.7 (größtenteils)
Opera (Presto)	9.5 (ab 8 teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Attribut *additive* wird angegeben, ob eine Animation additiv sein soll oder nicht.

Mögliche Werte sind:

sum

Die Animation ist additiv und wird zum darunterliegenden Wert addiert (einschließlich der möglichen Animationen mit niedrigerer Priorität).

replace

Die Animation ist nicht additiv und überschreibt den darunterliegenden Wert.

'replace' wird angenommen, wenn das Attribut nicht angegeben ist. Das Attribut wird ignoriert, falls das Animationsziel keine additive Animation erlaubt.

Ist $f_f(t)$ die einfache Animationsfunktion zusammen mit Wiederholungen und dem eingeforenen Wert am Ende der Animation, so ergibt sich die Animationseffektfunktion als:

$F(t,u) = u + f_f(t)$, wenn die Animation additiv ist,

und

$F(t,u) = f_f(t)$. wenn nicht.

49 `http://de.wikibooks.org/wiki/media%3ASVGanimateMotion_accumulate01.svg`
50 `http://de.wikibooks.org/wiki/media%3ASVG_Analoguhr02.svg`

Beispiele:

Abb. 262 Additive und kumulative Animationen

Abb. 263 Farbkanäle unabhängig voneinander animieren

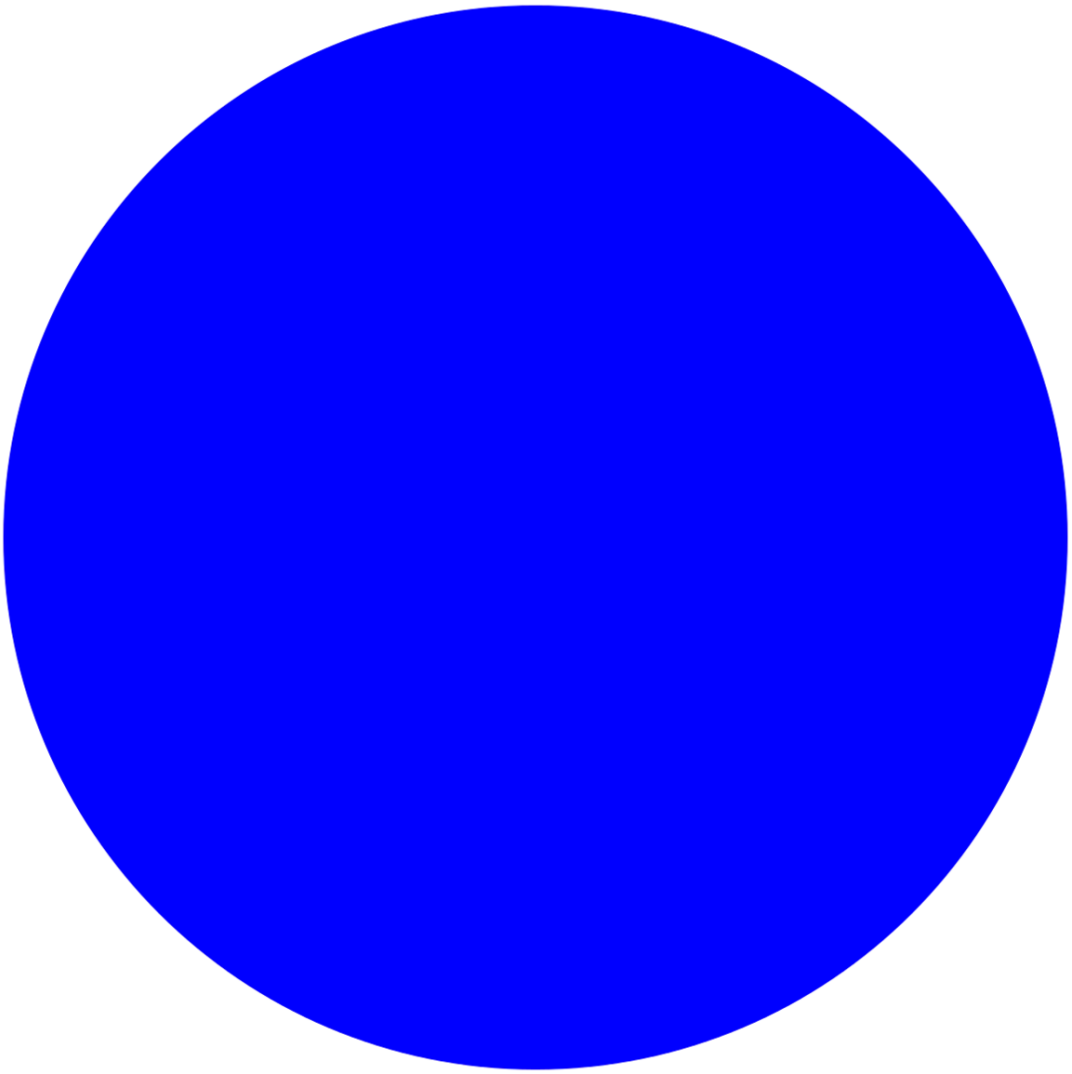


Abb. 264 Additive ***animateMotion***

```
<circle r="15" cx="100">
<animate attributeName="cx" dur="10s" repeatDur="40s"
additive="sum"
values="0;100;500;200;300;100" />
</circle>
```

Diese additive Animation hat also effektiv die Präsentationswerte 100;200;600;300;400;200 bei jedem Durchlauf.

```
<circle r="15" cx="100">
<animate attributeName="cx" dur="10s" repeatDur="20s"
values="400;-100" fill="freeze" />
<animate attributeName="cx" dur="10s" repeatDur="40s"
additive="sum" fill="freeze"
values="0;100;500;200;300;100" />
</circle>
```

Die erste Animation ist nicht additiv und überschreibt den darunterliegenden Wert von *cx* von 100 mit der Animation. Die zweite Animation ist additiv und zu jedem Zeitpunkt wird der darunterliegende Wert der anderen Animation hinzuaddiert. Die Animation beginnt also mit dem Wert 400 und endet mit 0.

```
<circle r="15" cx="100">
<animate attributeName="cx" dur="10s" repeatDur="20s"
values="400;-100" fill="freeze"
accumulate="sum" />
<animate attributeName="cx" dur="10s" repeatDur="40s"
additive="sum" fill="freeze"
values="0;100;500;200;300;100" />
</circle>
```

Anders als beim vorherigen Beispiel kumuliert hier die erste Animation, die Animation beginnt also wieder mit 400, endet aber letztlich mit -100.

Additive und kumulative Animationen[51]

Vergleich von *from-to*, *from-by* und *by* animate mit *values* animate.

Die *from-to*-, *from-by*- und *by*-Animationen angewendet auf die Attribute x, y, width und height der blauen Rechtecke werden mit den entsprechenden *values*-Animationen einschließlich additivem und kumulativem Verhalten für das unterliegende rote Rechteck verglichen. Ein zusätzliches dunkelrotes Rechteck simuliert das gleiche Verhalten, wobei immer *additive* 'replace' und *accumulate* 'none' verwendet wird. Die blauen Rechtecke verdecken alle roten Rechtecke. Daher ist ein Fehler aufgetreten, wenn etwas Rotes sichtbar wird.

Farbkanäle unabhängig voneinander animieren[52]

Die drei Kanäle rgb werden im großen Rechteck unabhängig voneinander animiert. In den kleinen Rechtecken sind die gleichen Animationen einzeln zu sehen.

Additive ***animateMotion***[53]

Ein Kreis wird entlang einiger Pfade mit einer Dauer von 10s entlang bewegt, wobei *additive* ***animateMotion*** verwendet wird. Die Superposition ergibt keine Animation. Der Kreis ist immer an der gleichen Stelle positioniert. Die Farbe des Kreises ändert sich von blau nach grau, nachdem die Animation beendet ist. Falls das rote Zentrum des Kreises sichtbar wird, ist ein Fehler aufgetreten.

51 http://de.wikibooks.org/wiki/media%3ASVGadditive01.svg
52 http://de.wikibooks.org/wiki/media%3ASVGanimateColor_additive02.svg
53 http://de.wikibooks.org/wiki/media%3ASVGanimateMotion_additive02.svg

14.11 Kontrolle des Zeitablaufes

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	-
Safari (Webkit)	4 (größtenteils)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Attribut *keyTimes* kann eine Liste mit Zeiten als Anteil der einfachen Animationsdauer angegeben werden, wann die in der *values*-Liste angebenen Werte angenommen werden sollen. Falls das Attribut angegeben ist, muss es exakt so viele Werte enthalten wie die *values*-Liste. Die Werte korrespondieren in der angegebenen Reihenfolge.

Jeder Wert der *keyTimes*-Liste ist eine Zahl zwischen 0 und 1. Die Werte in der Liste müssen monoton steigend notiert sein, der erste Wert muss 0 sein. Die Werte stellen einen Zeitversatz relativ zum Beginn der einfachen Animationsdauer dar in Einheiten der einfachen Animationsdauer.

Die Bedeutung und Notation hängt von der noch zu diskutierenden Interpolationsart (Attribut *calcMode*) ab:

Bei einer gleichmäßigen Animation (englisch: paced) wird das Attribut *keyTimes* ignoriert und diese Liste automatisch entsprechend der Abstände zwischen den *values*-Listenwerten berechnet.

Bei einer diskreten Animation gibt der *keyTimes*-Listenwert an, wann zum entsprechenden *values*-Listenwert gesprungen wird.

Sonst muss der letzte Wert 1 sein und die Interpolation erfolgt jeweils so, dass der zum *keyTimes*-Listenwert gehörige *values*-Listenwert jeweils exakt zum angegebenen Zeitpunkt angenommen wird. In der Zeit dazwischen wird gemäß der Interpolationsmethode interpoliert, welche gemäß *calcMode* angegeben oder impliziert ist.

Ist *keyTimes* nicht angegeben, so wird die einfache Animationsdauer entsprechend der Anzahl der *values*-Listenwerte in gleichlange Intervalle unterteilt. Bei einer diskreten Animation mit n Werten 1 bis n entspricht das also einer *keyTimes*-Liste mit den Werten (i-1)/n mit i von 1 bis n. Bei einer kontinuierlichen Animation mit n größer als 1 entsprechend (i-1)/(n-1). Eine entsprechende Liste wird dann gegebenenfalls impliziert.

Zu beachten ist, dass bei einer diskreten Animation der *values*-Listenwert zum *keyTimes*-Listenwert 1 nicht unbedingt sichtbar wird, insbesondere wenn die Animation nicht eingefroren ist und nicht kumulativ ist.

Wenn die einfache Animationsdauer unendlich ('indefinite') ist, wird das Attribut *keyTimes* ignoriert.

Gibt es Fehler in der Notation, so wird die Anzeige des Dokumentes bei SVG 1.1 abgebrochen. Bei SVG tiny 1.2 wird legiglich das Attribut ignoriert.

Beispiele:

```
<line x1="100" x2="400" y1="100" y2="400" stroke="black" stroke-width="10">
<animate attributeName="stroke-linecap" dur="30s"
```

```
        values="square;round;butt" fill="freeze"
        keyTimes="0;0.2;1" />
</line>
```

Die Animation von *stroke-linecap* ist immer diskret, was an den möglichen Werten liegt, die nicht interpolierbar sind. Zu Beginn ist *stroke-linecap* 'square', 0.2 * 30s = 6s springt es nach 'round'. Nach 30s wird der die Animation mit dem Endwert 'butt' eingefroren.

```
<line x1="100" x2="400" y1="100" y2="400" stroke="black" stroke-width="10">
<animate attributeName="stroke-linecap" dur="30s" repeatDur="60s"
        values="square;round;butt" fill="freeze"
        keyTimes="0;0.2;1" />
</line>
```

Die gleiche Animation wie zuvor, nur jetzt mit Wiederholung. Sie beginnt wieder mit 'square', springt nach 6s auf 'round'. Nach 30s setzt die Wiederholung ein, der Wert springt wieder auf 'square', nach 36s auf 'round' und wird nach 60s mit dem Endwert 'butt' eingefroren.

```
<circle r="15" cx="100">
<animate attributeName="cx" dur="50s"
values="0;100;500;200;300;100"
keyTimes="0;0.4;0.5;0.7;0.9;1" />
</circle>
```

cx eines Kreises wird kontinuierlich animiert. Entsprechend der *keyTimes* erfolgt die Zeiteinteilung. Die *cx* beginnt mit 0 und ändert sich kontinuierlich innerhalb von 20s nach 100, dann ändert sich der Wert nach 500 innerhalb von weiteren 5s, weiter nach 200 innerhalb von 10s, dann nach 300 in weiteren 10s und weiter zum Endwert 100 nach der einfachen Animationsdauer von 50s.

```
<circle r="15" cx="100">
<animate attributeName="cx" dur="50s"
values="0;500;400;200"
keyTimes="0;0.5;0.5;1" />
</circle>
```

Ähnliches Beispiel wie zuvor. Nur mit weniger Werten. Zudem wird hier nach der halben einfachen Animationsdauer in der ansonsten kontinuierlichen Animation ein diskreter Sprung hervorgerufen, weil zweimal dieselbe Wert in der *keyTimes*-Liste notiert ist. Dies ist erlaubt, nur nicht die Angabe kleinerer Werte. Die Interpolation erfolgt also in den ersten 25s von 0 nach 500, in den zweiten 25s von 400 nach 200.

```
<circle r="15" cx="100">
<animate attributeName="cx" dur="50s"
values="0;500;400;200" fill="freeze"
keyTimes="0;0.5;1;1" />
</circle>
```

Ähnliches Beispiel wie zuvor. Nur wird hier zweimal der *keyTimes*-Listenwert 1 verwendet, um zu bewirken, dass der eingefrorene Wert (200) ein anderer ist als der letzte Wert, der sich aus der kontinuierlichen Animation ergibt (400).

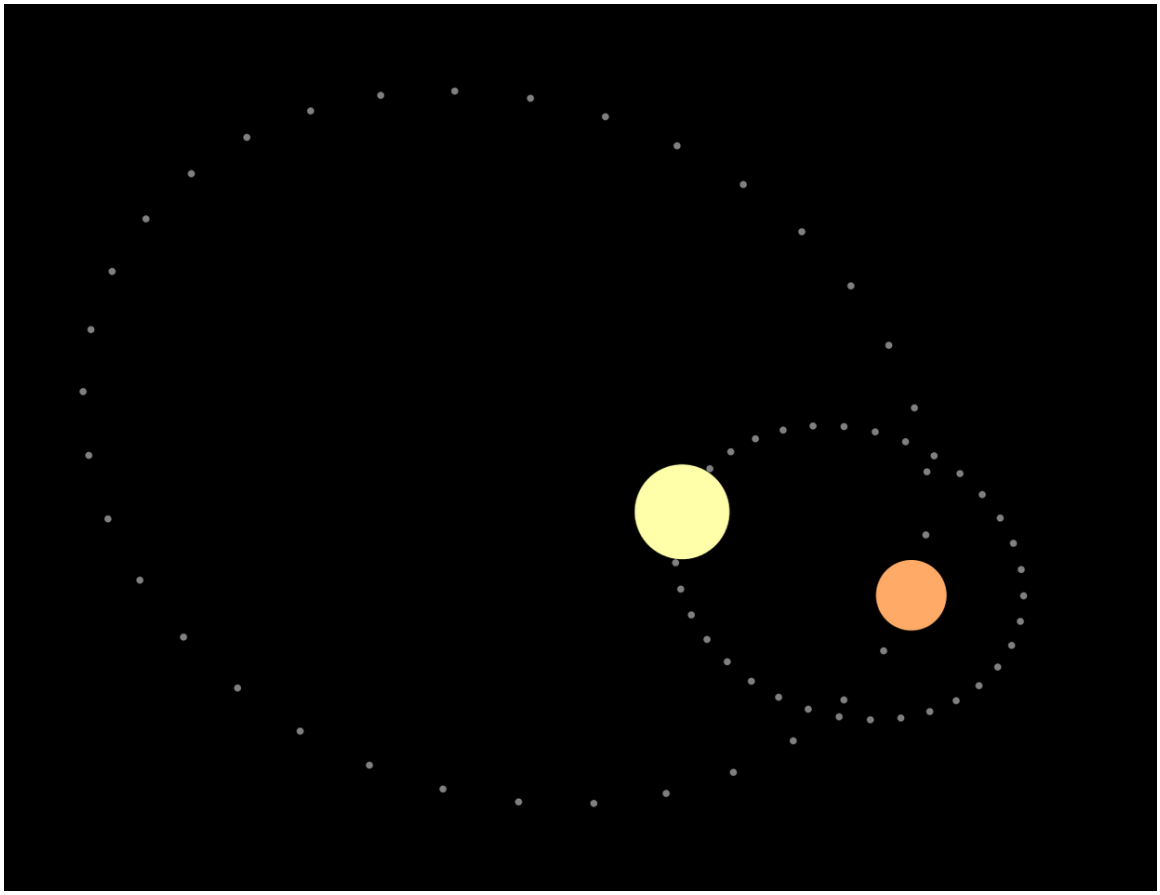


Abb. 265 Bewegung eines Doppelsternensystems

Anwendungsbeispiele aus dem Bereich der klassischen Physik:

Bewegung eines Doppelsternensystems[54]

Zwei Sterne bewegen sich um ihren gemeinsamen Schwerpunkt.

Das Problem ist in der Physik bekannt als Zweikörperproblem und kann zum Beispiel für das bei einem Doppelsternensystem vorliegende Gravitationspotential analytisch gelöst werden (klassische Näherung). Von der Lösung wird die Liste für *values* bestimmt und die für *keyTimes*. Bei der analytischen Lösung ist sowohl der Ort als auch die Zeit als Funktion eines Laufparameters gegeben, daher ist die Verwendung solcher Listen ideal, anders als eine explizite Zeitabhängigkeit des Ortes zu bestimmen.

Statt mit einer *values*-Liste ginge es noch genauer mit einer Bewegung entlang eines Pfades und einer zusätzlichen kubischen Interpolation der Bewegung, das ist allerdings deutlich arbeitsaufwendiger und ist für den visuellen Effekt nicht notwendig.

54 `http://de.wikibooks.org/wiki/media%3ASVG_Doppelsternensystem01.svg`

Beim Doppelsternensystem handelt es sich um einen gebundenen Zustand im Gravitationspotential, genauso sieht die klassische Lösung für das Coulombpotential für Teilchen mit entgegengesetzter Ladung aus. In der Praxis gibt es allerdings kaum so große geladene Objekte, dass die klassische Näherung, also ohne Berücksichtigung von Strahlung und Quantenphysik besonders hilfreich wäre. Anders ist das bei nicht gebundenen Zuständen, das sind dann Streulösungen. Weil die Gravitation immer attraktiv ist, tritt da nur eine Möglichkeit auf, beim Coulombpotential können auch gleiche Ladungen vorliegen, dann ist das Potential repulsiv und es gibt nur die Streulösung und keinen gebundenen Zustand.

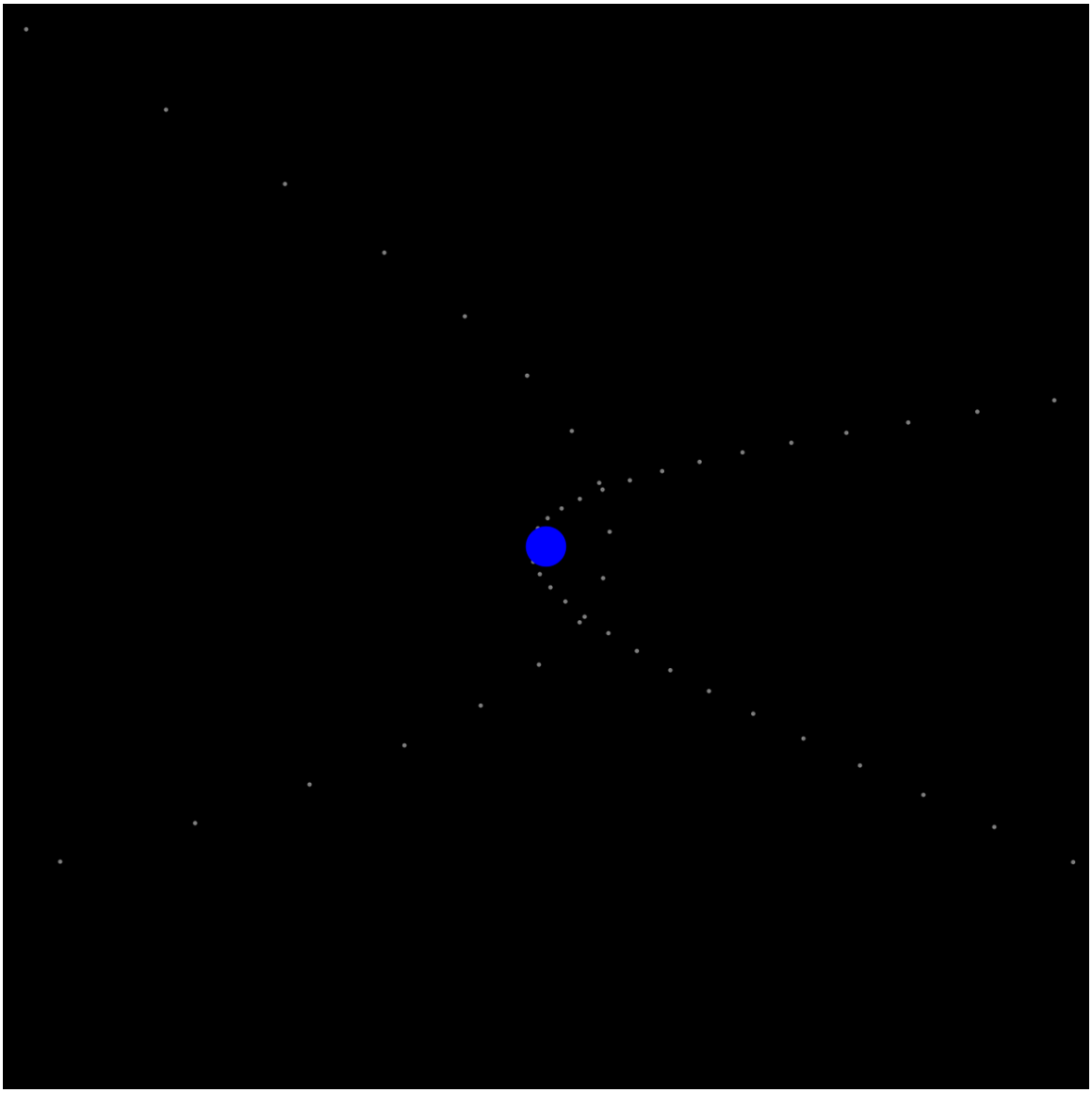


Abb. 266 Streuung im attraktiven Potential

Streuung im attraktiven Potential[55]

55 http://de.wikibooks.org/wiki/media%3ASVG_Streuung01.svg

Bewegung zweier Teilchen im attraktiven Coulombpotential oder im Gravitationspotential - klassische Näherung ohne Strahlung.

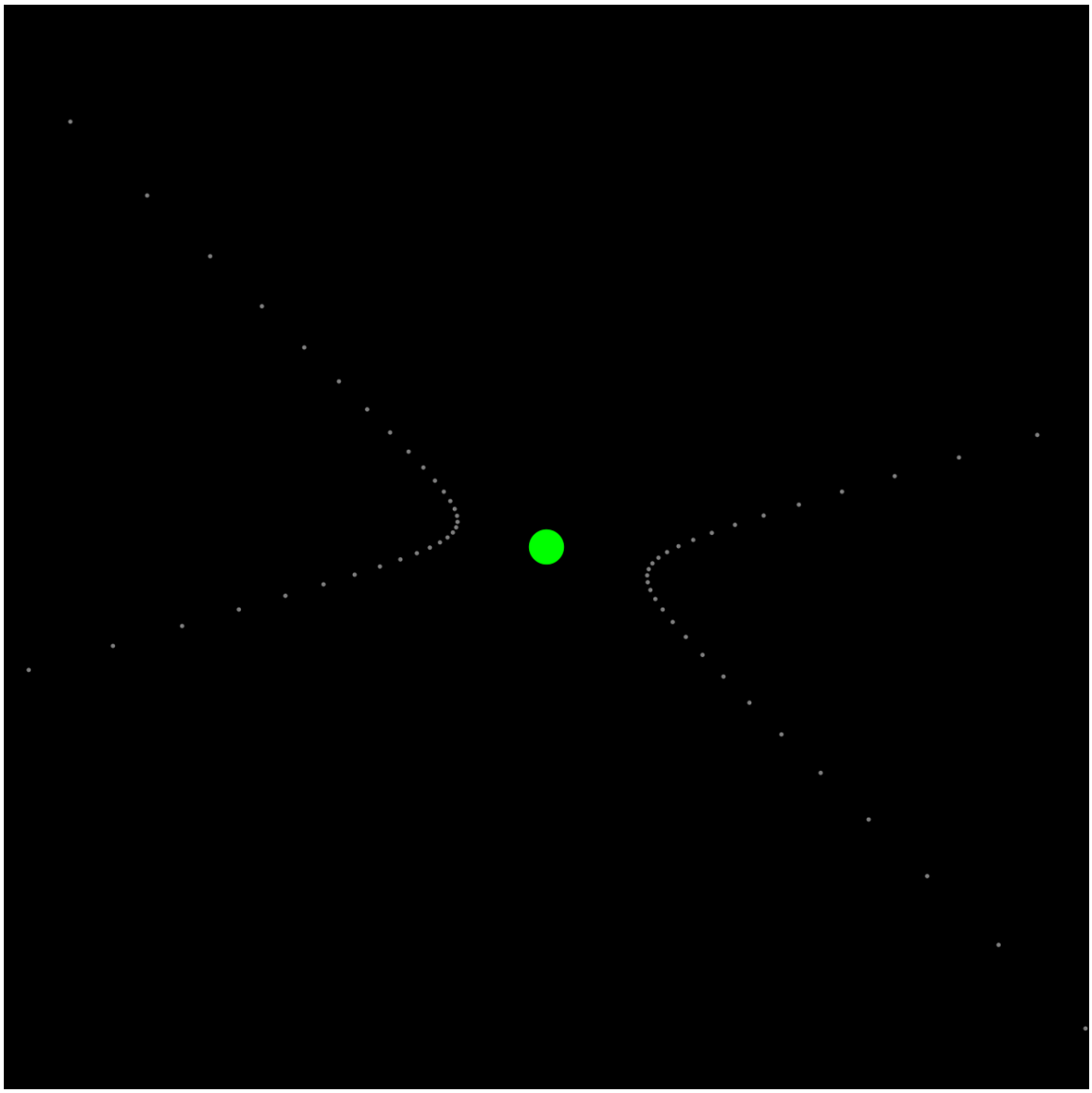


Abb. 267 Streuung im repulsiven Potential

Streuung im repulsiven Potential[56]

Bewegung zweier Teilchen im repulsiven Coulombpotential - klassische Näherung ohne Strahlung.

Bei anderen Potentialen gibt es oft keine analytische Lösung und solche Probleme werden dann oft numerisch gelöst, sind dann aber für die Animation auch nur für eine endliche Zeit berechenbar. Je nach Art der Rechnung kann das Ergebnis auch wieder eine explizite Zeitabhängigkeit des Ortes sein, so dass die Verwendung von *keyTimes* nicht notwendig ist. *keyTimes* sind also insbesondere dann

56 `http://de.wikibooks.org/wiki/media%3ASVG_Streuung02.svg`

hilfreich, wenn die Orte der Bewegung zu bestimmten Zeitpunkten bekannt sind, aber die Bewegung nicht gleichmäßig ist. Dies kann etwa auch hilfreich sein, wenn sich ein Objekt über längere Zeit gleichmäßig bewegt, also nur zwei Punkte in der Ebene notwendig sind, um die Bewegung zu beschreiben, sich dann aber in kurzer Zeit die Bewegungsrichtung ändert, wo also viele Punkte benötigt werden, um die Bewegung gut zu beschreiben und sich auch der Betrag der Geschwindigkeit gegebenenfalls drastisch ändert. Dies ist bei den meisten Streuproblemen der Fall, zum Beispiel auch bei Stößen von Atomen oder Molekülen, die sich meist mit konstanter Geschwindigkeit bewegen und nur wenn sie sich sehr nahekommen eine Ablenkung, also eine Änderung der Geschwindigkeit, stattfindet. Die Atome und Moleküle sind elektrisch neutral und haben daher nur kurzreichweitige Potentiale, anders als bei obigen Beispielen mit dem Colombpotential, welche hilfreich sein können, wenn Stöße von Ionen oder Elektronen (klassisch) betrachtet werden.

14.12 Wahl der Interpolationsmethode

Mit dem Attribut *calcMode* wird die Methode festgelegt, mit der zwischen zwei aufeinanderfolgenden Werte in der *values*-Werteliste interpoliert wird.

Ist für das Animationsziel keine kontinuierliche Interpolation möglich, so wird *calcMode* ignoriert und eine diskrete Animation durchgeführt. Zum einen wird dies angewendet, wenn die Werte des Animationszieles nie interpolierbar sind, wie etwa für *stroke-linejoin*, *stroke-linecap*, *fill-rule*, *font-family* etc. Das sind die einfachen Fälle für ein Darstellungsprogramme, die problemlos erkannt werden können.

Dann gibt es kompliziertere Fälle, wo die Interpolierbarkeit davon abhängt, welche Werte in *values* notiert sind. Dies tritt vor allem auf bei den etwas überfrachteten Eigenschaften *fill* und *stroke*, weil diese nicht nur interpolierbare Farbwerte haben können, sondern auch Maldienste referenzieren und den Wert 'none' annehmen können.

Enthält die *values*-Liste einen solchen nicht interpolierbaren Wert, so ist *calcMode* ebenfalls zugunsten einer diskreten Animation zu ignorieren. In der Praxis ist allerdings bei einigen Darstellungsprogrammen ein davon abweichendes fehlerhaftes Verhalten zu beobachten, wo zum Beispiel nicht interpolierbare Werte als schwarz (#000) interpretiert werden oder mitten in der Animation mit undefiniertem Zeitablauf nur zeitweilig zu einer diskreten Animation gewechselt wird. Ein solches Verhalten ist vom Autor nicht angebbar und ist daher als fehlerhaft einzustufen.

Autoren, die Wert auf eine definierte und einheitliche Präsentation legen, sollten also für eine Animation von *fill* und *stroke* das Attribut *calcMode* unbedingt explizit angeben, wenn die *values*-Liste nicht interpolierbare Werte enthält.

Ist bei ***animateMotion*** *calcMode* nicht angegeben, so erfolgt eine gleichmäßige Animation (Wert 'paced'), bei der der Betrag der Geschwindigkeit der Änderung konstant ist. Bei einer einzelnen Animation entspricht das auch einer Bewegung mit konstanter Geschwindigkeit. Werden mehrere Bewegungen auffaddiert, muss das nicht mehr der Fall sein.

Ist ansonsten das Attribut *calcMode* nicht angegeben, wird der Wert 'linear' impliziert. Ansonsten richtet sich der Zeitablauf nach den Angaben in *keyTimes*.

Mögliche Werte von *calcMode* sind:

discrete

Diskrete Animation; die Animationsfunktion springt von einem zum nächsten Wert der *values*-Liste ohne Interpolation.

linear

Lineare oder besser affine Interpolation; die Geschwindigkeit der Änderung für ein Intervall zwischen zwei aufeinanderfolgenden Werten der *values*-Liste ist konstant. Ansonsten richtet sich der Zeitablauf nach den Angaben in *keyTimes*.

paced

Gleichmäßige Animation; über die gesamte einfache Animationsdauer ist der Betrag der Geschwindigkeit der Änderung konstant.

spline

Interpolation durch eine kubische Bézier-Kurve; die Geschwindigkeit der Änderung für ein Intervall zwischen zwei aufeinanderfolgenden Werten der *values*-Liste wird bestimmt durch eine Funktion, welche entsprechend im Attribut *keySplines* anzugeben ist. Ansonsten richtet sich der Zeitablauf nach den Angaben in *keyTimes*.

14.12.1 Diskrete Animation

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	- (nicht testbar)
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Bei einer diskreten Animation gemäß *calcMode* 'discrete' ändert sich die einfache Animationsfunktion diskontinuierlich, sofern zwei aufeinanderfolgende Werte in der *values*-Liste unterschiedlich sind. Die Sprünge von einem Wert zum nächsten erfolgen gemäß der Angaben in *keyTimes*.

Anwendungsbeispiele:

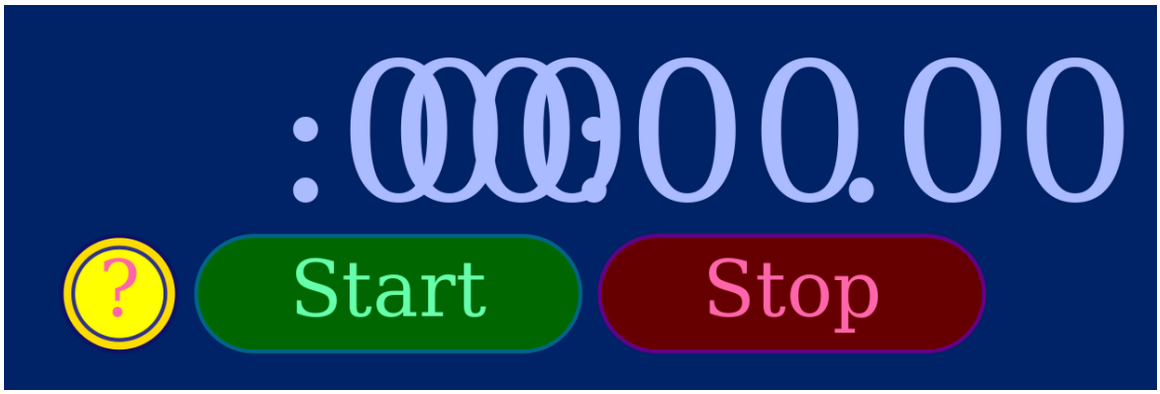


Abb. 268 Digitale Stopuhr

Digitale Stopuhr[57]

Einfacher Prototyp einer digitalen Stopuhr. Bei diesem Beispiel ist nicht intensiv getestet worden, wie genau die Stopuhr ist, eignet sich also nicht für ernsthafte Anwendungen ohne weitere Tests und Genauigkeitsabschätzungen.

Start und Stop kann mit verschiedenen Ereignissen erreicht werden: Klicken oder Aktivieren des entsprechenden Knopfes, beziehungsweise Verwendung der Taste b für Beginn/Start und e für Ende/Stop. Ebenfalls möglich ist die Verwendung des gelben Knopfes für beides: drücken für Start, loslassen für Stop.

Abb. 269 Digitaluhr

Ganz ähnlich funktioniert folgende Digialuhr:

Digitaluhr[58]

Eine einfache digitale Uhr wird animiert. Die Uhr wird mittels des wallclock-Wertes des begin-Attributes gestartet. Startzeitpunkt ist wallclock(00:00), was bedeutet, dass die Animation um Mitternacht nach lokaler Zeit beginnt.

57 http://de.wikibooks.org/wiki/media%3ASVG_Stopuhr01.svg
58 http://de.wikibooks.org/wiki/media%3ASVG_Digitaluhr01.svg

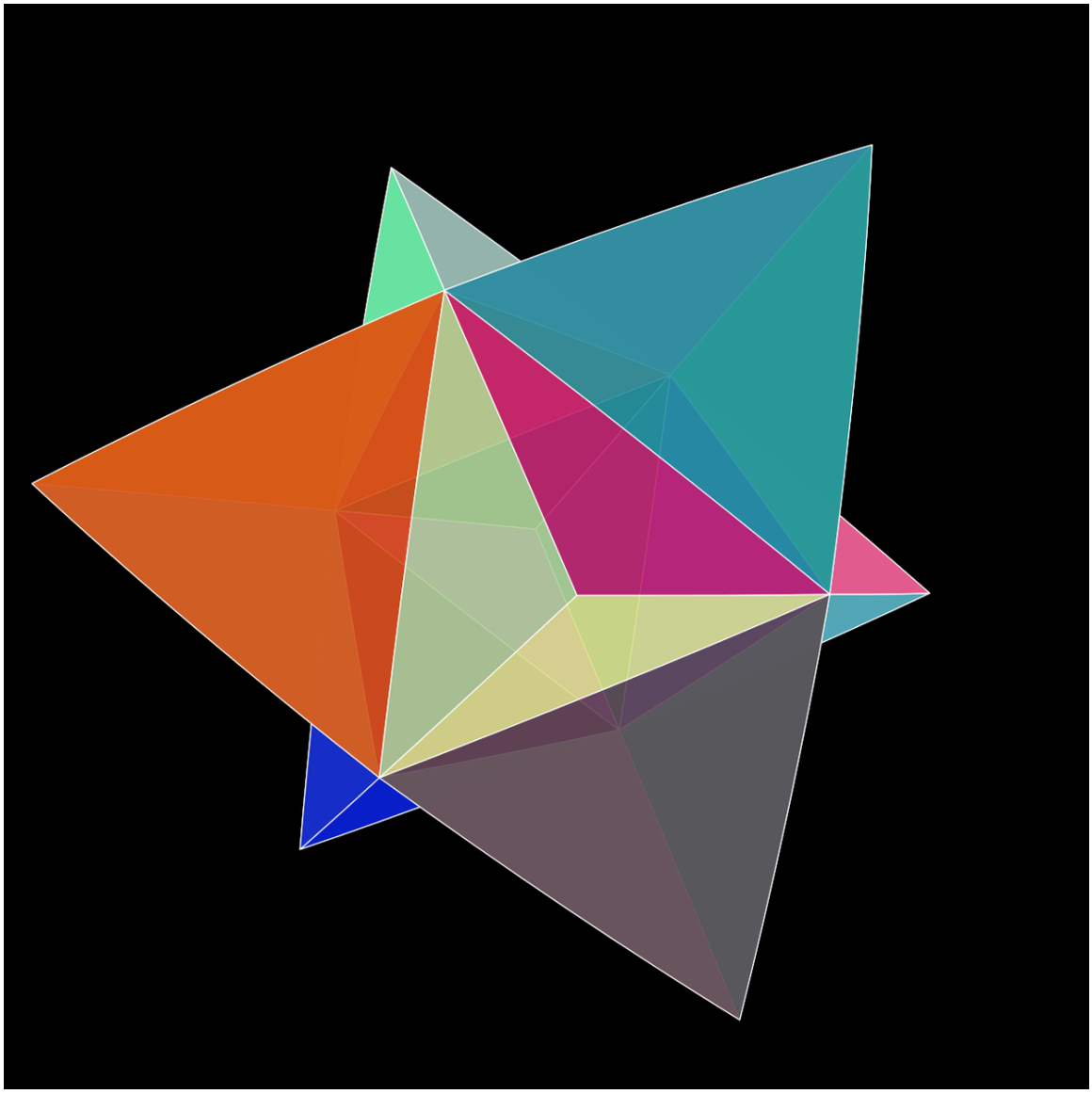


Abb. 270 Sterntetraeder

Ähnlich wird bei folgendem Beispiel die Reihenfolge der dargestellten Flächen diskret geändert, um die Illusion eines Körpers hervorzurufen:

Sterntetraeder[59]

Drehung eines Sterntetraeders um den Mittelpunkt des Bildes.

59 http://de.wikibooks.org/wiki/media%3ASVGzeichenreihenfolge06.svg

14.12.2 Lineare oder affine Animation

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	4
Konqueror (KSVG)	- (nicht testbar)
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Bei *calcMode* 'linear' ist die Interpolationsfunktion zwischen zwei aufeinanderfolgenden Werten erst a und dann b in der *values*-Liste i(u) = b * u + a * (1 - u) mit u als Parameter von 0 bis 1 für die relative Zeit zwischen den beiden korrespondierenden Werten in *keyTimes*.

15.1 Verweise

SVG stellt mit dem Element ***a*** und Attributen aus dem Namensraum von XLink[1] auch die Funktionalität von Verweisen (englisch: hyperlink) zur Verfügung. Dies kann sowohl dazu verwendet werden, Animationen zu starten als auch andere Dokumente aufzurufen oder einen anderen Ausschnitt oder allgemeiner eine transformierte Darstellung desselben Dokumentes zu betrachten.

Ferner sieht SVG vor, dass das Darstellungsprogramm Funktionalitäten bereitstellt, mit denen der Betrachter die Art der Anzeige beeinflussen kann. Die Anzeige kann vergrößert oder verkleinert werden und bei einem Ausschnitt kann dieser verschoben werden. Eine Drehung der Darstellung kann ebenfalls möglich sein.

15.1.1 URI, IRI

Im internet werden Ressourcen mit vereinheitlichen Identifizierern verfügbar gemacht, auch bekannt als URI (englisch: Unified Resource Identifier) oder IRI (englisch: Internationalized Resource Identifier). Im folgenden wird vereinfachend die Bezeichnung IRI für alle Möglichkeiten verwendet. Zum Beispiel etwa:

```
http://example.com/Bild.svg
```

IRIs sind etwas allgemeiner als URIs, IRIs werden allgemein aus Zeichen auch dem UNICODE-Bereich konstruiert, während die erlaubten Zeichen in URIs weiter eingeschränkt sind. Aus Autorensicht sind URIs also eher die sicherere Wahl, weil sie weniger Anforderungen an das Darstellungsprogramm bei der Verarbeitung stellen und auch von älteren Programmen korrekt interpretiert werden. In SVG 1.1 sind auch nur URIs vorgesehen, in SVG tiny 1.2 auch IRIs.

15.1.2 Externe Ressourcen referenzieren

```
http://example.com/Bild.svg
```

1 `http://www.w3.org/TR/xlink/`

ist eine absolute Angabe einer IRI. Die besteht aus einer Angabe des Protokolles, hier 'http', gefolgt von Angaben zur Identifikation der Ressource. Die Angaben können vom Protokoll abhängen. Für 'http' folgt hier nach '://' eine Domäne (englisch: domain) 'example.com', vereinfacht der Name oder auch die internet-Protokoll-Adresse des Rechners, auf dem die Datei zu finden ist, dann mit einem Schrägstrich abgeteilt eine Pfadangabe zur Datei. Hier ist dies einfach '/Bild.svg'. Liegt die Datei in einem Unterverzeichnis 'svg', wäre die Pfadangabe '/svg/Bild.svg'.

Die Pfadangabe entspricht in der Regel nicht dem absoluten Pfad auf dem referenzierten Rechner. Dort ist in der Regel in der Konfiguration des Programmes, welches Anfragen zum Protokoll 'http' beantwortet ein geeignetes Wurzelverzeichnis angegeben, auf welches sich die Pfadangabe bezieht.

Relative Angaben sind ebenfalls möglich, optional auch in Kombination mit *xml:base*. Das Darstellungsprogramm setzt relative Angaben dann entsprechend zu einer IRI zusammen. Neben der kürzeren Adresse hat dies auch den Vorteil, dass es einfacher ist, eine Gruppe zusammengehöriger Dokumente auf einen anderen Rechner zu verschieben, ohne die Angaben zur IRI in den Dateien ändern zu müssen.

Bei der Verwendung von *xml:base* gibt dies Attribut eine IRI an, mit welcher eine relative Angabe vervollständigt wird, jeweils durch Anhängen der relativen Angabe an den Attributwert. Also

```
http://example.com/Bild.svg
```

entspricht mit

```
xml:base="http://example.com/"
```

der Angabe

```
Bild.svg
```

Im Weiteren sei angenommen, dass *xml:base* nicht gesetzt ist.

Liegt das Dokument Bild.svg im Hauptverzeichnis des servers example.com, so ist vorheriges Beispiel äquivalent zu:

```
/Bild.svg
```

Liegen referenzierendes und referenziertes Dokument in einem Verzeichnis, so reicht einfach die Angabe des Dateinamens:

```
Bild.svg
```

oder

```
./Bild.svg
```

'.' steht immer für das aktuelle Verzeichnis.

'..' steht für das übergeordnete Verzeichnis.

Angenommen in einem Dokument erreichbar unter

```
http://example.com/Verzeichnis/Beispiele/Bild.svg
```

soll ein Dokument A.svg im Unterverzeichnis 'SVGs' des Verzeichnisses 'Beispiele' referenziert werden, so kann zum Beispiel angegeben werden:

```
http://example.com/Verzeichnis/Beispiele/SVGs/A.svg
```

```
SVGs/A.svg
```

```
./SVGs/A.svg
```

```
/Verzeichnis/Beispiele/SVGs/A.svg
```

Soll hingegen ein Dokument B.svg in dem Verzeichnis 'Verzeichnis' referenziert werden, so kann angegeben werden:

```
http://example.com/Verzeichnis/B.svg
```

```
../B.svg
```

```
/Verzeichnis/B.svg
```

15.1.3 Interne Fragmente referenzieren

Ferner ist es möglich, auf ein Element im selben Dokument zu verweisen, wofür Fragmentidentifizierer verwendet werden. Diese bestehen aus einem # gefolgt von einem Wert eines Attributes *id* oder *xml:id* im selben Dokument, zum Beispiel:

```
#Beispiel
```

15.1.4 Fragmente in externen Ressourcen referenzieren

Beide Methoden können auch kombiniert werden, um ein Element in einem anderen Dokument zu identifizieren, zum Beispiel:

```
http://example.com/Bild.svg#Beispiel
```

oder

```
SVGs/A.svg#Beispiel
```

Im engeren Sinne gehört das # und alles dahinter nicht mehr zur IRI selbst. In dieser Konstruktion sind also weder Sonderzeichen zu maskieren, noch gibt das Darstellungsprogramm diesen Teil überhaupt bei einem Aufruf weiter. Bei obigen Beispielen wird also nur der Teil vor dem # für eine Anfrage an einen server oder ein Dateisystem verwendet, der Teil dahinter wird vom Darstellungsprogramm zur Auffindung des angegebenen Fragmentes in der aufgerufenen Datei verwendet.

15.1.5 Methoden der Referenzierung

In SVG tauchen zwei verschiedene Methoden auf, um IRIs zu notieren.

Zum einen erfolgt dies direkt als Wert des Attributes *href* aus dem Namensraum XLink, zum Beispiel, sofern das Präfix des Namensraumes zuvor als 'xlink' festgelegt wurde:

```
xlink:href="http://example.com/Bild.svg"
```

Die andere Methode ist die sogenannte funktionale IRI, die aus CSS stammt und bei Eigenschaftswerten Anwendung findet. Der Wert beginnt dann mit 'url(' und endet mit ')'. Dazwischen steht die IRI. Zum Beispiel so:

```
fill="url(#Farbverlauf1)"
```

15.2 Möglichkeiten und Einschränkungen

Je nach Attribut gibt es Einschränkungen, welche Typen von IRIs notiert werden dürfen. Die jeweilige Definition des referenzierenden Elementes kann auch eine Angabe dazu enthalten, welche Formate die Dokumente haben dürfen, auf welche verwiesen wird. Teils ist auch angegeben, ob nur komplette Dateien oder nur Fragmente referenziert werden dürfen. Animationselemente dürfen etwa nur Fragmentidentifizierer enthalten, verweisen also immer auf Elemente im selben Dokument.

Mit Ausnahme von Bildern (Element ***image***) und Verweisen (Element ***a***) sind in SVG tiny 1.1 alle Referenzierungen lokal, die Ressourcen sind also im selben Dokument zu finden. Generell können Elemente wie ***image***, ***audio***, ***video***, ***animation*** und ***foreignObject*** nur komplette externe Dokumente referenzieren.

In SVG tiny 1.2 wird dies Prinzip der Lokalität etwas gelockert. ***use*** darf wie auch in SVG 1.1 externe Fragmente referenzieren, allerdings keine anderen ***svg***-Elemente und keine Fragmente mit Skripten, Verweise auf Animationen und auch keine weiteren ***use*** oder ***animation***-Elemente, welche externe Referenzen haben. ***prefetch*** und ***font-face-uri*** dürfen sowohl interne als auch externe Ressourcen referenzieren.

Eigenschaften wie *fill* oder *stroke* dürfen in SVG tiny 1.2 keine externen Maldienste referenzieren, das Element ***mpath*** auch keine externen Pfade. Navigationsattribute sind in SVG tiny 1.2 auch nur lokal gemeint.

Eine direkte oder indirekte Selbstreferenzierung, die also zu einer rekursiven Schleife führen täte, ist ein Fehler. Bei komplizierteren Schleifen kann allerdings der Abbruch der Interpretation abhängig vom Darstellungsprogramm sein, je nachdem, wann diesem die Schleife auffällt.

15.2.1 Pseudopotokoll 'data'

Das Pseudoprotokoll 'data' kann bei einigen Elementen verwendet werden, um Inhalte direkt in den Verweis einzubetten. Sinngemäß sieht das wie folgt aus:

```
xlink:href="-Dokumentdaten-"
```

Anwendbar ist das Pseudoprotokoll bei ***a***, ***image***, ***audio***, ***video***, ***animation***, ***foreignObject***, ***script***, ***handler***, ***font-face-uri***.

Die IRI beginnt mit der Angabe des Protokolles 'data', gefolgt von einem Doppelpunkt ':'. Dann folgt die Angabe des Inhaltstyps (ehemals MIME-Typ) des einzubettenden Dokumentes, zum Beispiel 'image/svg+xml' für ein SVG-Dokument, 'image/png' für ein PNG-Dokument, 'image/jpeg' für ein JPEG-Dokument, 'application/xhtml+xml' für ein XHTML-Dokument etc. Dann folgt ein Semikolon ';'.

Bei Textdateien kann dann eine Angabe zur Kodierung des Textes erfolgen, etwa 'charset=iso-8859-1', gefolgt von einem Komma ',' und dann den Daten des Dokumentes. Beispiel:

```
xlink:href="data:image/svg+xml;charset=iso-8859-1,-Dokumentdaten-"
```

oder

```
xlink:href="data:application/xhtml+xml;charset=utf-8,-Dokumentdaten-"
```

Da in einer IRI einige typisch in Dokumenten auftauchende Zeichen nicht einfach so auftreten dürfen, sind diese entsprechend zu maskieren (englisch: URL encode). Ein Konflikt tritt zum Beispiel bereits

auf, wenn in den Dokumentdaten die gleichen Anführungszeichen auftreten wie jene, mit denen der Attributwert umschlossen wird. Auch Leerzeichen sind in IRIs problematische Zeichen.

Mit der base64 gibt es eine weitere effektive Methode, um eine geeignete Maskierung vorzunehmen. In dem Falle wird hinter der Angabe des Inhaltstyps und des Semikolons 'base64' angegeben, gefolgt von einem Komma und den Dokumentdaten.

15.2.2 Exkurs: Hinweise zur Kodierung für das Pseudoprotokoll 'data'

Abb. 272 SVG mit per data eingebettetem PNG mit base64-Kodierung[a]

a `http://de.wikibooks.org/wiki/media%3ASvgTutorialEmbedbitmap.svg`

Serverseitige Skriptsprachen wie PHP enthalten geeignete Funktionen sowohl für base64 als auch für 'URL encode'.

Steht etwa in der PHP-Variablen $Beispiel ein mit base64 zu kodierendes Dokument, so kann verwendet werden:

```
$ausgabe=base64_encode($Beispiel);
```

Oder aber sofern eine URL-Kodierung ausreicht:

```
$ausgabe=urlencode($Beispiel);
```

Mittels perl kann eine Datei 'Beispiel.svg' wie folgt mit base64 kodiert werden, wobei dann die Ausgabe 'Beispiel.svg.base64' ist:

```
perl -MMIME::Base64 -0777 -ne "print encode_base64($_)" <Beispiel.svg
 >Beispiel.svg.base64
```

Das Ergebnis Beispiel.svg.base64 kann dann in die direkt in die IRI eingefügt werden.

Eine weitere Möglichkeit bei Unix/Linux-Betriebssystemen besteht darin, einfach in der Konsole einzugeben:

```
base64 Beispiel.svg >Beispiel.svg.base64
```

Eine weitere einfache Möglichkeit besteht darin, dass email-Programme ebenfalls die base64-Kodierung beherrschen. Es reicht also, sich selbst ein Dokument zuzuschicken und vom email-Programm zuvor solch eine Kodierung vornehmen zu lassen und dann das Ergebnis aus dem Quelltext der email zu kopieren.

Anhand des Quelltextes des Beispieles rechts neben diesem Abschnitt wird deutlich, warum das Einbetten von Pixelgraphik in SVG nur von eingeschränktem Nutzen ist. Einmal abgesehen davon, dass Pixelgraphik meist ein Problem aufwirft, wenn die Graphik skaliert wird, wird auch der Quelltext recht unübersichtlich und die Dateigröße schnell recht umfangreich, auch weil eine mit base64 kodierte Datei im Schnitt etwas größer ist als die unkodierte Datei.

15.3 Attribute von XLink

Für jene Elemente, welche eine Funktionalität mit XLink verfügbar machen, sind eine Reihe von Attributen aus dem Namensraum XLink verwendbar. Teilweise sind die Attributwerte in SVG festgelegt, eine explizite Angabe ist daher in bei einigen Attributen nicht notwendig oder gar komplett überflüssig. Um die Funktionalität von XLink zu gewährleisten, sind diese Attribute aber explizit in SVG, gegebenenfalls samt Wertvorgaben festgelegt.

15.3.1 Attribut *type*

Das Attribut *type* legt den Typ des Verweises fest. SVG verwendet nur den einfachen Typ von Verweisen im Sinne von XLink, insofern ist auch nur die einfache Funktionalität von XLink verfügbar. Der Wert von *type* ist folglich festgelegt auf 'simple'. Eine explizite Angabe ist nicht notwendig.

Das Attribut ist nicht animierbar.

15.3.2 Attribut *role*

Der Wert des Attributes ist eine IRI zu einem Dokument, welches die Rolle oder Funktion des Dokumentes oder Fragmentes beschreibt, auf welches verwiesen wird. Die Angabe ist optional. Hinsichtlich der Maschinenlesbarkeit siehe auch die Anmerkung zu Attribut *arcrole*.

Das Attribut ist nicht animierbar.

15.3.3 Attribut *arcrole*

Der Wert des Attributes ist eine IRI zu einem Dokument, welches die Rolle oder Funktion des Verweises zum Dokument oder Fragment beschreibt, auf welches verwiesen wird. Die Angabe ist optional.

Das Attribut ist nicht animierbar.

Die Funktion des Verweises ist in SVG vielfach durch die Definition des Elementes impliziert, in welchem das Attribut steht. Bei Animationselementen hat der Verweis offenbar die Funktion, das Animationsziel zu identifizieren. Dies muss nicht explizit angegeben werden. Es mag allerdings Fälle geben, wo Details relevant sind, etwa die Unterscheidung bei einem Verweis mit dem Element ***a***, welche eine Relation (Attribut *rel*) zum Ausdruck bringen soll - Werte wie Person, Mutter, Tochter hängen davon ab, was auf wen zu welchem Zwecke verweist. In solchen Fällen kann sich eine entsprechende Angabe der IRI eines definierenden Dokumentes lohnen. Die IRI einer bekannten Definition kann selbst ähnlich wie IRIs bei RDFa als maschinenlesbare Definition aufgefasst werden. Sofern die Definition im per *arcrole* (oder auch *role*) angegebenen Dokument also den interpretierenden Programm bekannt ist, kann dies gegebenenfalls auch Funktionalitäten und Hilfen bereitstellen, um mit der definierten Rolle umzugehen. Es kann sich aber auch nur um eine Definition der Bedeutung im semantischen Sinne handeln.

15.3.4 Attribut *title*

Das Attribut gibt dem Verweis einen Titel, ähnlich dem gleichnamigen Attribut in (X)HTML. Der Wert des Attributes ist eine Zeichenkette, die den Titel oder auch eine Nutzerhilfe zum Verweis repräsentiert. Titel und Nutzerhilfe beziehen sich also auf den Verweis selbst, nicht direkt auf das Verweisziel. Die Angabe ist optional.

Das Attribut ist nicht animierbar.

15.3.5 Attribut *show*

Das Attribut gibt an, wie das Verweisziel angezeigt werden soll. Bei verschiedenen Elementen ist der Wert impliziert, zum Beispiel wird das Verweisziel mit ***image***, ***audio***, ***video***, ***animation*** und ***foreignObject*** eingebunden. Bei Animationselementen liegt eine andere, in XLink nicht diskutierte Funktionalität vor, welche in SVG definiert ist. Entsprechend kann ein Darstellungsprogramm eine explizite Angabe ignorieren, falls diese im gegebenen Zusammenhang keinen Sinn ergibt. Sofern sie Sinn ergibt, soll die Angabe aber beachtet werden. Die Angabe ist optional.

Folgende Werte sind gemäß XLink möglich:

new

Das Verweisziel soll in einem neuen Anzeigebereich präsentiert werden

replace

Das Verweisziel soll im gerade aktuellen Anzeigebereich präsentiert werden (bei verschachtelten Dokumenten etwa XHTML in ***foreignObject*** in SVG in XHTML (frameset) etc ist das nicht eindeutig, welcher davon aktuell ist).

embed

Das Verweisziel soll in einem eigenen Anzeigebereich in das aktuelle Dokument eingebettet werden, wie etwa beim Element ***image***

other

Die Funktion des Elementes bestimmt die Art der Anzeige, dies ist bei SVG das typische Verhalten.

none

Keine Angabe oder direkte Anzeige des Verweiszieles.

In SVG 1.1 sind als Möglichkeiten nur 'new' und 'replace' angegeben, bezogen auf die Verwendung mit dem Element ***a***.

Das Attribut ist nicht animierbar.

15.3.6 Attribut *actuate*

Das Attribut legt den gewünschten Zeitablauf fest, in dem das Verweisziel darzustellen ist. Effektiv wird dies bei SVG impliziert. Formal wird der Wert 'onLoad' für Element wie ***image*** oder ***foreignObject*** impliziert, bei denen der Inhalt direkt angezeigt wird. Bei Elementen wie ***audio***, ***video*** oder ***animation*** trifft 'other' zu, weil das Verhalten in SMIL und SVG tiny festgelegt ist.

Das Wert für ***a*** ist 'onRequest'.

Das Attribut ist nicht animierbar.

15.3.7 Attribut *href*

Mit *href* wird die IRI des Verweiszieles, der Resource angegeben.

Das Attribut ist zumeist animierbar. Sofern das nicht zutrifft, wie etwa bei ***mpath***, ist dies bei der Elementdefinition vermerkt.

15.4 Element *a*

Squiggle (Batik)	1.7
Opera (Presto)	8
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Entsprechend dem Element ***a*** in (X)HTML definiert SVG das Element ***a*** für Verweise auf Dokumentfragmente und insbesondere auf andere Dokumente. Um die Funktionalität eines Verweises bereitzustellen, werden Attribute von XLink verwendet. Die von historischen Varianten von HTML bekannte Funktionalität eines Ankers stellt das Element ***a*** nicht speziell bereit, das wird wie auch in moderneren Varianten von (X)HTML mit einem Fragmentidentifizierer und dem Attribut *id* beziehungsweise *xml:id* bei jedem Element bereitgestellt.

Ansonsten ist das Element ***a*** in SVG ein Gruppierungselement wie das Element ***g*** auch. Die darin darzustellenden Elemente werden dazu verwendet, um den Verweis auszulösen (zum Beispiel durch Anklickern mit einem Zeigergerät).

In SVG 1.1 ist es (anders als in (X)HTML) erlaubt, ***a***-Elemente in anderen ***a***-Elementen zu verschachteln. Gewinnen tut dann gemäß Malermodell jenes Element, welches am weitesten innen in der Verschachtelung liegt. In SVG tiny 1.2 kann das Element ***a*** alle Elemente enthalten, die auch sein Elternelement enthalten darf, mit Ausnahme des Elementes ***a*** selbst. Eine Verschachtelung ist also nicht korrekt und sollte allerdings auch bei SVG 1.1 verbleiben, um Komplikationen zu vermeiden.

Es gibt durch das Darstellungsprogramm meist keine besonderen Hinweise darauf, dass ein Verweis verfügbar ist. Insofern sollte der Autor durch Dekoration und Beschreibung selbst dafür sorgen, dass dem Nutzer klar ist, wo Verweise zu finden sind (etwa mit einem dünnen blauen Strich um die betroffenen Elemente und einer eindeutigen Beschreibung in der Textalternative).

SVG tiny 1.2 legt genauer fest, dass Darstellungsprogramme erst auf eine Verweisfunktionalität mit einem visuellen Indikator hinweisen sollen, wenn das entsprechende Element im Bereich etwas eines Zeigergerätes ist oder im Fokus ist. Oft ändert sich etwa die Darstellung des Zeigers zu einer zeigenden Hand über einem Verweis.

Sofern CSS verwendet wird, ergibt sich natürlich auch mit den Pseudoklassen :link und :visited eine Möglichkeit für Autoren, auf die Erscheinung der Indikation eines Verweises Einfluss zu nehmen.

15.4.1 Attribute von XLink

Für das Element ***a*** sind die beschriebenen Attribute von XLink verfügbar. Die für ***a*** besonders relevanten Attribute von XLink sind *href*, *role*, *title* und *show*.

Mit *href* wird das Verweisziel angegeben. Der Wert ist eine IRI. *href* ist animierbar.

Mit *title* kann ein Titel oder eine Nutzerhilfe für den Verweis als einfacher Text als Attributwert angegeben werden.

Mit *role* kann ein Verweis zu einem Dokument bereitgestellt werden, welches die Rolle oder Funktion des Verweiszieles beschreibt. Dies kann insbesondere relevant sein, wenn das Verweisziel in einem Format ohne Textalternative ist. Eine weitere Relevanz ergibt sich, wenn die Funktion des Verweiszieles eine andere ist, als der Inhalt des Verweiszieles selbst vermuten lässt.

Mit *show* mit den möglichen Werten 'new' oder 'replace' kann angegeben werden, ob das Verweisziel in einem neuen Anzeigebereich dargestellt werden soll ('new') oder ob das verweisende Dokument ersetzt werden soll ('replace'). Da bei komplizierteren Verschachtelungen von Dokumente mehr als diese Möglichkeiten gibt, bietet das Attribut *target* eine bessere Alternative.

15.4.2 Attribut *target*

Das Attribut *target* gibt an, in welchem Anzeigebereich der referenzierte Inhalt angezeigt werden soll. Sofern sowohl *show* von XLink als auch *target* angegeben sind, hat *target* Vorrang. Das Element ist analog zum gleichnamigen von (X)HTML definiert.

Mögliche Werte sind zum einen die in (X)HTML reservierten Schlüsselwörter, zum andere Namen eines Rahmens, (englisch: frame) im Sinne von (X)HTML oder einer ähnlichen Konstruktion. Die reservierten Schlüsselwörter sind:

_replace

Das aktuelle SVG-Bild wird durch den referenzierten Inhalt im selben rechteckigen Bereich desselben Rahmens ersetzt

_self

Das aktuelle SVG-Bild wird durch den referenzierten Inhalt im selben Rahmen ersetzt. Es ersetzt also den gesamten Inhalt des Rahmens, nicht unbedingt nur das SVG-Bild (Voreinstellung)

_parent

Der direkte Elternrahmen wird ersetzt. Sofern es den nicht gibt, ist das Verhalten das gleiche wie für _self

_top

Der komplette Fensterbereich oder Bereich des Reiters (englisch: tab) wird ersetzt, samt aller Rahmen

_blank

Der Inhalt wird in einem neuen Fenster oder Rahmen dargestellt, soweit dies technisch möglich ist, wenn nicht, erfolgt das Verhalten von _top

Der alternativ angebbare Name richtet sich nach den vorhandenen Namen von Rahmen und ähnlichen Konstruktionen. Diese beginnen in der (X)HTML Konvention mit einem Buchstaben. Jedenfalls ist das '_' als erstes Zeichen reserviert, '_new' ist also kein erlaubter Wert.

SVG selbst stellt keine Rahmen bereit. Im Sinne von SVG stellen aber ***object*** oder ***img*** von (X)HTML rechteckige Bereiche innerhalb eines Rahmens dar, wie für '_replace' anwendbar. Wird das SVG-Dokument also so in ein (X)HTML-Dokument eingebettet, so wird mit '_replace' nur das SVG-Dokument ersetzt, mit '_self' auch das (X)HTML-Dokument. Der Wert '_replace' ist neu in SVG tiny 1.2 und schließt eine Lücke der HTML4-Definition, bei der noch nicht davon ausgegangen wurde, dass Bilder auch Verweise enthalten können.

Das Attribut *target* ist animierbar.

15.4.3 Beispiele Element *a*

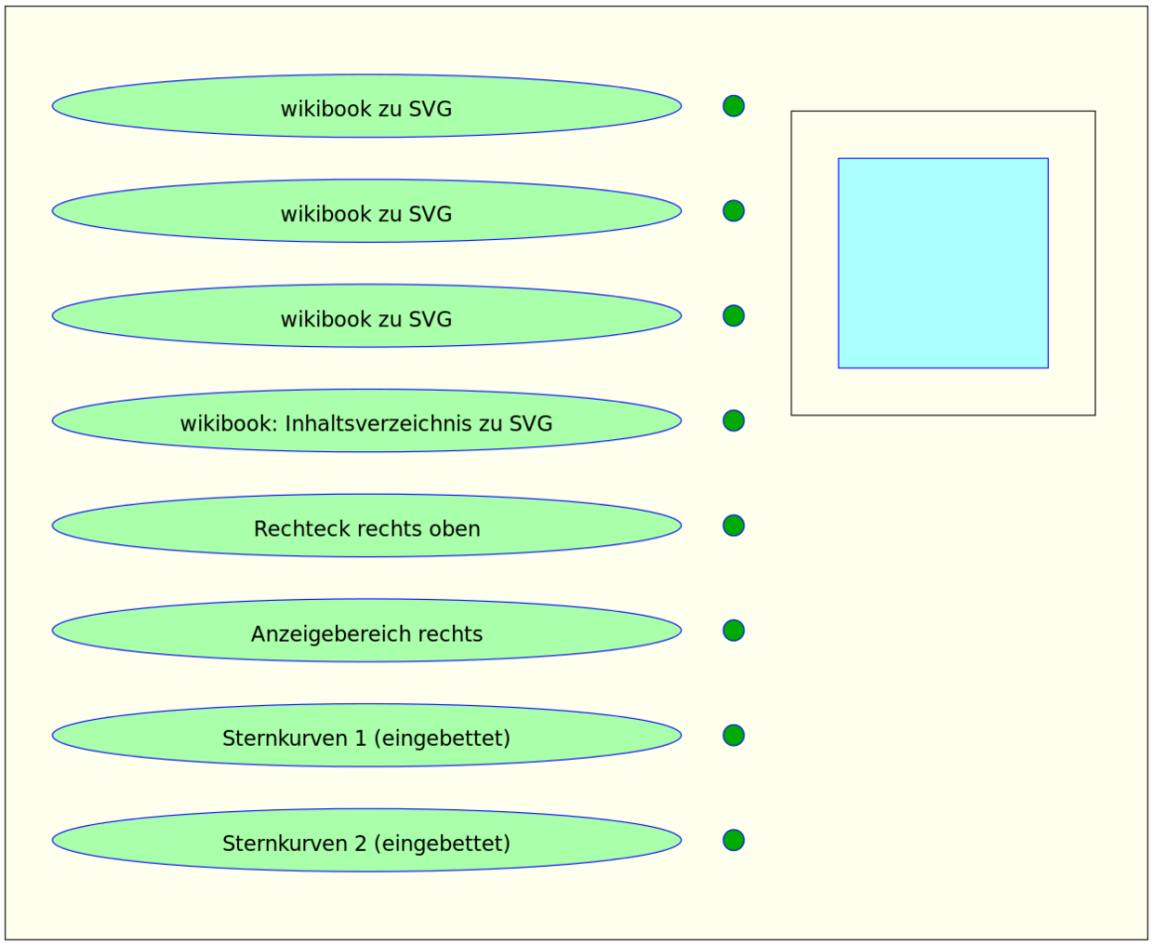


Abb. 273 Element *a*

Ein Verweis, welcher eine Ellipse, beschreibenden Text und einen Kreis umschließt, sie zum Beispiel wie folgt aus:

```
<a xlink:href="http://de.wikibooks.org/wiki/SVG#Inhalt"
   xlink:title="Verweis auf externes Fragment, neues Fenster mit target"
   target="_blank">
<ellipse cx="300" cy="400" rx="300" ry="30" fill="#afa" stroke="blue" />
<text x="300" y="410">wikibook: Inhaltsverzeichnis zu SVG</text>
<circle cx="650" cy="400" r="10" fill="#0a0" stroke="blue" />
</a>
```

Alle drei Kindelemente sind zum Aktivieren des Verweises geeignet, egal wo sie auf der Zeichenfläche positioniert sind Das geht, sofern sie nicht durch andere Elemente verdeckt werden, die ein Aktivieren oder Anklickern selbst registrieren. Auch wenn die Elemente nicht dargestellt sind, ist der Verweis nicht aktivierbar, wenn sie nur unsichtbar oder transparent sind, dann schon. Auch *pointer-events* kann die Aktivierbarkeit beeinflussen.

Element ***a***[2]

Einige Beispiele zur Verwendung von Verweisen auf externe Dokumente und Dokumentfragmente, solche, welche per Pseudoprotokoll data: eingebettet wurden und interne Fragmente.

Die blau umrandeten Objekte stellen jeweils eine Verweisfunktionalität bereit.

15.5 Verweise auf Inhalte in SVG-Dokumenten

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	9 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

In SVG ist es möglich, auf bestimmte Teile von SVG-Dokumenten zu verweisen.

Da der referenzierte Teil ein Bereich einer graphischen Darstellung ist, ist es erforderlich, einen gewissen Ausschnitt festzulegen, der dargestellt werden soll. Dies kann zum einen durch eine spezielle Konstruktion in der IRI erfolgen als in SVG 1.1 auch durch eine Vorgabe mit einem Element ***view***. Zudem kann ein einfacher Fragmentidentifizierer angegeben werden. Das entsprechende Element soll dann zentriert werden.

Die Darstellung in einem gewissen Ausschnitt erfolgt durch Angabe eines '#svgView', mit dem analog zur Angabe eines Fragmentidentifizierers ein darzustellender Bereich oder Ausschnitt angegeben werden kann.

Sei 'Bild.svg' das Dokument, von dem ein Ausschnitt dargestellt werden sollte. Der entsprechende Fragmentidentifizierer sieht wie folgt aus:

```
http://Bild.svg#svgView(*)
```

Dabei steht * für eine weitere Zeichenfolge. In SVG tiny 1.2 entspricht dies einer Liste mit dem Attribut transform mit einer Liste von Transformationen in Klammern. Die Wirkung entspricht der einer Transformation eines Elementes ***g***.

Zwei Listenelelemente der Attributliste werden jeweils mit einem Semikolon getrennt. Die im jeweiligen Attribut notierten Transformationen dürfen anders als in der Notation im Dokument keine (unmaskierten) Leerzeichen enthalten, entsprechend wird immer ein Komma als Trennzeichen verwendet.

Einfaches tiny 1.2 Beispiel also etwa:

```
http://Bild.svg#svgView(transform(scale(2),skewX(13),rotate(60,30,-70)))
```

Auch möglich:

```
ht
tp://Bild.svg#svgView(transform(scale(2));transform(skewX(13),rotate(60,30,-70)))
```

2 http://de.wikibooks.org/wiki/media%3ASVGa01.svg

Bei der Vollversion von SVG 1.1 sind weitere Angaben möglich. Neben *transform(...)* können dann auch die Parameter *viewBox(...)*, *preserveAspectRatio(...)*, *zoomAndPan(...)* und *viewTarget(...)* verwendet werden, mit entsprechenden Werten wie die korrespondierenden Attribute. Die Wirkung entspricht dabei der Anwendung derselben Attribute auf das Element ***view***.

Beispiel (mit einem Fragmentidentifizierer ID):

```
http://Bild
.svg#svgView(viewBox(-100,-200,500,600);preserveAspectRatio(none);viewTarget(ID))
```

Wird einfach ein Fragmentidentifizierer verwendet, so ergibt sich ein besonderes Verhalten, bei dem das identifizierte Element im Anzeigebereich zentriert wird. Paßt die umgebende Box nicht in den Anzeigebereich des nächstgelegenen Elementes ***svg***, wird eine entsprechende Skalierung vorgenommen. Ist die umgebende Box kleiner als der Anzeigebereich, wird nur zentriert und nicht skaliert.

Ist das Ziel des Verweises ein Animationselement oder allgemeiner ein zeitabhängiges Element, so löst dies eine Beginnzeit aus. Das ist näher beschrieben im Kapitel über Animation, im Abschnitt über den *begin*-Wert 'indefinite'.

In SVG 1.1 ist eine weitere Möglichkeit angegeben, einen Fragmentidentifizierer anzugeben. Dies ist eine Syntax gemäß XPOINTER. Da die Notation in SVG tiny 1.2 nicht mehr vorgesehen ist, empfiehlt es sich, darauf zu verzichten. Die Notation sieht wie folgt aus:

```
http://Bild.svg#xpointer(id('ID'))
```

und ist äquivalent zu:

```
http://Bild.svg#ID
```

15.5.1 Beispiel #svgView

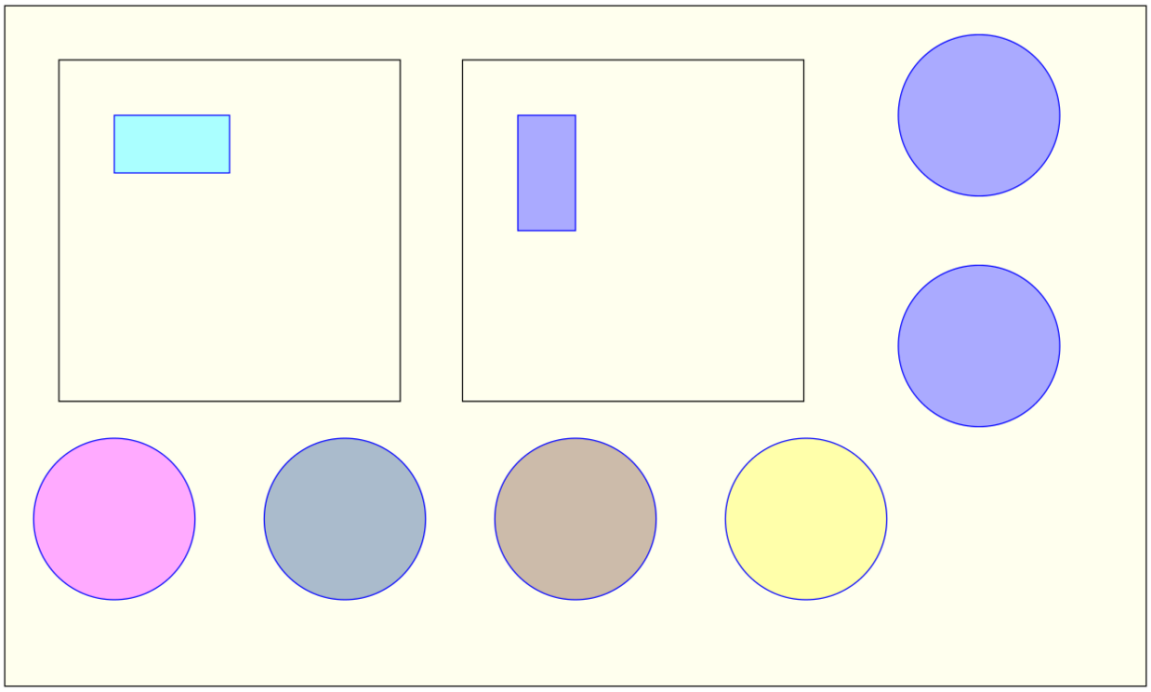


Abb. 274 Element ***a*** und #svgView

Element ***a*** und #svgView[3]

Einige Beispiele zur Verwendung von #svgView, um verschiedene Darstellungen des Dokumentes zu erreichen. Die Konstruktionen können in der URI auch variiert werden. Auch bei Dokumenten, die der Betrachter nicht selbst verändern kann, kann er die Darstellung ändern, indem er einen #svgView an die URI des Dokumentes anhängt.

Die blau umrandeten Objekte stellen jeweils eine Verweisfunktionalität bereit.

15.6 Element *view*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	4
Chrome (Webkit)	9
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Element ***view*** kann in der Vollversion von SVG 1.1 eine (alternative) Darstellung des Dokumentes vordefiniert werden.

Wird in einem Verweis auf ein Element ***view*** verwiesen, so überschreibt dieses mit seinen die Anzeige betreffenden Attributen die entsprechenden des nächstgelegenen ***svg***-Elternelement. Der entsprechend definierte Bereich wird angezeigt.

Da das Element selbst nicht angezeigt wird, wird es idealerweise im ***defs***-Element jenes ***svg***-Elementes notiert, für welches die Darstellung definiert werden soll.

Neben den allgemein verfügbaren Attributen können folgende Attribute notiert werden:

- *viewBox*
- *preserveAspectRatio*
- *zoomAndPan*
- *viewTarget*

Die Bedeutung und Wirkung der Attribute *viewBox*, *preserveAspectRatio* und *zoomAndPan* ist bereits für das Element ***svg*** im Kapitel Dokumentstruktur beschrieben. Dies ist entsprechend anwendbar.

15.6.1 Attribut *viewTarget*

Mit dem Attribut *viewTarget* kann ein Fragmentidentifizierer angegeben werden. Der Wert ist der Fragmentidentifizierer jenes Elementes, welches bei der Darstellung zentriert dargestellt werden soll.

Das Attribut ist nicht animierbar.

3 `http://de.wikibooks.org/wiki/media%3ASVGa02.svg`

15.6.2 Beispiele für *view*

Eine einfache Anwendung, mit der verschiedene Bereiche einer Graphik dargestellt werden können. Die Objekte in den Bereichen bilden einen Navigationsring, es ist also möglich, per Verweis von einem Bereich zum nächsten zu springen. Alternativ kann der jeweilige Bereich natürlich auch mit einem externen Verweis ausgewählt werden.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1"
     xmlns="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink"
     viewBox="-100 -100 1200 1200"
     xml:lang="de">

<title>Beispiel Element view</title>
<desc>
Variante A: blauer Kreis
Variante B: rotes Rechteck
Variante C: grüne Ellipse
Variante D: gelbe gekreuzte Linien, eine hervorgehoben
</desc>

<defs>
<view id="A" viewBox="-200 -200 400 400" />
<view id="B" viewBox="800 -200 400 400" zoomAndPan="disable" />
<view id="C" viewBox="-200 800 400 400" preserveAspectRatio="none" />
<view id="D" viewBox="800 800 400 400" viewTarget="L" />
</defs>

<a xlink:href="#B">
<circle r="50" fill="blue" />
</a>
<a xlink:href="#C">
<rect  x="950" y="-50" width="100" height="100" fill="red" />
</a>
<a xlink:href="#D">
<ellipse cy="1000" rx="60" ry="40" fill="green" />
</a>
<a xlink:href="#A">
<line id="L" x1="950" x2="1050" y1="950" y2="1050"
      fill="none"
      stroke="yellow" stroke-width="10" stroke-linecap="round" />
<line x2="950" x1="1050" y1="950" y2="1050"
      fill="none"
      stroke="yellow" stroke-width="10" stroke-linecap="round" />
</a>
</svg>
```

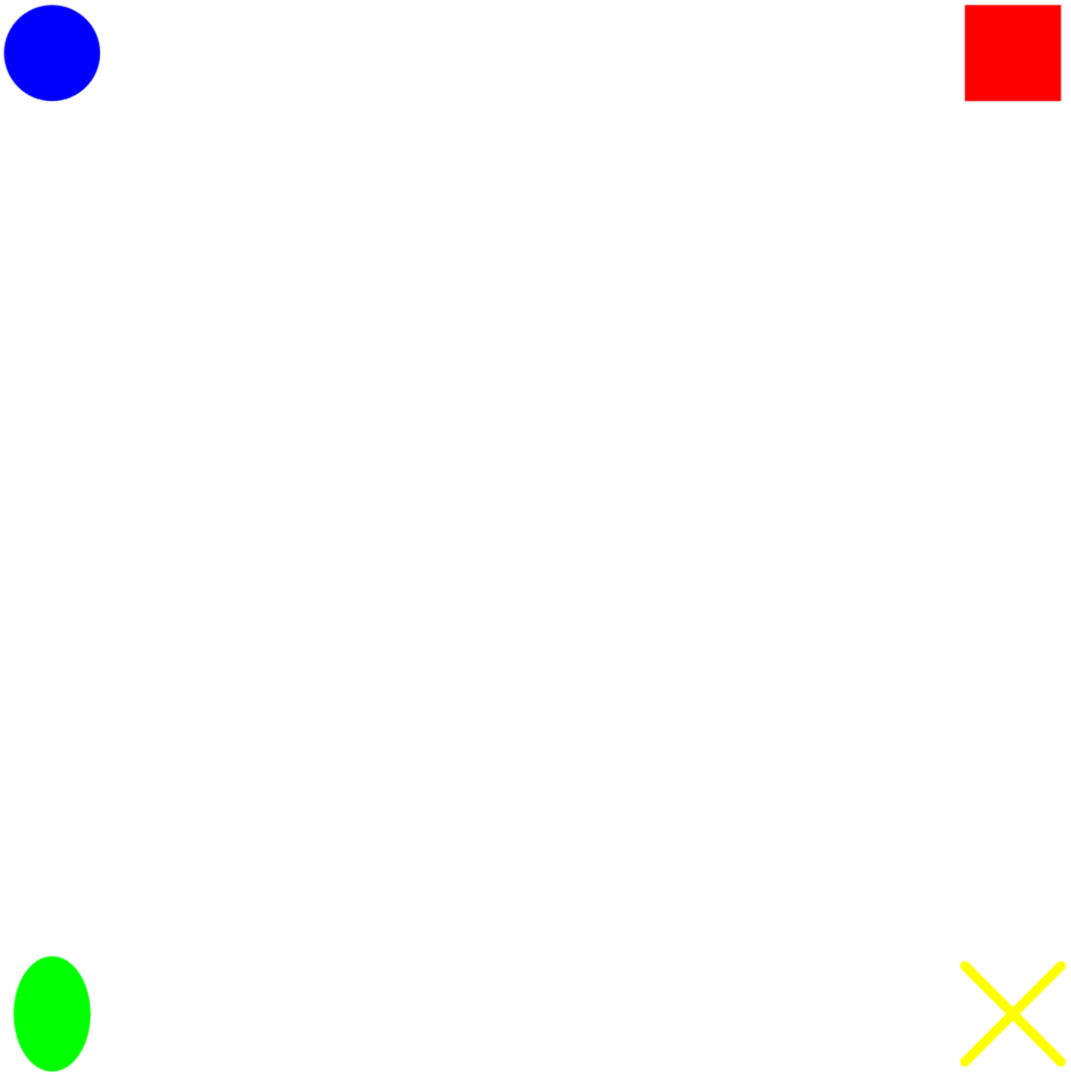


Abb. 275 Element *view*

Entsprechend zum Angucken mit ein paar mehr Alternativen: Element *view*[4]

Variante A: blauer Kreis

Variante B: rotes Rechteck

Variante C: grüne Ellipse

4 http://de.wikibooks.org/wiki/media%3ASVGview01.svg

Variante D: gelbe gekreuzte Linien, eine hervorgehoben

Variante E: A+B

Variante F: A+C

Variante G: C+D

Variante H: B+D

Variante Z: Eingangsausschnitt

Im Bereich der Dekoration oder des Designs von internet-Seiten mit CSS-Hintergrundbildern ist in letzter Zeit mehr und mehr die Idee aufgekommen, lieber wenige archivartige Bilder zu verwenden, statt viele Einzelbilder, weil die Ladezeiten und die Effektivität des Seitenaufbaus auch von der Anzahl der zu ladenden Dateien abhängt, nicht nur von der Dateigröße. Für solche Anwendungen eignen sich #svgView und ***view*** natürlich ideal. Dies funktioniert dann prinzipiell auch bereits mit den Möglichkeiten von CSS2 und nicht erst mit ähnlichen Funktionalitäten von CSS3.

16 Interaktivität

16.1 Interaktivität

SVG-Dokumente können interaktiv sein, also auf Ereignisse reagieren, die vom Betrachter oder Nutzer des Dokumentes ausgehen.

Sofern Skriptsprachen interpretiert werden, können Skripte die Erscheinung des Dokumentes zeitabhängig ändern, was durch den Nutzer beinflußbar sein kann, wofür spezielle Mechanismen im Dokumentobjektmodell verfügbar sind (englisch: listener, handler). Dies wird im Kapitel über Skripte behandelt.

Eine Interaktion kann auch den Beginn oder das Ende einer Animation hervorrufen. Dies wird im Kapitel über Animation behandelt. Die Listen für den Beginn und das Ende haben spezielle Ereigniswerte, mit denen ähnlich wie für Skriptsprachen Ereignisse genutzt werden können, damit der Betrachter die Zeitabhängigkeit des Dokumentes aktiv beeinflussen kann.

SVG stellt auch die Funktionalität von Verweisen (englisch: hyperlink) zur Verfügung. Dies wird im Kapitel Verweise erklärt.

SVG tiny 1.2 bietet ferner die Möglichkeit anzugeben, welche Elemente fokussiert oder editiert werden können und bietet eine Möglichkeit, einen Navigationsring anzugeben.

Mit der Eigenschaft *pointer-events* kann der Autor auch beeinflussen, in welcher Weise Interaktivität mit einem Zeigergerät Einfluss auf ein Element haben kann, ob dieses das Ereignis also registriert oder nicht. Die Eigenschaft ist beschrieben im Kapitel über allgemein verwendbare Eigenschaften (Kapitel Graphiken formatieren).

Sofern das Darstellungsprogramm CSS interpretiert, ist mit den dynamischen Pseudoklassen ':hover', ':active' und ':focus' ebenfalls eine Änderung der Darstellung durch Interaktivität möglich. Anders als die Ereignisse, welche für Skripte und Animationen verwendbar sind, sind Pseudoklassen jedoch nicht direkt an Ereignisse gebunden, sondern stellen einen aktuellen Status dar. Ändert ein Objekt seinen Status, können die mit einer solchen Pseudoklasse notierten Eigenschaften wirksam oder unwirksam werden. CSS definiert allerdings nicht präzise, wann sich der Status ändert, so dass das dynamische Verhalten je nach Darstellungsprogramm deutlich anders ausfallen kann (bei ':hover' etwa ist nicht klar, ob der Status nur eintritt, wenn sich ein Zeigergerät über das betroffene Element bewegt oder auch, wenn sich das betroffene Element unter das Zeigergerät bewegt und umgekehrt, wenn der Status wieder verloren geht). In der Praxis eignen sich daher Pseudoklassen nicht für präzise Interaktionen, sondern wie CSS ganz allgemein für dekorative Aspekte. Interessant dabei ist auch, dass Arbeitsentwürfe für CSS3 veröffentlicht wurden (2009), mit denen dekorative Übergänge und Animationen mittels CSS möglich werden.

16.2 Ereignisse

Ereignisse werden in SVG über das Dokumentobjektmodell ausgewertet und für Beginn- und Endlisten von Animationen. Insgesamt ergibt sich daraus eine recht umfangreiche Liste. Teile davon sind nur über Skripte und das Dokumentobjektmodell auswertbar.

16.2.1 Blubbern

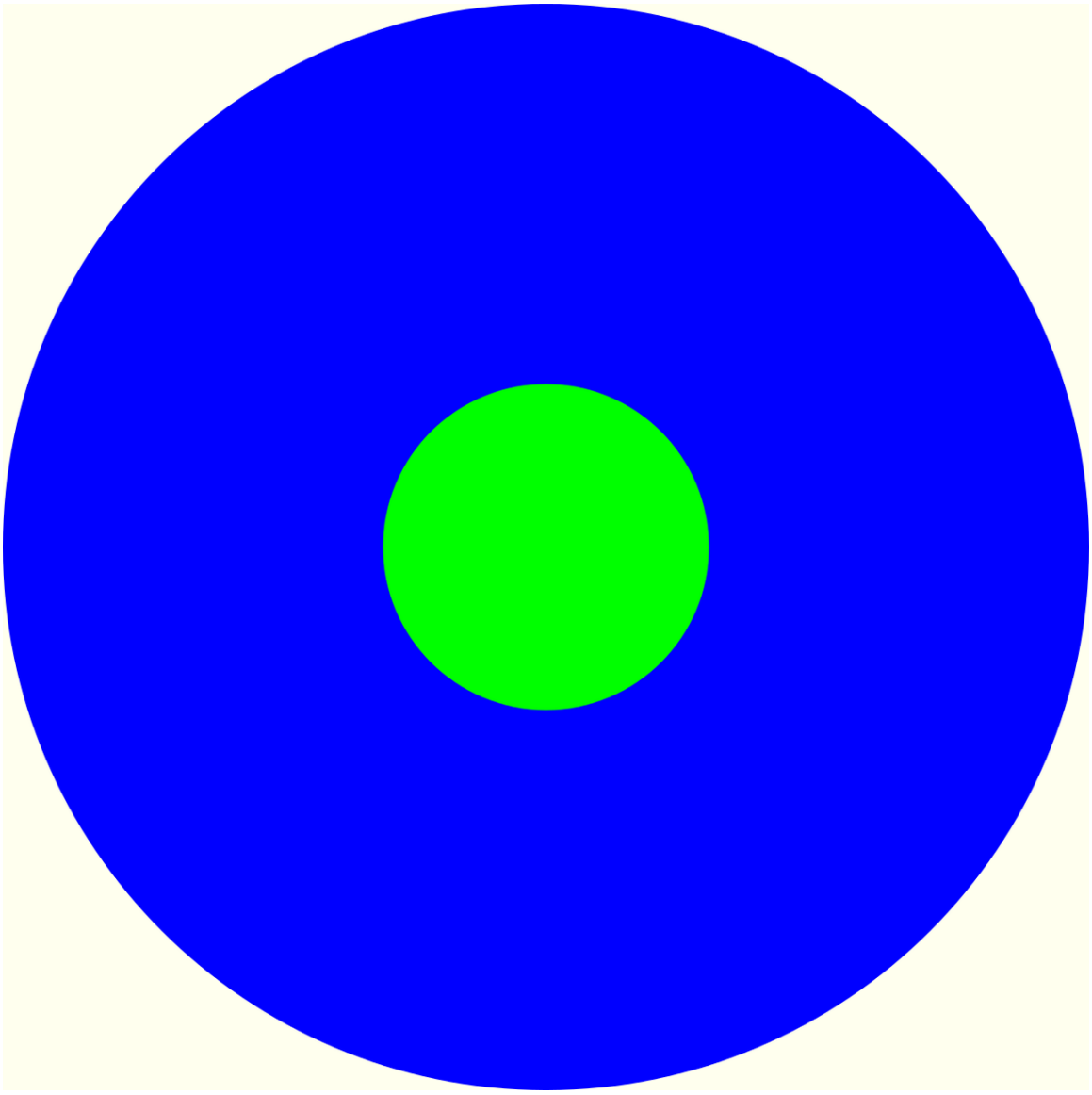


Abb. 276 Beispiel für Blubbern

Wenn ein Element ein Ereignis registriert (zum Beispiel einen Klick), so ist eine relevante Frage, ob auch das Elternelement das Klickereignis registriert oder ob das angeklickte Element das Ereignis vor den Elternelementen abschirmt. Wenn das Ereignis nicht abgeschirmt wird, sondern auch an das Elternelement weitergereicht wird, so wird von blubblern (englisch: bubble) gesprochen.

Etwa folgendes Beispiel für Blubbern[1]:

```
<g fill="blue">
<set attributeName="fill" to="green" begin="click" />
<circle r="100" />
<circle r="30" fill="green">
  <set attributeName="fill" to="yellow" begin="click" />
</circle>
</g>
```

Wird nun der gelbe Kreis angeklickert, ist die Frage des Blubberns relevant für die Farbe des größeren Kreises. Blubbert das Ereignis, wird *fill* vom Element ***g*** auf 'green' gesetzt und der große Kreis wird grün. Blubbert es nicht, so wird nur der kleine Kreis gelb, der große bleibt blau. Korrekt ist in diesem Falle das Blubbern, also wird der große Kreis grün, wenn der kleine angeklickert und gelb wird.

In der Tat blubbern fast alle Ereignisse, die für Beginn- und Endlisten von Animationen verfügbar sind. Ausnahme sind die Ereignisse 'zoom' und 'rotate'. Problematisch sind ferner die Ereignisse 'beginEvent', 'endEvent' und 'repeatEvent', weil in SMIL definiert ist, dass diese nicht blubbern (was dann auch auf SVG 1.1 zutrifft). SVG tiny 1.2 legt hingegen fest, dass diese blubbern.

Der Autor kann das Blubberverhalten nicht selbst festlegen. Allenfalls kann die Struktur des gesamten Dokumentes so angelegt werden, dass durch das Blubbern kein Problem entsteht.

Das Verhalten ist in der Tabelle im folgenden Abschnitt einzeln angegeben.

16.2.2 Ereignisliste

Eine komplette Liste (Tabelle) mit Ereignissen und in welchem Zusammenhang sie verwendet werden, sieht wie folgt aus. Abbrechen bezieht sich auf die Möglichkeit, ein Ereignis über das DOM (Dokumentobjektmodell) zu unterbinden (englisch: cancelable). Ein DOM-Ereignisattribut kann in SVG 1.1 eingesetzt werden, um ein Skript anzusteuern und wird dann im betroffenen Element notiert. SVG tiny 1.2 hat dafür spezielle Elemente (***listener*** und ***handler***).

1 http://de.wikibooks.org/wiki/media%3ASVG_bubble01.svg

Ereignisliste					
Ereignistyp	**Beschreibung**	**Name für Animation**	**Name DOM-Ereignisattribut (SVG1.1)**	**Blubbern**	**Abbrechen**
DOMFocusIn	Element erhält Fokus	focusin	onfocusin	ja	nein
DOMFocusOut	Element verliert Fokus	focusout	onfocusout	ja	nein
DOMActivate	Element wird aktiviert, etwa durch Anklickern oder Tastendruck	activate	onactivate	ja	ja
click	Element mit Zeigergerät angeklickert: mousedown - mouseup - click	click	onclick	ja	ja
mousedown	Knopf am Zeigergerät gedrückt	mousedown	onmousedown	ja	ja
mouseup	Knopf am Zeigergerät losgelassen	mouseup	onmouseup	ja	ja
mouseover	Zeigergerät auf Element bewegt	mouseover	onmouseover	ja	ja
mousemove	Zeigergerät über Element bewegt	mousemove	onmousemove	ja	ja
mouseout	Zeigergerät vom Element wegbewegt	mouseout	onmouseout	ja	ja
mousewheel	Rad vom Zeigergerät aktiviert (nur SVG tiny 1.2)	-	-	ja	ja
beginEvent	zeitabhängiges Element beginnt (*begin*-Liste)	beginEvent	onbegin	ja (SVG tiny 1.2), nein (SVG 1.1)	nein
endEvent	zeitabhängiges Element endet (*end*-Liste)	endEvent	onend	ja (SVG tiny 1.2), nein (SVG 1.1)	nein
repeatEvent	zeitabhängiges Element wiederholt sich	repeatEvent	onrepeat	ja (SVG tiny 1.2), nein (SVG 1.1)	nein
textInput	Text eingegeben (nur SVG tiny 1.2)	-	-	ja	ja
keydown	Taste gedrückt (nur SVG tiny 1.2)	-	-	ja	ja
keyup	Taste losgelassen (nur SVG tiny 1.2)	-	-	ja	ja
loadstart	Ladeoperation beginnt, nur SVG tiny 1.2	-	-	nein	nein
progress	Ladeoperation einer Resource macht Fortschritte, nur SVG tiny 1.2	-	-	nein	nein
loadend	Ladeoperation beendet, nur SVG tiny 1.2	-	-	nein	nein
SVGTimer	Zeitgeberintervall abgelaufen, nur SVG tiny 1.2	-	-	nein	nein
load	Element samt Inhalt und referenzierter Ressourcen geladen (nur SVG tiny 1.2)	load	-	nein	nein
SVGload	Dokument geladen, nur Element ***svg*** (veraltet in SVG tiny 1.2, load benutzen)	SVGload (in SVG 1.1)	onload	nein	nein

Ereignisliste					
Ereignistyp	**Beschreibung**	**Name für Animation**	**Name DOM-Ereignisattribut (SVG1.1)**	**Blubbern**	**Abbrechen**
resize	Größe des Anzeigebereiches verändern, nur Element ***svg*** (nur SVG tiny 1.2)	resize	-	ja	nein
SVGResize	Größe des Anzeigebereiches verändern, nur Element ***svg*** (veraltet in SVG tiny 1.2, resize benutzen)	SVGResize (in SVG 1.1)	onresize	ja	nein
scroll	Dokument rollen, nur Element ***svg*** (nur SVG tiny 1.2)	scroll	-	ja	nein
SVGScroll	Dokument rollen, nur Element ***svg*** (veraltet in SVG tiny 1.2, scroll benutzen)	SVGScroll (in SVG 1.1)	onscroll	ja	nein
SVGZoom	Dokument vergrößern oder verkleinern, nur Element ***svg***	zoom	onzoom	nein	nein
SVGRotate	Dokument drehen, nur Element ***svg***, nur SVG tiny 1.2	rotate	-	nein	nein
SVGUnload	Dokument aus Anzeige entfernt, nur Hauptelement ***svg*** (nur SVG 1.1)	SVGUnload	onunload	nein	nein
SVGAbort	Dokument laden abgebrochen, bevor ein Elmenent komplett geladen ist (nur SVG 1.1)	SVGAbort	onabort	ja	nein
SVGError	Element nicht richtig geladen oder fehlerhaft (nur SVG 1.1)	SVGError	onerror	ja	nein
DOMSubtreeModified	Allgemeine DOM-Änderung (nur SVG 1.1)	-	-	ja	nein
DOMNodeInserted	Kindknoten eingefügt (nur SVG 1.1)	-	-	ja	nein
DOMNodeRemoved	Kindknoten von anderem Knoten entfernt (nur SVG 1.1)	-	-	ja	nein
DOMNodeRemovedFromDocument	Kindknoten vom Dokument entfernt (nur SVG 1.1)	-	-	nein	nein
DOMNodeInsertedIntoDocument	Kindknoten in Dokument eingefügt (nur SVG 1.1)	-	-	nein	nein
DOMAttrModified	Attribut geändert (nur SVG 1.1)	-	-	ja	nein
DOMCharacterDataModified	Textinhalt eines Knotes geändert (nur SVG 1.1)	-	-	ja	nein

16.3 Fokussierbarkeit

Squiggle (Batik)	-
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

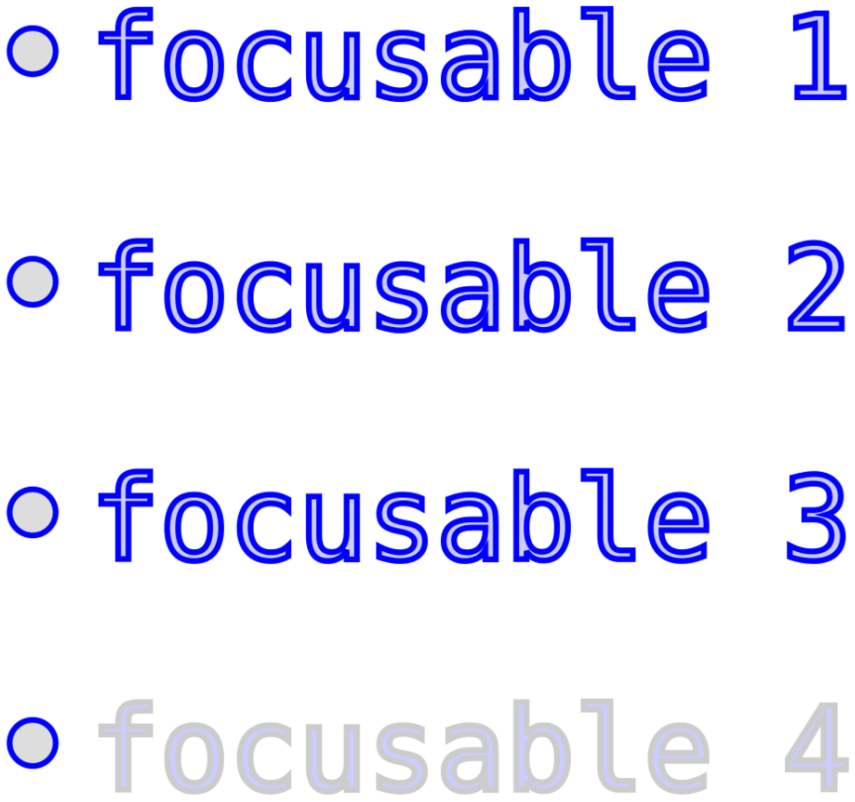


Abb. 277 Beispiel Fokussierbarkeit

In SVG tiny 1.2 kann mit dem Attribut *focusable* angegeben werden, ob ein Element fokussierbar ist oder nicht. Die korresondiert dann mit den Ereignissen 'focusin' und 'focusout'. Ist ein Element fokussierbar, so können diese Ereignisse bei diesem auftreten, sonst nicht.

Alle darstellbaren und gruppierenden Elemente (außer ***defs***) können fokussierbar. Ein fokussierbares Gruppierungselement kann auch fokussierbare Kindelemente enthalten.

Die möglichen Werte sind:

true

Das Element ist fokussierbar

false

Das Element ist nicht fokussierbar

auto

Das ist die Voreinstellung, das im folgenden angegebene Verhalten hängt vom Element ab und deckt sich mit dem, was in SVG 1.1 immer gilt.

Das Verhalten für 'auto' entspricht 'false', falls nicht eine der folgenden Ausnahmen auftritt:

- Das Element ***a***.
- Textelemente (die dargestellt werden; in tiny 1.2 ***text***, ***textArea*** und ***tspan***), welche das Attribut *editable* auf 'simple' gesetzt haben.
- Zielelemente von Animationen beziehungsweise damit verbundenen Ereignissen, falls die *begin*- und *end*-Listen 'focusin', 'focusout' oder 'activate' enthalten.
- Elemente mit einem Ereignisempfänger für die DOM-Ereignisse 'onfocusin', 'onfocusout', 'onactivate'.

focusable ist animierbar.

Beispiel Fokussierbarkeit[2]

Text mit blauem Strich sind fokussierbar. Ein grauer Strich zeigt an, dass der Text nicht fokussierbar ist. Strich und Fokussierbarkeit werden bei zwei Texten animiert.

Eine Fokussierung bewirkt eine Animation der Füllung des benachbarten Kreises.

16.4 Hervorhebung

Squiggle (Batik)	-
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

2 `http://de.wikibooks.org/wiki/media%3ASVGfocusable01.svg`

Abb. 278 Beispiel *focusHighlight*

Mit dem Attribut *focusHighlight* kann in SVG tiny 1.2 festgelegt werden, ob ein fokussiertes Element hervorgehoben werden soll oder nicht. Die Art der Hervorhebung kann abhängig vom Darstellungsprogramm sein, zum Beispiel könnte die Farbe verändert werden oder ein Rahmen um das Objekt gemalt werden.

Der Attributwert ist entweder 'auto' oder 'none'.

Bei 'none' sollen fokussierte Elemente nicht hervorgehoben werden. Bei 'auto' soll das fokussierte Element hervorgehoben werden. Dies ist die Voreinstellung.

Das Attribut ist nicht animierbar.

Beispiel *focusHighlight*[3]

Die Antworten auf die Frage 'Wie geht's?' können fokussiert werden, was im Falle der Antwort 'Gut' hervorgehoben werden sollte. Im Falle von 'Schlecht' bedingt durch focusHighlight sollte keine Hervorhebung stattfinden. Unabhängig davon wird der Strich der Rechtecke durch Fokussierung farblich verändert.

16.5 Editierbarkeit

Squiggle (Batik)	-
Opera (Presto)	9.5 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

3 `http://de.wikibooks.org/wiki/media%3ASVGfocusHighlight01.svg`

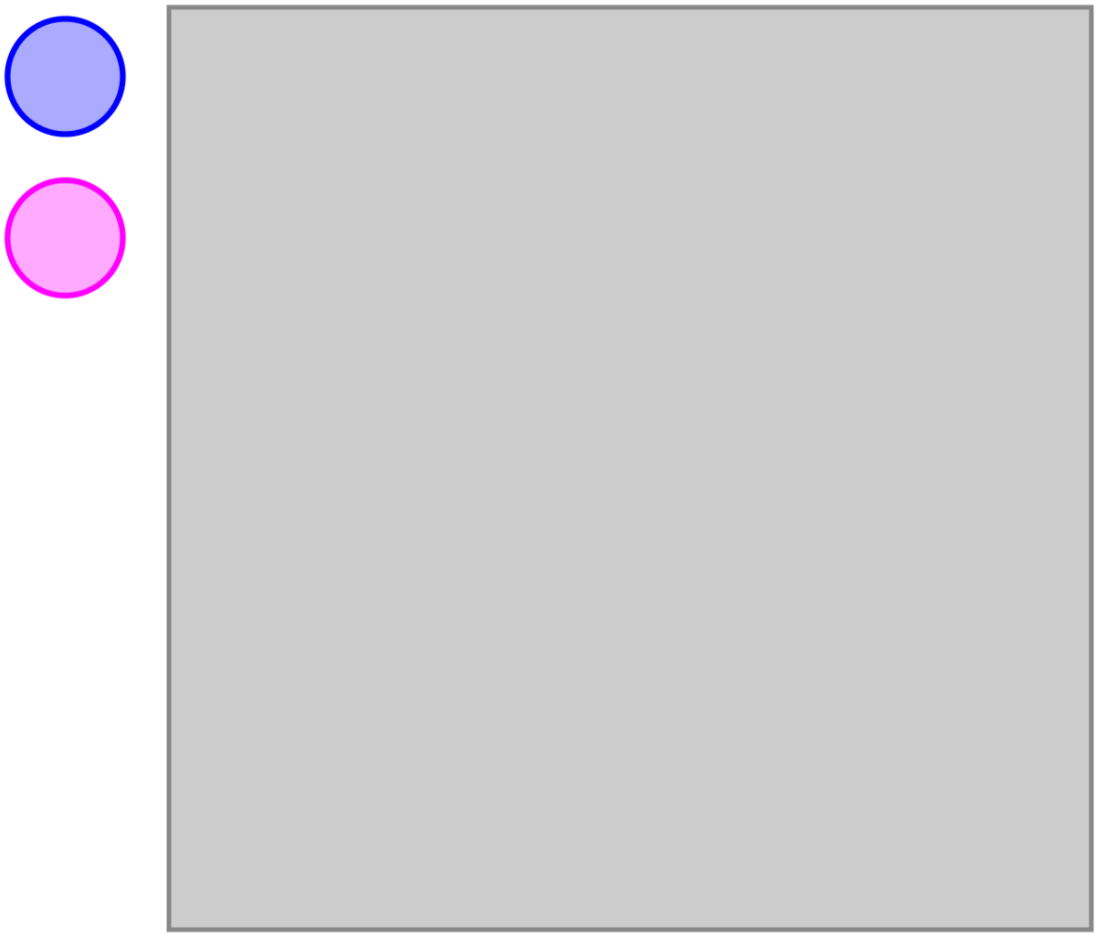


Abb. 279 Beispiel *editable*

In SVG tiny 1.2 kann mit dem Attriut *editable* angegeben werden, ob Textelemente (***text***, ***textArea*** und ***tspan***) editierbar sind oder nicht.

Jenseits von Skriptinteraktionen bietet SVG tiny 1.2 jedoch keine Möglichkeit für den Autor, das Dokument auf die Editierung reagieren zu lassen oder dies zum Beispiel mit einem Formular abschicken zu lassen. Eine formularartige Interaktion erfordert demzufolge Formularfunktionalitäten, wie sie zum Beispiel von XForms oder von XHTML zur Verfügung gestellt werden. Da eine Interpretation von Skripten beim Nutzer nicht sichergestellt werden kann, sollten damit keine relevanten Informationen verareitet werden, ohne die Zugänglichkeit der Verarbeitung anderweitig zu gewährleisten (Kriterium der Barrierefreiheit und Zugänglichkeit).

Die möglichen Werte sind 'none' oder 'simple'. Bei 'none' gibt es keine Editiermöglichkeit. Bei 'simple' stellt das Darstellungsprogramm eine einfache Eingabemöglicheit für dargetellten, nicht versteckten Text zur Verfügung, mit welcher der Text editiert werden kann.

Sofern eine Zwischenablage im Darstellungsprogramm verfügbar ist, muss sich auch damit einfach Text hin- und herkopieren lassen.

Im Falle von 'simple' ist *focusable* 'true'.

editable ist animierbar.

Beispiel *editable*[4]

Nach einem Aktivieren des blauen Kreises ist der Text für 10s editierbar, Dies wird mit einem blauen Strich des Rechtecks gekennzeichnet. Mit dem magenta Kreis kann die Editierbarkeit alternativ beendet werden.

16.6 Navigation

Zwischen fokussierbaren Elementen soll navigiert werden können. Dazu ist von den Darstellungsprogrammen ein geeigneter Mechanismus zur Verfügung zu stellen, oft zum Beispiel mit Tabulatortasten.

Im Dokument besteht ein Fokussring, in dem kann also von einem fokussierbaren Element zum nächsten gesprungen werden, vom letzten wieder zum ersten. Ohne weitere Angaben wird die Reihenfolge verwendet, wie die fokussierbaren Elemente im Quelltext stehen.

SVG tiny 1.2 bietet auch die Möglichkeit, die Reihenfolge zu ändern oder statt eines einfachen Ringes ein effektiveres Muster zu realisieren.

Ein Element mit *display*="none" ist nicht fokussierbar und ist daher nicht Bestandteils eines Navigationsmechnismus. Lediglich unsichtbare oder transparente Elemente sind hingegen fokussierbar. Es können auch Elemente fokussiert werden, die außerhalb des angezeigten Bereiches liegen. Wenn das Element fokussiert wird, soll der Anzeigebereich so verschoben werden, dass das fokussierbare Element im sichtbaren Bereich liegt.

Das Verhalten ist wie folgt:

- Beim Laden liegt der Fokus zuerst beim Darstellungsprogramm.
- Sobald das Darstellungsprogramm den Fokus freigibt, geht der Fokus an ein anderes Element über, das erste passende aus folgender Liste:
  - Das Wurzelelement ***svg***, falls dies fokussierbar ist.
  - Das Element, welches mit dem Attribut *nav-next* angegeben im Wurzelelement ***svg*** ist, sofern es angegeben ist.
  - Das erste fokussierbare Element im Dokument
  - Das Darstellungsprogramm
- Wenn ein Element fokussiert ist, dann wird das nächste fokussierte Element wie folgt bestimmt, die erste zutreffende Bedingung gilt:
  - Sofern im fokussierten Element das Attribut *nav-next* angegeben ist, das darin referenzierte Element
  - Das nächste fokussierbare Element in der Reihenfolge des Quelltextes
  - Das Darstellungsprogramm

4 `http://de.wikibooks.org/wiki/media%3ASVGeditable01.svg`

- Wenn ein Element fokussiert ist, dann wird das vorherige fokussierte Element wie folgt bestimmt, die erste zutreffende Bedingung gilt:
  - Sofern im fokussierten Element das Attribut *nav-prev* angegeben ist, das darin referenzierte Element
  - Das vorherige fokussierbare Element in der Reihenfolge des Quelltextes
  - Das Darstellungsprogramm

In alleinstehenden SVG-Dokumente, muss das Darstellungsprogramm immer ein aktuell fokussiertes Objekt haben. Hat kein Element im Dokument den Fokus, so hat das Dokument selbst den Fokus.

Wird ein SVG-Dokument in ein anderes Dokument eingebettet, nimmt das SVG-Dokument am Navigationsring des einbettenden Dokumentes teil, sofern dies nicht explizit anderweitig anders festgelegt ist.

Der Fokusring muss allerdings auch verlassen werden können, wozu das Darstellungsprogramm einen Mechanismus zur Verfügung stellen muss, der Fokus geht dann zurück an das Darstellungsprogramm.

16.6.1 Navigationsattribute

Squiggle (Batik)	-
Opera (Presto)	9.5 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Abb. 280 Beispiel Navigationsattribute *nav-next* und *nav-prev*

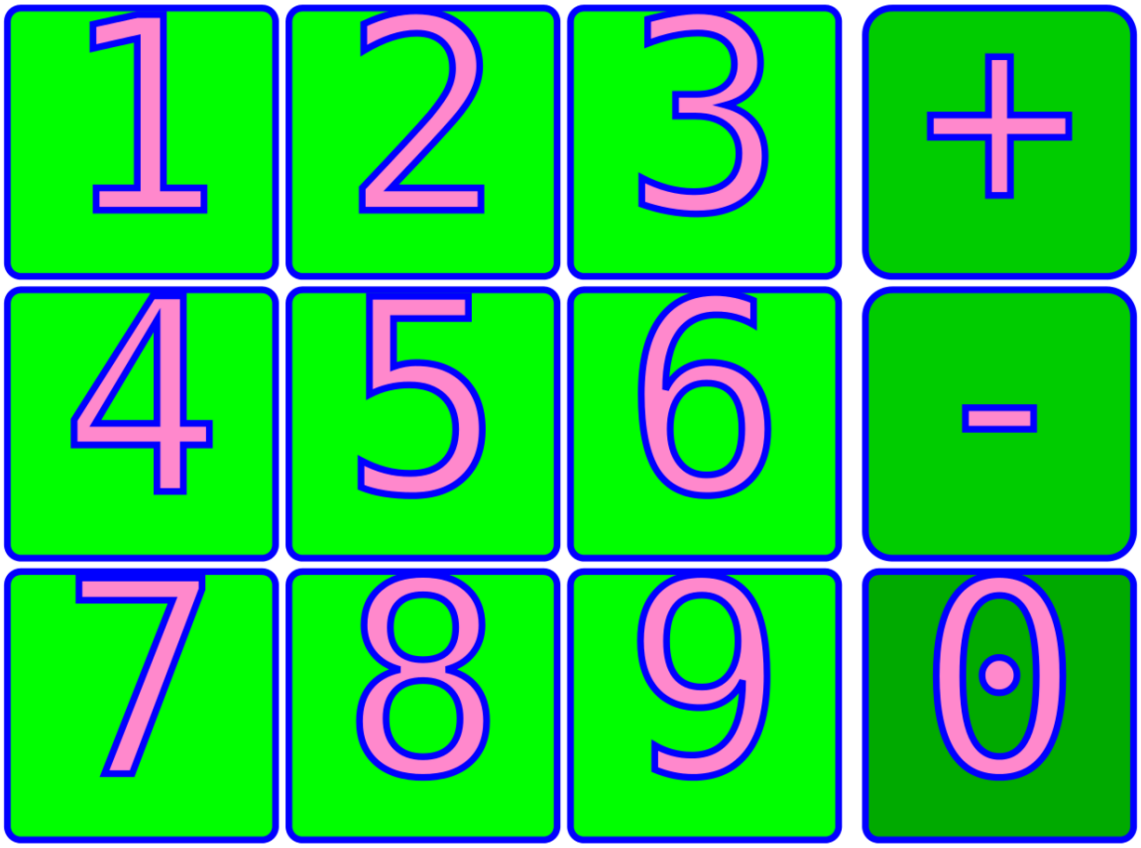


Abb. 281 Beispiel Navigationsattribute

Die Reihenfolge der Navigation in einem Fokusring kann mit folgenden Attributen festgelegt werden:

nav-next gibt das nächste Element an, *nav-prev* das vorherige, mögliche Werte sind:

self

Der Fokus bleibt beim Element selbst

auto

Verhalten, als ob das Attribut nicht angegeben ist, die Voreinstellung

Funktionale IRI

Fragmentidentifizierer des nächsten, beziehungsweise des vorherigen fokussierbaren Elementes im selben Dokument; also url(ID) mit ID dem Identifizierer

Die Attribute sind animierbar.

Weitere acht Attribute ermöglichen eine Navigation in bestimmte 'Richtungen' in einer gedachten zweidimensionalen Ebene. Die 'Richtung' ergibt sich aus dem Namen:

nav-up

Nach oben

nav-up-right

Nach oben rechts

nav-right

Nach rechts

nav-down-right

Nach unten rechts

nav-down

Nach unten

nav-down-left

Nach unten links

nav-left

Nach links

nav-up-left

Nach oben links

Mögliche Werte sind die gleiche wie für *nav-next* und *nav-prev*:

self

Der Fokus bleibt beim Element selbst

auto

Verhalten liegt beim Darstellungsprogramm, als ob das Attribut nicht angegeben ist, die Voreinstellung

Funktionale IRI

Fragmentidentifizierer des fokussierbaren Elementes in der angegebenen 'Richtung' im selben Dokument; also url(ID) mit ID dem Identifizierer

Die Attribute sind animierbar.

Beispiel Navigationsattribute *nav-next* und *nav-prev*[5]

Acht Textelemente 1 bis 8 sind nebeneinander angeordnet. Mittels nav-next and nav-prev kann von einem zum nächsten gewechselt werden, jeweils umgekehrt als die Zahlen angeben und die Elemente im Quelltext stehen.

Wenn ein nicht sichtbares Element fokussiert wird, wird der Anzeigebereich so verschoben, dass das fokussierte Element im sichtbaren Bereich liegt.

5 `http://de.wikibooks.org/wiki/media%3ASVGnavigation01.svg`

Beispiel Navigationsattribute[6]

Zehn Ziffern, + und - sind als Textelemente nebeneinander angeordnet. Mit Navigationselementen kann von einem zum nächsten gewechselt werden, für 1 bis 9 die Richtungenentsprechend der räumlichen Anordnung, für +, - und 0 mit vorwärts und rückwärts untereinander.

Ein fokussiertes Element sollte hervorgehoben sein.

16.7 Positioniererelement, Mauszeiger

Bei interaktiven Darstellungsprogrammen wird der Zeiger, die Maus oder das Positioniererelement oftmals abhängig von der aktuellen Funktion anders dargestellt. Dazu hat das Darstellungsprogramm einen Satz von Präsentationen, welche über die Eigenschaft *cursor* in der Vollversion von SVG 1.1 verfügbar ist. Die Eigenschaft ist im Kapitel Graphiken formatieren beschrieben. Neben den vordefinierten Formen kann der Autor auch eigene Präsentationen referenzieren. Damit Autoren die graphische Repräsentation auch innerhalb des SVG-Dokumentes mittels SVG oder eines Pixelgraphikformates realisieren können, gibt es das Element ***cursor***.

Weil die Eigenschaft *cursor* ganz allgemein in CSS definiert ist, stehen die per SVG realisierten Präsentationen auch für Dekorationen in anderen Formaten zur Verfügung, sofern das Darstellungsprogramm *cursor* und SVG interpretiert.

16.7.1 Element *cursor*

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	12
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Element ***cursor*** gibt es in der Vollversion von SVG 1.1 die Möglichkeit die Präsentation des Mauszeigers oder der Positionsmarke selbst und einheitlich für ein Dokument zu definieren.

Es wird ein komplettes Bild oder ein Element referenziert, welches den Zeiger darstellt. Dazu wird die präzise Zeigerposition angegeben.

Ansonsten ist das Element in der Spezifikation recht spärlich beschrieben. Bei der Referenzierung eines Elementes ist davon auszugehen, dass dieses einen eigenen Anzeigebereich mit definierter Höhe und Breite und vor allem definiertem Koordinatensystem haben sollte, also im Zweifelsfalle ein ***symbol*** oder ein ***svg*** sein sollte.

Damit der Zeiger überhaupt präsentiert werden kann, ist dieser vermutlich mit der Eigenschaft *cursor* zusammen mit optionalen weiteren Zeigern, gefolgt von einem generischen Zeiger anzugeben. Der jeweils folgende Zeiger wird in dem Fall verwendet, wenn der oder die davor angegebenen Zeiger nicht darstellbar sein sollten.

6 `http://de.wikibooks.org/wiki/media%3ASVGnavigation02.svg`

In der Dokumenttypdefinition des Elementes wurde ferner vermutlich vergessen anzugeben, dass das Element Animationselemente beinhalten kann, daher sind diese im Bedarfsfalle außerhalb zu notieren und das Element ist dann als Zielelement der Animation mittels *href* von XLink zu notieren. Erlaubter Inhalt des Elementes sind daher nur ***title***, ***desc*** und ***metadata***. Als nicht direkt dargestellter Inhalt sollte ***cursor*** im Element ***defs*** notiert werden. Andere Gruppierungselemente als Elternelemente sind allerdings technisch nicht ausgeschlossen.

Attribute *x* und *y*

Mit den Attributen *x* und *y* wird im Koordinatensystem des Zeigers die Position festgelegt, auf die gezeigt wird. *x* für die x-Koordinate, *y* für die y-Koordinate. Der Wert ist für beide Attribute jeweils eine Koordinate. Die Attribute sind animierbar.

Für ein nicht angegebenes Attribut wird jeweils 0 angenommen.

Attribute von XLink, insbesondere *href*

Mit dem Attribut *href* von XLink wird die Graphik oder das Element referenziert, welches als Zeiger dienen soll. Der Wert ist eine IRI der Graphik, beziehungsweise ein Fragmentidentifizierer des Elementes. Das Attribut ist animierbar.

Damit sind auch die anderen Attribute von XLink verfügbar.

Mindestens PNG muss als Format für die Graphik interpretiert werden.

Beispiele *cursor*

Abb. 282 PNG als Zeiger

Abb. 283 Symbol als Zeiger

PNG als Zeiger[7]

Der Zeiger wechselt alle 10s zwischen einem kreisartigen Kringel und einem Kreuz, die links oben dargestellt sind. Der Wechsel ist darunter dargestellt. Zudem wird die Position animiert.

Hier sind die PNGs gleich eingebettet, zudem als Erweiterung der Dokumenttypdeklaration. Es können natürlich auch ganz einfach externe Dateien verwendet wird, wenn kein Wert darauf gelegt wird, dass alles in einem Dokument verfügbar ist. Die Erweiterung der Dokumenttypdeklaration ist hier vorteilhaft, weil die IRI der Bilder aufgrund der Animation wiederholt werden, da lohnt sich wie hier eine Abkürzung einzuführen.

Symbol als Zeiger[8]

Der Zeiger wechselt alle 10s zwischen zwei Symbolen, die links oben dargestellt sind. Der Wechsel ist darunter dargestellt. Zudem wird die Position animiert.

Der Vorteil von einem Symbol liegt natürlich in der so verfügbaren Vektorgraphik und damit der höheren Qualität. Auch kann das Symbol einfach im selben Dokument untergebracht werden. Das gilt natürlich auch für ein ***svg***.

7 `http://de.wikibooks.org/wiki/media%3ASVGcursor01.svg`
8 `http://de.wikibooks.org/wiki/media%3ASVGcursor02.svg`

16.8 Interaktivität mit CSS

Mit den dynamischen Pseudoklassen ':hover', ':active' und ':focus' kann die Dekoration eines Dokumentes abhängig von einer Nutzerinteraktion modifiziert werden. Genauer, es kann angegeben werden, wie ein Element dekoriert werden soll, wenn der jeweilige Status angenommen wird.

Diese Pseudoklassen schließen sich nicht gegenseitig aus. Auf ein Element können mehrere davon gleichzeitig zutreffen.

CSS definiert nicht, welche Elemente sich in dem Status der jeweilige Pseudoklasse befinden können, oder wie ein Element in einen solchen Status gelangt oder ihn wieder verlässt. Das Verhalten kann also bei verschiedenen Darstellungsprogrammen bei der gleichen Aktion unterschiedlich ausfallen.

16.8.1 Pseudoklasse :hover

Squiggle (Batik)	-
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	4
Chrome (Webkit)	12
Microsoft Internet Explorer (Trident)	
librsvg	

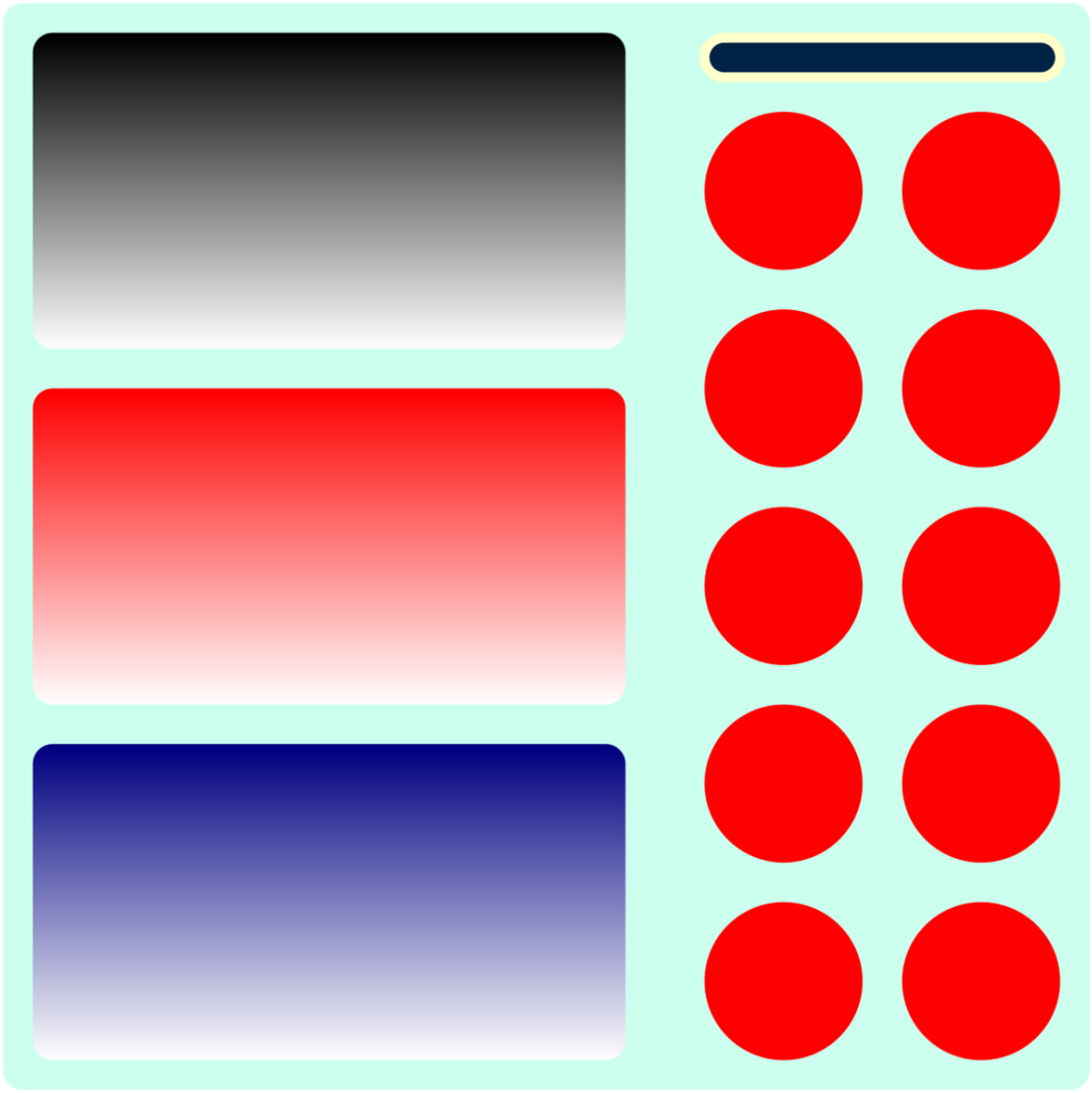


Abb. 284 Beispiel mit :hover

Die Pseudoklasse :hover findet Anwendung, wenn der Nutzer ein Element zum Beispiel mit einem Zeigergerät zwar markiert, aber nicht aktiviert. Ein Darstellungsprogramm kann :hover anwenden, wenn der Mauszeiger über ein Element gezogen wird. Darstellungsprogramme, die keine interaktiven Medien präsentieren, interpretieren auch diese Pseudoklasse nicht. Einige Zeigegeräte wie eventuell Stifte, die an sich interaktiv sind, sind gegebenenfalls auch nicht geeignet, um den Status :hover hervorzurufen.

Beispiel:

```
<defs>
        <linearGradient id="verlauf1" x1="50%" x2="50%" y1="0%" y2="100%">
                <stop offset="0" stop-color="black"/>
                <stop offset="1" stop-color="white"/>
        </linearGradient>
        <linearGradient id="verlauf2" x1="50%" x2="50%" y1="0%" y2="100%">
```

```
                <stop offset="0" stop-color="red"/>
                <stop offset="1" stop-color="white"/>
        </linearGradient>
        <linearGradient id="verlauf3" x1="50%" x2="50%" y1="0%" y2="100%">
                <stop offset="0" stop-color="navy"/>
                <stop offset="2" stop-color="white"/>
        </linearGradient>

<style type="text/css">
#r1{fill:url(#verlauf1)}
#r2{fill:url(#verlauf2)}
#r3{fill:url(#verlauf3)}

#r1:hover{fill:url(#verlauf2)}
#r2:hover{fill:url(#verlauf3)}
#r3:hover{fill:url(#verlauf1)}
</style>
</defs>

<rect width="300" height="100" x="50" y="20" rx="10" id="r1" />
<rect width="300" height="100" x="50" y="220" rx="10" id="r2"/>
<rect width="300" height="100" x="50" y="420" rx="10" id="r3"/>
```

Oder etwas ausgeschmückter mit Rechtecken, Kreisen und Gruppen und Ausnutzung von Spezifitäten:

Beispiel mit :hover[9]

Rechtecke und Kreise ändern ihre dekorativen Eigenschaften mit dem Status :hover

16.8.2 Pseudoklasse :active

Squiggle (Batik)	-
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

9 http://de.wikibooks.org/wiki/media%3ASVG_CSShover01.svg

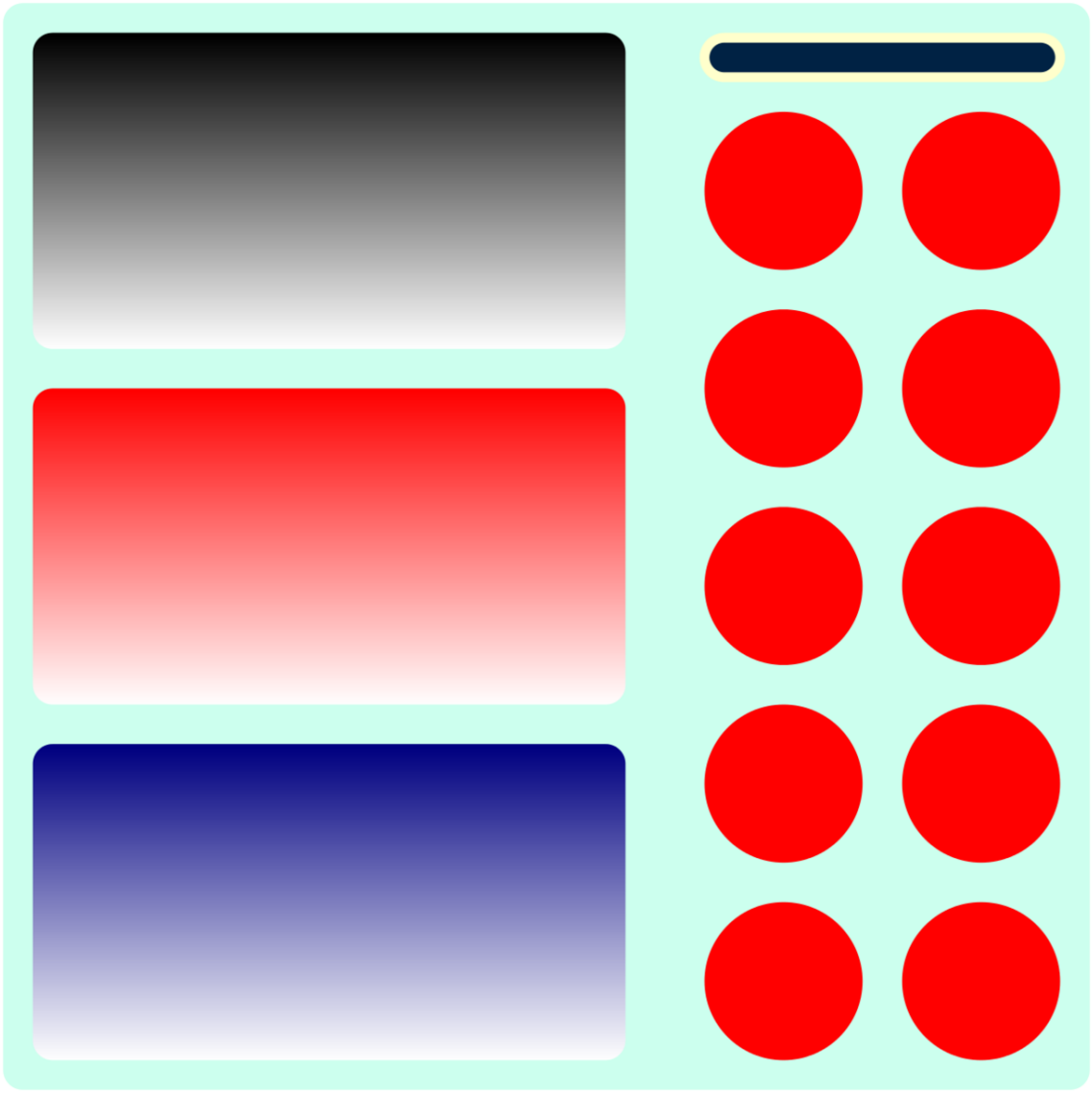


Abb. 285 Beispiel mit :active

Die Pseudoklasse :active findet Anwendung, während ein Element vom Benutzer aktiviert wird. Zum Beisiel könnte das der Fall sein, wenn der Nutzer die Maustaste drückt bis er sie wieder losläßt.

Beispiel mit :active[10]

Rechtecke und Kreise ändern ihre dekorativen Eigenschaften mit dem Status :active

10 http://de.wikibooks.org/wiki/media%3ASVG_CSSactive01.svg

16.8.3 Pseudoklasse :focus

Squiggle (Batik)	-
Opera (Presto)	-
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	12
Microsoft Internet Explorer (Trident)	
librsvg	

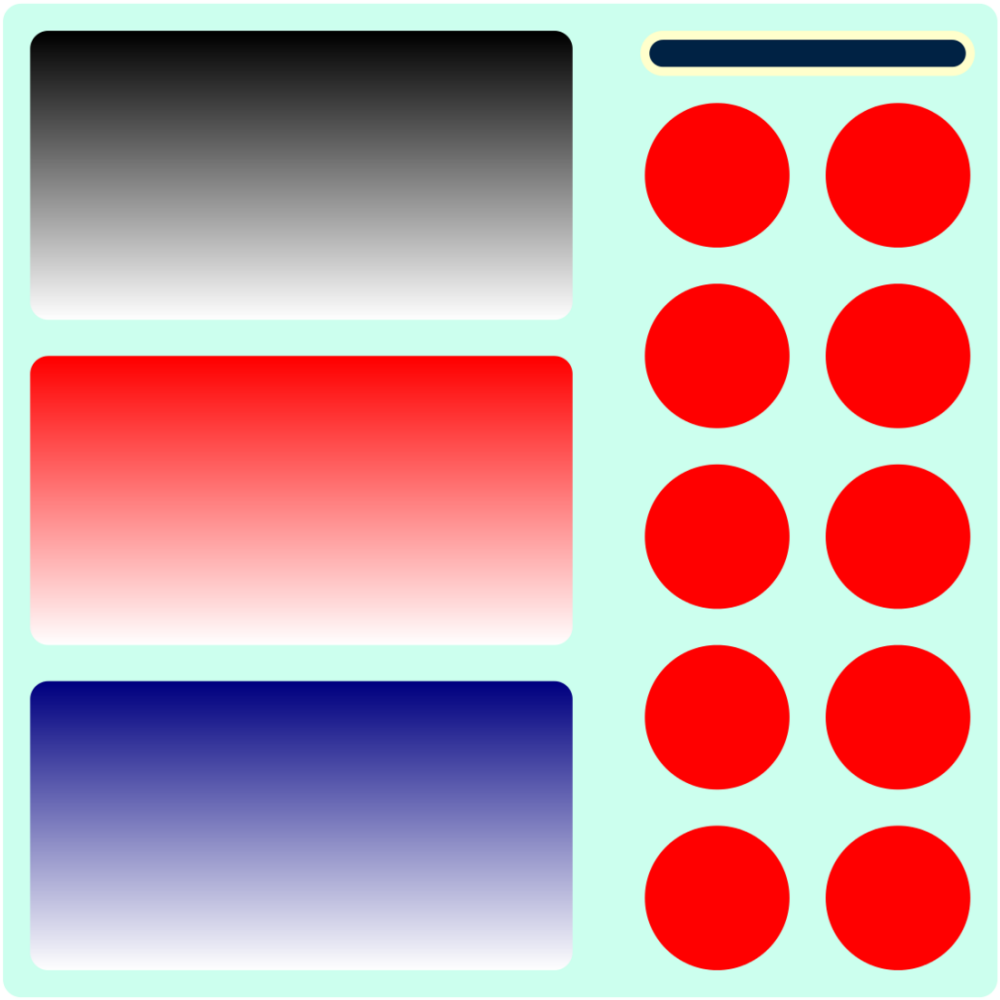


Abb. 286 Beispiel mit :focus

Die Pseudoklasse :focus findet Anwendung, während ein Element den Fokus hat, es also vom Nutzer ausgelöste Ereignisse akzeptiert, also in Ereignissen gesprochen zwischen 'focusin' und 'focusout', etwa vom Zeigergerät oder von der Tastatur ausgelöst, zum Beispiel bei einer Markierung.

Beispiel mit :focus[11]

Rechtecke und Kreise ändern ihre dekorativen Eigenschaften mit dem Status :focus

16.9 Weiterführende Literatur

- http://www.ibm.com/developerworks/library/x-svgint/
- http://www.ibm.com/developerworks/xml/library/x-matters42/index.html

11 http://de.wikibooks.org/wiki/media%3ASVG_CSSfocus01.svg

17 Multimedia

17.1 Multimedia

In SVG tiny 1.2 werden einige neue Elemente, Attribute und Eigenschaften definiert, welche der Präsentation von Multimediadateien dienen. Die Elemente sind den Medienelementen von SMIL ähnlich und haben einen eigenen Zeitablauf und bieten eine ähnliche Kontrolle des Zeitablaufes wie Animationselemente. Weil externe zeitabhängige Dateien referenziert werden, gibt es ferner eine Möglichkeit der Synchronisation verschiedener Multimediainhalte.

Für Dateien, die nur hörbare Information enthalten, Audiodateien mit einfachem Zeitablauf, gibt es das Element ***audio***. Für Dateien mit visuellen Inhalten und optional zusätzlich mit hörbaren Inhalten und einem einfachen Zeitablauf, umgangssprachlich Videos, gibt es das Element ***video***. Für beides kann mit der Eigenschaft *audio-level* Einfluss mit die Lautstärke genommen werden. Für komplette externe SVG-Dateien, optional mit zeitabhängigem Inhalt, gibt es das Element ***animation***.

Das Konzept unterscheidet sich damit deutlich von dem von SMIL, wo gleichnamige Elemente sich zwar in der semantischen Bedeutung unterscheiden, aber nicht in den jeweils damit präsentierbaren Medien. Als für mobile Geräte optimierte Version von SVG ist nachvollziehbar, dass der Funktionsumfang eines Elementes stärker eingegrenzt ist.

In SVG tiny 1.2 sind für ***audio*** oder ***video*** keine Formate angegeben, welche vom Darstellungsprogramm interpretiert werden müssen. Auch für gleichnamige Elemente im Arbeitsentwurf für 'HTML5' ist das bislang (Mitte 2012) nicht gelungen. Dies liegt daran, dass viele Formate Lizenzbedingungen unterliegen, also weder von allen interessierten Autoren frei genutzt werden können, noch in Darstellungsprogrammen frei implementiert werden können. Bei Formaten wie OGG (Containerformat, oft mit Audio-Codec Vorbis, FLAC oder Speex und Video-Codec Theora), die tatsächlich als patent- oder lizenzfrei gelten, verkompliziert sich die Lage dennoch, weil von einigen Firmen befürchtet wird, dass im amerikanischen Raum im Nachhinein bei hinreichend erfolgreichen freien Formaten sogenannte U-Boot-Patente auftauchen können, die also entweder nachträglich von Unbeteiligten eingereicht und unberechtigt anerkannt werden, oder wo alte Patente so uminterpretiert werden, dass es ausreicht, um damit gegen implementierende Firmen zu prozessieren, um damit Geld zu verdienen. Ob die Befürchtungen berechtigt sind, ist strittig und kann vom konkreten Format abhängen.

Auch wenn ein Autor mit dem Erwerb eines Produktes wie etwa einer digitalen Kamera die Lizenz erworben hat, damit Dokumente zu erzeugen und auch sichtbar zu machen, sind die Lizenzen oft eingeschränkt auf den Gebrauch mit dem konkreten Produkt oder auf den persönlichen Gebrauch, so dass daraus nicht automatisch eine lizenzfreie Veröffentlichung und Ansicht mit anderen Programmen im internet gefolgert werden kann. Nutzer von freien Betriebssystemen werden nicht immer die Möglichkeit oder den Willen haben, für lizenzierte Programme zu bezahlen, so dass die Darstellung lizenzbehafteter Medien letztlich nicht allgemein sicherzustellen ist. Insofern ist die Formatwahl für den Autor nicht trivial und aufgrund der Probleme umso wichtiger.

Wenn der Autor Dokumente in verschiedenen Formaten angeben kann, so kann dies in SVG mit bedingter Verarbeitung realisiert werden. Mittels *requiredFormats* können verschiedene Versionen angeboten werden, um die Wahrscheinlichkeit einer Anzeige zu erhöhen.

17.2 Zeitachsen für Medien und für das Dokument

Medieninhalte und das SVG-Dokument definieren einen eigenen Zeitablauf oder eine eigene Zeitachse. Referenzierte Medieninhalte und die Medienelemente im SVG-Dokumente sind daher zeitlich aufeinander abzustimmen.

Weil das Laden der Medien Zeit benötigt, kann mit dem Attribut *timelineBegin* vom Autor festgelegt werden, ob der Zeitablauf des SVG-Dokumentes bereits beginnt, wenn die Startmarkierung des ***svg***-Hauptelementes geladen ist ('onStart') oder erst, wenn das gesamte Dokument geladen ist ('onload').

Daneben kann auch mit dem Attribut *externalResourcesRequired* der Zeitablauf eines Fragmentes verzögert werden, bis externe Ressourcen verfügbar sind. Um Warte- und Ladezeiten zu optimieren oder zu verkürzen, steht ferner das Element ***prefetch*** zur Verfügung.

Die Bedeutung von Verfügbarkeit hängt davon ab, auf welchem Wege oder mit welchem Protokoll das externe Medium geladen wird. Im Falle des Protokolles HTTP jedenfalls ist vom Laden des kompletten Dokumentes auszugehen. Bei sich progressiv aufbauenden Formaten mag es auch reichen, sobald die Information für eine erste Darstellung verfügbar ist. Bei einem Datenstromprotokoll (englisches Stichwort: streaming, Protokolle wie RTSP/RTP) mag es auch ausreichen, wenn so viel von einem externen Medium geladen ist, dass aufgrund der bisherigen Datenrate von einer störungsfreien Darstellung des gesamten Mediums ausgegangen werden kann.

Beim Abspielen der Medien kann es zudem zu Verzögerungen und Störungen kommen. Mit von SMIL übernommenen Attributen können daher zusammengehörige Inhalte miteinander und zum SVG-Dokument synchronisiert werden. Zudem kann die Genauigkeit der Synchronisation vom Autor festgelegt werden.

17.3 Präsentation von Audio und Video beeinflussen

Autoren können im Bedarfsfalle die Präsentation von Audio- und Video-Kanälen getrennt beeinflussen.

Wird nichts besonderes angegeben, wird beides präsentiert. Wird die Lautstärke mit dem Attribut *audio-level* auf 0 gesetzt, so ist effektiv der Audiokanal abgestellt. Wird die Sichtbarkeit *visibility* auf hidden gesetzt, ist effektiv der Videokanal abgestellt, während der Audiokanal weiter wiedergegeben wird. Die komplette Präsentation oder Darstellung kann mit der Eigenschaft *display* abgestellt werden. Die Darstellung kann ferner auch über die Attribute zur Steuerung des Zeitablaufes unterbunden werden.

17.4 Element *audio*

Squiggle (Batik)	-
Opera (Presto)	9.5 (Zeitablauf teilweise, Audio nur mit speziellen Testversionen und Formaten); 10.60 (Containerformat OGG mit Vorbis oder WAVE mit PCM)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Hörbare Medien können mit dem Element ***audio*** eingebettet werden. Es gibt keine visuelle Präsentation des Elementes. Die mit dem Attribut *href* von XLink referenzierte Datei muss ein Audioformat sein, darf also nur hörbaren Inhalt enthalten.

Neben den im folgenden angegebenen Attributen sind auch die zur Synchronisation anwendbar, die in einem der folgenden Abschnitte erläutert werden. Der Zeitablauf wird mit den Attributen zur Kontrolle des Zeitablaufes beeinflusst, welche bereits im Kapitel über Animation erläutert wurden. Dies sind: *begin*, *end*, *dur*, *min*, *max*, *restart*, *repeatCount*, *repeatDur* und *fill*. Allerdings hat *fill* keinen hörbaren Effekt.

Sofern das referenzierte Dokument zwar relevante Information, aber selbst keine Textalternative enthält, ist es dringend aus Gründen der Zugänglichkeit geboten, die Textalternative innerhalb des SVG-Dokumentes anzugeben. Dazu eignen sich zum einen die Elemente ***title***, ***desc*** und ***metadata***, zum anderen auch die Attribute *title* und *role* von XLink.

17.4.1 Attribute von XLink, insbesondere *href*

Mit dem Attribut *href* von XLink wird auf die Audiodatei verwiesen, welche eingebunden werden soll. Die Voreinstellung ist ein leeres Attribut. Dieses und ungültige Angaben führen lediglich dazu, dass keine hörbare Präsentation stattfindet.

Das Attribut ist animierbar.

Bei einer Animation des Attributes oder sonst einer Änderung und wenn der Zeitablauf kontrolliert werden kann, wird der Zeitablauf nur erneut gestartet, wenn das Attribut *syncBehavior* auf 'independent' steht. Wenn der Zeitablauf nicht kontrollierbar ist, so ist der Zeitablauf unbeeinflusst von der Änderung.

Mit *href* sind auch die anderen Attribute von XLink verfügbar. Da es allerdings keine visuelle Präsentation des Elementes gibt, kann der Zugang zu den Attributen erschwert sein.

17.4.2 Attribut *type*

Das Attribut *type* erlaubt es dem Autor, einen Hinweis zu geben, in welchem Dateiformat das referenzierte Dokument vorliegt. Der Wert ist ein internet-Medientyp, zum Beispiel 'audio/oga'.

Sollte das Darstellungsprogramm 'audio/oga' nicht interpretieren können, kann es den Hinweis dazu benutzen, die Datei erst gar nicht anzufordern.

Das Attribut ist animierbar, weil auch *href* von XLink animierbar ist und sich somit während der Animation das Dateiformat ändern kann. Der Autor ist dann allerdings selbst dafür verantwortlich, dass die Relation zwischen den beiden Animationen ein sinnvoller und zutreffender Hinweis ist.

17.4.3 Beispiele *audio*

Nutzer rein dekorative Hintergrundmusik aufdrängen:

```
<audio xlink:href="gedudel.oga" type="audio/ogg"
       dur="media" repeatDur="indefinite" />
```

Die Musik beginnt automatisch beim Beginn des Dokumentes und wiederholt sich bis zum Ende. Die einfache Dauer ergibt sich automatisch aus der zeitlichen Länge des Mediums. Viele Nutzer mögen keine überraschende Musikberieselung ohne eigene Interaktion, daher wird es zumeist sinnvoll sein, einen Knopf zum Starten anzugeben.

Besser also die Audiodatei mit Knopf starten:

```
<g id="Knopf" role="button">
  <audio xlink:href="musike.oga"
       xlink:title="Martha Müller singt auf ihrem achzigsten Geburtstag,
                    begleitet von ihrem Enkel Klaus."
       type="audio/ogg"
       dur="media" begin="Knopf.click" />
  <rect width="150" height="50" x="0" y="0" rx="5"
      fill="#ffc" stroke="#009" stroke-width="5"/>
    <text x="75" y="30" text-anchor="middle" font-size="20"
      fill="#00f">Oma singt!</text>
</g>
```

Bedingte Verarbeitung mit verschiedenen Formaten:

```
<g stroke="blue" stroke-width="10"
   xmlns:xlink="http://www.w3.org/1999/xlink">
  <title>'Mein Hut, der hat drei Ecken' gesungen von Siggi Haase</title>
  <desc>
  Siggi Haase singt,
  begleitet auf der Mundharmonika von seiner Freundin Gunilla Schneider.
  Datum der Aufnahme: 2010-04-15
  Lied: Mein Hut, der hat drei Ecken
  Text:
  Mein Hut, der hat drei Ecken,
  drei Ecken hat mein Hut.
  Und hätt er nicht drei Ecken,
  so wär er nicht mein Hut.
  </desc>
  <switch>
    <g requiredFormats="audio/ogg">
      <audio xlink:href="Musik.oga" type="audio/ogg"
        begin="ogg1.click" end="ogg2.click; indefinite"
        repeatDur="indefinite" dur="media" />
      <circle xml:id="ogg1" fill="#ccf"
        r="80" cx="100" cy="100" role="button">
        <title>Startknopf</title>
      </circle>
      <circle xml:id="ogg2" fill="#fcc"
        r="80" cx="300" cy="100" role="button">
```

```
      <title>Endknopf</title>
    </circle>
  </g>
  <g requiredFormats="audio/mpeg">
    <audio xlink:href="Musik.mp3" type="audio/mpeg"
      begin="mp31.click" end="mp32.click; indefinite"
      repeatDur="indefinite" dur="media" />
    <circle xml:id="mp31" fill="#ccf"
      r="80" cx="100" cy="100" role="button">
      <title>Startknopf</title>
    </circle>
    <circle xml:id="mp32" fill="#fcc"
      r="80" cx="300" cy="100" role="button">
      <title>Endknopf</title>
    </circle>
  </g>
  <g requiredFormats="audio/x-wav">
    <audio xlink:href="Musik.wav" type="audio/x-wav"
      begin="wav1.click" end="wav2.click; indefinite"
      repeatDur="indefinite" dur="media" />
    <circle xml:id="wav1" fill="#ccf"
      r="80" cx="100" cy="100" role="button">
      <title>Startknopf</title>
    </circle>
    <circle xml:id="wav2" fill="#fcc"
      r="80" cx="300" cy="100" role="button">
      <title>Endknopf</title>
    </circle>
  </g>
  <g><!-- Alternativer Inhalt, wenn gar nichts läuft -->
     <text x="10" y="30" fill="#800">
           Keines der angebotenen Formate ist automatisch abspielbar.
     </text><text x="10" y="100" fill="#888">
           Manuell eines versuchen:
     </text>
    <a xlink:href="Musik.oga" xlink:show="new">
      <text x="10" y="140" fill="#00f">audio/ogg</text>
    </a>
    <a xlink:href="Musik.mp3" xlink:show="new">
      <text x="10" y="160" fill="#00f">audio/mp3</text>
    </a>
    <a xlink:href="Musik.wav" xlink:show="new">
      <text x="10" y="180" fill="#00f">audio/x-wav</text>
    </a>
  </g>
 </switch>
</g>
```

Bei den folgenden Beispielen sind gleich Audio- beziehungsweise Videodateien in das SVG-Dokument eingebettet. Im Rahmen dieses Buches ist das recht nützlich. Bei realen Anwendungen wird es zumeist effektiver sein, die Multimedia-Dokument getrennt vorzuhalten und diese nur zu referenzieren.

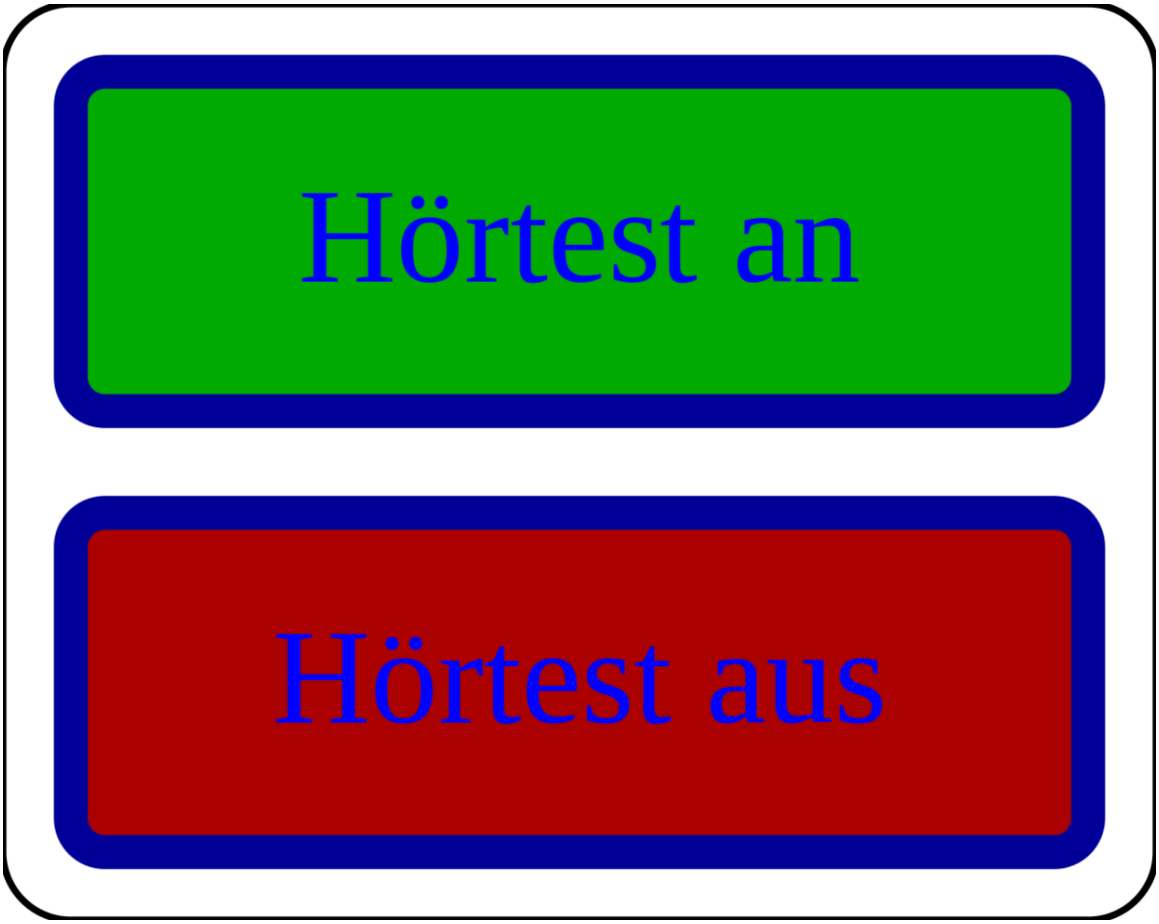


Abb. 287 Hörtest

Hörtest[1]

Einfacher Hörtest.

Voraussetzung ist die Präsentation des Formates audio/ogg (OGG+Vorbis).

Die Frequenz fällt von 20kHz auf 20Hz in 10s.

Der Ton begint mit dem Startknopf und endet mit dem Endknopf oder nach 10s.

Die Hintergrundfarbe wird über die aktive Dauer der Audioausgabe als visuelles Kennzeichen animiert.

1 http://de.wikibooks.org/wiki/media%3ASVGaudio01.svg

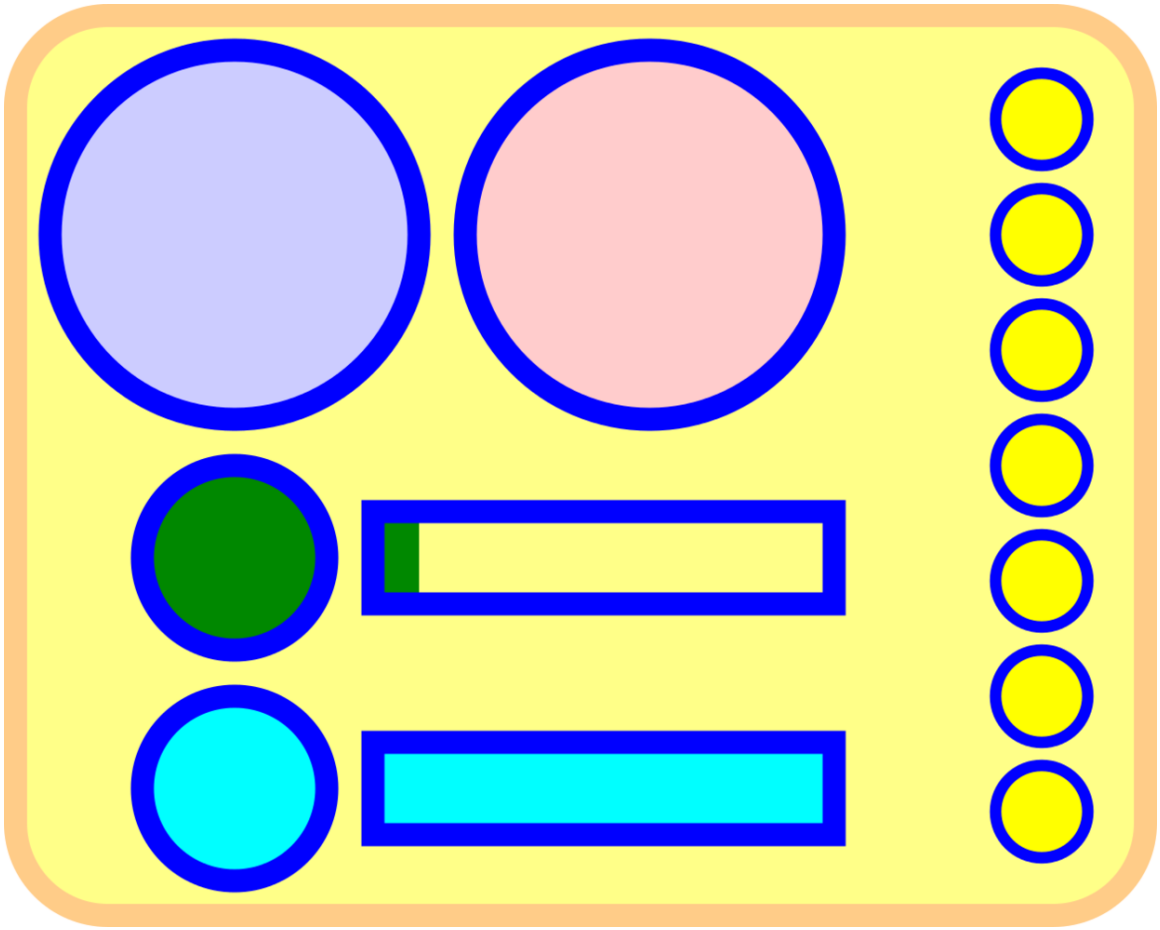


Abb. 288 Prototyp einer Musiktruhe

Prototyp einer Musiktruhe[2]

Eine Auswahl verschiedener Geräusche.

Voraussetzung ist die Präsentation des Formates audio/ogg (OGG+Vorbis).

Es gibt ein Anfangsgeräusch und sieben weitere Alternativen, auswählbar mit den gelben Knöpfen rechts.

Die großen Knöpfe oben dienen zum An- und Abschalten.

Die Konstruktion darunter dient dazu, die Lautstärke zu ändern.

2 `http://de.wikibooks.org/wiki/media%3ASVGaudio02.svg`

Starten mit den runden Knöpfen, stoppen mit den rechteckigen Indikatoren.

Die Titel der Knöpfe geben jeweils Auskunft über die Funktion.

Die Hintergrundfarbe wird über die aktive Dauer der Audioausgabe als visuelles Kennzeichen animiert.

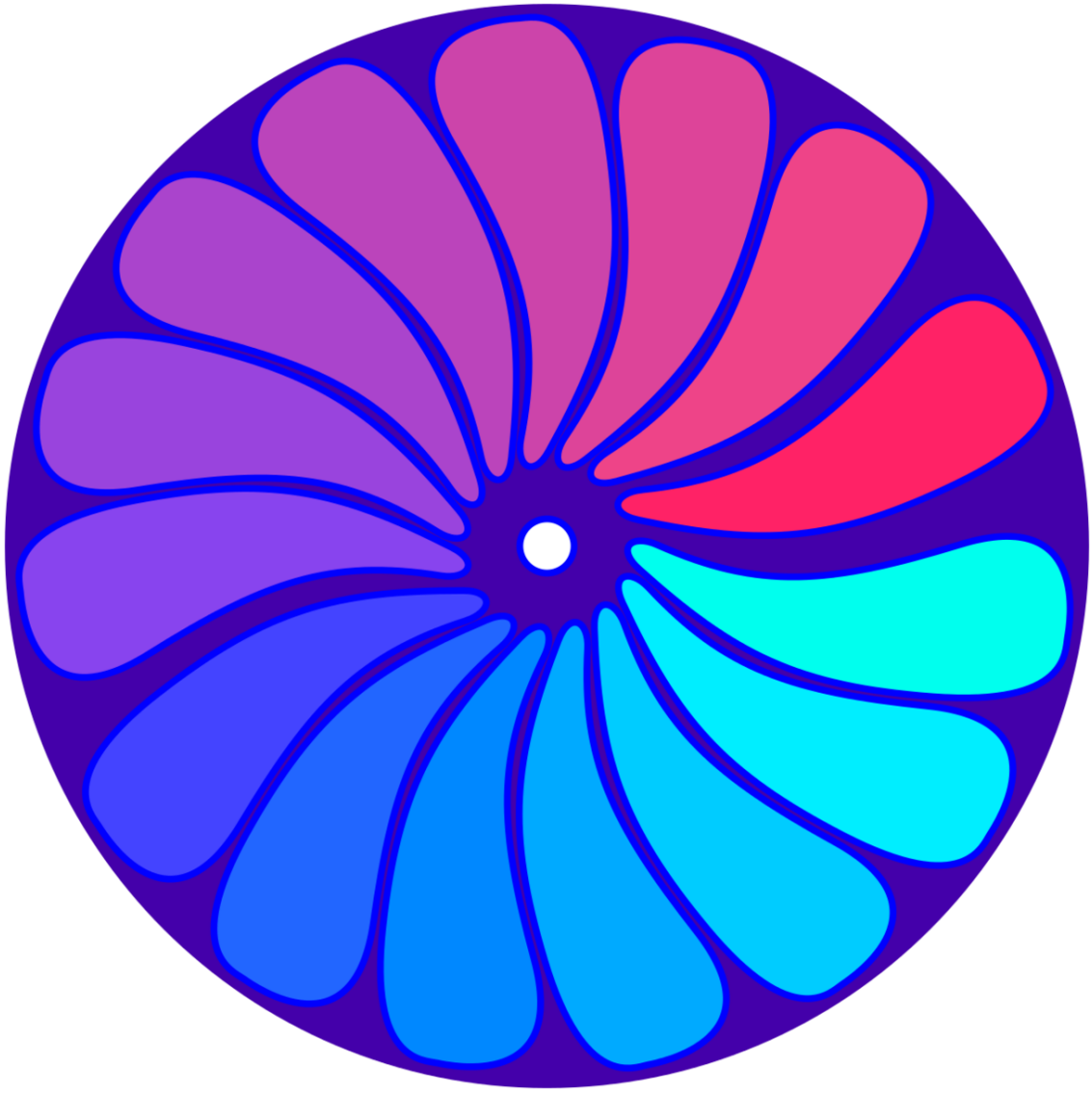


Abb. 289 Einfaches Musikinstrument

Einfaches Musikinstrument[3]

3 http://de.wikibooks.org/wiki/media%3ASVG_Musikinstrument01.svg

Ein rudimentäres Musikinstrument, welches mit Grundtönen im Format audio/ogg (OGG+Vorbis) arbeitet.

Tasten für zwei Oktaven der C-Dur-Tonleiter sind verfügbar.

Zur Aktivierung jeweils mit der Maus drücken, loslassen beendet den Ton.

Tonlänge ist also jeweils Länge des Tastendrucks.

Der Ton ist jeweils als Titel angegeben.

Ähnlich kann die Lautstärke mit dem kleinen mittleren Kreis eingestellt werden.

Lautstärke ändert sich von leise nach laut in 5s, symbolisiert mit einer Farbänderung von schwarz nach weiß.

Loslassen fixiert die Lautstärke auf dem aktuellen Wert.

Ein permanenter Ton etwa zur Lautstärkeeinstellung kann erreicht werden, wenn eine Taste gedrückt wird, die Maus aber erst außerhalb der Taste wieder losgelassen wird.

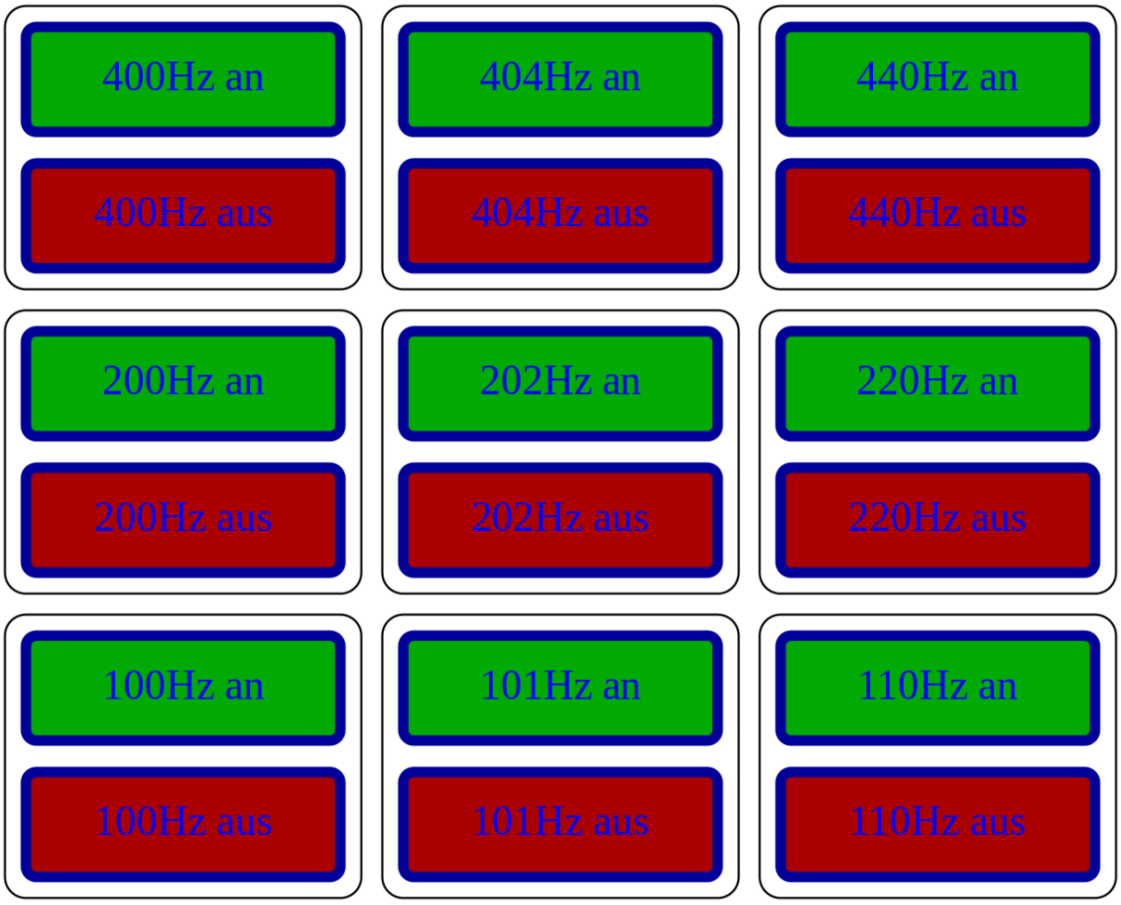


Abb. 290 Schwebungen

Schwebungen[4]

Neun verschiedene Töne können miteinander kombiniert werden, um Schwebungen hervorzurufen.

Voraussetzung ist die Präsentation des Formates audio/ogg (OGG+Vorbis).

Der Ton beginnt jeweils mit dem Startknopf und endet mit dem Endknopf.

Die Hintergrundfarbe wird über die aktive Dauer der Audioausgabe als visuelles Kennzeichen animiert.

Bekannte Mängel: eventuelle zusätzliche Knackgeräusche beim Beginn oder Ende einer Audiodatei, die allerdings nicht an der Audiodatei selbst liegen.

4 `http://de.wikibooks.org/wiki/media%3ASVGaudio03.svg`

17.5 Element *video*

Squiggle (Batik)	-
Opera (Presto)	9.5 (Zeitablauf teilweise, Video nur mit speziellen Testversionen); 10.60 (Containerformat OGG mit Theora und Vorbis oder Containerformat WebM mit VP8 und Vorbis)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Videos können mit dem Element ***video*** eingebettet werden.

Die mit dem Attribut *href* von XLink referenzierte Datei muss ein Videoformat sein und kann auch einen Audiokanal enthalten.

Neben den im folgenden angegebenen Attributen sind auch die zur Synchronisation anwendbar, die in einem der folgenden Abschnitte erläutert werden. Der Zeitablauf wird mit den Attributen zur Kontrolle des Zeitablaufes beeinflusst, welche bereits im Kapitel über Animation erläutert wurden. Dies sind: *begin*, *end*, *dur*, *min*, *max*, *restart*, *repeatCount*, *repeatDur* und *fill*.

video definiert einen eigenen Darstellungsbereich, der Wert des Attributes *viewBox* wird aus den Angaben der Videodatei impliziert. Entsprechend sind auch die Eigenschaften *viewport-fill* und *viewport-fill-opacity* anwendbar.

Sofern das referenzierte Dokument zwar relevante Information, aber selbst keine Textalternative enthält, ist es dringend aus Gründen der Zugänglichkeit geboten, die Textalternative innerhalb des SVG-Dokumentes anzugeben. Dazu eignen sich zum einen die Elemente ***title***, ***desc*** und ***metadata***, zum anderen auch die Attribute *title* und *role* von XLink.

17.5.1 Attribute von XLink, insbesondere *href*

Mit dem Attribut *href* von XLink wird auf die Videodatei verwiesen, welche eingebunden werden soll. Die Voreinstellung ist ein leeres Attribut. Dieses und ungültige Angaben führen lediglich dazu, dass keine Präsentation stattfindet.

Das Attribut ist animierbar.

Bei einer Animation des Attributes oder sonst einer Änderung und wenn der Zeitablauf kontrolliert werden kann, wird der Zeitablauf nur erneut gestartet, wenn das Attribut *syncBehavior* auf 'independent' steht. Wenn der Zeitablauf nicht kontrollierbar ist, so ist der Zeitablauf unbeeinflusst von der Änderung.

Mit *href* sind auch die anderen Attribute von XLink verfügbar.

17.5.2 Attribut *type*

Das Attribut *type* erlaubt es dem Autor, einen Hinweis zu geben, in welchem Dateiformat das referenzierte Dokument vorliegt. Der Wert ist ein internet-Medientyp, zum Beispiel 'video/ogv'. Sollte das Darstellungsprogramm 'video/ogv' nicht interpretieren können, kann es den Hinweis dazu benutzen, die Datei erst gar nicht anzufordern.

Das Attribut ist animierbar, weil auch *href* von XLink animierbar ist und sich somit während der Animation das Dateiformat ändern kann. Der Autor ist dann allerdings selbst dafür verantwortlich, dass die Relation zwischen den beiden Animationen ein sinnvoller und zutreffender Hinweis ist.

17.5.3 Attribute *x* und *y*

Der Wert ist jeweils eine Koordinate, also wo das Video innerhalb des jeweiligen Koordinatensystems dargestellt werden soll, *x* und *y* geben dann die linke obere Ecke an. Sofern nicht angegeben, wird jeweils 0 angenommen.

Sofern der Wert von *transformBehavior* auf einen anderen Wert als 'geometric' gesetzt ist, ist der angegebene Punkt allerdings nicht die linke obere Ecke, sondern der Befestigungspunkt für Transformationen. Der Befestigungspunkt ergibt sich aus der Transformation des angegebenen Punktes auf den nächstgelegenen Gerätepixel. Dies stellt dann den Mittelpunkt des Videos dar.

Die Attribute sind animierbar.

17.5.4 Attribute *width* und *height*

width und *height* geben die eigentliche Breite und Höhe des Darstellungsbereiches für das Video an. Der Wert ist jeweils eine Länge größer oder gleich 0. Das Video wird dann entsprechend skaliert. Bei 0 wird das Video nicht angezeigt. Negative Werte werden nicht unterstützt. Dies bedeutet, dass das Attribut mit dem unzulässigen Wert ignoriert wird.

width und *height* sind animierbar. Sofern nicht angegeben, wird als Wert 0 angenommen.

17.5.5 Attribut *preserveAspectRatio*

Wenn das Seitenverhältnis des Videos nicht zu *width* und *height* passt, kann mit *preserveAspectRatio* angegeben werden, wie es in den Anzeigebereich einzupassen ist.

Die möglichen Werte sind bereits entsprechend für das Element ***svg*** beschrieben.

Das Attribut ist animierbar.

17.5.6 Attribut *transformBehavior*

Der Autor kann entscheiden, wie Transformationen auf ein Video wirken. Insbesondere bei mobilen Geräten mit geringem Leistungsumfang kann die ansonsten übliche Transformation wie für sonstige graphische Elemente zu rechenaufwendig sein. Für Inhalte, welche insbesondere für solche Geräte

gedacht ist, kann es sinnvoll sein, vereinfachende Transformationsregeln anzuwenden. Dies geschieht mit dem Attribut *transformBehavior*.

Ist das Attribut angegeben und es ein anderer Wert als 'geometric' notiert, so geben *x* und *y* den Befestigungspunkt des Videos an, im Wesentlichen die Mitte. Bei Transformationen wird nur dieser transformiert. Form und Größe des Videos bleiben unverändert. Die Größe ist die in Gerätepixeln.

Mögliche Werte sind:

geometric

Das Video soll wie ein Rechteck in lokalen Koordinatensystem behandelt werden. Das ist die Voreinstellung.

pinned

Das Video wird nicht gedreht

pinned90

Das Video wird um 90 Grad um den Befestigungspunkt gedreht

pinned180

Das Video wird um 180 Grad um den Befestigungspunkt gedreht

pinned270

Das Video wird um 270 Grad um den Befestigungspunkt gedreht

Das Attribut ist nicht animierbar.

17.5.7 Attribut *overlay*

Neben einer Einschränkung hinsichtlich Transformationen kann auch eine Einschränkung hinsichtlich der Komposition mit anderen Elementen hilfreich sein. Geräte mit stärker begrenzter Leistungen können eventuell das Video nur ganz oben in der Zeichenreihenfolge darstellen. Ob das der Fall ist, kann mit der Eigenschaftszeichenkette 'http://www.w3.org/Graphics/SVG/feature/1.2/#ComposedVideo'[5] und bedingter Verarbeitung berücksichtigt werden.

Für Darstellungsprogramme, welche keine solche Einschränkung haben, kann der Autor das Verhalten explizit mit dem Attribut *overlay* angeben, um die Effizienz zu erhöhen.

Möglich sind die Werte 'top' oder 'none'.

Bei 'top' wird das Video in der Zeichenreihenfolge immer ganz oben platziert. Bei 'none' folgt die Darstellung der für andere Elemente gemäß dem Malermodell. Letzteres ist die Voreinstellung. Wenn es mehrere sich überlappende Videos mit dem Wert 'top' gibt, so folgt deren Darstellungsreihenfolge der Reihenfolge im Quelltext.

Das Attribut ist nicht animierbar.

5 `http://www.w3.org/Graphics/SVG/feature/1.2/#ComposedVideo'`

17.5.8 Attribut *initialVisibility*

Es gibt auch die Möglichkeit zu entscheiden, ob das Medienobjekt vor dem Beginn seiner aktien Dauer dargestellt werden soll. Ist dies der Fall, soll das erste Bild (der erste Rahmen, englisch: frame) dargestellt werden, bis die aktive Dauer beginnt.

initialVisibility hat die möglichen Werte 'whenStarted' und 'always'.

'whenStarted' ist die Voreinstellung. Das Video ist dann vor der aktiven Dauer nicht sichtbar.

'always' bedeutet, dass das Video auch vor der aktiven Dauer sichtbar ist.

Das Attribut ist nicht animierbar.

17.5.9 Beispiele *video*

Video sofort zu Dokumentbeginn starten:

```
<video xlink:href="Kaminfeuer.ogv" type="video/ogg"
       x="50" y="50" width="320" height="240"
       dur="media" repeatDur="indefinite">
       <title>Kaminfeuer</title>
       <desc>
        In einem steinernen Kamin entzündet sich aus dem Dunklen heraus
        ein Feuer aus Buchenholz ohne ersichtlichen Grund und brennt
        über Stunden langsam herunter und erlischt endlich.
        Der Vorgang wiederholt sich.
       </desc>
</video>
```

Hilfreicher ist es natürlich, das Video mit dem Knopf zu starten:

```
<video xlink:href="EinHundGingInDieKueche.ogv"
       xlink:title="Martha Müller singt
        auf ihrem achzigsten Geburtstag am 2008-03-21
        'Ein Hund ging in die Küche',
        auf dem Klavier begleitet von ihrem Enkel Klaus."
       type="video/ogg"
       initialVisibility="always"
       x="50" y="50" width="320" height="240"
       dur="media" begin="Knopf.click" />
<g id="Knopf" role="button">
  <title>Knopf zum Starten des Videos</title>
  <rect width="320" height="50" x="50" y="300" rx="5"
      fill="#ffc" stroke="#009" stroke-width="5"/>
  <text x="210" y="330" text-anchor="middle" font-size="20"
      fill="#00f">Oma singt!</text>
</g>
```

Die bedingte Verarbeitung funktioniert analog zu dem Beispiel zum Element ***audio***.

Abb. 291 Plasmakugel

Plasmakugel[6]

Ein Video mit einer Plasmakugel kann mit den angegebenen Knöpfen gestartet und gestoppt werden.

Davor und danach sind das erste, beziehungsweise das letzte (aktive) Bild des Videos sichtbar.

Voraussetzung ist die Präsentation des Formates video/ogg (OGG+Theora+Vorbis).

Bei der dargestellten Plasmakugel handelt es sich um eine Glaskugel, die mit einem speziellen Gasgemisch gefüllt ist. Durch eine kleinere Kugel in der Mitte wird durch eine hochfrequente Wechselspannung ein Plasma erzeugt, in welchem sich aufgrund der Kugelsymmetrie zufällig anordnende Entladungsschläuche zur Glaskugel entstehen, welche leuchten, nahe der kleinen Kugel und der Glaskugel eher rot-orange, in der Mitte eher bläulich-weiß. Die Entladungsschläuche

6 `http://de.wikibooks.org/wiki/media%3ASVGvideo01.svg`

bestehen gewöhnlich mehrere Sekunden bis Minuten lang und ändern ihre Form und Ausrichtung zufällig, jedoch untereinander bedingt durch die in den Schläuchen befindlichen Ladungen nicht komplett unkorreliert. Anzahl, Lebensdauer und Verhalten der Entladungsschläuche hängen von den Eigenschaften der angelegten Wechselspannung und vom Gasgemisch ab.

Das Video selbst dauert nur etwa eine Sekunde und wiederholt sich lediglich.

17.6 Element *animation*

Squiggle (Batik)	-
Opera (Presto)	9.5 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Während in SVG 1.1 andere SVG-Dokumente mit dem Element ***image*** eingettet werden können, bei animierten SVGs aber keine definierte Synchronisation des Zeitablaufes gegeben ist, ist in SVG tiny das Element ***image*** für Pixelgraphik reserviert. In SVG tiny 1.2 dient das Element ***animation*** dazu, SVG-Dokumente, auch animierte einzubetten, wobei der Zeitablauf und die Synchronisation genau wie für ***video*** und ***audio*** definiert sind.

Die mit dem Attribut *href* von XLink referenzierte Datei muss ein SVG-Dokument sein.

Neben den im folgenden angegebenen Attributen sind auch die zur Synchronisation anwendbar, die in einem der folgenden Abschnitte erläutert werden. Der Zeitablauf wird mit den Attributen zur Kontrolle des Zeitablaufes beeinflusst, welche bereits im Kapitel über Animation erläutert wurden. Dies sind: *begin*, *end*, *dur*, *min*, *max*, *restart*, *repeatCount*, *repeatDur* und *fill*.

animation definiert einen eigenen Darstellungsbereich, der Wert des Attributes *viewBox* wird aus den Angaben der des referenzierten SVG-Dokumentes impliziert. Entsprechend sind auch die Eigenschaften *viewport-fill* und *viewport-fill-opacity* anwendbar.

17.6.1 Attribute von XLink, insbesondere *href*

Mit dem Attribut *href* von XLink wird auf die SVG-Datei verwiesen, welche eingebunden werden soll. Die Voreinstellung ist ein leeres Attribut. Dieses und ungültige Angaben führen lediglich dazu, dass keine Präsentation stattfindet.

Das Attribut ist animierbar.

Bei einer Animation des Attributes oder sonst einer Änderung und wenn der Zeitablauf kontrolliert werden kann, wird der Zeitablauf nur erneut gestartet, wenn das Attribut *syncBehavior* auf 'independent' steht. Wenn der Zeitablauf nicht kontrollierbar ist, so ist der Zeitablauf unbeeinflusst von der Änderung.

Mit *href* sind auch die anderen Attribute von XLink verfügbar.

17.6.2 Attribute *x* und *y*

Der Wert ist jeweils eine Koordinate, also wo das SVG innerhalb des jeweiligen Koordinatensystems dargestellt werden soll, *x* und *y* geben dann die linke obere Ecke an. Sofern nicht angegeben, wird jeweils 0 angenommen.

Die Attribute sind animierbar.

17.6.3 Attribute *width* und *height*

width und *height* geben die eigentliche Breite und Höhe des Darstellungsbereiches für das SVG an. Der Wert ist jeweils eine Länge größer oder gleich 0. Das SVG wird dann entsprechend skaliert. Bei 0 wird das SVG nicht angezeigt. Negative Werte werden nicht unterstützt. Dies bedeutet, dass das Attribut mit dem unzulässigen Wert ignoriert wird.

width und *height* sind animierbar. Sofern nicht angegeben, wird als Wert 0 angenommen.

17.6.4 Attribut *preserveAspectRatio*

Wenn das Seitenverhältnis des SVGs nicht zu *width* und *height* passt, kann mit *preserveAspectRatio* angegeben werden, wie es in den Anzeigebereich einzupassen ist.

Die möglichen Werte sind bereits entsprechend für das Element ***svg*** beschrieben.

Das Attribut ist animierbar.

17.6.5 Attribut *initialVisibility*

Es gibt auch die Möglichkeit zu entscheiden, ob das SVG vor dem Beginn seiner aktien Dauer dargestellt werden soll. Ist dies der Fall, soll der Zeitpunkt 0 im Zeitablauf des referenzierten Dokumentes dargestellt werden, bis die aktive Dauer beginnt.

initialVisibility hat die möglichen Werte 'whenStarted' und 'always'.

'whenStarted' ist die Voreinstellung. Das SVG ist dann vor der aktiven Dauer nicht sichtbar.

'always' bedeutet, dass das SVG auch vor der aktiven Dauer sichtbar ist.

Das Attribut ist nicht animierbar.

17.6.6 Beispiele *animation*

SVG sofort zu Dokumentbeginn starten:

```
<animation xlink:href="Beispiel.svg"
       x="50" y="50" width="600" height="400"
       dur="indefinite" />
```

Es kann auch sinnvoll sein, das SVG mit dem Knopf zu starten:

```
<animation xlink:href="Beispiel.svg"
       xlink:title="Vereinfachtes Modell des Sonnensystems"
       initialVisibility="always"
       x="50" y="50" width="600" height="400"
       dur="indefinite" begin="Knopf.click" />
<g id="Knopf" role="button">
  <title>Knopf zum Starten des Videos</title>
  <rect width="600" height="50" x="50" y="460" rx="5"
      fill="#ffc" stroke="#009" stroke-width="5"/>
  <text x="350" y="490" text-anchor="middle" font-size="20"
      fill="#00f">Sonnensystem starten</text>
</g>
```

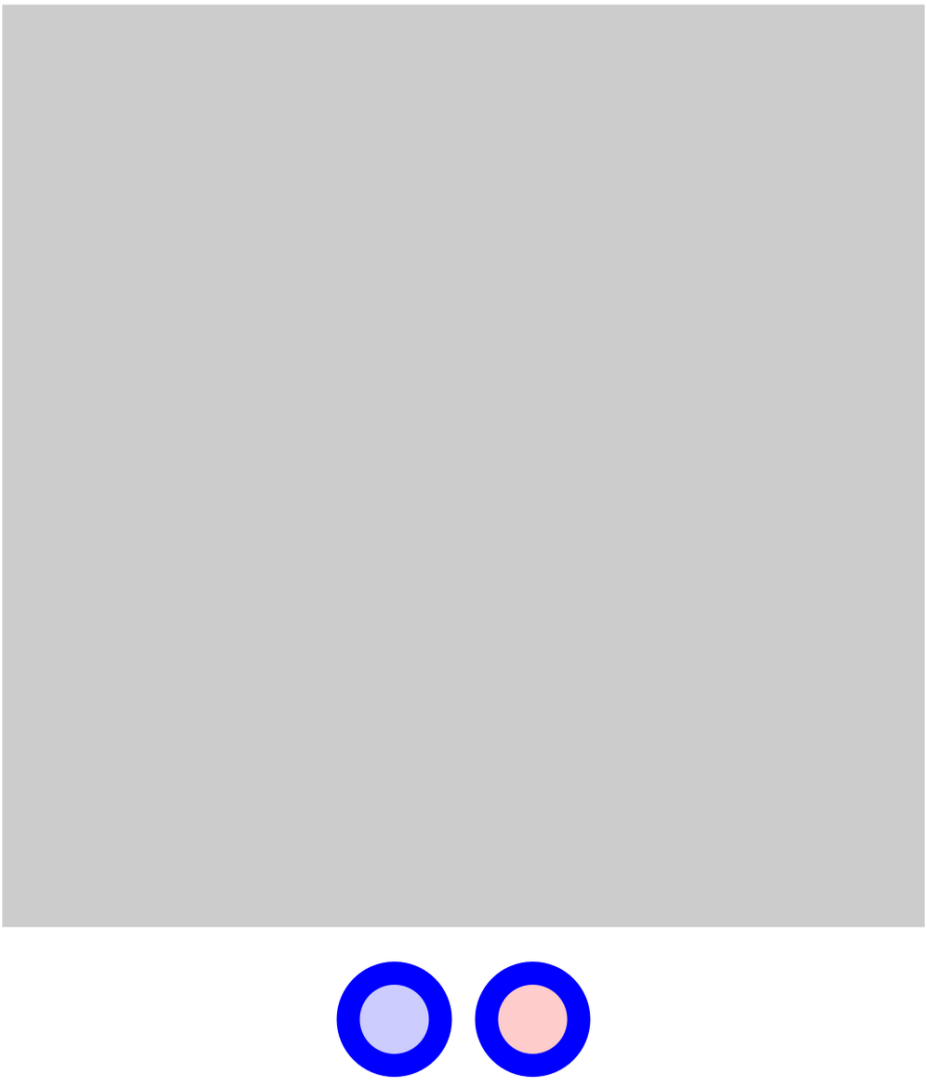


Abb. 292 Pulsierender Kreis

Pulsierender Kreis[7]

Einfache Animationen eines pulsierenden Kreises werden referenziert. Start der Animation mit dem blauen Knopf, stoppen mit dem roten. Der blaue Kreis verdeckt immer einen roten, sonst stimmt die Synchronisation nicht.

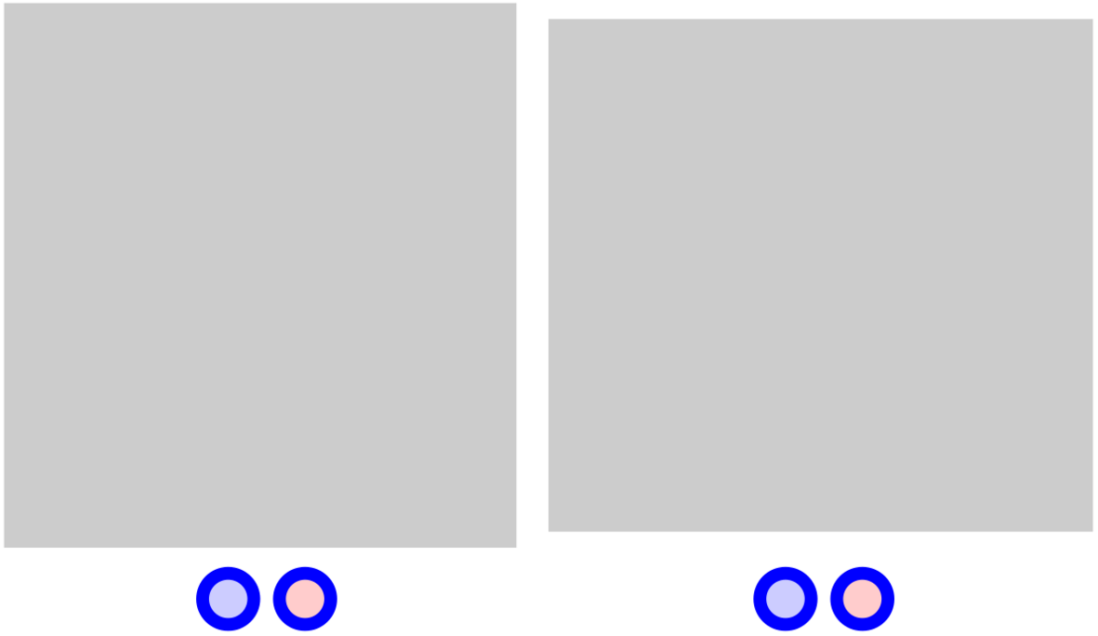


Abb. 293 Unabhängiger Zeitablauf

Unabhängiger Zeitablauf[8]

Zwei (identische) SVG-Dokumente werden eingebettet. Start der Anzeige des jeweiligen Dokumentes mit dem blauen Knopf, stoppen und entfernen mit dem roten. Beide haben einen unabhängigen Zeitablauf. Nach dem jeweiligen Start kann in dem jeweiligen Dokument die Bewegung eines jeden Rechtecks durch Anklickern gestartet werden. Wiederholtes Anklickern wiederholt den bisherigen Zeitablauf des eingebetteten Dokumentes, nicht des einbettenden oder des anderen eingebetteten Dokumentes.

7 `http://de.wikibooks.org/wiki/media%3ASVGanimation01.svg`
8 `http://de.wikibooks.org/wiki/media%3ASVGanimation02.svg`

Abb. 294 Auswahlliste

Auswahlliste[9]

Aus einer Vorschauliste rechts kann ausgewählt werden, welches animierte SVG abgespielt werden soll. Mit dem grünen Knopf kann gestartet werden, mit dem roten gestoppt (Siehe auch Titel der Knöpfe). Die Aktivierung eines anderen Dokumentes beendet die Darstellung des aktuellen, welches in die Vorschauliste zurückkehrt.

Analog kann natürlich auch eine Vorschauliste mit dem Element ***video*** angelegt werden, um mit Vorschau zwischen verschiedenen Videos auszuwählen. Dabei bietet sich wie bei den meisten anderen Beispiele auch, für realen Anwendungen die Dateien nicht einzubetten, sondern nur zu referenzieren.

9 `http://de.wikibooks.org/wiki/media%3ASVGanimation03.svg`

17.7 Eigenschaft *audio-level*

Squiggle (Batik)	-
Opera (Presto)	spezielle Sonderversionen ab 9.62video und allgemein ab 10.60
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Die Lautstärke von Audiokanälen kann mit der Eigenschaft *audio-level* beeinflusst werden, mit der Einschränkung, dass ein Kindelement nicht lauter eingestellt werden kann als ein Elternelement.

Die Eigenschaft kann angewendet werden auf die Medienelemente ***audio***, ***video*** und ***animation***, ferner auf ***use*** und Gruppierungselemente wie ***g***.

Mögliche Werte sind 'inherit' oder eine Zahl. Voreinstellung ist 1, das entspricht der Lautstärke des Elternelementes. 0 entspricht Ton aus. Die Eigenschaft wird nicht vererbt und ist animierbar.

Werte außerhalb des Bereiches 0 bis 1 werden auf den nächstgelegenen Wert aus diesem Intervall, also 0 für kleinere Werte und 1 für größere gerundet.

Die Lautstärke des Elementes ist das Produkt der Lautstärke des Elternelementes mit dem gerundeten Wert von *audio-level*.

Die Ausgabesignalstärke wird logarithmisch berechnet. Ist v der aus dem Elternelement und *audio-level* bestimmte Lautstärkewert, so ist die Ausgabelautstärke:

0 oder 'aus', falls v=0 ist, sonst:

dB-Änderung der Signalstärke = 20 * $\log_{10}$(v)

Beispiel zur Einstellung der Lautstärke per Animation:

```
<video id="Video" xlink:href="Dampflok.ogv"
       xlink:title="Dampflokomotive kommt aus Tunnel"
       type="video/ogg"
       initialVisibility="always"
       x="50" y="50" width="320" height="240"
       dur="media" begin="Knopf.click"
       audio-level="1">
  <animate attributeName="audio-level"
       from="1"
       to="0"
       dur="20s"
       begin="c1.click"
       end="c2.click; indefinite"
       fill="freeze" />
</video>
<g id="Knopf" role="button">
  <title>Knopf zum Starten des Videos</title>
  <rect width="320" height="50" x="50" y="300" rx="5"
      fill="#ffc" stroke="#009" stroke-width="5"/>
  <text x="210" y="330" text-anchor="middle" font-size="20"
      fill="#00f">fahren lassen!</text>
</g>
<circle id="c1" fill="#ccf" stroke="blue" stroke-width="10"
```

```
r="25" cx="75" cy="400" role="button">
<title>Knopf zum Leiser regeln</title>
</circle>
<circle id="c2" fill="#ccf" stroke="blue" stroke-width="10"
r="25" cx="140" cy="400" role="button">
<title>Knopf zum Leiser regeln stoppen</title>
</circle>
<rect x="200" y="375" width="120" height="50" fill="#f0f">
  <animate attributeName="width"
       from="120"
       to="0"
       dur="20s"
       begin="c1.click"
       end="c2.click; indefinite"
       fill="freeze" />
</rect>
```

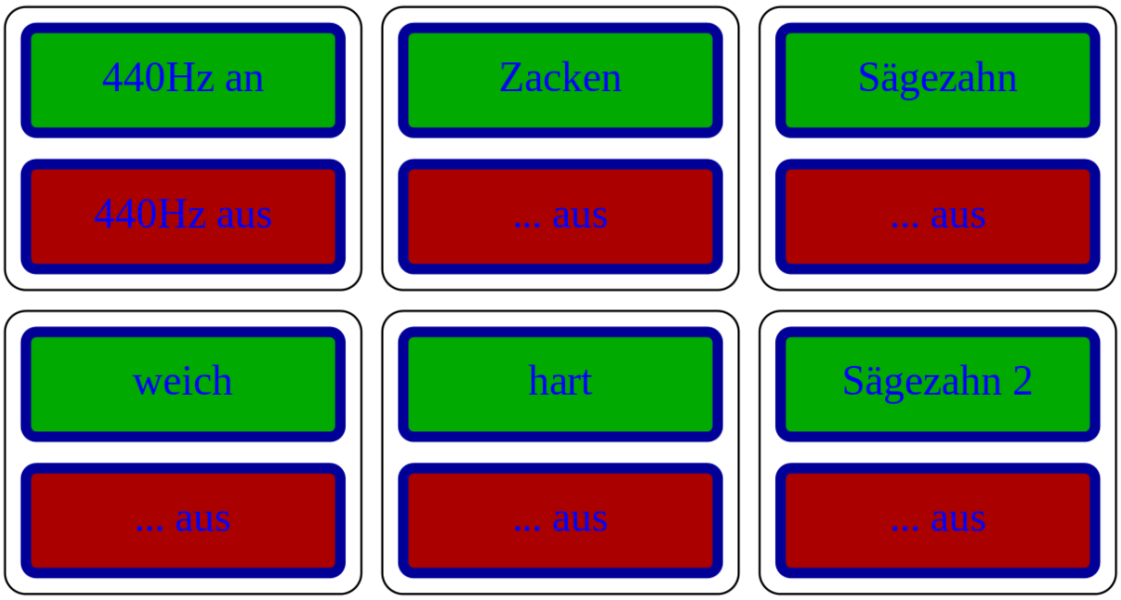


Abb. 295 Amplitudenmodulation

Amplitudenmodulation[10]

Amplitudenmodulation durch Animation von audio-level.

Voraussetzung ist die Präsentation des Formates audio/ogg (OGG+Vorbis).

Der Ton begint jeweils mit dem Startknopf und endet mit dem Endknopf.

Die Hintergrundfarbe wird über die aktive Dauer der Audioausgabe als visuelles Kennzeichen animiert.

10 http://de.wikibooks.org/wiki/media%3ASVGaudio-level01.svg

Bekannte Mängel: eventuelle zusätzliche Knackgeräusche beim Beginn oder Ende einer Audiodatei, die allerdings nicht an der Audiodatei selbst liegen.

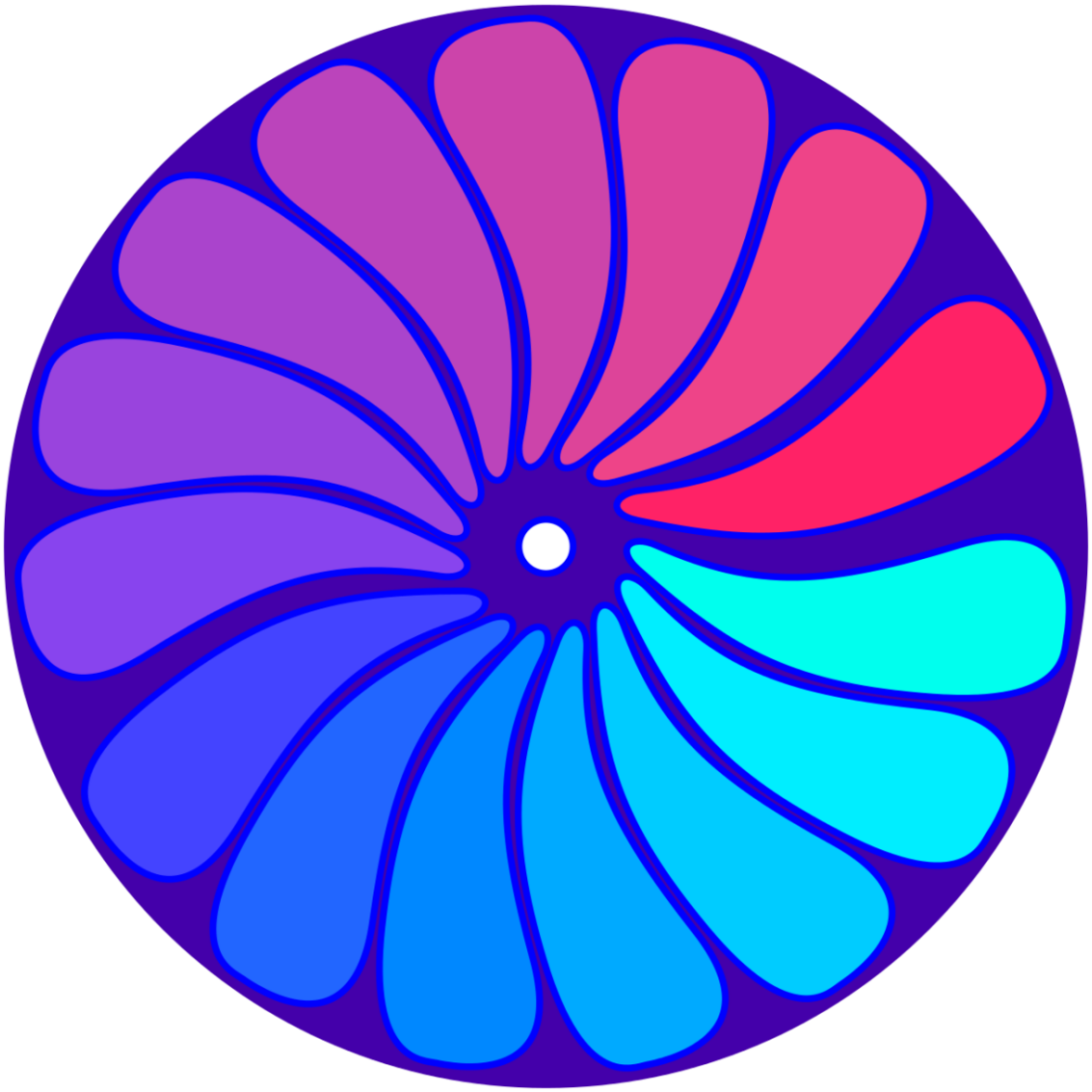


Abb. 296 einfaches Musikinstrument

Ein weiteres Beispiel ist das bereits oben behandelte einfache Musikinstrument[11].

Die Lautstärke mit dem kleinen mittleren Kreis eingestellt werden.

Lautstärke ändert sich von leise nach laut in 5s, symbolisiert mit einer Farbänderung von schwarz nach weiß.

11 `http://de.wikibooks.org/wiki/media%3ASVG_Musikinstrument01.svg`

Loslassen fixiert die Lautstärke auf dem aktuellen Wert.

Ein permanenter Ton etwa zur Lautstärkeeinstellung kann erreicht werden, wenn eine Taste gedrückt wird, die Maus aber erst außerhalb der Taste wieder losgelassen wird.

17.8 Attribute zur Laufzeitsynchronisation

Squiggle (Batik)	-
Opera (Presto)	9.5 (teilweise)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

SVG tiny 1.2 übernimmt von SMIL fünf Attribute zur Laufzeitsynchronisation von Medienelementen.

syncBehavior, *syncTolerance* und *syncMaster* können bei den Elementen ***audio***, ***video*** und ***animation*** angegeben werden. *syncBehaviorDefault* und *syncToleranceDefault* können beim Hauptelement ***svg*** angegeben werden.

17.8.1 Attribut *syncBehavior*

Das Attribut legt das Synchronisationsverhalten eines Medienelementes fest.

Mögliche Werte sind:

canSlip

Erlaubt dem Element einen eigenen Zeitblauf gegenüber dem Hauptelement ***svg*** (oder dem *syncMaster*). Ein angegebenes Attribut *syncTolerance* wird ignoriert.

locked

Bindet das Zeitverhalten fest an das Hauptelement ***svg*** (oder dem *syncMaster*).

Toleranzen können mit dem Attribut *syncTolerance* angegeben werden.

independent

Unabhängiger Zeitablauf, eigenständige Zeitlinie. Suchabläufe bei den Eltern werden ignoriert.

default

Die Laufzeitsynchronisation wird bestimmt durch das Attribut *syncBehaviorDefault*.

Das ist die Voreinstellung

Das Attribut ist nicht animierbar.

17.8.2 Attribut *syncTolerance*

Das Attribut gibt die Toleranz der Laufzeitsynchronisation an. Der Wert ist entweder ein Zeitwert (wie im Kapitel Animation definiert) oder der Wert 'default'. Ein einfacher Wert wäre etwa '1s', um eine Verschiebung der Zeitachse relativ zur Hauptachse um eine Sekunde zu erlauben. Bei der Angabe von 'default' wird der Wert von *syncToleranceDefault* verwendet. Das ist die Voreinstellung.

Das Attribut ist nicht animierbar.

syncTolerance hat nur einen Effekt, wenn für *syncBehavior* der Wert 'locked' angegeben ist. Solange die Abweichung der Zeiten kleiner als der angegebene Wert ist, wird keine Synchronisation vorgenommen. Ist die Abweichung größer, wird das Zeitverhalten an die Hauptzeitachse angepasst.

17.8.3 Attribut *syncMaster*

Mit dem Attribut wird angegeben, welches Element die Hauptzeitachse oder die maßgebliche Zeitinformation für die Synchronisation liefert. Der Wert ist entweder 'true' oder 'false'. 'false' ist die Voreinstellung.

Im Falle von 'true' liefert das Element selbst die maßgebliche Zeitinformation. Im Falle von 'false' wird es gegebenenfalls bei Abweichungen gegenüber der Hauptzeitachse synchronisiert.

Das Attribut ist nicht animierbar.

Es ist zu empfehlen, nur einen *syncMaster* zu setzen. Bei mehreren ist der im Quelltext zuerst auftretende und aktive der *syncMaster*. Ist kein *syncMaster* angegeben oder alle 'false', so ist das Hauptelement ***svg*** der *syncMaster*.

17.8.4 Attribut *syncBehaviorDefault*

Das im Hauptelement ***svg*** notierte Attribut legt die Voreinstellung für *syncBehavior* fest.

Mögliche Werte sind:

canSlip

Erlaubt dem Element einen eigenen Zeitblauf gegenüber dem Hauptelement ***svg*** (oder dem *syncMaster*). Ein angegebenes Attribut *syncTolerance* wird ignoriert.

locked

Bindet das Zeitverhalten fest an das Hauptelement ***svg*** (oder dem *syncMaster*).

Toleranzen können mit dem Attribut *syncTolerance* angegeben werden.

independent

Unabhängiger Zeitablauf, eigenständige Zeitlinie. Suchabläufe bei den Eltern werden ignoriert.

inherit

Das Verhalten wird vom Elternelement geerbt. Gibt es kein Elternelement, so ist das Verhalten vom Darstellungsprogramm abhängig.

'inherit' ist die Voreinstellung. Das Attribut ist nicht animierbar.

17.8.5 Attribut *syncToleranceDefault*

Das Attribut gibt die Toleranz der Laufzeitsynchronisation an. Der Wert ist entweder ein Zeitwert (wie im Kapitel Animation definiert) oder der Wert 'inherit'. Ein einfacher Wert wäre etwa '1s', um eine Verschiebung der Zeitachse relativ zur Hauptachse um eine Sekunde zu erlauben. Bei der Angabe von 'inherit' wird der Wert vom Elternelement geerbt. Das ist die Voreinstellung. Gibt es kein Elternelement, so ist das Verhalten vom Darstellungsprogramm abhängig, soll aber nicht größer als 2s sein.

Das Attribut ist nicht animierbar.

17.8.6 Beispiele zur Laufzeitsynchronisation

```
<video xlink:href="Kaminfeuer.ogv" type="video/ogg"
       x="50" y="50" width="320" height="240"
       dur="media" repeatDur="indefinite"
       audio-level="0.5"
       syncMaster="false" syncTolerance="0.2s">
       <title>Kaminfeuer</title>
       <desc>
        In einem steinernen Kamin entzündet sich aus dem Dunklen heraus
        ein Feuer aus Buchenholz ohne ersichtlichen Grund und brennt
        über Stunden langsam herunter und erlischt endlich.
        Der Vorgang wiederholt sich.
       </desc>
</video>
<audio xlink:href="Kuschelrockzusammenstellung.oga" type="audio/ogg"
       dur="media" repeatDur="indefinite"
       syncMaster="true" />
```

Die Kuschelrockzusammenstellung liefert also die Hauptzeitachse, welcher das Kaminfeuer mit einer Genauigkeit von 0.2s folgen muss. Gegebenenfalls müssen beim Kaminfeuer also Einzelbilder ausgelassen oder etwas länger angezeigt werden, um die Synchronsisation zu realisieren.

Abb. 297 Synchronisation zweier SVG-Dokumente

Synchronisation zweier SVG-Dokumente[12]

SVG-Dokumente mit Iterierten Funktionensystemen (IFS) werden eingebettet. Typisch dauert es bis zur Darstellung solcher Graphiken etwas länger.

Das kompliziertere IFS links ist syncMaster. Das einfachere IFS rechts muss also warten, bis auch das links dargestellt wird, bevor die Animation darin beginnen kann. Diese ist eine Animation der Strichfarbe und beginnt mit rot.

Ein Viertelkreis in einer Ecke hat jeweils eine diskrete Farbanimation mit einem Wechsel zwischen blau und grün. Bei einer Synchronisation müssen die Farben in beiden Dateien gleichzeitig umspringen.

12 `http://de.wikibooks.org/wiki/media%3ASVG_Laufzeitsynchronisation01.svg`

18 Erweiterbarkeit

18.1 Erweiterbarkeit

Als XML-Format lässt sich SVG durch andere XML-Formate erweitern. Prinzipiell ist eine nahezu beliebige Mischung verschiedener Formate möglich. SVG schränkt jedoch die Möglichkeiten insoweit ein, als außerhalb von bestimmten Elementen Elemente aus anderen Namensräumen bei der Darstellung ignoriert werden sollen. Neben den dafür vorgesehenen und in anderen Kapiteln beschriebenen Elementen ***title***, ***desc*** und ***metadata*** ist insbesondere ***foreignObject*** dafür vorgesehen, Elemente aus anderen Namensräumen aufzunehmen, die zur Präsentation gedacht sind. Allerdings sieht SVG selbst kein spezielles anderes Format vor, welches von einem Darstellungsprogramm interpretiert werden muss. Naheliegend sind allerdings die Formate XHTML und MathML. Sofern das Darstellungsprogramm auch CSS interpretiert, ist es naheliegend, dass beliebiger mit CSS dekorierter und mit XML ausgezeichneter Text dargestellt werden kann. In SVG tiny 1.2 wird sogar erwartet, dass Darstellungsprogramme allgemein die Fähigkeit haben, mit CSS oder XSL dekorierten und mit einem XML-Format ausgezeichneten Text auch innerhalb von ***foreignObject*** entsprechend anzuzeigen. Dies gewährleistet jedenfalls eine komfortable Möglichkeit, automatisch umgebrochenen Fließtext in SVG einzubetten.

Erweiterungen erfolgen jedenfalls in diesem Sinne immer in einem eigenen Namensraum, niemals durch eine Erweiterung der Dokumenttypdeklaration von SVG durch den Autor oder einfach so durch Einfügen von Elementen ohne eigenen Namensraum. Daraus ergibt sich insbesondere, dass Formate, welche keinen eigenen Namensraum definieren, ungeeignet für eine Integration in SVG sind.

Sofern eine Validierung erfolgen soll, empfiehlt sich, das SVG-Dokument und den fremden Inhalt getrennt zu validieren und dann erst zusammenzufügen, weil Validatoren meistens solchen Dokumenten mit gemischten Namensräumen nicht gewachsen sind, besonders jene Validatoren nicht, die mit DTDs arbeiten.

18.2 *foreignObject*

Squiggle (Batik)	-(bislang kein Format gefunden, welches interpretiert würde, um dies testen zu können)
Opera (Presto)	9.5 (teilweise und oft falsch)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (teilweise)
Konqueror (KSVG)	-
Safari (Webkit)	4 (teilweise)
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Das Element ***foreignObject*** dient dazu, um Inhalte aus anderen Namensräumen zur Darstellung aufzunehmen. Das Element erzeugt für den fremden Inhalt einen eigenen Anzeigebereich, in welchem gegebenenfalls der fremde Inhalt dargestellt wird. Insbesondere stammt der direkte Inhalt des Elementes nicht aus dem Namensraum von SVG. Eine Ausnahme davon sind die Elemente ***title***, ***desc***, ***metadata*** und ***switch*** in SVG tiny 1.2. Sollte allerdings der fremde Inhalt wiederum ein Dokumentfragment aus dem Namensraum von SVG enthalten, so ist laut der Empfehlung SVG 1.1 mindestens dieses Fragment darzustellen.

In SVG 1.1 wird zudem erwartet, dass ***foreignObject*** zusammen mit dem Element ***switch*** verwendet wird, also nur als Kindelement von ***switch*** auftritt, wobei die bedingte Verarbeitung sinnvollerweise durch Verwendung des Attributes *requiredExtensions* erfolgt. In SVG tiny 1.2 sind auch andere Gruppierungselemente als Elternelemente erlaubt. Ob die Verwendung von *requiredExtensions* angebracht ist oder nicht, hängt von vielerlei Dingen ab. Wenn der fremde Inhalt nur semantisch relevante Elemente enthält, die zudem mit CSS dekoriert werden, so erfordert die Präsentation nicht, dass das Darstellungsprogramm das Format selbst kenn, es muss nur die Dekoration durchführen können. Bietet das fremde Format in dem Fragment hingegen eine technisch relevante Funktionalität, so ist eine bedingte Verarbeitung sicherlich angebracht.

In SVG 1.1 wird der fremde Inhalt direkt im Element notiert. In SVG ist es alternativ auch möglich, ein komplettes Dokument zu referenzieren, welches in das ***foreignObject*** eingebettet wird.

18.2.1 Attribute *x* und *y*

x repräsentiert die x-Koordinate der linken oberen Ecke des ***foreignObject***. *y* repräsentiert die y-Koordinate der linken oberen Ecke des ***foreignObject***. Der Wert ist jeweils eine Koordinate. Wenn der Attributwert nicht angegeben ist, wird jeweils 0 angenommen.

Die Attribute sind animierbar.

18.2.2 Attribute *width* und *height*

width repräsentiert die Breite des ***foreignObject***. *height* repräsentiert die Höhe des ***foreignObject***. Der Wert ist jeweils eine nicht negative Länge. Der Wert 0 unterbindet die Anzeige des Elementes. In SVG 1.1 ist die Angabe des Attributes erforderlich, sonst oder bei einem negativen Wert liegt ein

Fehler vor, der zum Abbruch der Anzeige des Dokumentes führt. Wenn der Attributwert in SVG tiny 1.2 nicht angegeben oder negativ ist, wird 0 angenommen.

Die Attribute sind animierbar.

18.2.3 Attribute aus dem Namensraum XLink, insbesondere *href*

In SVG tiny 1.2 dient das Attribut *href* aus dem Namensraum XLink dazu, eine externe Datei zu referenzieren, welche in das ***foreignObject*** eingebettet wird. Damit sind auch die anderen Attribute von XLink verfügbar.

Ist *href* von XLink angegeben, ist das externe Dokument einzubetten und nicht der gegebenenfalls im Element angegeben fremde Inhalt darzustellen. Ist das Attribut nicht angegeben, so ist der Inhalt gegebenenfalls darzustellen.

Das Attribut ist animierbar.

18.2.4 Beispiele

XHTML

Ein Beispiel in SVG tiny 1.2 mit einem externen Dokument sieht wie folgt aus:

```
<foreignObject xmlns:xlink="http://www.w3.org/1999/xlink"
   xlink:href="beispiel.xhtml"
   x="100" y="100" width="400" height="400">
   <title>Beispiel für XHTML in SVG</title>
   <desc>Beispieltext 'Hallo Welt!'.</desc>
</foreignObject>
```

Da das Element nicht der bedingten Verarbeitung unterliegt, dienen Titel und Beschreibung als Hilfe und SVG-eigene Textalternative, sofern der referenzierte Inhalt nicht darstellbar sein sollte.

Ein Beispiel mit direkt integriertem Inhalt aus dem Namensraum von XHTML und bedingter Verarbeitung mit SVG-Alternative:

```
<switch>
<foreignObject requiredExtensions="http://www.w3.org/1999/xhtml"
 x="100" y="100" width="400" height="400">
 <html xmlns="http://www.w3.org/1999/xhtml">
   <head>
     <title>Beispiel für XHTML in SVG</title>
     <style type="text/css">
       body {font-family:sans-serif}
       h1 {font-size:130%}
     </style>
   </head>
   <body>
     <h1>Beispiel für XHTML in SVG</h1>
     <p>Hallo Welt!</p>
   </body>
 </html>
</foreignObject>
<text font-family="sans-serif" font-size="20"
      x="120" y="120">Hallo Welt!</text>
</switch>
```

Leider ist es bei wikibooks nicht möglich, ein Beispiel mit XHTML direkt zum Angucken hochzuladen. Daher noch ein komplettes Beispiel zum Kopieren:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg viewBox="0 0 700 500"
    xmlns="http://www.w3.org/2000/svg" version="1.1"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xml:lang="de">
<title>foreignObject und XHTML</title>
<desc>
'Hallo Welt'-Beispiel mit XHTML in SVG.

Interessant ist für die Brauchbarkeit der Darstellung,
ob der XHTML-Inhalt wirklich nutzbar ist, etwa durch Rollbalken,
wenn Inhalt übersteht, was hier getestet wird.
Eine andere in SVG nicht verfügbare Funktionalität ist zum Beispiel
die eines Formulares - nach dem Absenden sollten die Formulardaten
als GET-Parameter verfügbar sein.
</desc>
<rect x="0" y="0" width="700" height="500" fill="#ffc" />
<g font-family="sans-serif" font-size="20">
<switch>
<foreignObject requiredExtensions="http://www.w3.org/1999/xhtml"
 x="100" y="100" width="500" height="260">
 <html xmlns="http://www.w3.org/1999/xhtml">
   <head>
     <title>Beispiel für XHTML in SVG</title>
     <style type="text/css">
       body {display:block; font-family:sans-serif;
             height:250px; width:500px; overflow: auto}
       h1 {display:block; font-size:130%}
       p,div {display:block}
     </style>
   </head>
   <body>
     <h1>Beispiel für XHTML in SVG</h1>
     <p>Hallo Welt!</p>

     <div>
       <form action="" method="get" target="_blank">
        <p>Text:<input name="Text" type="text"
                 size="10" maxlength="40" /></p>
        <p>Mehr Text:<textarea name="Mehr_Text"
                cols="10" rows="2"></textarea></p>
        <p>Auswahl:
          <select name="Auswahl" size="1">
           <option>A</option>
           <option>B</option>
           <option>C</option>
           <option>D</option>
           <option>E</option>
          </select>
        </p>
        <p>Radio:
          <input type="radio" name="Radio" value="1"/>1
          <input type="radio" name="Radio" value="2"/>2
          <input type="radio" name="Radio" value="3"/>3
        </p>
        <p>Auswahl:
          <input type="checkbox" name="checkbox" value="!"/>!
          <input type="checkbox" name="checkbox" value="?"/>?
          <input type="checkbox" name="checkbox" value="%"/>%
        </p>
        <p>
          <input type="hidden"
```

```
                name="versteckt" value="guckguck" />
          <input type="submit" value="Absenden " />
        </p>
       </form>
    </div>
   </body>
 </html>
</foreignObject>
<text font-family="sans-serif" font-size="20"
      x="120" y="120">Hallo Welt!</text>
</switch>
</g>
</svg>
```

SMIL

Beispiel mit SMIL mit bedingter Verarbeitung mit SVG-Alternative:

```
<switch xmlns:xlink="http://www.w3.org/1999/xlink">
<foreignObject requiredExtensions="http://www.w3.org/ns/SMIL"
   x="100" y="100" width="400" height="400">
  <smil xmlns="http://www.w3.org/ns/SMIL" version="3.0" baseProfile="Language">
   <body>
    <seq begin="3s">
     <img src="erstesbild.png"
             alt="Bild 1" dur="10s" end="click;activate" />
     <img src="zweitesbild.png"
             alt="Bild 2" dur="10s" end="click;activate" />
     <img src="drittesbild.png"
             alt="Bild 3" dur="10s" end="click;activate" />
    </seq>
   </body>
  </smil>
</foreignObject>
<g>
  <image xlink:href="drittesbild.png" xlink:title="Bild 3"
         width="400" height="300" display="none">
    <set id="s3" attributeName="display"
         to="inline" begin="s2.end" end="click;activate" dur="10s" />
  </image>
  <image xlink:href="zweitesbild.png" xlink:title="Bild 2"
          width="400" height="300" display="none">
    <set id="s2" attributeName="display"
         to="inline" begin="s1.end" end="click;activate" dur="10s" />
  </image>
  <image xlink:href="erstesbild.png" xlink:title="Bild 1"
          width="400" height="300" display="none">
    <set id="s1" attributeName="display"
         to="inline" begin="3s" end="click;activate" dur="10s" />
  </image>
</g>
</switch>
```

Das wäre zum Beispiel auch eine Möglichkeit, formal korrekt Audio- und Videodateien in SVG 1.1 einzubinden, indem die entsprechenden Elemente ***audio*** und ***video*** von SMIL genutzt werden, welche im Namensraum von SVG erst mit Version SVG tiny 1.2 verfügbar sind.

```
<switch xmlns:xlink="http://www.w3.org/1999/xlink">
<foreignObject requiredExtensions="http://www.w3.org/ns/SMIL"
   x="100" y="100" width="400" height="400">
  <smil xmlns="http://www.w3.org/ns/SMIL" version="3.0" baseProfile="Language">
   <body>
    <seq begin="3s">
```

```
     <video src="erstesvideo.ogv"
            alt="Video 1" dur="200s" end="click;activate" />
     <video src="zweitesvideo.ogv"
            alt="Video 2" dur="200s" end="click;activate" />
     <video src="drittesvideo.ogv"
            alt="Video 3" dur="200s" end="click;activate" />
    </seq>
   </body>
  </smil>
</foreignObject>
<g>
  <a xlink:href="erstesvideo.ogv" target="_blank" title="Zum Video 1">
     <image xlink:href="erstesbild.png" xlink:title="Bild 1"
         width="40" height="30" />
  </a>
  <a xlink:href="zweitesvideo.ogv" target="_blank" title="Zum Video 2">
     <image xlink:href="zweitesbild.png" xlink:title="Bild 2"
         x="50" width="40" height="30" />
  </a>
  <a xlink:href="drittesvideo.ogv" target="_blank" title="Zum Video 3">
     <image xlink:href="drittesbild.png" xlink:title="Bild 3"
         x="100" width="40" height="30" />
  </a>
</g>
</switch>
```

DAISY

Abb. 298 Ich habe dich so lieb (Joachim Ringelnatz)

Oder ein Beispiel im Forma DAISY, hier ein Gedicht von Joachim Ringelnatz. Weil DAISY kein eigenes Element ***style*** hat, wird zur Dekoration entweder das von SVG verwendet oder eine externe Stilvorlage mit einer Stilvorlagenverarbeitungsanweisung referenziert. Allerdings hat DAISY wie auch LML im Gegensatz zu etwa XHTML Elemente zur Auszeichnung von Gedichten, weswegen diese Formate bei solchem Inhalt semantisch reichhaltiger als XHTML sind und daher diesem vorzuziehen. Andererseits ist die Wahrscheinlichkeit einer Interpretation geringer. Da es sich wiederum im folgenden Beispiel nur um ausgezeichneten und dekorierten Text ohne besondere technische Funktionalität handelt, kann es vorteilhaft sein, auf die bedingte Verarbeitung zu verzichten. In

diesem Falle ist die Alternative lediglich ein einfacher Verweis zu einer anderen Quelle für das Gedicht.

```
<g font-family="sans-serif" font-size="20">
<switch xmlns:xlink="http://www.w3.org/1999/xlink">
<foreignObject requiredExtensions="http://www.daisy.org/z3986/2005/dtbook/"
   x="100" y="100" width="600" height="800">
<dtbook xmlns="http://www.daisy.org/z3986/2005/dtbook/" version="2005-3"
 xml:lang="de">
  <head profile="http://dublincore.org/documents/2008/08/04/dc-html/">
      <link rel="schema.dc" href="http://purl.org/dc/elements/1.1/" />
      <meta name="dc:identifier"
 content="wikibooks;2010-04-29;Ringelnatz;Ich_habe_dich_so_lieb" />
      <meta name="dc:relation"
 content="http://de.wikisource.org/wiki/Ich_habe_dich_so_lieb"/>
      <meta name="dc:title" content="Ich habe dich so lieb"/>
      <meta name="dc:creator" content="Joachim Ringelnatz"/>
      <meta name="dc:publisher" content="Dr. Olaf Hoffmann"/>
      <meta name="dc:date" content="2010-04-29"/>
      <meta name="dc:type" content="Text"/>
      <meta name="dc:format" content="ANSI/NISO Z39.86-2005"/>
  </head>
  <book><bodymatter><level>
  <poem>
    <title>Ich habe dich so lieb</title>
    <author>Joachim Ringelnatz</author>
    <byline>aus: Allerdings</byline>
    <dateline>1928</dateline>
    <linegroup>
      <line>Ich habe dich so lieb!</line>
      <line>Ich würde dir ohne Bedenken</line>
      <line>Eine Kachel aus meinem Ofen</line>
      <line>Schenken.</line>
    </linegroup>
    <linegroup>
      <line>Ich habe dir nichts getan.</line>
      <line>Nun ist mir traurig zu Mut.</line>
      <line>An den Hängen der Eisenbahn</line>
      <line>Leuchtet der Ginster so gut.</line>
    </linegroup>
    <linegroup>
      <line>Vorbei - verjährt -</line>
      <line>Doch nimmer vergessen.</line>
      <line>Ich reise.</line>
      <line>Alles, was lange währt,</line>
      <line>Ist leise.</line>
    </linegroup>
    <linegroup>
      <line>Die Zeit entstellt</line>
      <line>Alle Lebewesen.</line>
      <line>Ein Hund bellt.</line>
      <line>Er kann nicht lesen.</line>
      <line>Er kann nicht schreiben.</line>
      <line>Wir können nicht bleiben.</line>
    </linegroup>
    <linegroup>
      <line>Ich lache.</line>
      <line>Die Löcher sind die Hauptsache</line>
      <line>An einem Sieb.</line>
    </linegroup>
    <linegroup>
      <line>Ich habe dich so lieb.</line>
    </linegroup>
  </poem>
  </level></bodymatter></book>
</dtbook>
</foreignObject>
```

```
<a xlink:href="http://de.wikisource.org/wiki/Ich_habe_dich_so_lieb"
   xlink:title="Externer Verweis in neuem Fenster zu de.wikisource.org"
 target="_blank">
<text x="120" y="120">"Ich habe dich so lieb"; Joachim Ringelnatz, 1928</text>
</a>
</switch>
</g>
```

Ausgeschmücktes Beispiel: Ich habe dich so lieb (Joachim Ringelnatz)[1]

MathML

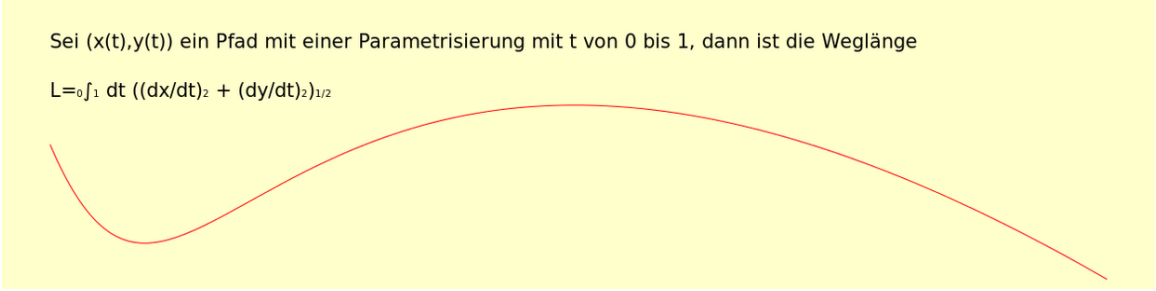


Abb. 299 Weglänge

Ein Beispiel mit MathML und bedingter Verarbeitung mit Alternative in SVG:

```
<g font-family="sans-serif" font-size="20">
<switch>
<foreignObject requiredExtensions="http://www.w3.org/1998/Math/MathML"
   x="50" y="50" width="1100" height="100">
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mtext>Sei (x(t),y(t)) ein Pfad
 mit einer Parametrisierung mit t von 0 bis 1,
 dann ist die Weglänge L=</mtext>
  <msubsup>
    <mo>&#8747;</mo>
    <mn>0</mn>
    <mn>1</mn>
  </msubsup>
  <msqrt>
    <msup>
      <mrow>
        <mo>(</mo>
        <mfrac>
          <mi>dx</mi>
          <mi>dt</mi>
        </mfrac>
        <mo>)</mo>
      </mrow>
      <mn>2</mn>
    </msup>
    <mo>+</mo>
    <msup>
      <mrow>
        <mo>(</mo>
        <mfrac>
          <mi>dy</mi>
```

1 http://de.wikibooks.org/wiki/media%3AichHabeDichSoLieb.svg

```
          <mi>dt</mi>
        </mfrac>
        <mo>)</mo>
      </mrow>
      <mn>2</mn>
    </msup>
  </msqrt>
  <mi>dt</mi>
</math>
</foreignObject>
<g>
<text x="50" y="50">Sei (x(t),y(t)) ein Pfad
mit einer Parametrisierung mit t von 0 bis 1,
dann ist die Weglänge</text>
<text x="50" y="100">L=<tspan baseline-shift="sub"
 font-size="10">0</tspan>&#8747;<tspan
  baseline-shift="160%" font-size="10">1</tspan> dt ((dx/dt)<tspan
 baseline-shift="super"
  font-size="10">2</tspan> + (dy/dt)<tspan baseline-shift="super"
  font-size="10">2</tspan>)<tspan baseline-shift="10"
 font-size="10">1/2</tspan></text>
</g>
</switch>
</g>
```

Ausgeschmücktes Beispiel: Weglänge[2]

18.3 Verweise

- CDF[3] (gemischte Formate)
- XHTML+RDFa[4], (X)HTML[5]
- SMIL[6]
- MathML[7]
- DAISY[8]
- LML[9]

2 http://de.wikibooks.org/wiki/media%3ASVGforeignObject_MathML.svg
3 http://www.w3.org/2004/CDF/
4 http://www.w3.org/TR/rdfa-syntax/
5 http://www.w3.org/MarkUp/
6 http://www.w3.org/AudioVideo/
7 http://www.w3.org/Math/
8 http://www.daisy.org/
9 http://purl.oclc.org/net/hoffmann/lml/

19 SVG-Zeichensätze

19.1 Zeichensätze und Schriften in SVG

Wie bereits im Kapitel 'Graphik formatieren' erläutert, kann in SVG angegeben werden, welcher Zeichensatz für einen Text verwendet werden soll. Dazu wurden Eigenschaften von CSS übernommen. Diese erlauben es, Schriften namentlich zu notieren. Ist für das Darstellungsprogramm der namentlich genannte Zeichensatz verfügbar, so wird dieser dann verwendet. Allerdings sind nicht alle namentlich bekannten Zeichensätze auch auf jedem Darstellungsprogramm verfügbar.

Schriften werden nach Familien eingeordnet. Der Autor kann auch angeben, dass ein Zeichensatz aus einer bestimmten generischen Schriftfamilie angegeben werden soll. Ein Darstellungsprogramm muss für jede in CSS angegebene Schriftfamilie wenigstens eine Schrift verfügbar haben, die dann verwendet wird. Somit dient die (alternative) Angabe einer generischen Schriftfamilie auch immer dazu, zu gewährleisten, dass der Text überhaupt angezeigt wird, wenn ein namentlich genannter Zeichensatz nicht verfügbar ist.

Da bei SVG aber auch die präzise graphische Präsentation von Text wichtig sein kann und ein Autor hohe Anforderungen an die Präsentation haben mag, kann per CSS auch ein Zeichensatz direkt als Dokument oder Dokumentfragment referenziert werden, welcher verwendet werden soll.

Nun gibt es wiederum diverse Formate, in denen Zeichensätze vorliegen können und von einem Darstellungsprogramm kann nicht erwartet werden, dass dieses alle Formate interpretieren kann.

Ein Autor, welcher exakt die Präsentation von Text in einem SVG-Dokument festlegen will, hätte so also nur die Möglichkeit, den Text in Pfade zu konvertieren. Dabei ginge allerdings die komplette Textinformation verloren, weswegen dies eine schlechte, nicht barrierefreie und daher nicht zu empfehlende Strategie ist. Dies sollte daher vom Autor vermieden werden, schon auch weil die Nachbearbeitung des Textes dadurch enorm erschwert wird, selbst wenn das Zugänglichkeitsproblem damit gelöst wird, dass der Text im Element ***desc*** des Pfades notiert wird.

Damit ein Autor mit solch hohen Anforderungen sowohl den Pfad speichern kann, als auch den Text als Text im Dokument stehenlassen kann, gibt es in SVG 1.1 und SVG tiny 1.2 die Möglichkeit, für die relevanten Zeichensätze, für jeden einzelnen Glyphen den Pfad abzulegen. Dies geschieht im selben Dokument oder ein externes Dokument mit den Glyphen des Zeichensatzes wird referenziert.

Ein solcher SVG-Zeichensatz kann dann per CSS referenziert werden - nicht zwangsläufig nur in SVG-Dokumenten. Wie und was von einem Zeichensatz als SVG-Zeichensatz gespeichert werden kann, ist auf die graphische Präsentation in SVG-Dokumenten optimiert, nicht unbedingt für Fließtexte in (X)HTML, welche wohlmöglich mit kleiner Auflösung gelesen werden sollen. Die mit SVG notierbaren Eigenschaften des Zeichensatzen entsprechen dann auch dem, was in CSS angegeben werden kann.

Prinzipiell kann für die Definition von Glyphen in SVG-Zeichensätzen der komplette Vorrat an Kommandos verwendet werden, also auch kubische Kurven und in SVG 1.1 auch elliptische Bögen.

Viele andere Formate für Zeichensätze haben nicht so eine reichhaltige Auswahl an Kurvenformen. Üblich sind oft nur quadratische Kurven. Somit ist es technisch also problemlos möglich, die Glyphen von einem anderen Format nach SVG zu konvertieren, nicht aber unbedingt umgedreht. Andererseits haben andere Formate oft Eigenschaften verfügbar, welche die Glyphen vereinfachen und besser lesbar machen, wenn sie sehr klein dargestellt werden sollen. Solche Informationen können bei der Konversion zu SVG nicht gerettet werden.

Aufgrund der eingeschränkten Kommandovielfalt kann andersherum ein SVG-Zeichensatz nicht exakt in ein Format konvertiert werden, in dem bestimmte Kurvenformen nicht darstellbar sind. Bei einer Konversion muss dann eine Näherung auf die einfachen Formen erfolgen, was dann eine Reduzierung der Qualität und eine Vergrößerung der Datenmenge zur Folge hat.

Bei einer Formatkonversion ist zudem zu bedenken, dass Zeichensätze dem Urheberrecht unterliegen und die Autoren der Zeichensätze die Nutzung unter bestimmte Lizenzen setzen können, die den Zeichensatz zum Beispiel an ein bestimmtes Format koppeln, an ein bestimmtes Betriebssystem oder Darstellungsprogramm. Neben den Fragen der technischen Machbarkeit sind also bei der Konversion von Zeichensätzen anderer Autoren auch noch rechtliche Fragen zu klären, bevor Dokumente veröffentlicht werden, welche solche konvertierten Zeichensätze enthalten.

Das Festlegen, welche Schrift, welcher Zeichensatz definiert wird, kann über das definierende Element ***font*** erfolgen und bei Verwendung von CSS2 in SVG1.1 auch mit der CSS2-Regel *@font-face*.

19.2 Koordinatensystem für SVG-Zeichensätze

Anders als bei sonstigem Inhalt in SVG liegt für Glyphen von Zeichensätzen ein anderes Koordinatensystem vor. Für die Glyphen wird ein Rechtssystem verwendet, bei dem also die positive y-Achse nach oben zeigt, nicht wie sonst in SVG nach unten. Dies wurde vor allem so festgelegt, um die Kompatibilität mit anderen gängigen Formaten für Zeichensätze zu gewährleisten, welche zumeist das gleiche System verwenden.

Autoren von SVG-Zeichensätzen müssen bei der Erstellung der Pfade also umdenken. Es kann sich als praktisch erweisen, einfach bei der optischen Kontrolle der Pfade als Pfade, nicht als Glyphen, eine Achsenspiegelung des Kontrolldokumentes an der x-Achse vorzunehmen, um die Glyphen richtig herum und lesbar zu sehen.

19.3 Einfache Beispiele der Einbindung

Im Element ***font*** wird ein Zeichensatz einer Schriftfamilie definiert.

Der Aufbau, um einen Zeichensatz zu definieren, ist etwas komplizierter, daher wird erst einmal die prinzipielle Struktur diskutiert.

Mit dem Element ***font-face*** in ***font*** wird angegeben, welche speziellen Eigenschaften der Zeichensatz repräsentiert. Mittels ***missing-glyph*** wird ein Zeichen angegeben, welches für einen fehlenden Glyphen steht, oft wie hier ein Quadrat, am besten jedenfalls ein Objekt, welches nicht mit einem normalen Glyphen verwechselt wird, damit der Leser erkennen kann, dass ein Problem vorliegt. Mit ***glyph*** wird dann jeweils der Pfad für eine Glyphe angegeben. In SVG tiny 1.2 ist das exakt ein Pfad

im Attribut *d*, welches dem von ***path*** entspricht. In SVG 1.1 kann es sich alternativ um anderen graphischen Inhalt handeln, welcher als Inhalt des Dokumentes notiert wird.

Identifiziert wird der Zeichensatz dann anhand des Namens, der mittels *font-family* im Element ***font-face*** notiert ist und der weiteren dort notierten Eigenschaften, die denen entsprechen, mit denen der jeweilige Zeichensatz dann verwendet wird.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg viewBox="0 0 300 80"
  version="1.1"
  xmlns='http://www.w3.org/2000/svg'>
  <title>Beispiel SVG-Zeichensatz</title>
  <defs>
    <font id="F" horiz-adv-x="1000">
      <font-face font-family="MeinSatz" font-weight="bold" font-style="normal"
          units-per-em="1000" cap-height="600" x-height="400"
          ascent="700" descent="300"
          alphabetic="0" mathematical="350" ideographic="400" hanging="500" />
      <missing-glyph><path d="M0,0h200v200h-200z"/></missing-glyph>
      <glyph unicode="A"><!-- Elemente für Glyphen 'A' --></glyph>
      <glyph unicode="v"><!-- Elemente für Glyphen 'v' --></glyph>
      <!-- etc, mehr Glyphen -->
      <hkern u1="A" u2="v" k="10"/>
      <!-- etc, mehr Überschneidungsregeln -->
    </font>
  </defs>
  <text x="10" y="50" font-size="20"
        font-family="'MeinSatz', fantasy"
        font-weight="bold"
        font-style="normal">Textbeispiel</text>
</svg>
```

Das Referenzieren eines externen Zeichensatzes kann zum Beispiel wie folgt aussehen:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg viewBox="0 0 300 80"
  version="1.1"
  xmlns='http://www.w3.org/2000/svg'>
  <title>Beispiel: Zeichensatz referenzieren</title>
  <defs>
    <font-face
      font-family="BeispielSchrift"
      unicode-range="U+0-7F">
      <font-face-src>
        <font-face-name name="BeispielSchrift"/>
        <font-face-uri xlink:href="MeinSatz.svg#F">
          <font-face-format string="svg"/>
      </font-face-src>
    </font-face>
  </defs>
  <text x="10" y="50" font-size="20"
        font-family="'BeispielSchrift', fantasy">Textbeispiel</text>
</svg>
```

Entsprechend das gleiche Beispiel mit CSS:

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg viewBox="0 0 300 80"
  version="1.1"
```

```
  xmlns='http://www.w3.org/2000/svg'>
  <title>Beispiel: Zeichensatz referenzieren</title>
  <defs>
   <style type="text/css">
      <![CDATA[
        @font-face {
          font-family: 'BeispielSchrift';
          src: url("MeinSatz.svg#F") format(svg)
        }

        text.meiner {font:family:'BeispielSchrift', fantasy}
      ]]>
   </style>
  </defs>
  <text class="meiner" x="10" y="50"
        font-size="20">Textbeispiel</text>
</svg>
```

Elemente ***font*** und ***font-face***[1]

Verschiedene Methoden, eine eigene Schrift zu verwenden, werden ausprobiert. Ganz oben ist der Testtext mit einer Serifenschrift angegeben, ganz unten der gleiche Text rein aus Pfaden zusammengesetzt, was dann eigentlich kein Text ist.

'MeinSatz' ist eine Schrift, bei der die betreffenden Glyphen direkt im Element ***font*** notiert sind.

'Ref' referenziert diese Schrift mit dem Element ***font-face-uri***.

'CSS' referenziert diese Schrift mit der @font-face-Regel von CSS.

19.4 Element *font*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Das Element ***font*** definiert einen Zeichensatz. Das Element muss als direktes Kindelement ein Element ***font-face*** haben. Weitere mögliche Kindelemente sind neben ***title***, ***desc***, ***metadata*** und ***switch*** die für dies Element speziellen Elemente ***glyph***, ***hkern***, ***missing-glyph*** und ***vkern*** in SVG 1.1.

Der Inhalt des Elementes wird nicht direkt dargestellt, daher empfiehlt es sich, dieses im Element ***defs*** zu notieren.

1 http://de.wikibooks.org/wiki/media%3ASVGfont02.svg

19.4.1 Attribute *horiz-origin-x* und *horiz-origin-y*

Das Attribut *horiz-origin-x* gibt die x-Koordinate des Ursprungs im Koordinatensystem des Zeichensatzes an, welcher für horizontal ausgerichteten Text verwendet wird. Das gilt für alle Glyphen eines Zeichensatzes.

Das Attribut *horiz-origin-y* gibt entsprechend die y-Koordinate an. *horiz-origin-y* ist in SVG tiny 1.2 nicht verfügbar.

Der Wert ist jeweils eine Zahl. Ist ein Attribut nicht angegeben, wird für dieses 0 angenommen.

Die Attribute sind nicht animierbar.

19.4.2 Attribute *horiz-adv-x* und *vert-adv-y*

Das Attribut *horiz-adv-x* gibt die Voreinstellung an für den Fortschritt in horizontaler Richtung nach der Darstellung des Glyphen, also praktisch die Breite des für einen Glyphen reservierten Platzes. *vert-adv-y* entsprechend in vertikaler Richtung. Die Breite eines Glyphen darf nicht negativ sein, auch wenn die Schreibrichtung von rechts nach links erfolgt, wie bei hebräisch oder arabisch.

vert-adv-y ist in SVG tiny 1.2 nicht verfügbar.

Der Wert ist jeweils eine Zahl. Ist ein Attribut nicht angegeben, wird für dieses 0 angenommen.

Die Attribute sind nicht animierbar.

19.4.3 Attribute *vert-origin-x* und *vert-origin-y*

Das Attribut *vert-origin-x* gibt die X-Koordinate des Ursprungs im Koordinatensystem des Zeichensatzes an, welcher für vertikal ausgerichteten Text verwendet wird. Das Gilt für alle Glyphen eines Zeichensatzes.

Das Attribut *vert-origin-y* gibt entsprechend die Y-Koordinate an. Die Attribute sind in SVG tiny 1.2 nicht verfügbar.

Der Wert ist jeweils eine Zahl. Ist das Attribut *vert-origin-x* nicht angegeben, wird für dieses die Hälfte des Effektivwertes von *horiz-adv-x* angenommen. Ist das Attribut *vert-origin-y* nicht angegeben, wird für diese die Postion angenommen, welches sich aus dem Attribut *ascent* ergibt.

Die Attribute sind nicht animierbar.

19.5 Element *font-face*

Squiggle (Batik)	1.7 (SVG, nicht die CSS-Variante)
Opera (Presto)	9 (teilweise, eingebettete Glyphen)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Wird ein Zeichensatz direkt im Element ***font*** definiert und nicht nur referenziert, so wird empfohlen, das Element ***font-face*** als direktes Kindelement von ***font*** zu notieren.

Mögliche Kindelemente von ***font-face*** sind neben ***title***, ***desc***, ***metadata*** und ***switch*** die für dies Element spezifische Elemente ***font-face-src*** und ***definition-src***.

Die Funktion des Elementes entspricht genau der CSS2-Regel @*font-face*. Es beschreibt die Eigenschaften eines Zeichensatzes.

Sofern der Zeichensatz innerhalb desselben Elternelementes ***font*** im selben Dokument definiert wird, sollte ***font-face*** auch diesen und keinen anderen Zeichensatz beschreiben. In dem Falle werden auch sämtliche Elemente ***font-face-src*** innerhalb von ***font-face*** ignoriert, weil angenommen wird, dass ***font-face*** die Eigenschaften des lokal definierten Zeichensatzes definiert.

19.5.1 Attribute entsprechend der CSS2-Regel @*font-face*

Die Attribute von ***font-face*** entsprechen den in CSS2 definierten Eigenschaften der CSS2-Regel @*font-face*.

Die Attribute sind: *font-family*, *font-style*, *font-weight*, *unicode-range*, *units-per-em*, *panose-1*, *stemv*, *stemh*, *slope*, *cap-height*, *x-height*, *accent-height*, *ascent*, *descent*, *widths*, *bbox*, *ideographic*, *alphabetic*, *mathematical*, *hanging*, *underline-position*, *underline-thickness*, *strikethrough-position*, *strikethrough-thickness*, *overline-position* und *overline-thickness*.

In SVG 1.1 gibt es zusätzlich zu den obigen, die in SVG tiny 1.2 verfügbar sind, noch die Attribute: *font-variant*, *font-stretch*, *font-size*, *v-alphabetic*, *v-mathematical* und *v-hanging*.

Die Attribute sind allesamt nicht animierbar.

Die Bedeutungen sind die gleichen wie bei den entsprechenden Eigenschaften, mit denen ein Zeichensatz ausgewählt wird. Nur dass bei diesem Element nicht ausgewählt wird, sondern die im Element definierten Glyphen einem bestimmten Zeichensatz mit entsprechenden Eigenschaften zugeordnet wird und dann wiederum als solcher für Textelemente ausgewählt werden kann.

19.5.2 *font-family*

Angabe der Schriftfamilie.

Wert entsprechend der gleichnamigen Eigenschaft. Sind mehrere Schriftfamilien angegeben, treffen alle auf die Glyphen zu.

19.5.3 *font-style*

Angabe des Stiles.

Mögliche Werte sind entweder 'all' als Voreinstellung für alle oder eine Liste mit den möglichen Werten 'normal', 'italic', 'oblique'. Was angegeben ist, trifft jeweils auf den definierten Zeichensatz zu.

19.5.4 *font-variant*

Angabe der Variante.

Möglicher Wert ist eine Liste mit den möglichen Werten 'normal', der Voreinstellung und 'small-caps'.

19.5.5 *font-weight*

Angabe des Gewichtes.

Möglicher Wert ist entweder 'all', die Voreinstellung für alle Gewichte, oder eine Liste von Gewichten mit den möglichen Werten 'normal', 'bold', '100', '200', '300', '400', '500', '600', '700', '800', '900'.

19.5.6 *font-stretch*

Angabe der Dehnung.

Möglicher Wert ist entweder 'all' oder eine Liste von Dehnungen mit den möglichen Werten 'normal', 'ultra-condensed', 'extra-condensed', 'condensed', 'semi-condensed', 'semi-expanded', 'expanded', 'extra-expanded', 'ultra-expanded'. Voreinstellung ist 'normal'.

19.5.7 *font-size*

Angabe der Schriftgröße.

Möglicher Wert ist entweder 'all' oder eine Liste von absoluten Längen.

19.5.8 *unicode-range*

Angabe des abgedeckten Unicode-Bereiches. Damit wird vermeiden, dass der Zeichensatz überhaupt geladen wird, wenn nur Zeichen außerhalb des angegebenen Bereiches benötigt werden. Der Wert ist eine Liste von Unicode-Bereichen. Voreinstellung ist 'U+0-7FFFFFFF' (Typisch wird der selbst definierte Zeichensatz nicht all diese Zeichen enthalten, daher ist zu empfehlen, den zutreffenden Bereich korrekt anzugeben).

Ein Bereich wird in der hexadezimalen Schreibweise notiert, für ein Zeichen etwa U+215A Dabei gibt es wiederum verschiedene Möglichkeiten. Zum einen kann ein '?' als Platzhalter für die letzte Ziffer verwendet werden, also zum Beispiel U+215?, das deckt dann alle Zeichen von U+2150 bis U+215F ab. Auch zwei Fragezeichen sind erlaubt, also zum Beispiel U+00?? geht dann von U+0000 bis U+00FF (Latin-1). Größere Bereiche können mit zwei Nummern verbunden mit einem Bindestrich notiert werden, Beispiel: U+15C00-15FFF entspricht dem Bereich U+15C00 bis U+15FFF (aztekisch).

19.5.9 *units-per-em*

Angabe, wievielen Einheiten in lokalen Koordinaten die Einheit em entspricht.

Der Wert ist eine Zahl. !!!Die Angabe ist bei Verwendung vieler anderer Attribute erforderlich, damit diese einen Sinn ergeben.

19.5.10 *panose-1*

Angabe der PANOSE-Nummern.

Dabei handelt es sich um Klassifikationsnummern für 'TrueType', insgesamt 10 Zahlen, welche die Schlüsseleigenschaften einer lateinischen Schrift festlegt. Der Wert besteht auch zehn ganzen Zahlen, jeweils durch Leerzeichen getrennt. Die Voreinstellung sind zehn Nullen, was bedeutet, dass jeder Zeichensatz zu der Nummernkombination passt.

19.5.11 *stemv*

Angabe der vertikalen Stengelbreite (englisch: stem).

Gemeint ist damit eine Typische Breite einer vertikalen Struktur bei den Glyphen, zum Beispiel die Breite des vertikalen Stiches von 'L', 'I', 'd', 'p' etc.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

Der Wert ist eine Zahl. Voreinstellung ist undefiniert. Ist der Wert undefiniert, wird das Attribut ignoriert.

19.5.12 *stemh*

Angabe der horizontale Stengelbreite (englisch: stem).

Gemeint ist damit eine Typische Breite einer horizontalen Struktur bei den Glyphen, zum Beispiel die Breite von Serifen oder des horizontalen Stiches von 'H', 'L', '-', 'z', 't' etc.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

Der Wert ist eine Zahl. Voreinstellung ist undefiniert. Ist der Wert undefiniert, wird das Attribut ignoriert.

19.5.13 *slope*

Angabe des Winkels von vertikalen Strukturen.

Sind vertikale Strukturen in einem Zeichensatz nicht exakt vertikal, sondern gekippt, etwa beim Typ 'italic' oder 'oblique'. Es handelt sich um einen Winkel entgegen dem Uhrzeigersinn von der Vertikalen aus zur vertikalen Struktur

Der Wert ist eine Zahl. Die Voreinstellung ist 0.

19.5.14 *cap-height*

Angabe der typischen Höhe von großen Buchstaben, von der Basislinie aus gemessen.

Sofern vorhanden ist das zum Beispiel die Höhe von 'M' oder 'H'.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

Der Wert ist eine Zahl. Voreinstellung ist undefiniert. Ist der Wert undefiniert, wird das Attribut ignoriert.

19.5.15 *x-height*

Angabe der typischen Höhe von kleinen Buchstaben, von der Basislinie aus gemessen.

Sofern vorhanden ist das zum Beispiel die Höhe von 'x' oder 'a'. Die Größe ist insbesondere wichtig für die Eigenschaft *font-size-adjust*, weil die Berechnung des Anpassungsfaktors auf dieser Größe basiert. Daher sollte der Wert angegeben werden.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

Der Wert ist eine Zahl. Voreinstellung ist undefiniert. Ist der Wert undefiniert, wird das Attribut ignoriert.

19.5.16 *ascent*

Angabe der maximalen Höhe von Buchstaben, von der Basislinie aus gemessen, aber ohne Akzentzeichen.

Sofern vorhanden ist das zum Beispiel die Höhe von 'l', 'b' oder 'h'.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

Der Wert ist eine Zahl. Voreinstellung ist undefiniert. Ist der Wert undefiniert, wird das Attribut ignoriert.

19.5.17 *descent*

Angabe der maximalen Tiefe von Buchstaben, von der Basislinie aus gemessen, aber ohne Akzentzeichen.

Sofern vorhanden ist das zum Beispiel der Teil unter der Grundlinie von 'j', 'g' oder 'y'.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

Der Wert ist eine Zahl. Voreinstellung ist undefiniert. Ist der Wert undefiniert, wird das Attribut ignoriert.

19.5.18 *accent-height*

Angabe der Höhe von Zeichen mit Akzent, im Koordinatensystem des Zeichensatzes angegeben.

Der Wert ist eine Zahl. Ist das Attribut nicht angegeben, wird der Wert von *accent* angenommen. Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.19 *widths*

Angabe der Breite von Glyphen.

Der Wert ist eine Liste von Kombinationen von optionalen Unicode-Bereichen und einer Zahl oder mehreren Zahlen. Der Listenseparator ist ein Komma.

Die Notation des Unicode-Bereiches ist bereits für das Attribut *unicode-range* beschrieben. Ist kein Bereich für eine Zahl angegeben, wird U+0-7FFFFFFF angenommen.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.* Wenn nicht genug Glyphenbreiten angegeben sind, gilt für die restlichen Glyphen der letzte Wert. Sind zuviele angegeben, werden die überschüssigen Breiten ignoriert.

Beispiel:

```
widths="U+4E00-4E1F 1736 1874 1692"
```

oder

```
widths="U+1A?? 1490, U+215? 1473 1838 1927 1684 1356 1792
    1815 1848 1870 1492 1715 1745 1584 1992 1978 1770"
```

19.5.20 *bbox*

Angabe des maximalen begrenzenden Rechtecks.

Das begrenzende Rechteck ist das kleinste (horizontal und vertikal) ausgerichtete Rechteck, welches jeweils einen jeden Glyphen des Zeichensatzes komplett umschließt.

Der Wert besteht aus einer Liste von vier Zahlen, jeweils mit einem Komma als Separator. Die erste Zahl gibt den x-Wert unten links an, die zweiten den y-Wert unten links, die dritte den x-Wert oben rechts, die vierte den y-Wert oben rechts.

19.5.21 *ideographic*

Angabe zur ideographischen Ausrichtung bezüglich der Grundlinie.

Der Wert ist eine Zahl, die einen Versatz im Koordinatensystem des Zeichensatzes angibt.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.22 *alphabetic*

Angabe alphabetischen Ausrichtung bezüglich der Grundlinie.

Der Wert ist eine Zahl, die einen Versatz im Koordinatensystem des Zeichensatzes angibt.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.23 *mathematical*

Angabe mathematischen Ausrichtung bezüglich der Grundlinie.

Der Wert ist eine Zahl, die einen Versatz im Koordinatensystem des Zeichensatzes angibt.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.24 *hanging*

Angabe hängende Ausrichtung bezüglich der Grundlinie.

Der Wert ist eine Zahl, die einen Versatz im Koordinatensystem des Zeichensatzes angibt.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.25 *v-ideographic*

Angabe zur ideographischen Ausrichtung bezüglich der Grundlinie für vertikal angeordnete Glyphen.

Der Wert ist eine Zahl, die einen Versatz relativ zu *vert-origin-x* im Koordinatensystem des Zeichensatzes angibt.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.26 *v-alphabetic*

Angabe alphabetischen Ausrichtung bezüglich der Grundlinie für vertikal angeordnete Glyphen.

Der Wert ist eine Zahl, die einen Versatz relativ zu *vert-origin-x* im Koordinatensystem des Zeichensatzes angibt.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.27 *v-mathematical*

Angabe mathematischen Ausrichtung bezüglich der Grundlinie für vertikal angeordnete Glyphen.

Der Wert ist eine Zahl, die einen Versatz relativ zu *vert-origin-x* im Koordinatensystem des Zeichensatzes angibt.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.28 *v-hanging*

Angabe hängende Ausrichtung bezüglich der Grundlinie für vertikal angeordnete Glyphen.

Der Wert ist eine Zahl, die einen Versatz relativ zu *vert-origin-x* im Koordinatensystem des Zeichensatzes angibt.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.29 *underline-position*

Angabe der y-Position von Unterstreichungen im Koordinatensystem des Zeichensatzes.

Der Wert ist eine Zahl.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.30 *underline-thickness*

Angabe der Dicke von Unterstreichungen im Koordinatensystem des Zeichensatzes.

Der Wert ist eine Zahl, eine Länge repräsentierend.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.31 *strikethrough-position*

Angabe der y-Position von Durchstreichungen im Koordinatensystem des Zeichensatzes.

Der Wert ist eine Zahl.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.32 *strikethrough-thickness*

Angabe der Dicke von Durchstreichungen im Koordinatensystem des Zeichensatzes.

Der Wert ist eine Zahl, eine Länge repräsentierend.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.33 *overline-position*

Angabe der y-Position von Überstreichungen im Koordinatensystem des Zeichensatzes.

Der Wert ist eine Zahl.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.5.34 *overline-thickness*

Angabe der Dicke von Überstreichungen im Koordinatensystem des Zeichensatzes.

Der Wert ist eine Zahl, eine Länge repräsentierend.

Wenn dies Attribut verwendet wird, muss auch *units-per-em' verwendet werden.*

19.6 Element *font-face-src*

Das Element entsprecht dem CSS2-Deskriptor 'src' der CSS2-Regel @*font-face*.

font-face-src ist, sofern angegeben, direktes Kindelement von ***font-face***. ***font-face-uri*** und ***font-face-name*** sind wiederum direkte Kindelemente von ***font-face-src***.

font-face-src wird verwendet, wenn der Zeichensatz nicht selbst im Element ***font*** definiert wird, sondern außerhalb davon in einem anderen Dokument oder einem anderen Dokumentfragment.

19.7 Element *font-face-uri*

font-face-uri referenziert die Ressource für den Zeichensatz.

Sofern ein SVG-Zeichensatz referenziert wird, wird der Fragmentidentifizierer des entsprechenden Elementes ***font*** mit angegeben.

19.7.1 Attribut *href* von XLink

Zur Referenzierung der Ressource wird in ***font-face-uri*** das Attribut *href* von XLink verwendet. Der Wert ist eine IRI samt Fragmentidentifizierer für ein Element ***font***.

Das Attribut ist für dieses Element nicht animierbar.

Mit *href* sind auch die anderen Attribute von XLink verfügbar.

19.8 *Element font-face-format*

Direktes Kindelelement von ***font-face-uri*** ist das Element ***font-face-format*** mit dem Attribut *string*, der Wert ist ein Liste von Zeichenfolge, mit dem Komma als Separator. Diese Zeichenfolgen sind beispielhaft für einige Formate in CSS2.0 festgelegt (oder in einem CSS3-Modul). Für bekannte Formate ergibt sich folgende Zuordnung einer Zeichenfolge zu einem Format:

svg

SVG-Zeichensatz, typische Dateiendungen: .svg, .svgz

type-1

PostScript Type 1, typische Dateiendungen: .pfb, .pfa

truetype

TrueType, typische Dateiendung: .ttf

truetype-gx

TrueType mit GX-Erweiterungen, typische Dateiendung: .ttf

truetype-aat

TrueType mit vorgeschrittenen Erweiterungen, typische Dateiendung: .ttf

opentype

OpenType, typische Dateiendungen: .ttf, .otf

embedded-opentype

Embedded OpenType, typische Dateiendung: .eot

truedoc-pfr

TrueDoc Portable Font Resource, typische Dateiendung: pfr

speedo

Speedo

intellifont

Intellifont

19.9 Element *font-face-name*

Mit dem als Kindelement von ***font-face-src*** notierbaren Element ***font-face-name*** kann dem Zeichensatz ein Name gegeben werden. Der Name wird im Attribut *name* notiert. Dieser Name kann dazu dienen, in der Eigenschaft oder im Präsentationsattribut *font-family* notiert zu werden, um diese Schrift zu verwenden. ***font-face*** selbst kann natürlich ebenfalls ein Attribut *font-family* mit der Angabe des Namens haben.

Das Element ***font-face-name*** ist in SVG tiny 1.2 nicht verfügbar.

19.10 Element *definition-src*

Das Element ***definition-src*** entspricht dem CSS2-Descriptor 'definition-src'. Es ist, sofern angegeben, direktes Kindelement von ***font-face***.

Das Element kann Attribute von XLink enthalten, insbesondere *href* ist relevant, um die Ressource zu referenzieren. Diese Ressource ist enthält Angaben zu: *panose-1*, *stemv*, *stemh*, *slope*, *cap-height*, *x-height*, *ascent* und *descent*.

Offenbar ist das entweder ein CSS-Dokument, wenn diese Syntax verwendet wird oder der Identifizierer für ein ***font-face*** mit den Angaben in den entsprechenden Attributen.

Das Element ***definition-src*** ist in SVG tiny 1.2 nicht verfügbar.

19.11 Element *glyph*

Squiggle (Batik)	1.7
Opera (Presto)	9 (teilweise: Attributvariante)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Das Element ***glyph*** ist ein direktes Kindelement des Elementes ***font***. Mit dem Element wird jeweils eine Glyphe eines Zeichensatzes definiert.

Das Koordinatensystem von ***glyph*** wird von den Attributen des Elternelementes ***font*** definiert.

In SVG tiny 1.2 wird exakt ein Pfad im Attribut *d* notiert, welches dem von ***path*** entspricht. In SVG 1.1 kann alternativ anderer graphischer Inhalt als Inhalt des Dokumentes notiert werden.

Enthält das Element in SVG 1.1 Kindelemente, entspricht das Verhalten dem vom ***use***, welches ein Symbol referenziert.

Eigenschaften werden nicht vom Elternelement geerbt, sondern von jenem Element, in dem der Text notiert ist, welcher mit den Glyphen dargestellt werden soll.

Ist ferner sowohl das Attribut *d* notiert als auch Kindelemente im ***glyph***, so wird zunächst der Pfad entsprechend *d* dargestellt, dann die Kindelemente.

19.11.1 Attribut *unicode*

Mit dem Attribut *unicode* wird eine Kombination von Unicode-Zeichen angegeben, die die Glyph repräsentieren soll.

Ist ein Unicode-Zeichen angegeben, so repräsentiert die Glyphe dieses Zeichen. Sind mehrere Zeichen angegeben, so repräsentiert die Glyphe diese Kombination. Dies ist eine Möglichkeit, eine Ligatur zu notieren. Zum Beispiel ergibt *unicode*="ffl" eine Glyphe, welche durch die drei aufeinanderfolgenden Zeichen 'f', 'f' und 'l' notiert wird. Das heißt, wenn im Text 'ffl' notiert wird, so gilt dies als eine einzige Glyphe.

Eine hexadezimale oder dezimale Notation ist ebenfalls möglich also "ffl" ist das gleiche wie "ffl" oder "ffl".

Das Attribut ist nicht animierbar.

19.11.2 Attribut *glyph-name*

Mit dem Attribut kann dem Glyphen ein Name gegeben werden. Namen von Glyphen sollten eindeutig innerhalt eines Zeichensatzes sein. Dies kann helfen, wenn die Unicode-Angabe nicht ausreichend ist, um die korrekte Glyphe zu bezeichnen. Das kann passieren, wenn es mehrere Glyphen für eine Unicode-Angabe gibt. Insbesondere kann dieser Name im Zusammenhang mit Angaben zur Proportionalschrift sein (englisch: kerning), weil es da sinnvoll sein kann, bei verschiedenen Kombinationen von Glyphen leichte Modifikationen vorzunehmen.

Der Wert ist eine Liste von Zeichenketten.

Das Attribut ist nicht animierbar.

19.11.3 Attribut *d*

Im Attribut *d* kann der Pfad für den Glyphen notiert werden. In SVG tiny 1.2 ist dies die einzige Möglichkeit, die Darstellung des Glyphen festzulegen.

Der Wert ist derselbe wie für das gleichnamige Attribut des Elementes ***path***.

Das Attribut ist nicht animierbar.

19.11.4 Attribut *arabic-form*

Bei arabischen Glyphen gibt es vier verschiedene Formen, in denen eine Glyphe repräsentiert werden kann. Mit diesem Attribut kann angegeben werden, welche davon gemeint ist.

Die möglichen Werte sind 'initial', 'medial', 'terminal' und 'isolated'. Sofern für eine arabische Glyphe nicht angegeben, für welche diese Information benötigt wird, wird 'isolated' angenommen. Das hat zur Folge, dass sofern nicht separat angegeben keine Glyphen für die anderen Formen verfügbar sind (und also Glyphe verwendet wird, die für fehlende Glyphen angegeben ist).

Ist die Angabe nicht erforderlich, hat sie keinen Effekt.

Das Attribut ist nicht animierbar.

19.11.5 Attribut *lang*

Es kann vorkommen, dass Glyphen nur für bestimmte Sprachen anwendbar sind. Mit dem Attribut wird angegeben, welche Sprachen anwendbar sind. Dies erfolgt mit einer Liste von Sprachmarkierungen als Wert des Attributes. Der Wert entspricht also dem von *xml:lang*. Die Glyphe wird verwendet, wenn beim Zieltext das Attribut *xml:lang* exakt mit einer der Sprachmarkierungen übereinstimmt. Die Glyphe findet auch Anwendung, wenn das Attribut *lang* nur ein Präfix angibt (also zum Beispiel 'de' und nicht 'de-at') und *xml:lang* dasselbe Präfix enthält.

Kurzum ist beim Text per *xml:lang* 'de' angegeben und in *lang* 'de-de en-gb', so ist die Glyphe nicht anwendbar. Umgekehrt ist per *xml:lang* 'de-at' angegeben und in *lang* 'de-de', so ist die Glyphe nicht anwendbar. Ist in *lang* 'de' angegeben, so ist die Glyphe auf alle deutschsprachigen Texte anwendbar, also zum Beispiel sowohl auf 'de-de' als auch 'de-at' etc.

Das Attribut ist nicht animierbar.

19.11.6 Attribute *horiz-adv-x* und *vert-adv-y*

Das Attribut *horiz-adv-x* gibt den Fortschritt in horizontaler Richtung nach der Darstellung des Glyphen an, also praktisch die Breite des für einen Glyphen reservierten Platzes. *vert-adv-y* entsprechend in vertikaler Richtung. Die Breite eines Glyphen darf nicht negativ sein, auch wenn die Schreibrichtung von rechts nach links erfolgt, wie bei hebräisch oder arabisch.

vert-adv-y ist in SVG tiny 1.2 nicht verfügbar.

Der Wert ist jeweils eine Zahl. Ist ein Attribut nicht angegeben, wird der Wert verwendet, der beim gleichnamigen Attribut von ***font*** notiert ist.

Die Attribute sind nicht animierbar.

19.11.7 Attribut *orientation*

Das Attribut wird nur verwendet für eine bestimmte Richtung, in der der Text fortschreitet. Mögliche Werte sind 'h' für horizontal und 'v' für vertikal. Die Glyphe wird dann nur verwendet, wenn der Text in der angegebenen Richtung fortschreitet. Ist das Attribut nicht angegeben, ist es für beide Richtungen verwendbar.

Das Attribut ist in SVG tiny 1.2 nicht verfügbar.

Das Attribut ist nicht animierbar.

19.11.8 Attribute *vert-origin-x* und *vert-origin-y*

Das Attribut *vert-origin-x* gibt die X-Koordinate des Ursprungs im Koordinatensystem des Glyphen an, welcher für vertikal ausgerichteten Text verwendet wird.

Das Attribut *vert-origin-y* gibt entsprechend die Y-Koordinate an. Die Attribute sind in SVG tiny 1.2 nicht verfügbar.

Der Wert ist jeweils eine Zahl. Ist das Attribut nicht angegeben, wird der Wert des gleichnamigen Attributes vom Elternelement verwendet.

Die Attribute sind nicht animierbar.

19.12 Element *missing-glyph*

Squiggle (Batik)	1.7
Opera (Presto)	9 (teilweise: Attributvariante)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Element wird die Präsentation der Glyphe definiert, die angezeigt wird, wenn keine passende Glyphe verfügbar ist.

Die Attribute sind dieselben wie für das Element ***glyph***.

Das Element ***missing-glyph*** ist ein direktes Kindelement des Elementes ***font***.

Das Koordinatensystem von ***missing-glyph*** wird von den Attributen des Elternelementes ***font*** definiert.

In SVG tiny 1.2 wird exakt ein Pfad im Attribut *d* notiert, welches dem von ***path*** entspricht. In SVG 1.1 kann alternativ anderer graphischer Inhalt als Inhalt des Dokumentes notiert werden.

Enthält das Element in SVG 1.1 Kindelemente, entspricht das Verhalten dem vom ***use***, welches ein Symbol referenziert.

Eigenschaften werden nicht vom Elternelement geerbt, sondern von jenem Element, in dem der Text notiert ist, welcher mit den Glyphen dargestellt werden soll.

19.13 Auswahl des richtigen Glyphen

Die Reihenfolge der Angabe der Glyphen ist entscheidend für die Auswahl. Der erste passende Glyph wird verwendet.

Gibt es zum Beispiel eine Ligatur für 'ff', so ist diese vor der Glyphe von 'f' zu notieren, sonst wird 'f' verwendet und nicht 'ff'.

Entsprechendes gilt für die Sprachauswahl.

Es ist also immer der speziellere Fall vor dem allgemeinen zu notieren, damit die speziellere Glyphe überhaupt verwendet wird.

19.14 Elemente *hkern* und *vkern*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mögliche weitere direkte Kindelemente von ***font*** sind ***hkern*** und ***vkern*** (letzteres nicht in SVG tiny 1.2 verfügbar).

Mit dem Element ***hkern*** wird das horizontale Unterschneiden von Glyphenpaaren bei Proportionalschriften definiert (englisch: kerning). Mit ***vkern*** entsprechend das vertikale.

Zum Beispiel bei Paaren wie 'Te', 'Ta', 'Av', 'aj' etc ist es üblich, solche Paare etwas dichter zusammenzusetzen als für die Glyphen unabhängig angegegeben. Bei 'Ga', 'Ge', 'Am', 'ag' wird hingegen eher der normale Abstand verwendet. Das Unterschneiden reduziert einen optisch ungleichmäßig wirkenden Eindruck, wenn bedingt durch die Form der Einzelglyphen zwischen diesen ungewöhnlich viel freier Platz bleibt.

Die Reihenfolge der anzugebenden Glyphen ist von links nach rechts, unabhängig von der späteren Schreibrichtung und wird dann auch von links nach rechts angewendet. Entscheidend bei der Anwendung ist also immer die Abfolge der Glyphen von links nach rechts, auch wenn die Schreibrichtung von rechts nach links sein sollte.

Entsprechend ist die Reihenfolge von ***vkern*** immer von oben nach unten.

19.14.1 Attribute *u1* und *u2*

u1 gibt den oder die ersten Glyphen für ein Überschneidungspaar an, *u2* die zweiten. Im einfachsten Falle ist der Wert jeweils ein Unicode-Zeichen. Es können allerdings auch mehrere oder Bereiche angegeben werden. Dann gelten die Angaben zur Überschneidungen für alle daraus resultierenden Paare von *u1* und *u2*.

Es gibt also die Möglichkeit, als Wert eine Liste von Zeichen oder Bereichen zu notieren, die mit einem Komma voneinander zu separieren sind. Details zur Notation siehe Attribut *unicode-range* des Elementes ***font-face***.

Die Attribute sind nicht animierbar.

19.14.2 Attribute *g1* und *g2*

Sofern den Glyphen Namen gegeben wurden, können auch diese mit *g1* und *g2* zur Angabe des oder der ersten beziehungsweise zweiten Glyphen für ein Überschneidungspaar verwendet werden.

Der Wert ist eine Liste von Glyphennamen, wie beim Element ***glyph*** mittels *glyph-name* festgelegt. Dann gelten die Angaben zur Überschneidungen für alle daraus resultierenden Paare von *g1* und *g2*.

Die Attribute sind nicht animierbar.

Es muss mindestens ein Attribut von *u1* und *g1* und eines von *u2* und *g2* notiert werden.

19.14.3 Attribut *k*

Mit dem Attribut *k* wird angegeben, um wie viel der Abstand zwischen dem Glyphenpaar für die Überschneidung verkleinert werden soll. Der Wert ist eine Zahl (in implizierten Einheiten im Koordinatensystem des Zeichensatzes).

Ein negativer Wert vergrößert also den Abstand zwischen den beiden Glyphen, ein positiver verkleinert ihn. Bei 0 bleibt er unverändert wie ohne Überschneidung.

Das Attribut ist nicht animierbar.

19.15 Alternative Glyphen, Element *altGlyph*

Squiggle (Batik)	1.7 (falsch implementiert)
Opera (Presto)	10 (falsch implementiert)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

In SVG 1.1 können alternative Glyphen definiert werden, wenn es notwendig ist, die normale Erscheinung von Glyphen durch eine andere Repräsentation zu ersetzen.

Zum Beispiel könnten zur Darstellung von Noten die korrespondierenden Buchstaben notiert werden, welche dann durch Glyphen ersetzt werden, welche die Noten darstellen. So kann die übliche graphische Repräsentation einer Melodie erreicht bleiben, wofür nur herkömmliche Buchstaben und Zahlen im Quelltext stehen, was die Melodie leichter editierbar macht und leichter ohne spezielle Programme kopierbar.

Für Text, welcher im Element ***altGlyph*** notiert wird, können alternative Glyphen verwendet werden. Folglich ist ***altGlyph*** in einem Textelement zu notieren, also ***text***, ***tspan***, ***textPath***.

19.15.1 Attribut *href* von XLink

Mit dem Attribut *href* von XLink kann ein Element ***glyph*** oder ***altGlyphDef*** referenziert werden, welches die alternativen Glyphen repräsentiert. Dazu wird als Wert der Fragmentidentifizierer des entsprechenden Elementes als Verweisziel notiert.

Mit *href* sind auch die anderen Attribute von XLink verfügbar.

Wird ein ***glyph*** referenziert, wird diese Glyphe anstatt jener in ***altGlyph*** angegebenen Zeichen verwendet, die entsprechend passen.

Wird ein ***altGlyphDef*** referenziert, wird ein entsprechender Satz von Glyphen von diesem Element verwendet anstatt jener in ***altGlyph*** angegebenen Zeichen, die entsprechend passen.

Sofern *href* notiert ist, wird dies zur Identifikation der Glyphe verwendet. *glyphRef* wird dann ignoriert.

href ist für dieses Element nicht animierbar.

19.15.2 Attribut *glyphRef*

Das Attribut identifiziert die zu ersetzende Glyphe. Der Wert ist vom verwendeten Format abhängig.

Das Attribut ist nicht animierbar.

19.15.3 Attribut *format*

Das Attribut gibt an, welches Format verwendet wird. Die Bedeutung ist die gleiche wie für das gleichnamige Attribut von ***font-face-format***.

Das Attribut ist nicht animierbar.

19.15.4 Attribute *x* und *y*

Die Attribute geben die x- beziehungsweise y-Koordinaten der Glyphen an. Die Bedeutung ist die gleiche wie die der gleichnamigen Attribute von ***tspan***. Die Werte ersetzen bei den zu ersetzenden Glyphen die mit ***tspan*** oder ***text*** definierten Positionen.

Die Attribute sind animierbar.

19.15.5 Attribute *dx* und *dy*

Die Attribute geben die x- beziehungsweise y-Verschiebungen der Glyphen an. Die Bedeutung ist die gleiche wie die der gleichnamigen Attribute von ***tspan***. Die Werte ersetzen bei den zu ersetzenden Glyphen die mit ***tspan*** oder ***text*** definierten Verschiebungen.

Die Attribute sind animierbar.

19.15.6 Attribut *rotate*

Das Attribut gibt die Drehung der Glyphen an. Die Bedeutung ist die gleiche wie die des gleichnamigen Attributes von ***tspan***. Die Werte ersetzen bei den zu ersetzenden Glyphen die mit ***tspan*** oder ***text*** definierten Drehungen.

Das Attribut ist animierbar (nicht additiv).

19.16 Element *altGlyphDef*

Das Element ***altGlyphDef*** definiert einen Satz möglicher alternativer Glyphen zur Ersetzung. Als nicht direkt dargestellter Inhalt empfhielt es sich, dies Element innerhalb von ***defs*** zu notieren.

Im einfachsten Falle enthält ***altGlyphDef*** eines oder mehrere Elemente ***glyphRef***. Jedes ***glyphRef*** referenziert einen einzelnen Glyphen einer bestimmten Schrift. Die referenzierten Glyphen werden dann anstatt der Zeichen verwendet, welche im referenzierenden Element ***altGlyph*** notiert sind, sofern für alle eine Alternative verfügbar ist. Sind nicht für alle Zeichen Ersatzglyphen vorhanden, ist das Verhalten das gleiche, als sei kein ***altGlyph*** angegeben, die Zeichen werden also mit dem aktuell gültigen Zeichensatz dargestellt.

Im komplizierteren Falle enthält ***altGlyphDef*** eines oder mehrere Elemente ***altGlyphItem***. Jedes ***altGlyphItem*** enthält eines oder mehrere Elemente ***glyphRef***. Jedes ***glyphRef*** referenziert einen einzelnen Glyphen einer bestimmten Schrift.

Es wird das erste ***altGlyphItem*** verwendet, mit dem alle angegebenen Zeichen ersetzt werden können. Gibt es einen solchen Satz nicht, so ist das Verhalten das gleiche, als sei kein ***altGlyph*** angegeben, die Zeichen werden also mit dem aktuell gültigen Zeichensatz dargestellt.

Die referenzierten Glyphen werden sofern anwendbar dann anstatt der Zeichen verwendet, welche im referenzierenden Element ***altGlyph*** notiert sind.

19.17 Element *altGlyphItem*

Das Element ***altGlyphItem*** definiert einen Satz möglicher alternativer Glyphen zur Ersetzung. Der erste passende Satz wird verwendet. Die referenzierten Glyphen werden dann anstatt der Zeichen verwendet, welche im referenzierenden Element ***altGlyph*** notiert sind.

19.18 Element *glyphRef*

Das Element definiert einen gegebenenfalls verwendbaren Glyphen.

19.18.1 Attribut *href* von XLink

Mit dem Attribut *href* von XLink wird ein Element ***glyph*** referenziert, welche die alternativen Glyphe repräsentiert. Dazu wird als Wert der Fragmentidentifizierer des entsprechenden Elementes als Verweisziel notiert.

Mit *href* sind auch die anderen Attribute von XLink verfügbar.

Sofern *href* notiert ist, wird dies zur Identifikation der Glyphe verwendet. *glyphRef* wird dann ignoriert.

href ist für dieses Element nicht animierbar.

19.18.2 Attribut *glyphRef*

Das Attribut identifiziert die zu ersetzende Glyphe. Der Wert ist vom verwendeten Format abhängig.

Das Attribut ist nicht animierbar.

19.18.3 Attribut *format*

Das Attribut gibt an, welches Format verwendet wird. Die Bedeutung ist die gleiche wie für das gleichnamige Attribut von ***font-face-format***.

Das Attribut ist nicht animierbar.

19.18.4 Attribute *x* und *y*

Die Attribute gibt eine neue x- beziehungsweise y-Koordinate des Glyphen im Koordinatensystem des Zeichensatzes an.

Die Attribute sind nicht animierbar.

19.18.5 Attribute *dx* und *dy*

Die Attribute geben die x- beziehungsweise y-Verschiebungen des Glyphen im Koordinatensystem des Zeichensatzes an.

Die Attribute sind nicht animierbar.

19.19 Beispiele alternative Glyphen

Die Glyphen für ! und ? werden ersetzt:

```
<svg viewBox="0 0 400 150"
  xmlns="http://www.w3.org/2000/svg" version="1.1"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xml:lang="de">
<title>Element altGlyph</title>
<!-- desc ... -->
<defs>
   <font id="F2" horiz-adv-x="500">
      <font-face font-family="Beispiel"
          units-per-em="1000" cap-height="600" x-height="400"
          ascent="700" descent="300"
          alphabetic="0" mathematical="350" ideographic="400" hanging="500" />
      <missing-glyph d="M100,100h100v100h-100z" />
      <glyph id="glyph1" unicode="!" d="M100,0 h200v400h-200z
 M100,-300h200v200h-200z" />
      <glyph id="glyph2" unicode="?" d="M50,0 h300l-100 400h-100z
 M150,-300h100v100h-100z" />
   </font>
   <altGlyphDef id="agd1">
     <glyphRef xlink:href="#glyph1" />
     <glyphRef xlink:href="#glyph2" />
   </altGlyphDef>
</defs>

<text x="100" y="100" font-size="20" font-family="serif">
<!-- wird nicht ersetzt, weil einige Ersatzglyphen fehlen -->
<altGlyph xlink:href="#agd1" x="0" y="40">SVG!?! -*</altGlyph>
<!-- wird ersetzt -->
<altGlyph xlink:href="#agd1" x="0" y="80">!??!!</altGlyph>
</text>

<!-- mehr ... -->
</svg>
```

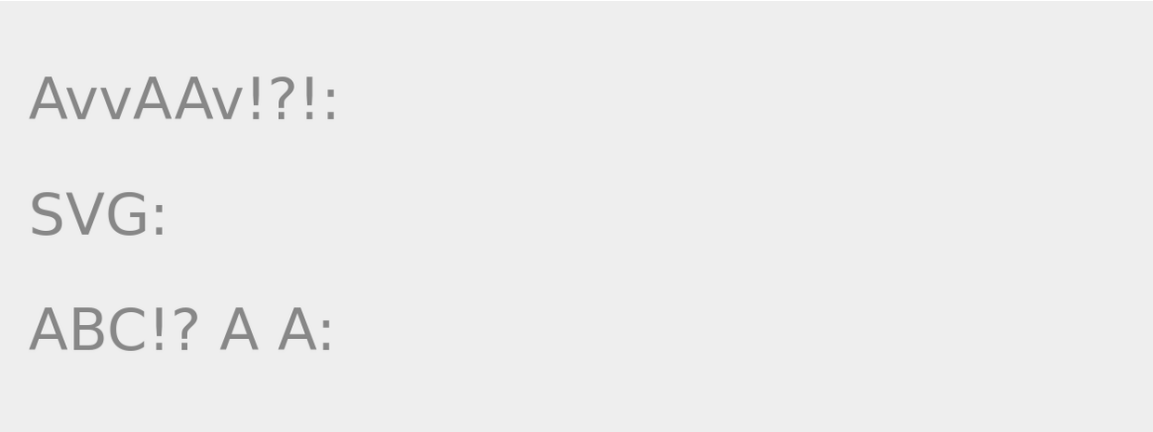


Abb. 300 Beispiel alternative Glyphen

Beispiel alternative Glyphen[2]

2 http://de.wikibooks.org/wiki/media%3ASVGaltGlyph01.svg

Mit der einfachen Methode werden gegebenenfalls Glyphen durch alternative Glyphen ersetzt. Links in grau ist der jeweilige Testtext angegeben, rechts daneben das Resultat. Nur Testtexte, die ausschließlich die Zeichen A, v, ! und ? enthalten, sind zu ersetzen.

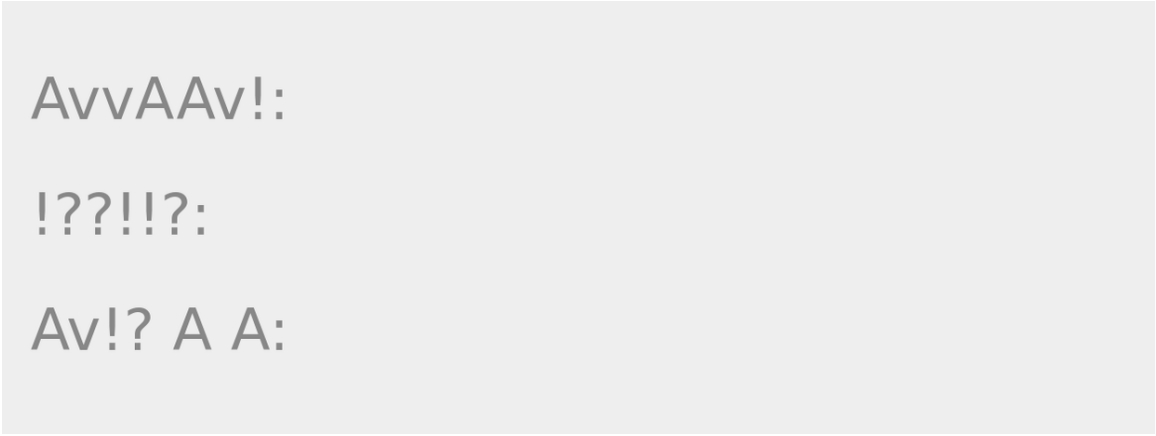


Abb. 301 Beispiel (2) alternative Glyphen

Beispiel (2) alternative Glyphen[3]

Mit der komplizierteren Methode mit Sätzen von Glyphen werden gegebenenfalls Glyphen durch alternative Glyphen ersetzt. Links in grau ist der jeweilige Testtext angegeben, rechts daneben das Resultat. Wegen fehlender Glyphen wird der Text 'Av!?' nicht ersetzt, die anderen schon.

19.20 Literatur

- deutsche Übersetzung CSS2, Kapitel Schriften[4]
- Schriften in SVG 1.1[5] (en)
- Schriften in SVG tiny 1.2[6] (en)
- Über Schriften in SVG[7] (en)

3 http://de.wikibooks.org/wiki/media%3ASVGaltGlyph02.svg
4 http://www.edition-w3.de/TR/1998/REC-CSS2-19980512/kap15.html
5 http://www.w3.org/TR/SVG11/fonts.html
6 http://www.w3.org/TR/SVGTiny12/fonts.html
7 http://nimbupani.com/about-fonts-in-svg.html

20 Filter-Effekte

20.1 Filtereffekte mit SVG

SVG-Filter sind ein mächtiges Werkzeug, um in SVG Licht und Schatten, einfache 3D-Effekte, Plastizität, Weichzeichner und vieles mehr zu realisieren. Filter lassen sich beliebig verschachteln und kombinieren.

Ausgehend von darstellbaren SVG-Elementen, allgemeiner einer Quellgraphik, wird die graphische Repräsentation für einen Filtereffekt in einer Zwischenablage abgelegt. Von solch einer Vorlage werden Farbkanäle und Transparenz verwendet, um die Präsentation nach der Anwendung des Filters zu berechnen, welcher also eine Änderung der Präsentation bewirkt.

Filter sind so definiert, dass ein Autor sehr leicht die inhaltliche Struktur und Information des Dokumentes beibehalten und trotzdem die graphische Präsentation des Inhaltes mit Filtern 'aufhübschen' kann. Ähnlich wie bei Stilvorlagen und Skripten liegt der Trick darin, das Dokument mit der eigentlichen inhaltlichen Information zunächst komplett und ohne Einsatz von Filtern fertigzustellen und dann die Filter in einem zweiten Schritt hinzuzufügen. So bleibt das Dokument und die beabsichtigte Information, vor allem Textinhalt auch mit solchen Darstellungsprogrammen zugänglich, die keine Filter interpretieren.

Weil Filtereffekte praktisch pixelweise in der Auflösung des Bildschirmes berechnet werden müssen, ist dies für ein Darstellungsprogramm eine vergleichsweise anspruchsvolle Aufgabe, welche hohe Anforderungen an den Prozessor stellen kann. Nicht immer haben die Darstellungsprogramme auch einen Zugriff auf leistungsfähige Graphikprozessoren, die derartige Filter viel effektiver berechnen können als herkömmliche Prozessoren, was eine weitere Komplikation darstellt.

Daher können Filtereffekte (eventuell auch noch kombiniert mit Animationen oder Skripten) Darstellungsprogramme auf alten Rechnern oder mobilen Geräten schnell überfordern, weswegen Filter in den dafür gedachten Profilen gar nicht oder nur eingeschränkt verfügbar sind. Damit auch mit diesen Geräten eine Darstellung von Dokumenten sinnvoll möglich ist, welche Filter verwenden, ist dabei immer bereits beschriebenes Vorgehen zu empfehlen, bei dem der Inhalt auch ohne Interpretation von Filtern verfügbar bleibt.

Folgendes Beispiel zeigt, wie sich ein Filter auf eine Quellgraphik auswirkt. Links die Quellgraphik, rechts das Ergebnis nach Anwendung eines Filters, welcher Unschärfe bewirkt.

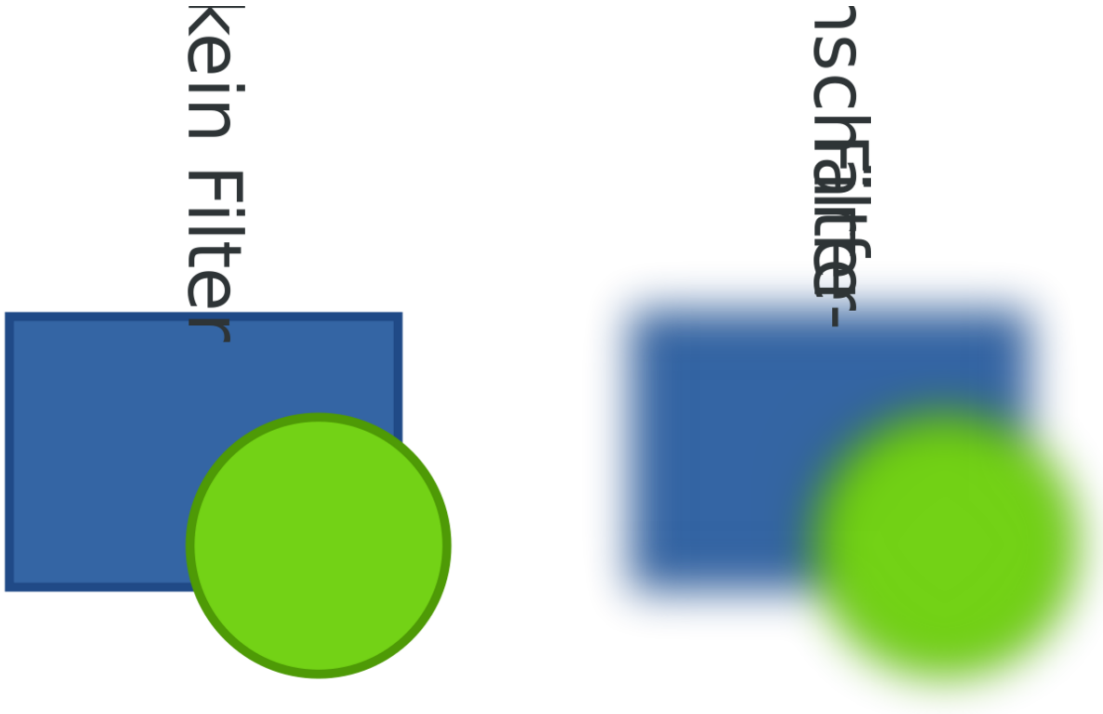


Abb. 302 Quellgraphik und Ergebnis eines Gaußfilters

In den tiny-Varianten von SVG sind also Filter nicht vorgesehen, in der basic-Variante von SVG 1.1 werden nur einige definiert. Nur die Vollversion von SVG 1.1 bietet alle Möglichkeiten.

Funktion	Elemente	SVG Basic
Kombination, Überblendung	*feBlend*	✓**Ja**
Farb-Matrix	*feColorMatrix*	✓**Ja**
Farbkanaltransferfunktionen	*feComponentTransfer*, *feFuncR*, *feFuncG*, *feFuncB*, *feFuncA*	✓**Ja**
Komposition	*feComposite*	✓**Ja**
Faltung	*feConvolveMatrix*	✗**Nein**
Diffuse Beleuchtung	*feDiffuseLighting*	✗**Nein**
Lokale Verschiebung	*feDisplacementMap*	✗**Nein**
Gleichmäßige Füllung	*feFlood*	✓**Ja**
Gaußscher Weichzeichner	*feGaussianBlur*	✓**Ja**
Bild einfügen	*feImage*	✓**Ja**
Zusammenfügen, fusionieren	*feMerge*, *feMergeNode*	✓**Ja**
Verdicken oder verdünnen	*feMorphology*	✗**Nein**
Räumlicher Versatz	*feOffset*	✓**Ja**

Glanzlicht	***feSpecularLighting***	**✗Nein**
Kacheln	***feTile***	**✓Ja**
Turbulenz	***feTurbulence***	**✗Nein**

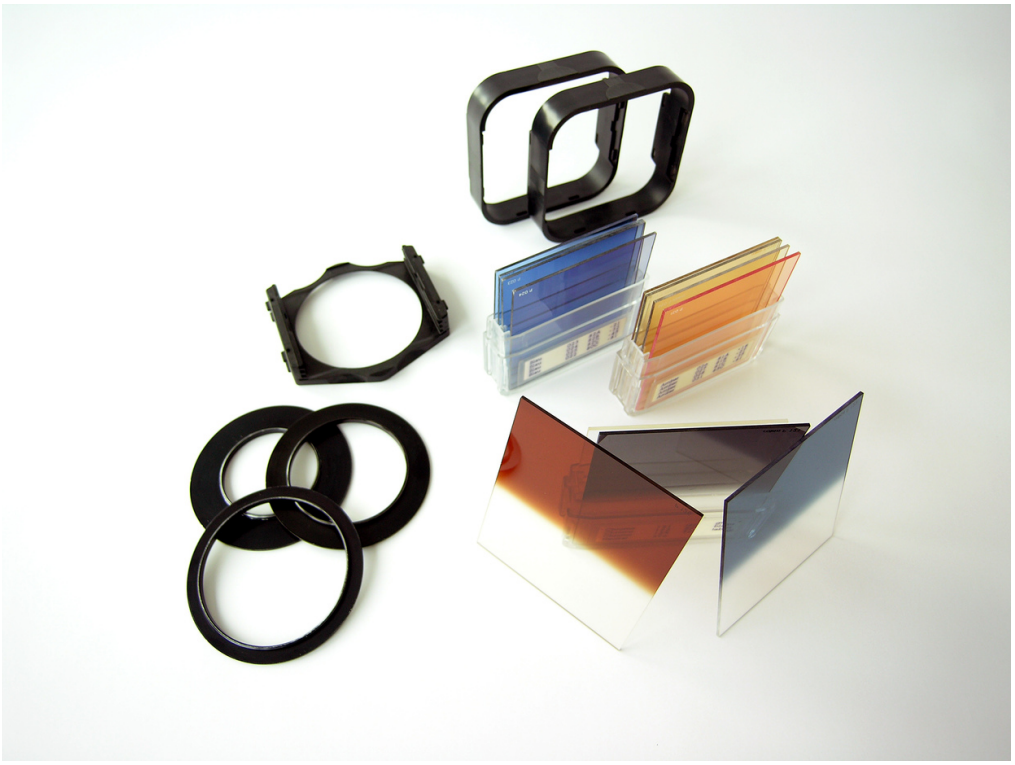


Abb. 303 Filtersystem für Kameras

Als nicht direkt dargestellte Elemente sollten SVG-Filter innerhalb des Elementes ***defs*** notiert werden. Die jeweiligen Einzelfilter sind im Element ***filter*** untergebracht. Soll ein derart definierter Filter auf ein graphisches Element oder ein Gruppierungselement angewendet werden, so wird dafür die Eigenschaft oder das Präsentationsattribut *filter* bei jeweiligen Element verwendet, um den Filter zu referenzieren.

Ein Filter in SVG hat eine gewisse Analogie zu einem Filter in der Photographie, als eine Glasplatte, die vor das Objektiv geschraubt oder gehalten wird, um einen bestimmten optischen Effekt zu erreichen.

Die Ausgabe eines Filters ist eine Pixelgraphik, die allerdings je nach Vergrößerung neu berechnet wird, so dass ein pixelige Wiedergabe weitestgehend vermieden wird.

20.2 Eingangsbeispiel

Abb. 304 Eingangsgraphik dieses Buches

Abb. 305 Variation zum Eingangsbeispiel

Folgendes Beispiel entspricht der Eingangsgraphik dieses Buches[1]. Der profane Inhalt besteht aus Ellipsen und dem Text 'SVG', in der Mitte der Ellipsen angeordnet. Die Füllung von zwei Ellipsen ist teiltransparent weiß ausgelegt.

Die Filter werden jeweils mit dem Element ***filter*** umschlossen. Filter sind immer rechteckig, somit gibt es dafür die entsprechenden Attribute. In diesem Beispiel gibt es zwei ***filter***.

Der erste Filter ist ein einfacher Weichzeichner für zwei Ellipsen. Darüber werden nochmal die gleichen Ellipsen ohne Filter gesetzt, was hier explizit erfolgt, aber auch mit einem Filter erledigt werden könnte. Der Effekt ist ein etwas weicherer Rand für die beiden Ellipsen.

Der zweite Filter ist deutlich komplizierter. Dieser täuscht eine Lichtquelle mit Schattenwurf, so dass der Eindruck erweckt wird, als schwebe der Inhalt über der Zeichenfläche und werfe einen Schatten auf diese. Ferner wird eine reliefartige Oberflächenstruktur vorgetäuscht, einschließlich einiger Glanzlichter von der Lichtquelle, links oben angeordnet.

Dazu wird zunächst der Alpha-Kanal des Inhaltes verwendet, also der Kanal für Transparenz, dies Bild wird mit einem Weichzeichner behandelt und nach unten rechts verschoben, grob den Effekt einer unendlich entfernten, ausgedehnten Lichtquelle simulierend. Die teiltransparente Füllung sorgt lediglich für eine Abschwächung des Lichtes, während die Striche und der Text komplett undurchsichtig sind. Bedingt durch den Weichzeichner ergibt sich allerdings kein scharfer Schatten, also eine Simulation einer ausgedehnten Lichtquelle, welche sehr weit weg ist, wie die Sonne.

1 `http://de.wikibooks.org/wiki/media%3ASVG_Buch01.svg`

Ein weiteres Abbild des Inhaltes wird mit einem Weichzeichner behandelt. Dies wird verwendet, um Glanzlichter eine Punktlichtquelle und eine Oberflächenstruktur vorzutäuschen. Von dem Ergebnis wird aber nur der Alpha-Kanal verwendet. Dieser wird schließlich vereint mit dem Schatten, welcher bereits zuvor mit der Originalgraphik vereint wurde.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg viewBox="0 150 1000 700"
version="1.1"
xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink"
xml:lang="de">

<title>SVG</title>
<desc>
...
</desc>

<defs>
<filter id="weich"
  filterUnits="userSpaceOnUse"
  x="0" y="150" width="1000" height="700">
  <feGaussianBlur in="Source" stdDeviation="2" />
</filter>

<filter id="LichtUndSchatten"
  filterUnits="userSpaceOnUse"
  x="0" y="150" width="1000" height="700">

<feGaussianBlur in="SourceAlpha" stdDeviation="12"
x="0" y="150" width="1000" height="700"  result="Gauss" />
<!-- der diffuse Schatten wird aus der Alphamaske erzeugt
     und nach rechts unten verschoben -->
<feOffset in="Gauss" dx="20" dy="40" result="OffsetGauss"/>

<!-- Der diffuse Schatten wird mit der Graphik vereint -->
<feComposite in="SourceGraphic" in2="OffsetGauss"
   operator="over" result="Composite"/>

<!-- Ein weiteres diffuses Abbild wird erstellt... -->
<feGaussianBlur in="SourceAlpha"
   stdDeviation="6" result="blur"/>

<!-- ... und mit Glanzlichtern versehen... -->
<feSpecularLighting in="blur" surfaceScale="4"
   specularConstant=".8" specularExponent="16"
   lighting-color="#def" result="specOut">
<!-- ... und eine Punktlichtquelle wird positioniert -->
    <fePointLight x="-500" y="-1000" z="1200"/>
  </feSpecularLighting>

<!-- Die Glanzlichter werden mit der Alphamaske maskiert...  -->
<feComposite in="specOut" in2="SourceAlpha"
   operator="in" result="specOut2"/>
<!-- ... und dieses wird mit der Komposition
     aus Graphik und Schatten vereint  -->
<feComposite in="Composite" in2="specOut2"
   operator="arithmetic" k1="0" k2="1" k3="1" k4="0"
   result="litPaint"/>
</filter>
</defs>

<g filter="url(#LichtUndSchatten)">

<g filter="url(#weich)">
```

```
  <ellipse cx="480" cy="480" rx="430" ry="290"
    fill="#fff" fill-opacity="0.2"
    stroke="#04a" stroke-width="10" />
  <ellipse cx="480" cy="480" rx="420" ry="280"
    fill="none"
    stroke="#039" stroke-width="5" />
</g>

<ellipse cx="480" cy="480" rx="430" ry="290"
    fill="#fff" fill-opacity="0.2"
    stroke="#04a" stroke-width="8" />
<ellipse cx="480" cy="480" rx="420" ry="280"
    fill="none"
    stroke="#039" stroke-width="3" />

<text x="480" y="600" text-anchor="middle"
  font-size="336" stroke-linecap="square"
  stroke-width="12" stroke="#035"
  fill="none">SVG</text>
</g>

</svg>
```

Variation zum Eingangsbeispiel[2]

Hier werden etwas andere Attributwerte verwendet, andere Filter, zudem wird die Farbe rotiert.

20.3 Element *filter*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (teilweise)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Das Element ***filter*** kann verwendet werden, um darin eine Gruppe von Filtereffekten zu notieren. Das Element legt Größe und Lage des rechteckigen Filterbereiches fest.

filter werden nicht direkt dargestellt, sollten daher in ***defs*** notiert werden. Solch ein Filter wird angewendet, indem er mit der Eigenschaft *filter* beim betroffenen Element referenziert wird. Als nicht direkt dargestelltes Element hat die Eigenschaft *display* keinen Einfluss auf die Funktion und Verfügbarkeit des Filters.

Eigenschaften werden vom Elternelement geerbt, nicht vom referenzierenden Element.

Die enthaltenen Urfilter werden der Reihe nach abgearbeitet. Die Ausgabe des letzten wird als Ausgabe verwendet.

20.3.1 Attribut *filterUnits*

Das Attribut definiert das Einheitensystem für die Attribute *x*, *y*, *width*, *height*.

2 http://de.wikibooks.org/wiki/media%3ASVG_Filtereffekte01.svg

Mögliche Werte sind 'userSpaceOnUse' und 'objectBoundingBox'; letzteres wird angenommen, wenn das Attribut nicht angegeben ist. Das Attribut ist animierbar.

Bei 'userSpaceOnUse' wird das lokale Koordinatensystem verwendet, welches für das referenzierende Element gilt.

Bei 'objectBoundingBox' bestimmt das umgebende Rechteck (englisch: boundingBox) die Einheiten.

20.3.2 Attribut *primitiveUnits*

Das Attribut bestimmt das Einheitensystem für die Urfilter und für die Attribute, welche die Filterregion definieren.

Mögliche Werte sind 'userSpaceOnUse' und 'objectBoundingBox'; ersteres wird angenommen, wenn das Attribut nicht angegeben ist. Das Attribut ist animierbar.

Bei 'userSpaceOnUse' wird das lokale Koordinatensystem verwendet, welches für das referenzierende Element gilt.

Bei 'objectBoundingBox' bestimmt das umgebende Rechteck (englisch: boundingBox) die Einheiten.

20.3.3 Attribute *x*, *y*, *width* und *height*

Die Attribute definieren einen rechteckigen Bereich der Zeichenebene, auf welchen der Filter anzuwenden ist.

x ist der linke Rand der Region, *y* der obere Rand, *width* die Breite, *height* die Höhe. *x* und *y* sind Koordinaten, *width* und *height* sind Längen. Negative Werte für *width* und *height* sind ein Fehler, welcher zum Abbruch der Darstellung des Dokumentes führt. Ein Wert '0' unterbindet die Anzeige des Filterergebnisses, also letztlich die Darstellung des referenzierenden Elementes.

Sind *x* oder *y* nicht angegeben, wird für das fehlende Attribut jeweils '-10%' angenommen. Sind *width* oder *height* nicht angegeben, wird für das fehlende Attribut jeweils '120%' angenommen.

Die Attribute sind animierbar.

Die benötigte Verarbeitungszeit und der notwendige Speicher für einen Filter sind direkt abhängig von der Filterfläche, also dem Produkt und Breite und Höhe. Verarbeitungszeit und Speicher hängen natürlich auch von der Anzahl und der Art der verwendeten Urfilter ab. Um den Aufwand für einen Filtereffekt so gering wir möglich zu halten, ist es also dringend zu empfehlen, den Filterbereich so klein wie möglich zu wählen. Andererseits gibt es Urfilter, die Bereich benötigen, die außerhalb des zu filternden Inhaltes liegen, wie etwa Weichzeichner. Für diese ist ein etwas größerer Bereich zu verwenden, sonst werden Teile des Effektes abgeschnitten.

Ebenfalls eine Verlängerung der Verarbeitungszeit kann sich aus bestimmten Transformationen des Koordinatensystems ergeben, welches für die Filter anzuwenden ist, wie Scherungen, Drehungen um andere Winkel als ganzzahlige vielfache von 90 Grad und andere Transformationen, bei denen die Achsen nicht exakt entlang der x- und y-Achsen des Ausgabegerätes ausgerichtet sind.

20.3.4 Attribut *filterRes*

Mit dem Attribut wird die Auflösung von Zwischenablagen festgelegt, die benötigt werden, um Zwischenergebnisse von Filtereffekten abzulegen. Hohe Auflösung bedeutet einen großen Speichergebrauch und eine hohe Prozessorlast.

Der Wert ist eine Zahl, optional gefolgt von einer weiteren Zahl (mit Leerzeichen dazwischen). Die Zahl bedeutet jeweils die Anzahl der Pixel für die Breite und die Höhe bei einer Zahl. Bei zwei Zahlen bedeutet die erste Zahl die Anzahl der Pixel für die Breite, die zweite für die Höhe.

Voreinstellung für Monitore ist die Anzahl der Pixel auf dem Monitor für die Abmessung des Filterbereiches. Bei Druckern ist eine angemessene Auflösung als Voreinstellung anzunehmen, also zum Beispiel eine Angabe wie die Anzahl der Pixel, die 400dpi entsprechen.

Mit einer Angabe von weniger Pixeln kann ein pixeliger Effekt provoziert werden, wo also die Pixel der Zwischenablage klar zu erkennen sind. Größere Werte können die Verarbeitung unangemessen verlängern wie auch den Speicherverbrauch drastisch ansteigen lassen.

Negative Werte sind ein Fehler, welcher zum Abbruch der Darstellung des Dokumentes führt. Ein Wert '0' unterbindet die Anzeige des Filterergebnisses, also letztlich die Darstellung des referenzierenden Elementes.

Das Attribut ist animierbar. Ein Wechsel von niedriger Pixelzahl zur Voreinstellung kann einen Effekt wie das Einblenden des Inhaltes bewirken, umgekehrt entsprechend ein Ausblenden.

20.3.5 Attribut *href* von XLink

Mit dem Attribut *href* von XLink kann ein anderer Filter im selben Dokumentfragment referenziert werden, welcher verwendet werden kann. Sind Attribute im referenzierten Filter, aber nicht im referenzierenden angegeben, so werden sie übernommen, sonst nicht. Falls der referenzierende Filter keine Urfilter enthält, aber der referenzierte oder dieser selbst einen Filter referenziert, dann erbt der referenzierende Filter die Urfilter.

Das Attribut ist anmierbar.

Mit *href* sind auch die anderen Attribute von XLink verfügbar.

20.3.6 Beispiele *filter*

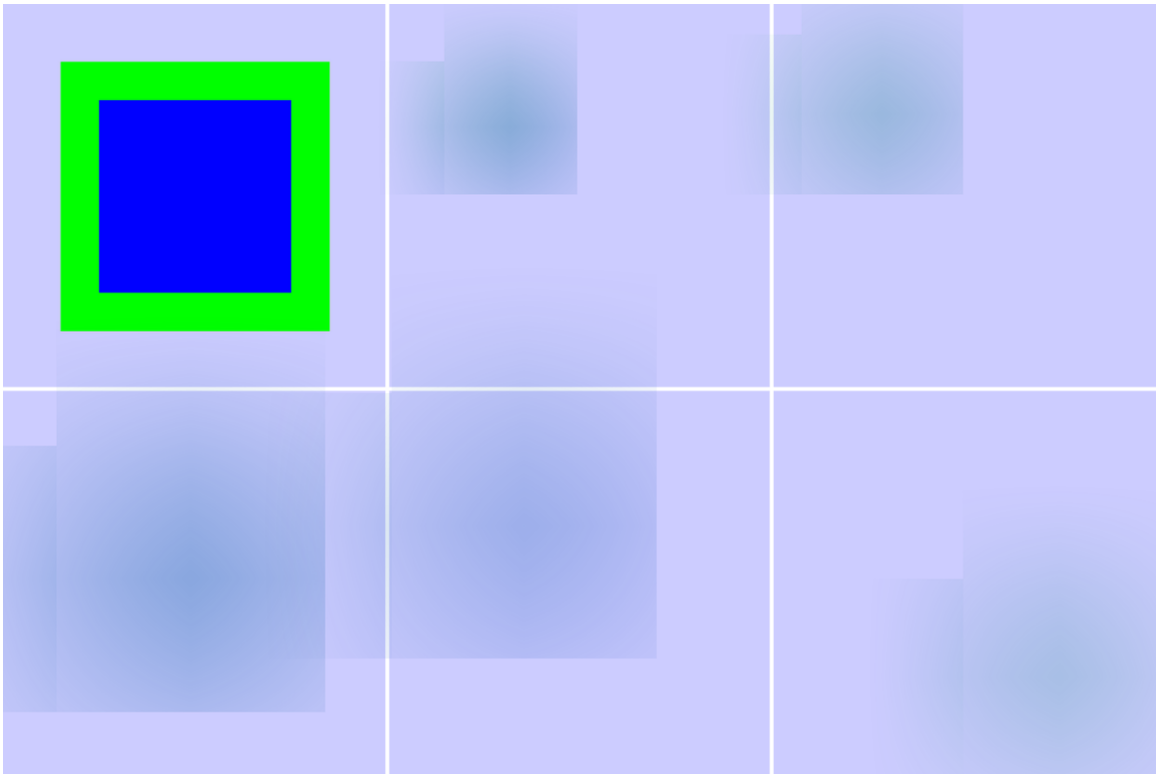


Abb. 306 relative Koordinaten (1)

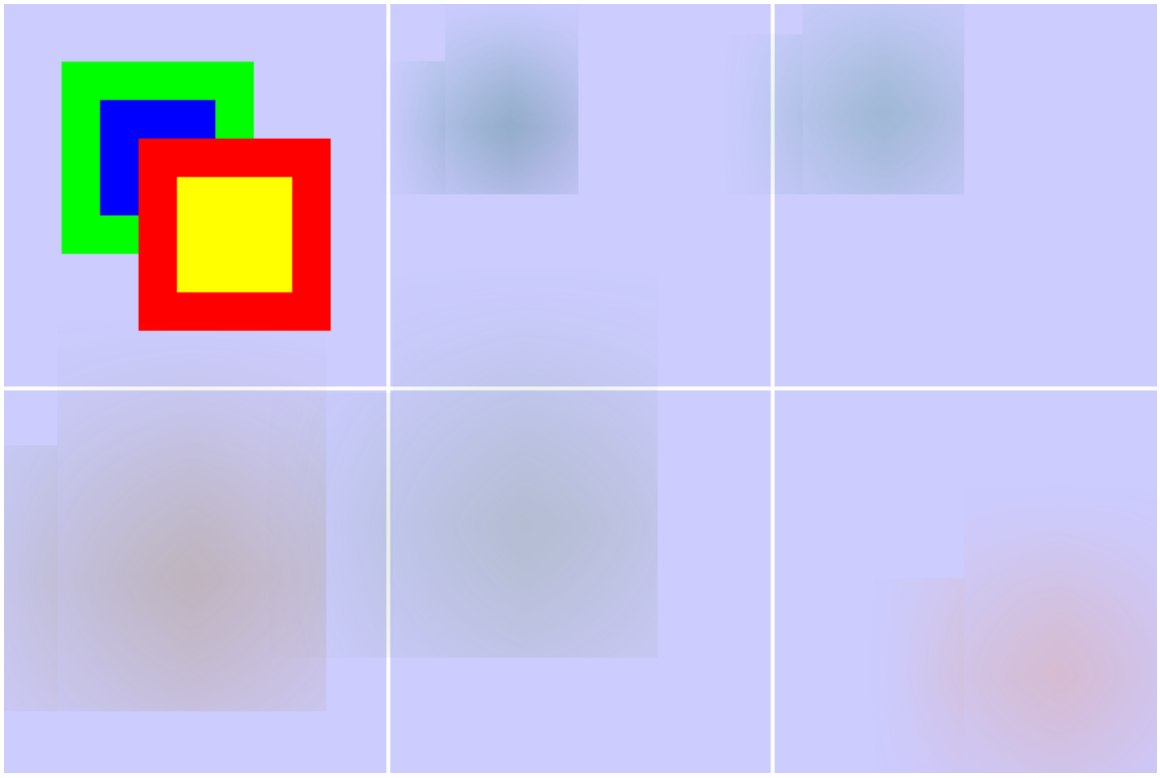


Abb. 307 relative Koordinaten (2)

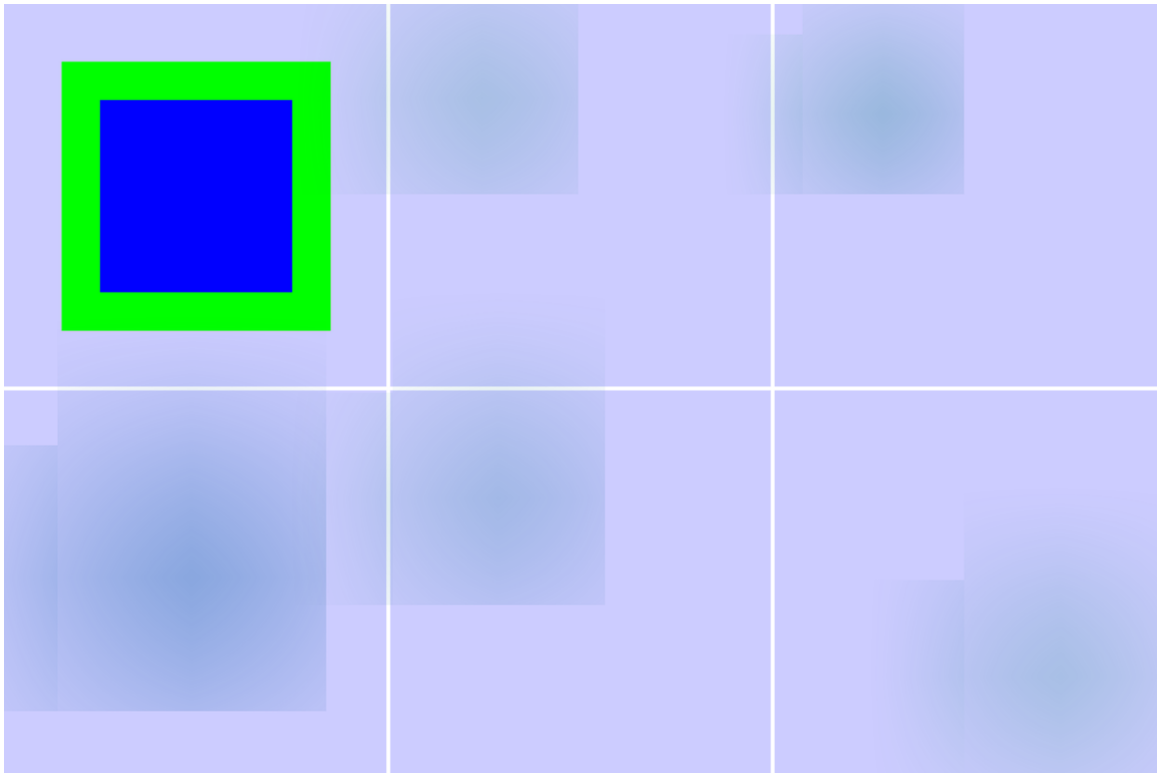


Abb. 308 lokale Koordinaten (1)

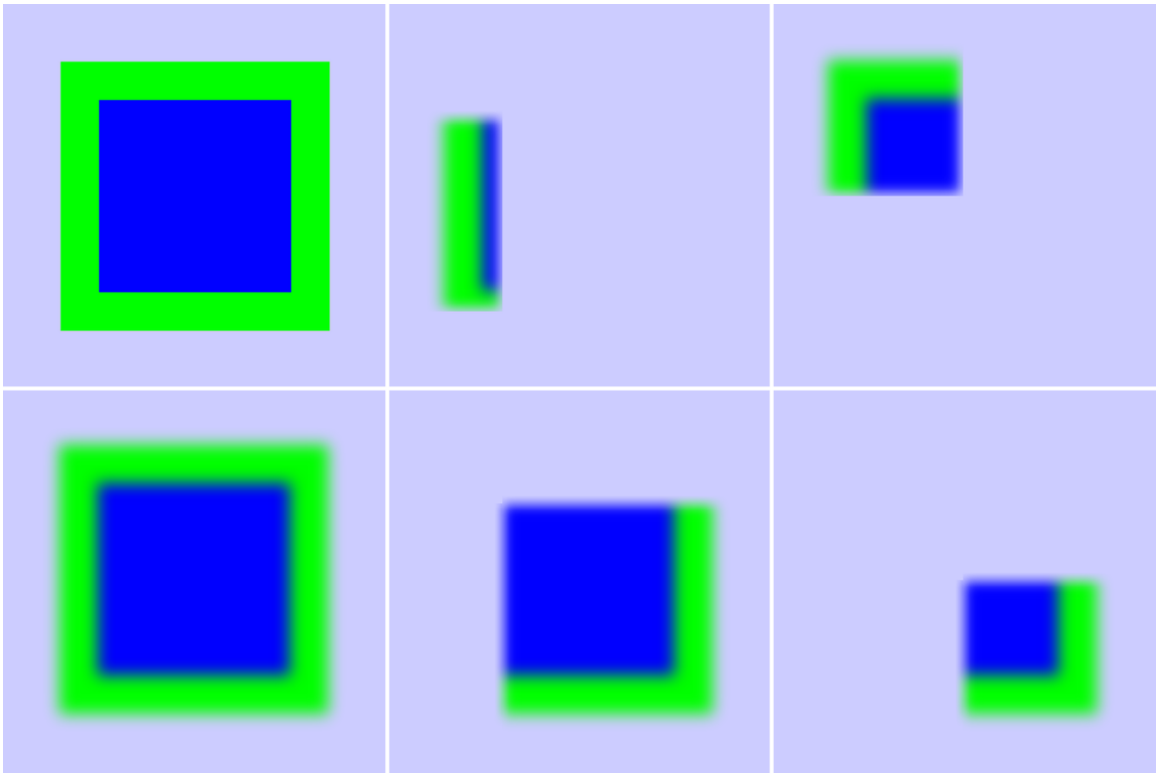


Abb. 309 lokale Koordinaten (2)

In den Abschnitten zu den Urfiltern finden sich zusätzlich zahlreiche weitere Beispiele. Hier gibt es nur ein paar einfache Beispiele zu den Attributen von ***filter***, wobei lediglich ein einfacher Gaußschner Weichzeichner als Urfilter verwendet wird. Es kann also für das Verständnis der Attribute des Urfilters nützlich sein, die entsprechenden, noch folgenden Abschnitte zu kennen, dies ist aber nicht Schwerpunkt dieser Beispiele.

Links oben ist jeweils das ungefilterte Rechteck zum Vergleich angegeben.

relative Koordinaten (1)[3]

Auswirkung verschiedener Filterregionen und Auflösungen auf ein Rechteck mit Koordinatenangaben relativ zur Größe des Elementes.

relative Koordinaten (2)[4]

Auswirkung verschiedener Filterregionen und Auflösungen auf eine Gruppe mit zwei Rechtecken mit Koordinatenangaben relativ zur Größe des Elementes.

3 `http://de.wikibooks.org/wiki/media%3ASVGfilter01.svg`
4 `http://de.wikibooks.org/wiki/media%3ASVGfilter02.svg`

lokale Koordinaten (1)[5]

Auswirkung verschiedener Filterregionen in lokalen Koordinaten auf ein Rechteck. Der Urfilter verwendet nach wie vor relative Koordinaten.

lokale Koordinaten (2)[6]

Auswirkung verschiedener Filterregionen in lokalen Koordinaten auf ein Rechteck. Auch der Urfilter verwendet hier lokale Koordinaten.

20.4 Eigenschaft *filter*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

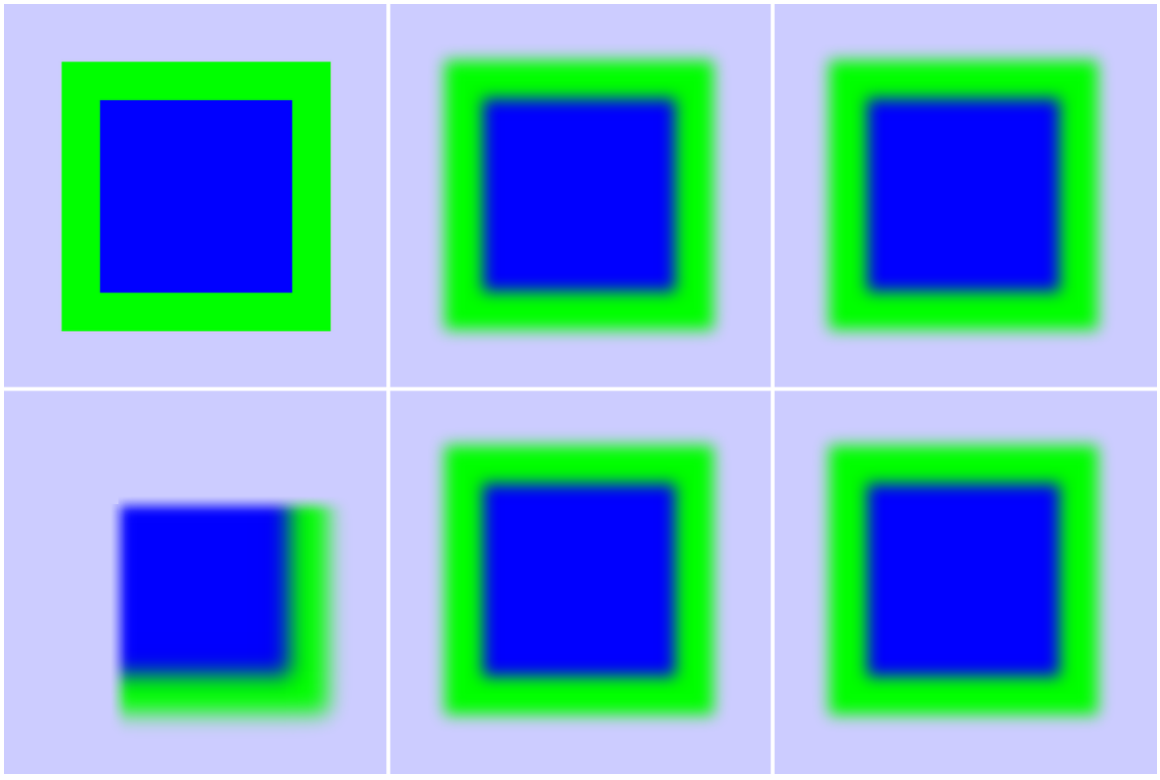


Abb. 310 *filter*-Beispiele

5 http://de.wikibooks.org/wiki/media%3ASVGfilter03.svg
6 http://de.wikibooks.org/wiki/media%3ASVGfilter04.svg

Mit der Eigenschaft *filter* kann ein Filter bei einem Element referenziert werden. Anwendbar ist die Eigenschaft bei graphischen Elementen und Gruppierungselementen.

Mögliche Werte sind:

inherit

geerbt

none

Filter wird nicht verwendet

MFunktionale IRI

IRI des Filters, der verwendet werden soll

Die Eigenschaft wird nicht vererbt und ist animierbar.

filter-Beispiele[7]

Sechs verschiedene Möglichkeiten, die Filtereigenschaft auf ein Rechteck anzuwenden, werden durchprobiert.

20.5 Hintergrundbild, Eigenschaft *enable-background*

Squiggle (Batik)	-
Opera (Presto)	9 (teilweise, kein Ausschnitt)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Als Eingabe für Urfilter kann der Hintergrund der Filterregion dienen. Dies benötigt allerdings Speicherbedarf, weswegen es notwendig ist, dies explizit festzulegen. Dazu wird die Eigenschaft *enable-background* bei einem Gruppierungselement angegeben, bei dem der Filter angewendet werden soll. Damit wird ein Schnappschuß der Filterregion angelegt, wie sie vor Anwendung eines Filters aussieht.

Mögliche Werte sind:

inherit

geerbt

accumulate

der Hintergrund kumuliert, die Voreinstellung, kein Schnappschuß

new

7 `http://de.wikibooks.org/wiki/media%3ASVGfilter05.svg`

Ein Schnappschuß wird angelegt und ist für die gesamte Filterregion verfügbar

new x y width height

Lediglich ein Bereich der Filterregion wird für einen Schnappschuß verwendet

Die Eigenschaft ist nicht animierbar.

Der Wert 'new' ermöglicht es, dass Kindelemente des Gruppierungselementes, bei welchem die Eigenschaft notiert ist, den Hintergrund zu verwenden. In einen neuen Bereich werden somit alle Kindelemente des Gruppierungselementes als Schnappschuß zusätzlich dargestellt.

Bei der Voreinstellung 'accumulate' hängt das Verhalten vom Kontext ab. Hat ein Vorfahre *enable-background*="new" notiert, so werden sowohl alle graphischen Elemente des Gruppierungselementes im Hintergrund des Elternelementes dargestellt, als auch in der direkten graphischen Ausgabe. Sonst erfolgt nur die direkte graphische Ausgabe.

Ist kein Hintergrund abgelegt, wird aber angefordert, so wird ein leerer Hintergrund zurückgegeben.

Folgen dem Wert 'new' weitere vier Angaben 'x y width height', so wird damit nur ein Unterbereich der Filterregion festgelegt, der für den Schnappschuß verwendet wird. Also die Koordinate 'x' für die linke Kante, 'y' für die obere Kante, die Länge 'width' für die Breite, 'height' für die Höhe. Es wird immer nur ein kleinerer Ausschnitt verwendet. Ein negativer Wert für 'width' oder 'height' bewirkt, dass ein Schnappschuß erfolgt, ebenso wie eine falsche Anzahl von Parametern.

20.6 Urfilter - Überblick und Gemeinsamkeiten

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	9 (größtenteils)
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (teilweise)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Ein Filter wird konstruiert aus einzelnen Urfiltern, auch Filtereffekte oder Primitive genannt. Soweit nicht anders angegeben, agieren alle Urfilter mit vormultiplizierten RGBA-Mustern, also mit den drei Farbkanälen und einem Alpha-Kanal für die Transparenz.

Nach Anwendung jeden Urfilters wird jeder der vier Kanäle auf den zulässigen Bereich eingeschränkt. Mit der Eigenschaft *color-interpolation-filters* kann festgelegt werden, in welchem Farbraum interpoliert wird. Per Voreinstellung wird eine lineare Interpolation (linearRGB) verwendet, anders als bei sonstigen Interpolationen, für welche 'sRGB' verwendet wird, siehe die Eigenschaft *color-interpolation* im Kapitel Graphik formatieren.

Während es für Farben in SVG generell diverse Notationen und Einheiten gibt, wird für Filter bei der Rechnung mit Farbkanälen von einem Bereich von 0 bis 1 als zulässigem Bereich ausgegangen. Also 0 für einen leeren Farbkanal, 1 für einen vollen. Es handelt sich dann also auch für die drei Farbkanäle um den gleichen Zahlenbereich wie für die Opazität im vierten Kanal, dem Alpha-Kanal.

Wenn von vormultiplizierten Werten die Rede ist, so wird angenommen, dass der Wert aus dem Alpha-Kanal an jeden Farbkanal dranmultipliziert wird, bevor eine Rechnung innerhalb eines Filters statt-

findet. Die passiert bei allen Urfiltern mit Ausnahme von ***feColorMatrix*** und ***feComponentTransfer***. Bei diesen wird der Alpha-Kanal nicht vor der Rechnung an die Farbkanäle dranmultipliziert.

Eine weitere öfter auftretende Redewendung ist 'transparent schwarz' - das bedeutet einfach, dass bei allen vier Kanälen der Wert 0 vorliegt. In der konkreten Darstellung ist das erstmal nicht zu unterscheiden von einer Situation, in der eine andere Farbe angenommen wird und nur der Alpha-Kanal 0 ist. Bei Filtern kann sich aber durchaus ein Unterschied ergeben, insbesondere bei jenen, wo keine vormultiplizierten Werte verwendet werden.

Urfilter haben eine Reihe von Attributen gemeinsam, welche in folgenden Abschnitten beschrieben werden.

20.6.1 Attribute *x*, *y*, *width* und *height*

Die Attribute definieren einen rechteckigen Unterbereich des Filters, auf den der jeweilige Filter angewendet wird.

x ist der linke Rand der Region, *y* der obere Rand, *width* die Breite, *height* die Höhe. *x* und *y* sind Koordinaten, *width* und *height* sind Längen. Negative Werte für *width* und *height* sind ein Fehler, welcher zum Abbruch der Darstellung des Dokumentes führt. Ein Wert '0' unterbindet die Anwendung des Urfilters.

Sind *x* oder *y* nicht angegeben, wird für das fehlende Attribut jeweils '0%' angenommen. Sind *width* oder *height* nicht angegeben, wird für das fehlende Attribut jeweils '100%' angenommen.

Die Attribute sind animierbar.

20.6.2 Attribut *result*

Das Attribut gibt dem Ergebnis des Urfilters einen Namen. Der Wert ist also eine Zeichenkette, so dass das Ergebnis als Eingabe bei einem anderen Urfilter verwendet werden kann, indem der Name in dessen Attribut *in* notiert wird.

Ist kein Ergebnisname angegeben, so ist das Ergebnis nur im nächsten Urfilter verwendbar, falls dieser kein Attribut *in* notiert hat.

Der Name gilt nur im selben Filter und derselbe Name kann beliebig oft wiederholt werden. Tritt ein Name mehrfach auf, so wird jeweils das Ergebnis des ersten, dem aufrufenden Urfilter vorhergehenden Urfilters verwendet.

Das Attribut ist animierbar.

20.6.3 Attribut *in*

Das Attribut gibt an, was als Eingabe bei einem Urfilter verwendet werden soll. Das Attribut ist nur zulässig und sinnvoll für Urfilter, die eine Eingabe brauchen, also etwa nicht für ***feFlood*** oder ***feImage*** oder ***feTurbulence***.

Mögliche Werte sind:

SourceGraphic

Die graphischen Elemente des Gruppierungselementes im Filterbereich

SourceAlpha

Nur der Alpha-Kanal der graphischen Elemente des Gruppierungselementes im Filterbereich

BackgroundImage

Das Hintergrundbild, welches mittels *enable-background* verfügbar gemacht wurde

BackgroundAlpha

Der Alpha-Kanal des Hintergrundbildes, welches mittels *enable-background* verfügbar gemacht wurde

FillPaint

Füllung mit dem Wert der Eigenschaft *fill* des Zielelementes des Filters

StrokePaint

Füllung mit dem Wert der Eigenschaft *stroke* des Zielelementes des Filters

Name

Name eines vorhergehenden Urfilterergebnisses im selben Filter

Ist kein Wert angegeben, so wird das Ergebnis des vorherigen Urfilters verwendet. Beim ersten wird 'SourceGraphic' verwendet.

Das Attribut ist animierbar.

20.6.4 Beispiele Urfilterattribute

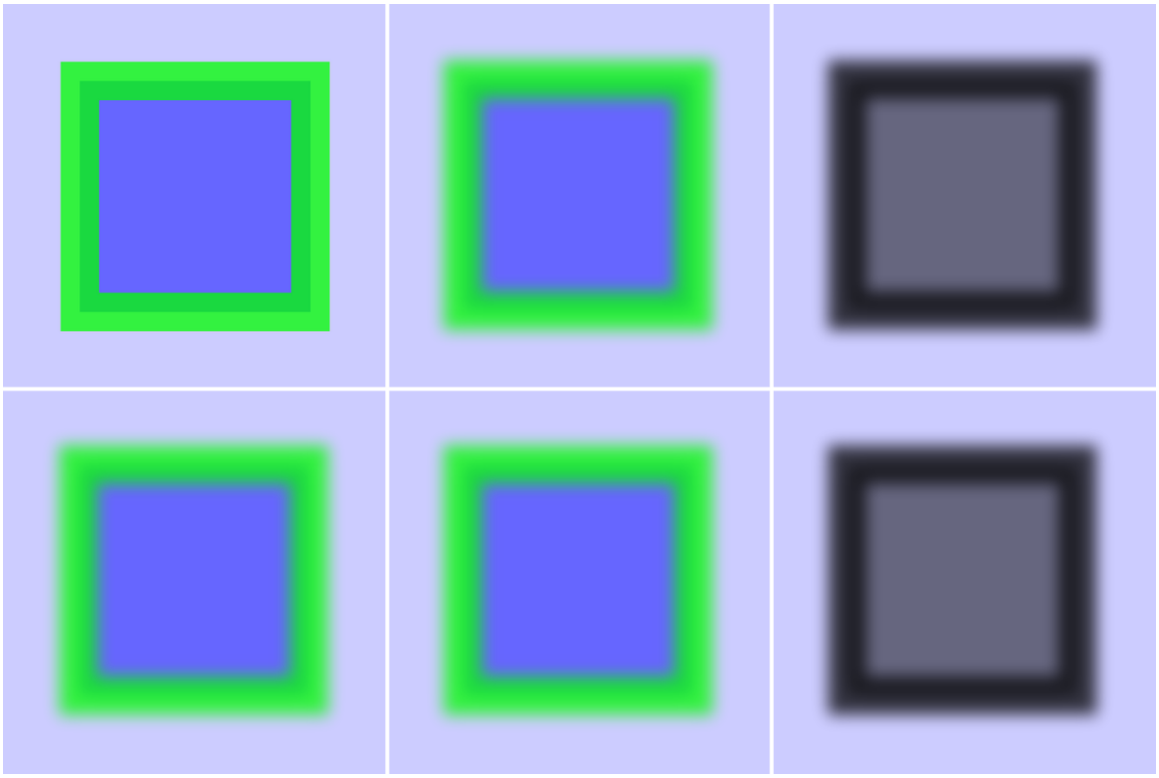


Abb. 311 Eingabemöglichkeiten (1)

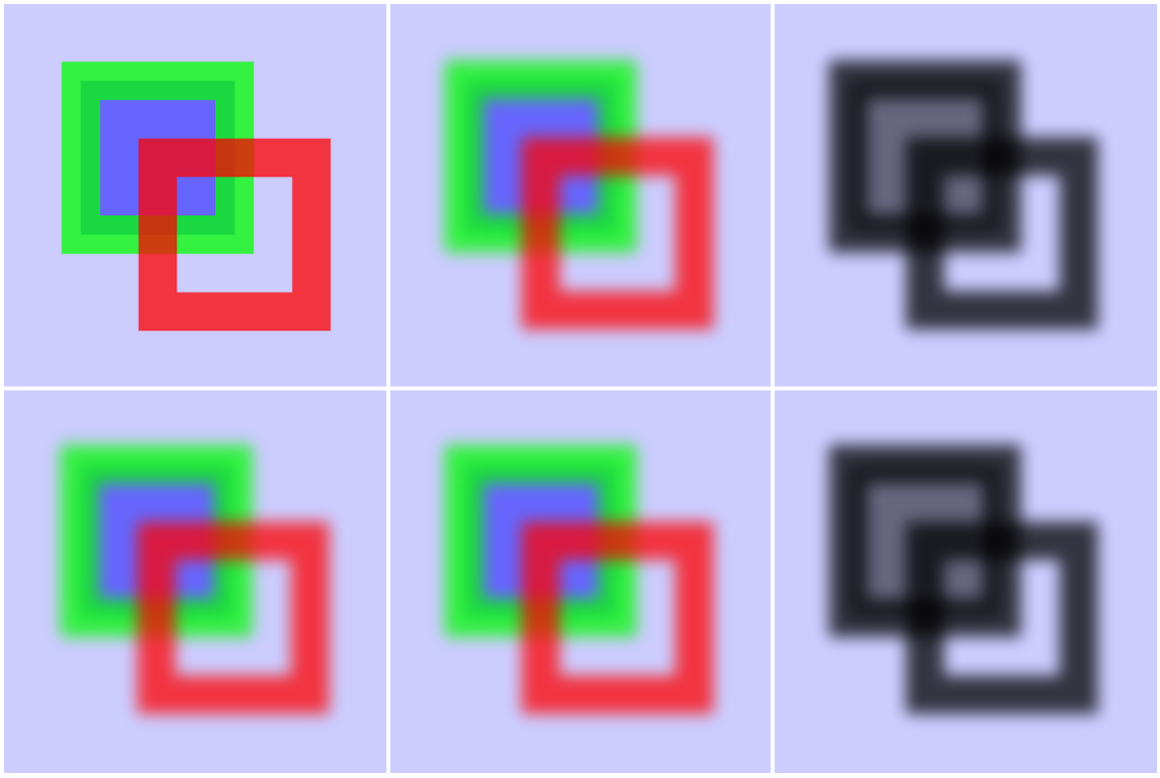


Abb. 312 Eingabemöglichkeiten (2)

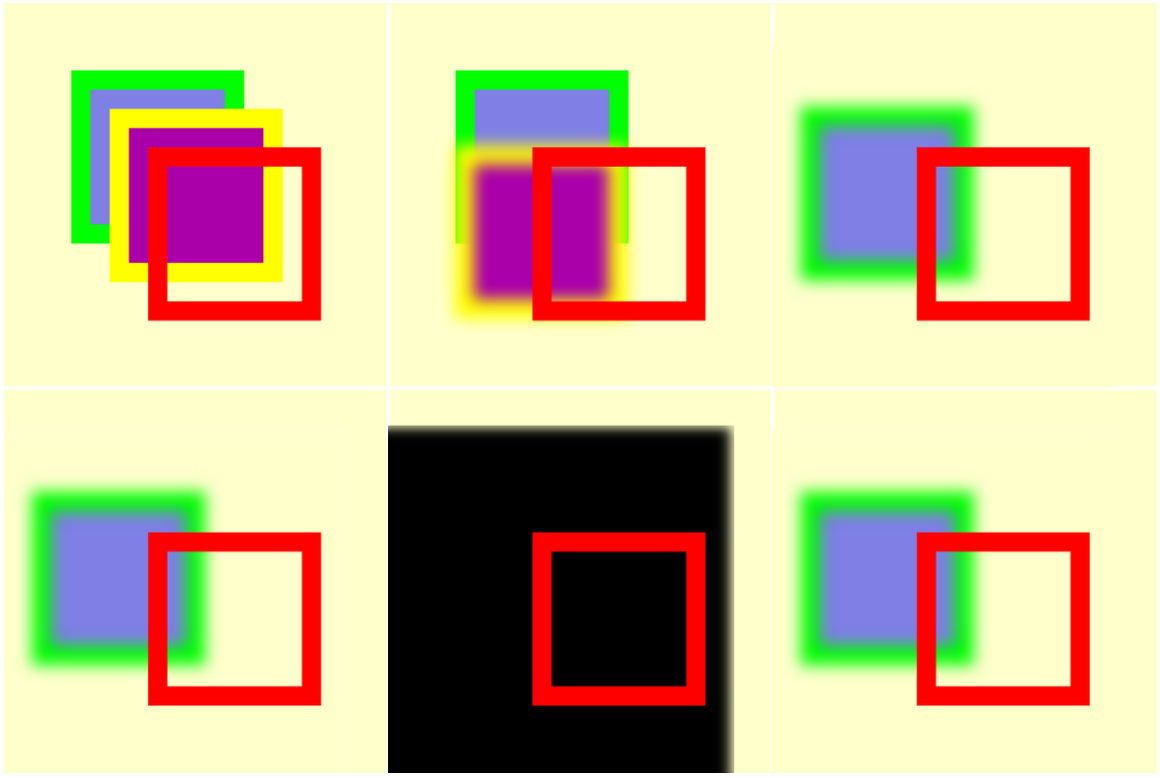


Abb. 313 Verwendung des Hintergrundbildes

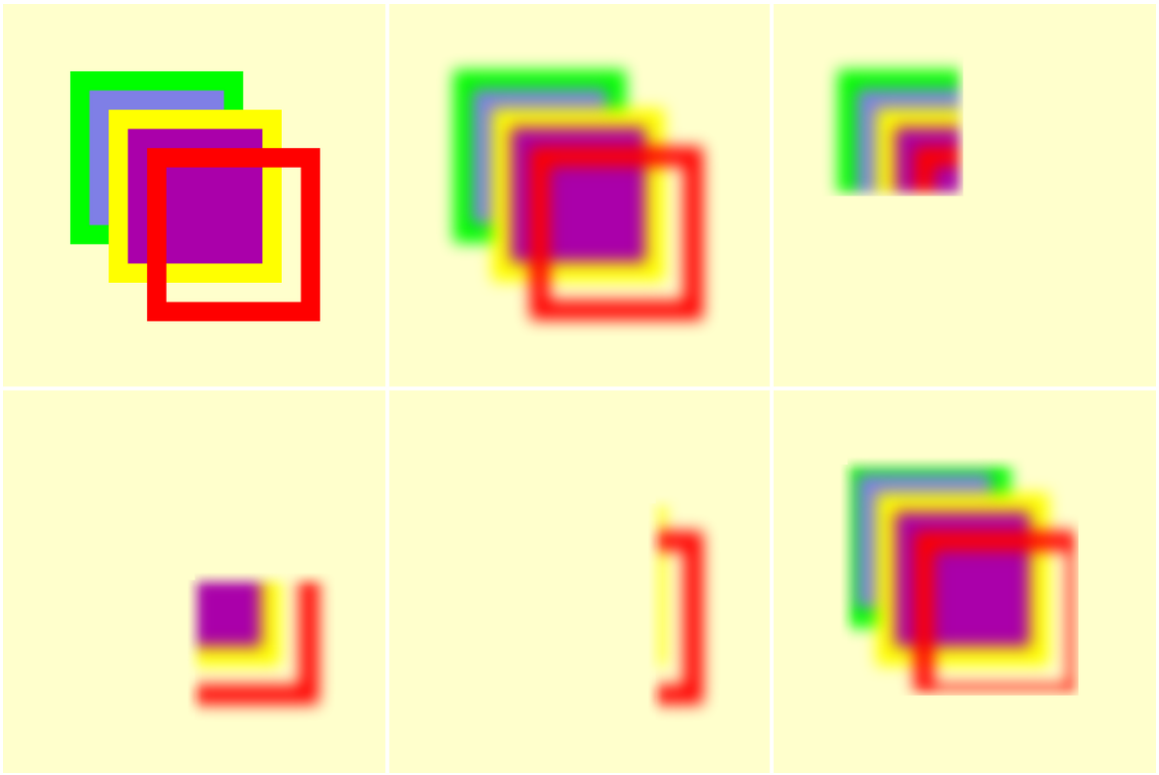


Abb. 314 Filterunterregion

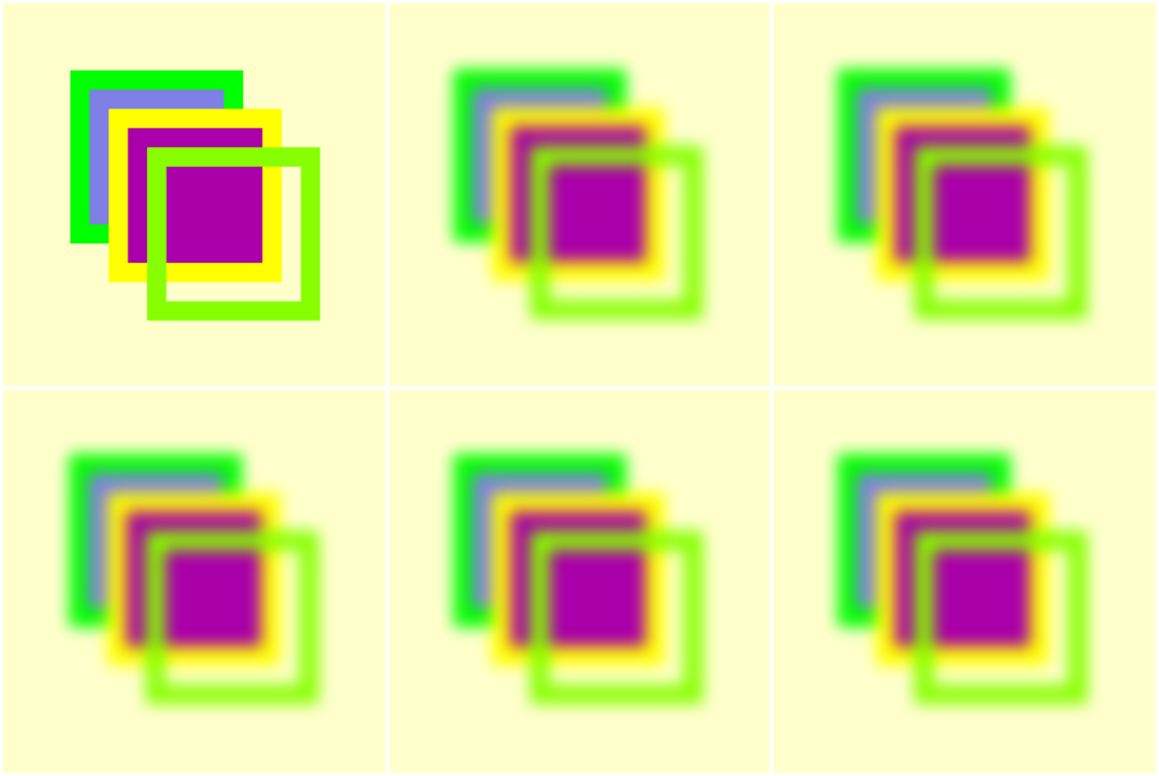


Abb. 315 Richtiges Ergebnis

Eingabemöglichkeiten (1)[8]

Auswirkung der Eingabe auf das Filterergebnis. Links oben ist das ungefilterte Rechteck zum Vergleich angegeben. Dann kommt oben in der Mitte SourceGraphic, rechts SourceAlpha, unten links FillPaint, in der Mitte StrokePaint und rechts unten wird als Eingabe das Ergebnis eines vorherigen Filters verwendet, hier SourceAlpha von einem Versatz um 0.

Eingabemöglichkeiten (2)[9]

Auswirkung der Eingabe auf das Filterergebnis. Links oben sind die ungefilterten Rechtecke zum Vergleich angegeben. Dann kommt oben in der Mitte SourceGraphic, rechts SourceAlpha, unten links FillPaint, in der Mitte StrokePaint und rechts unten wird als Eingabe das Ergebnis eines vorherigen Filters verwendet, hier SourceAlpha von einem Versatz um 0.

Verwendung des Hintergrundbildes[10]

Das Vergleichbild ist oben links dargestellt.

Gefiltert mit einem Weichzeichner wird immer das Quadrat mit violetter Füllung. Daß der Hintergrund verwendet werden soll, wird jeweils bei einem Gruppierungselement um alle drei Quadrate notiert. Oben in der Mitte wird ein kumulierender Hintergrund verwendet, sonst ein neuer Hintergrund, rechts unten nur ein Ausschnitt links oben. Links unten wird die Einstellung von einem weiteren Gruppierungselement geerbt, unten in der Mitte wird nur der Alphakanal verwendet.

Filterunterregion[11]

Auswirkung verschiedener Filterregionen in lokalen Koordinaten auf ein Rechteck. Als Filter wird ein einfacher Gaußscher Weichzeichner verwendet.

Links oben ist die ungefilterten Rechtecke zum Vergleich angegeben.

Richtiges Ergebnis[12]

Verschiedene Kombinationen - immer das gleiche Ergebnis ein Weichzeichner, kein roter Bereich.

Links oben sind die ungefilterten Rechtecke zum Vergleich angegeben.

8 http://de.wikibooks.org/wiki/media%3ASVG_Urfilter01.svg
9 http://de.wikibooks.org/wiki/media%3ASVG_Urfilter02.svg
10 http://de.wikibooks.org/wiki/media%3ASVG_Urfilter03.svg
11 http://de.wikibooks.org/wiki/media%3ASVG_Urfilter04.svg
12 http://de.wikibooks.org/wiki/media%3ASVG_Urfilter05.svg

20.7 Eigenschaft *color-interpolation-filters* Farbinterpolation für Urfilter

Squiggle (Batik)	1.7
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Abb. 316 Farbinterpolation bei diffuser Streuung

Abb. 317 Farbinterpolation bei reflektierender Streuung

Anwendbar auf Urfilter gibt es eine eigene Eigenschaft zur Farbinterpolation *color-interpolation-filters*. Diese hat die gleichen möglichen Werte wie *color-interpolation*:

inherit

geerbt

auto

Das Darstellungsprogramm kann selbst entscheiden, dem Autor ist es egal

sRGB

Interpolation erfolgt im Standard-Farbraum

linearRGB

Interpolation erfolgt im linearisierten Farbraum

Der Initialwert ist allerdings 'linearRGB' (anders als bei *color-interpolation*), die Eigenschaft wird ebenfalls vererbt und ist animierbar.

Insbesondere kann die Eigenschaft also beim Elternelement ***filter*** notiert werden und wirkt dann auf alle Urfilter darin - oder auch bei einem anderen Gruppierungselement notiert werden, besonders sinnvoll eventuell beim Elternelement aller Filter, also ***defs*** oder gar dem Hauptelement ***svg***.

Beispiele:

Unterschiedliche Einstellungen der Farbinterpolation für Filter. Links oben ist das Ausgangsbild zu sehen. In der Mitte oben wird linearRGB verwendet. Rechts oben wird sRGB verwendet. Links unten wird auto verwendet. In der Mitte unten wird inherit verwendet. Rechts unten wird für die Lichtquelle sRGB verwendet und für die Komposition die Voreinstellung linearRGB.

Bei diffuser Streuung[13]

Bei reflektierender Streuung[14]

20.8 Urfilter *feOffset* - Räumlicher Versatz

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Der Urfilter dient dazu, einen räumlichen Versatz hervorzurufen. Es wird eine relative Verschiebung zur aktuellen Position notiert.

Dies wird häufig verwendet, um den Effekt eines Schattens hervorzurufen, welcher gleichgroß wie die schattenwerfenden Elemente ist. Das passt recht gut zu einer Lichtquelle, welche unendlich weit entfernt ist und Elementen, die in kleinem Abstand parallel über der Zeichenebene zu schweben scheinen. Ist hingegen eher geplant, eine endlich weit entfernte Lichtquelle vorzutäuschen, so ist der Schatten größer als die Elemente. Da bietet es sich eher an, mit ***use*** eine Kopie zu erstellen und diese mit *transform* zu vergrößern und zu verschieben, bevor darauf ein Filter angewendet wird, um etwa über den Alpha-Kanal einen Schatten zu erzeugen.

Da Filter generell aufwendiger sind als die Verwendung von ***use*** und *transform*, kann es in vielen Fällen sinnvoll sein, so vorzuarbeiten, statt alles allein mit dem Filter zu lösen.

20.8.1 Attribute *dx* und *dy*

Mit den Attributen wird die Verschiebung angegeben, *dx* in Richtung der x-Achse, *dy* in Richtung der y-Achse.

Der Wert ist eine Zahl in den Einheiten, die mit *primitiveUnits* von ***filter*** angegeben sind. Für ein nicht angegebenes Attribut wird jeweils 0 angenommen.

Die Attribute sind animierbar.

13 http://de.wikibooks.org/wiki/media%3ASVGcolor-interpolation-filters01.svg
14 http://de.wikibooks.org/wiki/media%3ASVGcolor-interpolation-filters02.svg

20.8.2 Beispiele *feOffset*

Abb. 318 Ein Bild wird mehrfach und versetzt dargestellt

Abb. 319 Einfache Schatten

Es wird der Alpha-Kanal um 50 nach links und um 100 nach unten verschoben. Das Ergebnis wird in 'Versatz' abgelegt.

```
<feOffset in="SourceAlpha" result="Versatz" dx="50" dy="100" />
```

Mehrfach versetzte Kopien[15]

Eine Vorlage aus zwei Ellipsen und den Buchstaben 'SVG' wird mehrfach an verschiedene Positionen versetzt.

Einfache Schatten[16]

Anhand der von Ellipsen umgebenden Buchstaben 'SVG' wird gezeigt, wie mehrere einfache Schatten von punktförmigen, unendlich weit entfernten Lichtquellen vorgetäuscht werden.

Als Schatten dient einfach der Alphakanal und wird um ein Stück verschoben. Das passiert für jeden Einzelschatten. Wie dunkel ein Schatten ist, hängt vor allem von der Intensität aller anderen Lichtquellen ab. Dies kann indirekt über den Alphakanal des jeweiligen Schattens eingestellt werden.

15 http://de.wikibooks.org/wiki/media%3ASVGfeOffset01.svg
16 http://de.wikibooks.org/wiki/media%3ASVGfeOffset02.svg

Anschließend werden die Schatten unter die Graphik gelegt. Hier mit einem Urfilter zum Zusammenfügen, weil es sich um mehrere Schatten handelt.

Bei einem Schatten ist der zum Überblenden oder der zur Komposition dafür ebenfalls geeignet.

20.9 Urfilter *feFlood* - Gleichmäßige Füllung

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit diesem Urfilter kann die Filterunterregion mit einer Farbe teiltransparent gefüllt werden. Die Filterunterregion wird definiert durch die allgemeinen Attribute *x*, *y*, *width* und *height*.

20.9.1 Eigenschaft *flood-color*

Mit der Eigenschaft wird die Füllfarbe angegeben. Der Wert entspricht dem der Eigenschaften *fill* und *stroke*, allerdings ohne Maldienste. Möglich sind also neben einem Farbwert auch die Werte 'currentColor' und 'inherit'. Der Ausgangswert ist schwarz. Die Eigenschaft wird nicht vererbt und ist animierbar.

20.9.2 Eigenschaft *flood-opacity*

Mit der Eigenschaft wird die Opazität oder Teiltransparenz angegeben. Der Wert entspricht dem der Eigenschaften *opacity*, *fill-opacity* und *stroke-opacity*. Der Wert ist also entweder 'inherit' oder ein Wert zwischen 0 (durchsichtig) und 1 (undurchsichtig), beziehungsweise der Wert wird auf den nächstgelegenen Wert aus diesem Intervall korrigiert, wenn falsch angegeben. Der Ausgangswert ist 1. Die Eigenschaft wird nicht vererbt und ist animierbar.

20.9.3 Beispiele *feFlood*

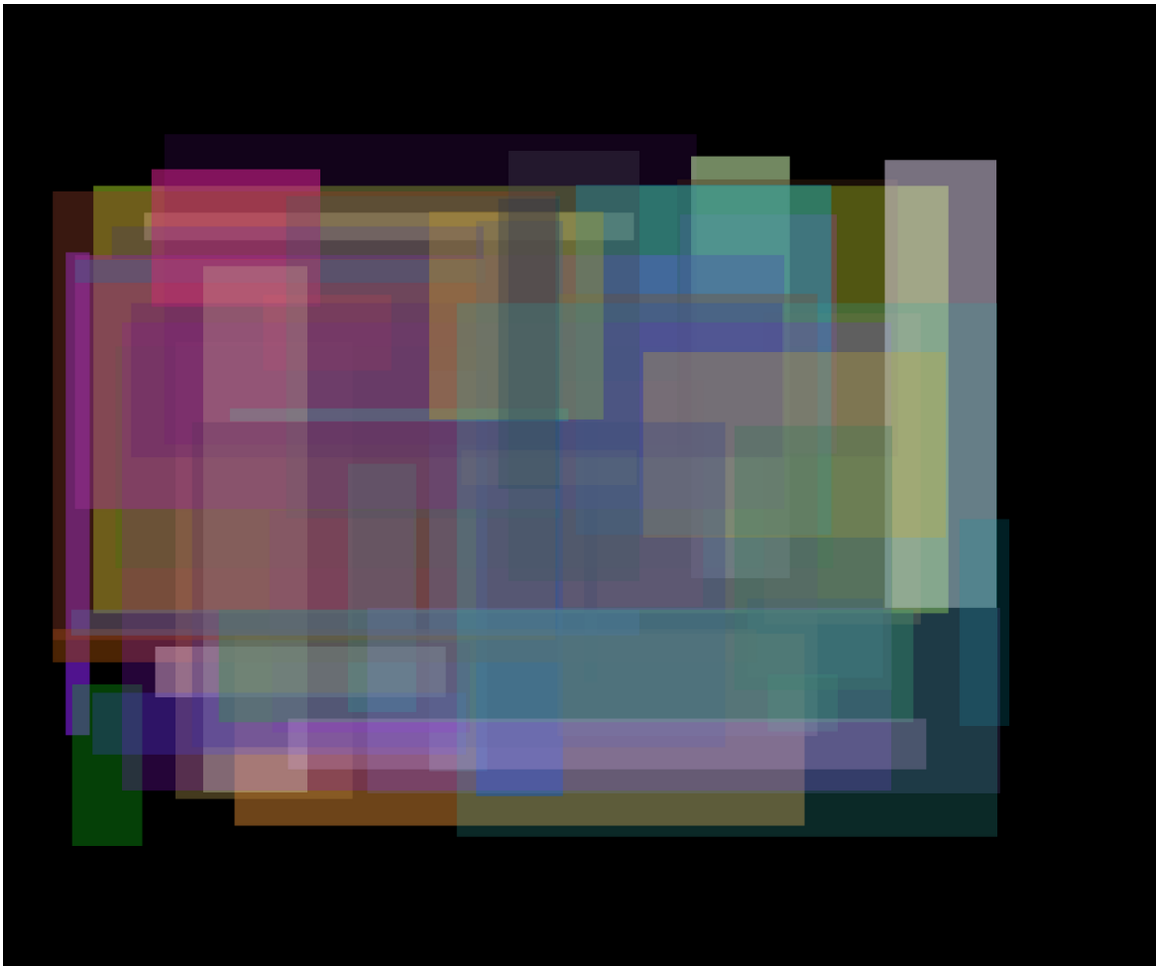


Abb. 320 Geflutete Filterunterregionen

Ein quadratischer Bereich wird halbtransparent dunkelblau gefüllt. Das Ergebnis wird als 'Hintergrund' abgelegt.

```
<feFlood result="Hintergrund"
         x="50" y="50" width="200" height="200"
         flood-color="#008" flood-opacity="0.5" />
```

Geflutete Filterunterregionen[17]

Zufällige Anordnung von rechteckigen Farbflächen.

17 http://de.wikibooks.org/wiki/media%3ASVGfeFlood01.svg

20.10 Urfilter *feGaussianBlur* - Weichzeichner

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

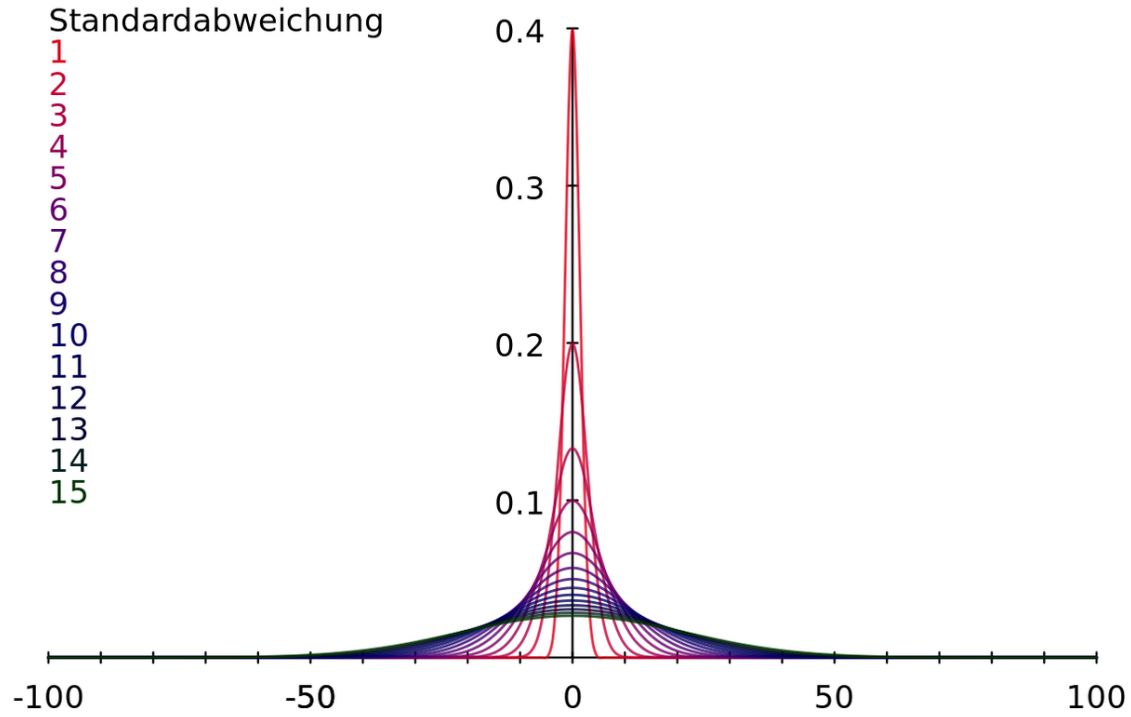


Abb. 321 Gaußkurven mit verschiedenen Standardabweichungen

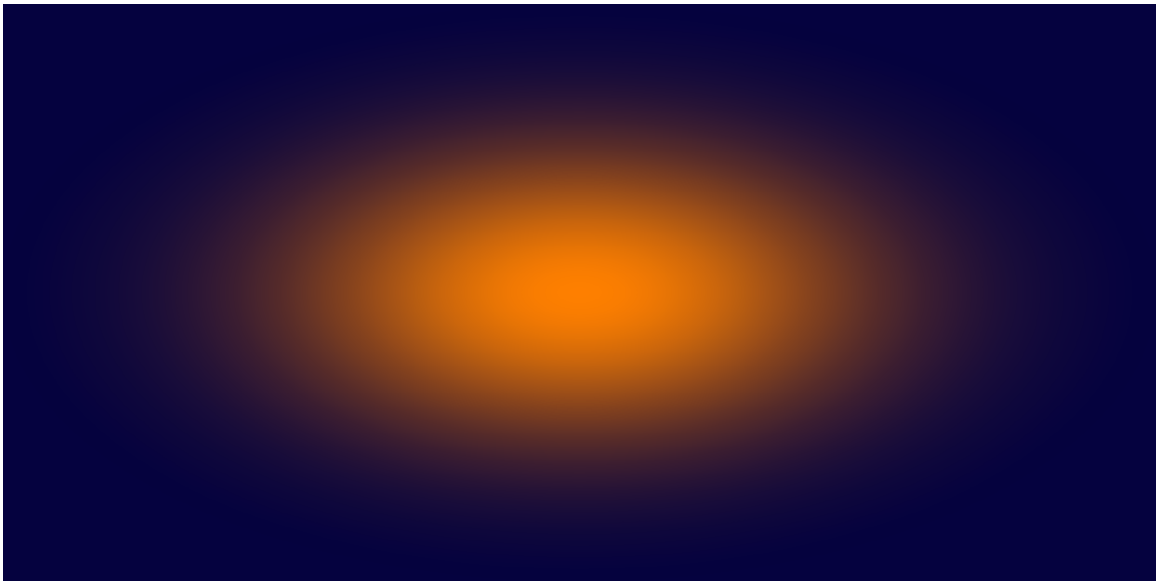


Abb. 322 zweidimensionale Gaußkurve als Farbverlauf mit verschiedenen Standardabweichungen in x- und y-Richtung

Eine Faltung mit einer Gaußkurve wird durchgeführt, um den Effekt eines Weichzeichners zu erreichen.

Die Faltungsfunktion ist:

$$F(x,y)=G(x)*H(y)$$

mit

$$G(x)=\exp(-x^2/(2a^2)) / (2*\pi*a^2)^{1/2}$$

$$H(y)=\exp(-y^2/(2b^2)) / (2*\pi*b^2)^{1/2}$$

Dabei geben 'a' und 'b' die Standardabweichungen in den Richtungen x und y an.

Als kurze und vereinfachte Erklärung, wie eine Faltung funktioniert, so ergibt das Integral über obige Funktion über den Filterbereich den gefilterten Wert am Punkt 0. Für einen beliebigen Punkt sind entsprechend x und y zu verschieben. Bei diskreten Objekten wie einem Raster aus Pixeln wird anstatt des Integrals die Summe über alle Pixel verwendet, geteilt durch die Anzahl der Pixel.

Die Bilder neben dem Text zeigen Gaußkurven, einmal eindimensional als Graph (Gaußkurven mit verschiedenen Standardabweichungen[18]) und einmal zweidimensional als Farbverlauf (zweidimensionale Gaußkurve als Farbverlauf mit verschiedenen Standardabweichungen in x- und y-Richtung[19]). Die Fläche (beziehungsweise bei zweidimensionalen Kurven das Volumen) unter der Kurve ist immer gleich. Das bedeutet, bei einer kleinen Standardabweichung tragen neben dem originalen Punkt und

18 http://de.wikibooks.org/wiki/media%3ASVG_Gausskurve01.svg
19 http://de.wikibooks.org/wiki/media%3ASVG_Gausskurve02.svg

wenige Nachbarn zum Ergebnis bei. Bei einer großen Standardabweichung tragen viele Nachbarn aus einer größeren Umgebung zum Ergebnis bei, welches dann also zu einer starken räumlichen Verschmierung führt.

Ferner ist zu beachten, dass die Hauptachsen für die Standardabweichungen immer in x- und y-Richtung des lokalen Koordinatensystems ausgerichtet sind. Wird dies anders benötigt, wäre das zu filternde Objekt also erst geeignet zu drehen, dann der Filter anzuwenden und dann das gefilterte Objekt zurückzudrehen (was durch Verschachtelung von Gruppierungselementen erreichbar ist).

Der Urfilter wird oft zusammen mit ***feOffset*** verwendet, um aus der Eingabe 'SourceAlpha' etwas zu erzeugen, was einen diffusen Schatten einer ausgedehnten, unendlich weit entfernen Lichtquelle vorzutäuschen. Durch Verwendung von teiltransparenten Schattenvorlagen können auch Schatten mehrerer Lichtquellen vorgetäuscht werden, die weiter voneinander entfernt sind.

20.10.1 Attribut *stdDeviation*

Mit dem Attribut werden die Standardabweichungen in x- und y-Richtung angegeben. Der Wert des Attributes besteht aus ein oder zwei Zahlen, mit Leerzeichen voneinander getrennt. Sind zwei Zahlen angegeben, ist die erste die Standardabweichung in x-Richtung, die zweite in y-Richtung. Ist nur eine Zahl angegeben, wird diese für die die Standardabweichung in x- und y-Richtung verwendet.

Negative Werte sind ein Fehler, der zum Abbruch der Darstellung des Dokunentes führt. In SVG 1.1 unterbindet die Angabe eines Wertes 0 die Darstellung des Filters. Dies entspricht allerdings nicht der Erwartung, die sich aus angegebener Formel ergibt. Danach wäre das Ergebnis eher kein Weichzeichnereffekt in der Richtung mit Standardabweichung 0. Eine derartige Änderung ist im Gespräch, entweder als Fehlerkorrektur für SVG 1.1 oder erst als Änderung bei einem neuen Filtermodul.

Ist *stdDeviation* nicht angegeben, wird 0 angenommen.

Das Attribut ist animierbar.

20.10.2 Beispiele *feGaussianBlur*

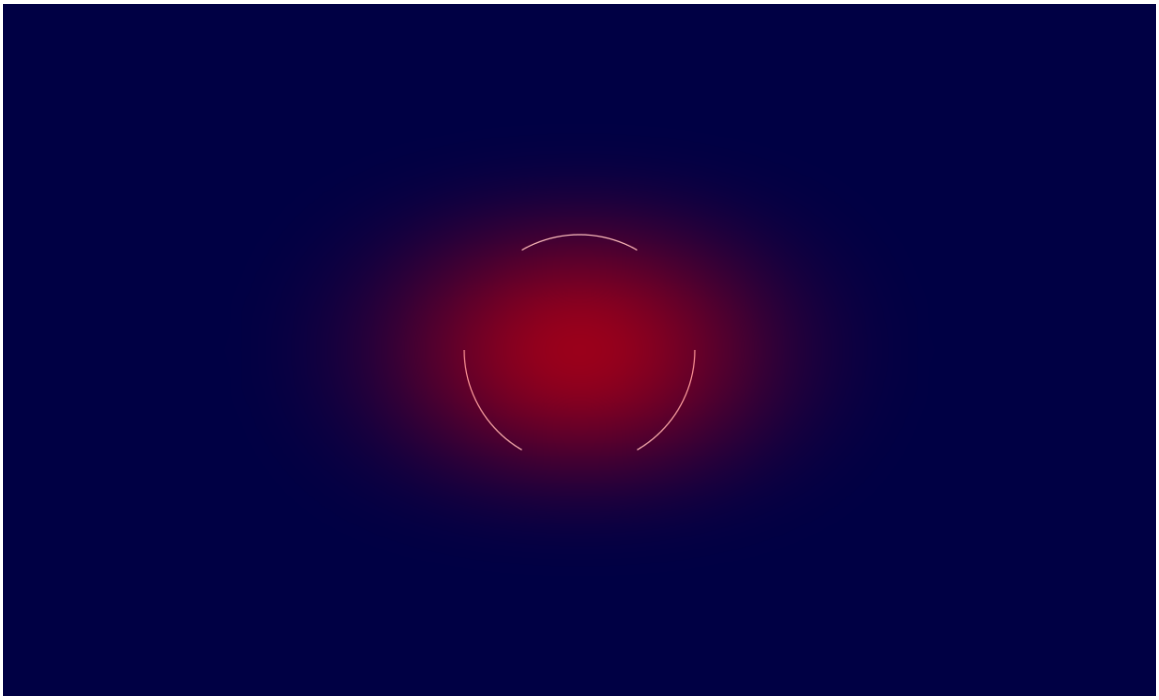


Abb. 323 Anwendung auf einen roten Kreis

Abb. 324 Schatten von ausdehnten Lichtquellen

Alpha-Kanal weichzeichnen und dann verschieben:

```
<feGaussianBlur in="SourceAlpha" stdDeviation="20 5"
x="0" y="0" width="500" height="500"/>
<feOffset  dy="20" dx="10" result="Schatten" />
```

Verschmierter roter Kreis [20]

Anwendung einer Faltung mittels feGaussianBlur auf einen roten Kreis. Die Standardabweichungen in x- und y-Richtung der Gaußkurve werden animiert.

Die Abmessung des ungefalteten Kreises ist weiß gestrichelt angedeutet.

Schatten von ausdehnten Lichtquellen[21]

Anhand der von Ellipsen umgebenden Buchstaben 'SVG' wird gezeigt, wie mehrere einfache Schatten von ausgedehnten Lichtquellen vorgetäuscht werden.

Nach Anwendung des Gaußschen Weichzeichners auf den Alphakanal wird der Schatten um ein Stück verschoben. Die Standardabweichungen entscheiden über die Ausdehnung der Lichtquelle.

20 http://de.wikibooks.org/wiki/media%3ASVGfeGaussianBlur01.svg
21 http://de.wikibooks.org/wiki/media%3ASVGfeGaussianBlur02.svg

Das passiert für jeden Einzelschatten. Wie dunkel ein Schatten ist, hängt vor allem von der Intensität aller anderen Lichtquellen ab. Dies kann indirekt über den Alphakanal des jeweiligen Schattens eingestellt werden.

Anschließend werden die Schatten unter die Graphik gelegt. Hier mit einem Urfilter zum Zusammenfügen, weil es sich um mehrere Schatten handelt.

Bei einem Schatten ist der zum Überblenden oder der zur Komposition dafür ebenfalls geeignet.

20.11 Urfilter *feImage* - Bild einfügen

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (teilweise, kein SVG)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Urfilter kann ein Bild eingefügt werden, statt der ansonsten üblichen Eingabe. Das Bild kann sowohl eine externe Datei sein, zum Beispiel PNG, JPEG/JFIF oder SVG, aber auch ein SVG-Fragment aus einer anderen Datei oder der aktuellen Datei.

Bei externen Dateien entspricht die Verarbeitung der des Elementes ***image***, bei SVG-Fragmenten der von ***use***. Dies schließt die Attribute *x*, *y*, *width*, *height* und das ebenfalls verfügbare *preserveAspectRatio* mit ein.

20.11.1 Attribut *href* von XLink

Mit dem Attribut *href* von XLink wird das Bild beziehungsweise Fragment referenziert.

Das Attribut ist anmierbar.

Mit *href* sind auch die anderen Attribute von XLink verfügbar.

20.11.2 Beispiele *feImage*

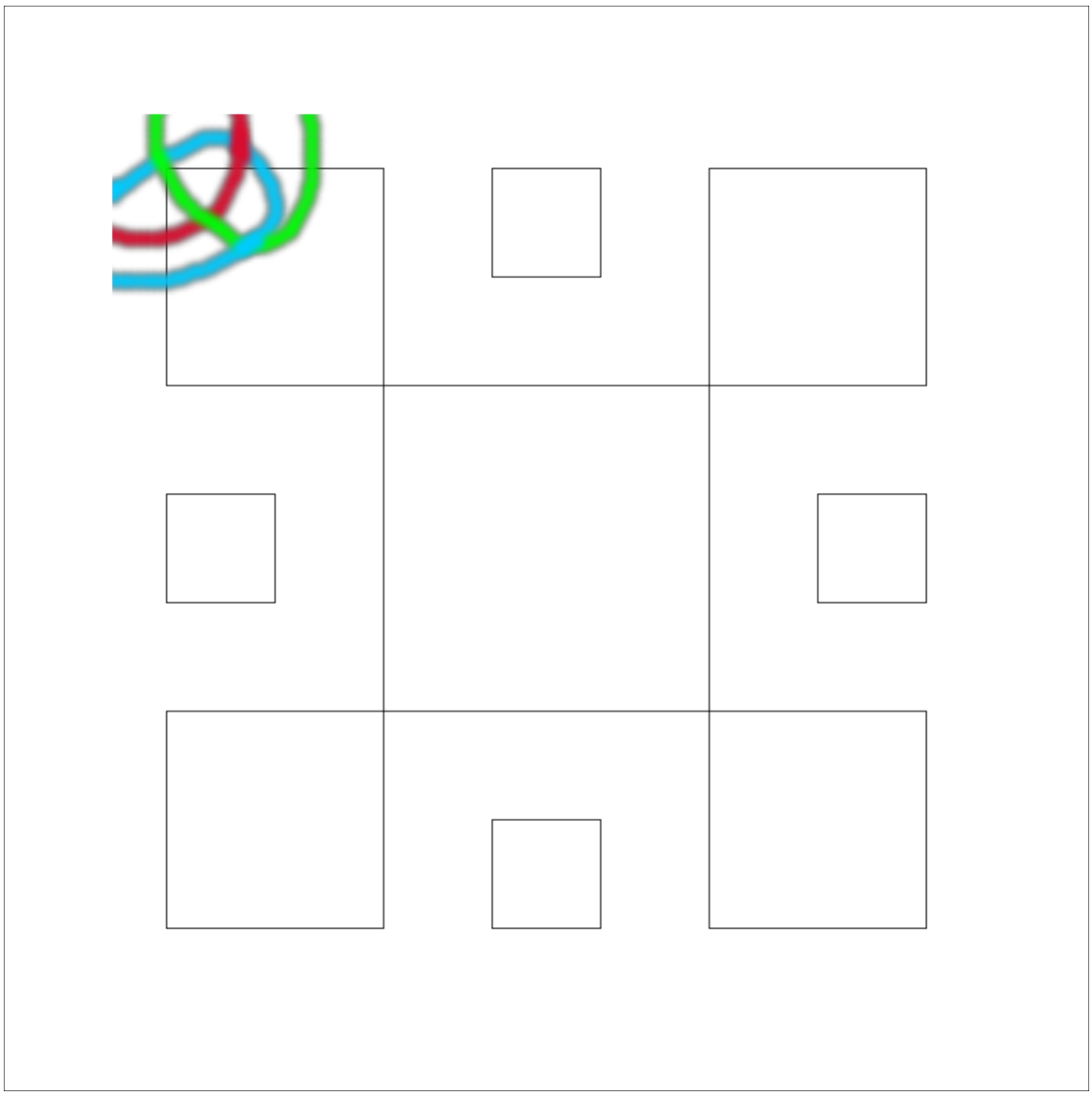


Abb. 325 Bilder einbetten (1)

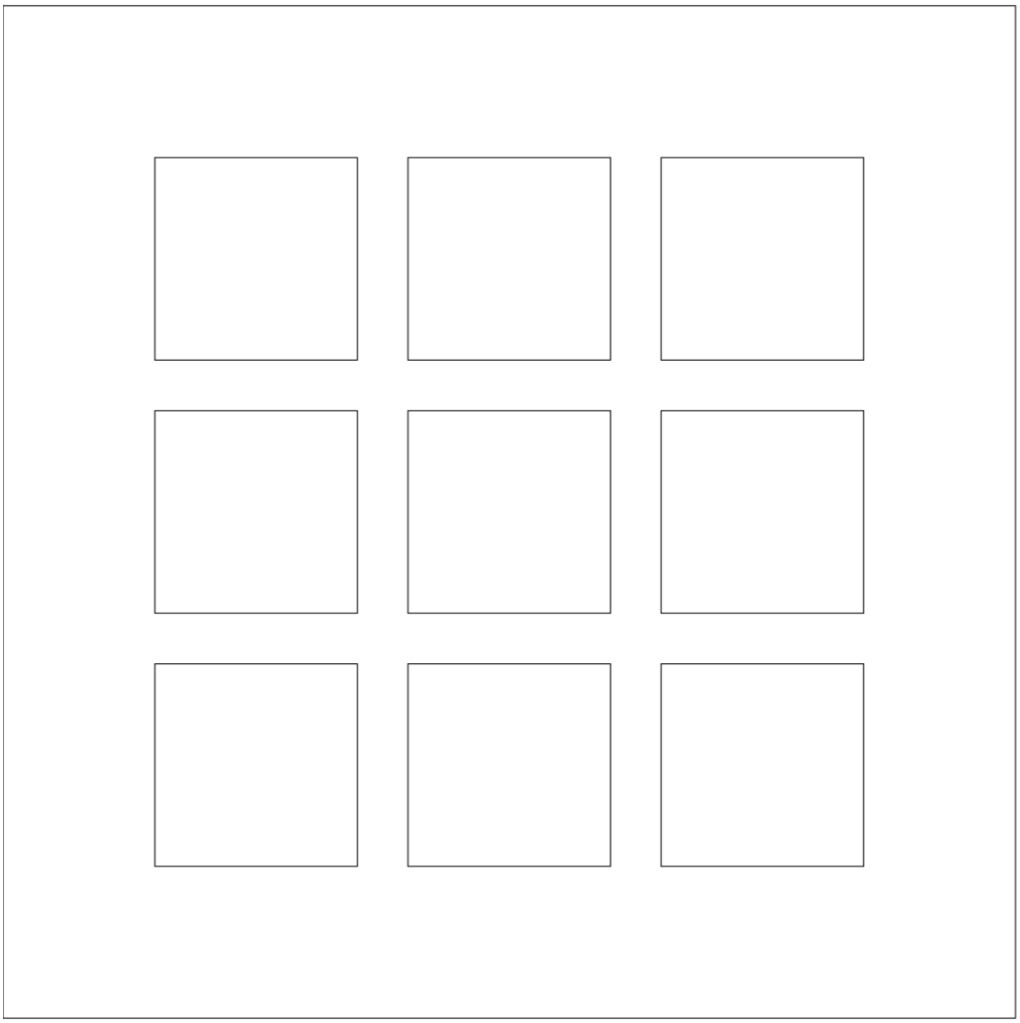


Abb. 326 Bilder einbetten (2)

Ein Bild namens 'Beispiel01.png' wird referenziert und als Ergebnis 'Beispiel' verfügbar gemacht.

```
<feImage xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="Beispiel01.png"
x="0.2" y="0.2" width="0.6" height="0.6" result="Beispiel"/>
```

Ein SVG-Fragment wird referenziert und mit einem Weichzeichner bearbeitet und von einem leeren Element ***g*** als Filter verwendet:

```
<defs xmlns:xlink="http://www.w3.org/1999/xlink">
<g id="g1">
<rect x="100" y="100" width="300" rx="50" ry="50" height="300"
      fill="#44ff44" />
```

```
<ellipse cx="250" cy="250" rx="150" ry="100"
      fill="#f00" fill-opacity="0.5"
      stroke="#00f" stroke-width="50" />
</g>

<filter id="f1" filterUnits="userSpaceOnUse"
x="0" y="0" width="500" height="500">
  <feImage id="fi" xlink:href="#g1"
           x="0" y="0" width="500" height="500"/>
  <feGaussianBlur id="fGB" stdDeviation="4 4"
           x="0" y="0" width="500" height="500"/>
</filter>
</defs>
<g filter="url(#f1)" />
```

Bilder einbetten (1)[22]

Es werden Kopien von Bildern zu einem Gesamtfilterergebnis zusammengefügt. Ein PNG-Bild mit drei handgemalten Kringeln wird als eingebettete Datei per Pseudoprotokoll data referenziert. Ein SVG-Fragment mit eigenem Anzeigebereich und mit drei gegeneinander verdrehten Ellipsen wird als weiteres Bild referenziert.

Die richtigen Positionen und Größen werden als Quadrate mit schwarzem Strich angedeutet.

Bilder einbetten (2)[23]

Ein SVG-Fragment ohne eigenen Anzeigebereich und mit drei gegeneinander verdrehten Ellipsen wird als Bild referenziert.

Es wird jeweils das lokale Koordinatensystem verwendet, das referenzierte Fragment muss also komplett in der Filterregion liegen, damit es korrekt angzeigt wird. Entsprechend skaliert das Fragment also auch nicht mit dem angegeben Unterbereich des Filters, welcher die passenden Abmessungen des referenzierten Fragmentes haben muss.

Der eigentliche Inhalt wird bei diesem Beispiel nicht verwendet, also auch nicht angezeigt. Dies wäre ein einfaches schwarzes Quadrat. Da andererseits der Hintergrund eines Filterergebnisses durchsichtig ist, können auch mehrere Filter übereinandergelegt werden.

Die richtigen Positionen und Größen werden als Quadrate mit schwarzem Strich angedeutet.

20.12 Urfilter *feTile* - Kacheln

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

22 http://de.wikibooks.org/wiki/media%3ASVGfeImage01.svg
23 http://de.wikibooks.org/wiki/media%3ASVGfeImage02.svg

Abb. 327 SVG-Kacheln

Die Unterregion des Filters, welche durch *x*, *y*, *width* und *height* gegeben ist, wird mit einem Muster gekachelt.

Die jeweilige Kachel des Musters ergibt sich im Regelfalle aus der Ausgabe eines anderen Urfilters, zum Beispiel ***feImage***, diese legt ihrerseits eine Unterregion des Filters mittels ihrer *x*, *y*, *width* und *height* fest.

Eine Kachel beginnt bei *x*, *y* die anderen dann jeweils um ganzzahlige Vielfache von *width* und *height* verschoben, die komplette Unterregion von ***feTile*** ausfüllend.

20.12.1 Beispiele *feTile*

Ein Bild 'Beispiel01.png' wird referenziert und als Kachel verwendet. Der gekachelte Bereich ist als Ergebnis 'Gekacheltes' verfügbar.

```
<feImage xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="Beispiel01.png"
x="0.05" y="0.05" width="0.1" height="0.1"/>
<feTile x="0.1" y="0.1" width="0.8" height="0.8" result="Gekacheltes"/>
```

SVG-Kacheln[24]

24 http://de.wikibooks.org/wiki/media%3ASVGfeTile01.svg

Muster mit ***feTile*** erstellen, hier besteht das Muster aus zwei Ellipsen und den Buchstaben 'SVG', welches alles gedreht ist.

20.13 Urfilter *feMerge* mit *feMergeNode* - Zusammenfügen

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

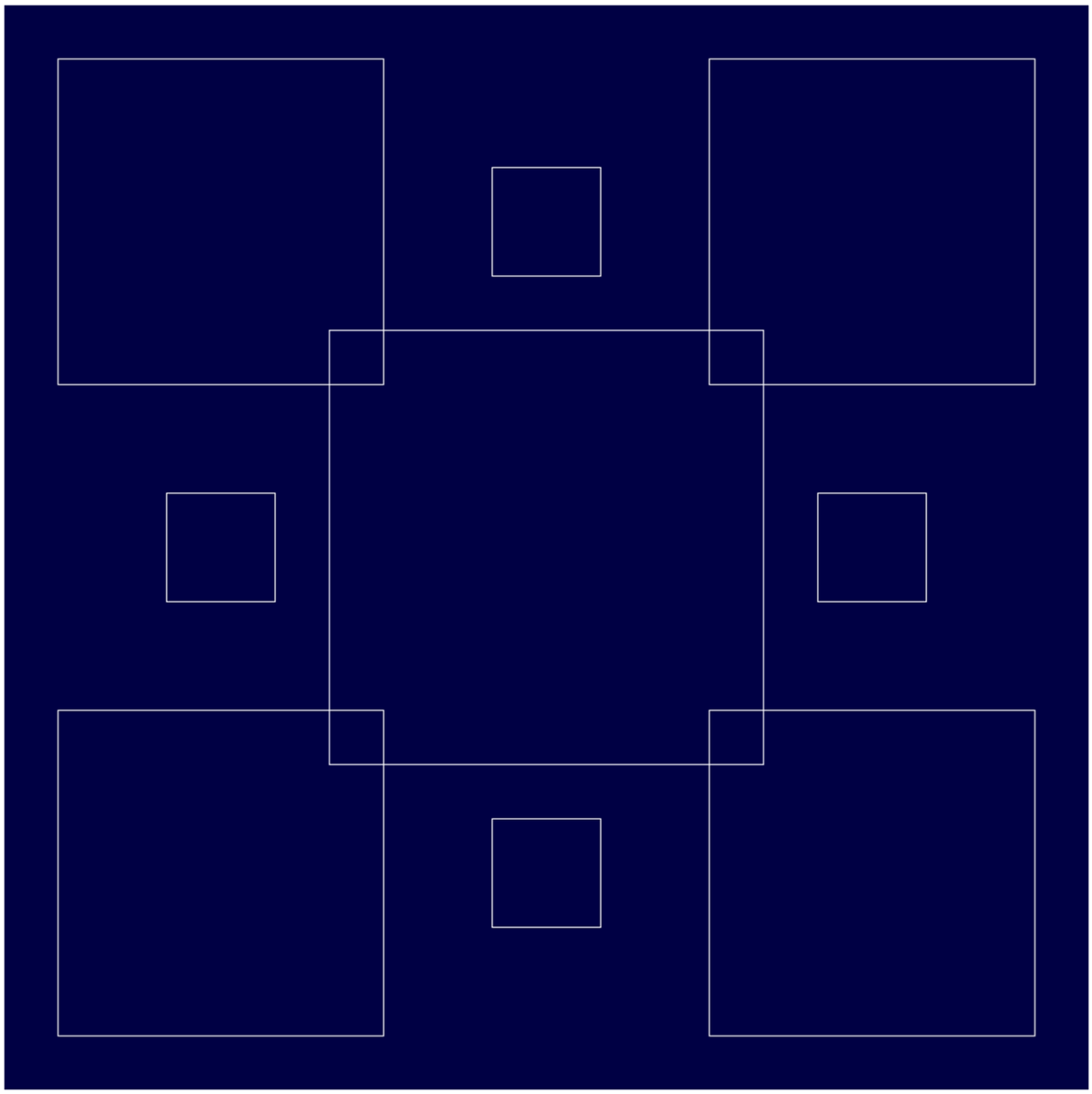


Abb. 328 Zusammenfügen

Mittels ***feMerge*** können Ergebnisse verschiedener Urfilter oder Eingabekanäle zusammengefügt werden. Die zusammenzufügenden Eingaben oder Ergebnisse werden jeweils im Unterelement ***feMergeNode*** notiert. Die Eingaben werden in der Reihenfolge übereinandergelegt, wie sie notiert sind.

20.13.1 Beispiele *feMerge* und *feMergeNode*

Einige Filterergebnisse werden mittels ***feMerge*** und ***feMergeNode*** zusammengefügt:

```
<defs>
 <filter id="f1" filterUnits="userSpaceOnUse"
    x="0" y="0" width="500" height="500">
  <feGaussianBlur in="sourceGraphic" stdDeviation="2 2"
    result="i1" />
  <feOffset in="sourceAlpha" dx="50" dy="50"
    result="i2" />
  <feFlood flood-color="#f0f" flood-opacity="0.5"
    result="i3" />

  <feMerge id="fm" x="0" y="0" width="500" height="500">
    <feMergeNode in="i3" />
    <feMergeNode in="i1" />
    <feMergeNode in="i2" />
  </feMerge>
 </filter>
</defs>
```

Zusammenfügen[25]

Neun animierte SVG-Fragmente werden zu einem Filterergebnis zusammengefügt.

Die richtigen Positionen und Größen werden als Quadrate mit weißem Strich angedeutet.

20.14 Urfilter *feBlend* - überblenden

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Urfilter ***feBlend*** kann eine Komposition aus zwei Eingaben erreicht werden. Die beiden Eingaben werden pixelweise miteinander zur Ausgabe vermischt oder überblendet.

20.14.1 Attribut *in2*

Mit dem Attribut wird die zweite Eingabe festgelegt. Der Attributwert ist der gleiche wie für *in*.

25 http://de.wikibooks.org/wiki/media%3ASVGfeMerge01.svg

Das Attribut ist animierbar.

20.14.2 Attribut *mode*

Mit dem Attribut wird die Art der Vermischung, Überblendung oder Komposition festgelegt.

Die möglichen Werte sind:

normal

Normal

multiply

Multiplizieren

screen

Bildschirm

darken

Abdunkeln

lighten

Aufhellen

Die Voreinstellung ist 'normal'. Der Modus 'normal' entspricht dem Zusammenfügen von ***feMerge*** oder dem Typ 'over' von ***feComposite***. Das entspricht dem was passiert, wenn ohne Filter teiltransparente Elemente übereinanderliegen.

Das Attribut ist animierbar.

Die Überblendung erfolgt gemäß folgender Regeln:

Für den Alpha-Kanal ist die Überblendung immer gleich:

$q_r = 1 - (1-q_a)*(1-q_b)$

Dabei ist q_r die Teiltransparenz des Ergebnisses, q_a und q_b jeweils die Teiltransparenz der erster beziehungsweise zweiten Eingabe.

Für die folgenden Formeln gelten zudem folgende Abkürzungen:

c_r die RGB-Farbe des Ergebnisses, c_a und c_b die RGB-Farbe der erster beziehungsweise zweiten Eingabe.

Min und Max sind jeweils Funktionen, die das Minimum, beziehungsweise Maximum der darin mit Komma separierten Werte bestimmen.

normal

$c_r = (1 - q_a) * c_b + c_a$

multiply

$c_r = (1-q_a)*c_b + (1-q_b)*c_a + c_a*c_b$

screen

$c_r = c_b + c_a - c_a * c_b$

darken

$c_r = \text{Min}\ ((1 - q_a) * c_b + c_a, (1 - q_b) * c_a + c_b)$

lighten

$c_r = \text{Max}\ ((1 - q_a) * c_b + c_a, (1 - q_b) * c_a + c_b)$

20.14.3 Beispiele *feBlend*

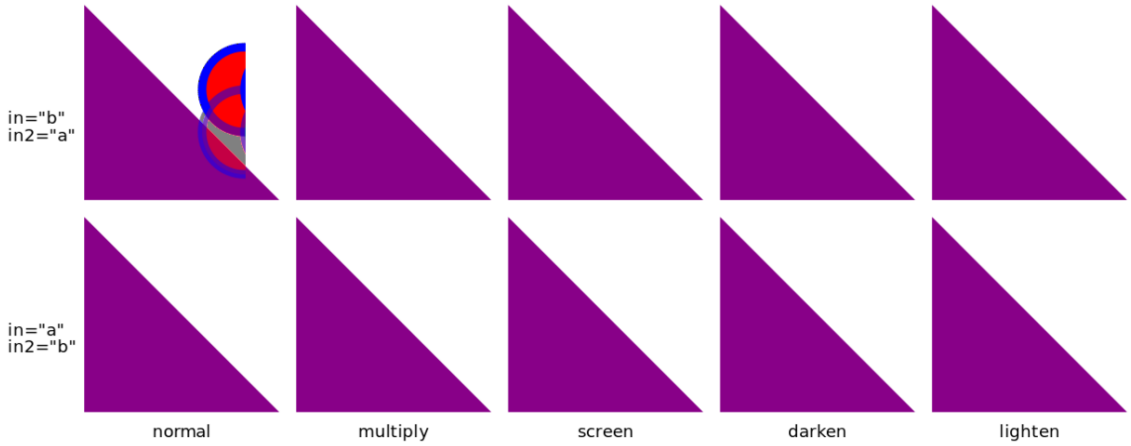


Abb. 329 Überblenden (1)

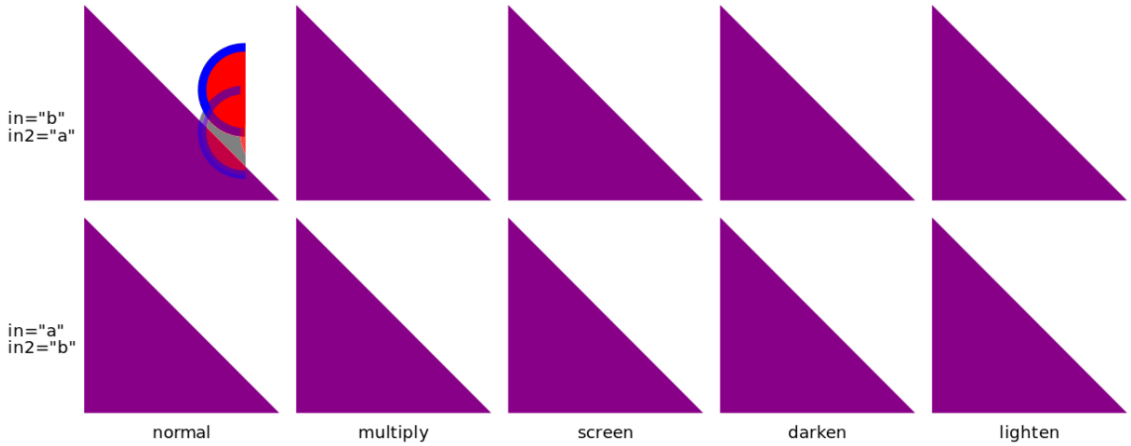


Abb. 330 Überblenden (2)

Zwei Bilder werden überblendet. Per Animation wird die Überblendmethode verändert.

```
<filter id="f1" xmlns:xlink="http://www.w3.org/1999/xlink"
    filterUnits="userSpaceOnUse"
    x="0" y="0" width="1" height="1">
  <feImage id="fi1" xlink:href="Beispiel01.png"
    x="0.2" y="0.2" width="0.4" height="0.4" result="i1" />
  <feImage id="fi2" xlink:href="Beispiel02.png"
    x="0.4" y="0.4" width="0.4" height="0.4" result="i2" />
  <feBlend id="fb" in="i1" in2="i2" mode="multiply">
    <animate attributeName="mode"
        values="normal;screen;darken;lighten"
        begin="3s"
        dur="12s" />
  </feBlend>
</filter>
```

Überblenden (1)[26]

Die zehn verschiedenen Möglichkeiten der Überblendung von zwei Bildern werden dargestellt. Das erste Bild a zeigt zwei rote Kreise mit blauem Rand, einer davon teiltranparent. Das zweite Bild b zeigt zwei grüne Kreise mit blauem Rand, einer davon teiltranparent. Die Kreise überlappen jeweils etwas.

Überblenden (2)[27]

Wie das vorherige Beispiel, hier nur für b mit gelben Kreisen mit rotem Rand, um zu sehen, was passiert, wenn a und b gemeinsame Farbkanäle haben.

20.15 Urfilter *feColorMatrix* - Farb-Matrix

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Der Urfilter ermöglicht eine Matrixtransformation im RGBA-Raum einschließlich Verschiebung. Die Matrix wird in Form einer 5x5-Matrix verwendet, also jeweils eine Dimension für die Kanäle R (rot), G (grün), B(blau), A (Alpha, Transparenz) und einer zusätzlichen Dimension für die Verschiebung.

Aus den Kanälen **E** = (R,G,B,A,1) vor der Transformation ergeben sich die Kanäle **A** = (R',G',B',A',1) nach der Transformation und T_{ae} die Matrix beziehungsweise die Komponenten davon (a und e von 0 bis 4 und $T_{44} = 1$ und sonst $T_{a4} = 0$

$\mathbf{A} = T_{ae}\ \mathbf{E}$

Die Transformation erfolgt dann jeweils für jeden Pixel.

26 http://de.wikibooks.org/wiki/media%3ASVGfeBlend01.svg
27 http://de.wikibooks.org/wiki/media%3ASVGfeBlend02.svg

20.15.1 Attribut *type*

Mit dem Attribut *type* wird angegeben, welche Transformation ausgeführt werden soll. Das ist vor allem eine Vereinfachung, die es einem erspart, für immer wiederkehrende Transformationen immer alle Matrixkomponenten anzugeben.

Mögliche Werte sind:

matrix

Die Matrix wird komplett angegeben

saturate

Sättigung

hueRotate

Farbtondrehung

luminanceToAlpha

Leuchtdichte in Alpha-Kanal

Das Attribut ist animierbar. Die einzelnen Möglichkeiten werden im Folgenden näher definiert. Die Voreinstellung ist 'matrix'.

20.15.2 Attribut *values*

Das Attribut gibt die Komponenten oder Parameter der Transformation in Form einer Liste von Zahlen an.

Das Attribut ist animierbar. Voreinstellung hängt von *type* ab, entspricht aber immer der Identitätstransformation, Ausgabe ist also gleich der Eingabe.

Der genaue Wert von *values* hängt von *type* ab:

values für *type*='matrix'

Der Wert ist eine Liste mit genau zwanzig Zahlen, die den T_{ae} entsprechen:

T_{00}, T_{01}, T_{02}, T_{03}, T_{04}, T_{10}, T_{11}, T_{12}, T_{13}, T_{14}, T_{20}, T_{21}, T_{22}, T_{23}, T_{24}, T_{30}, T_{31}, T_{32}, T_{33}, T_{34}

Voreinstellung ist in diesem Falle '1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0' - die Einheitsmatrix.

values für *type*='saturate'

Der Wert ist exakt eine Zahl zwischen 0 bis 1 jeweils einschließlich, im folgenden s genannt:

T_{00} = 0.213+0.787s

T_{01} = 0.715-0.715s

T_{02} = 0.072-0.072s

T_{10} = 0.213-0.213s

T_{11} = 0.715+0.285s

T_{12} = 0.072-0.072s

T_{20} = 0.213-0.213s

T_{21} = 0.715-0.715s

T_{22} = 0.072+0.928s

T_{33} = 1

T_{44} = 1

Die restlichen Komponenten sind 0. Voreinstellung ist in diesem Falle s = 1.

***values* für *type*='hueRotate'**

Der Wert ist exakt eine Zahl, einen Drehwinkel in Grad repäsentierend, im folgenden a genannt:

T_{00} = 0.213 + cos(a)*0.787 - sin(a)*0.213

T_{01} = 0.715 - cos(a)*0.715 - sin(a)*0.715

T_{02} = 0.072 - cos(a)*0.072 + sin(a)*0.928

T_{10} = 0.213 - cos(a)*0.213 + sin(a)*0.143

T_{11} = 0.715 + cos(a)*0.285 + sin(a)*0.140

T_{12} = 0.072 - cos(a)*0.072 - sin(a)*0.283

T_{20} = 0.213 - cos(a)*0.213 - sin(a)*0.787

T_{21} = 0.715 - cos(a)*0.715 + sin(a)*0.715

T_{22} = 0.072 + cos(a)*0.928 + sin(a)*0.072

T_{33} = 1

$T_{44} = 1$

Die restlichen Komponenten sind 0. Voreinstellung ist in diesem Falle die a = 0.

***values* für *type*='luminanceToAlpha'**

Für diesen Type ist *values* nicht anwendbar.

$T_{30} = 0.2125$

$T_{31} = 0.7154$

$T_{32} = 0.0721$

$T_{44} = 1$

Die restlichen Komponenten sind 0.

20.15.3 Beispiele *feColorMatrix*

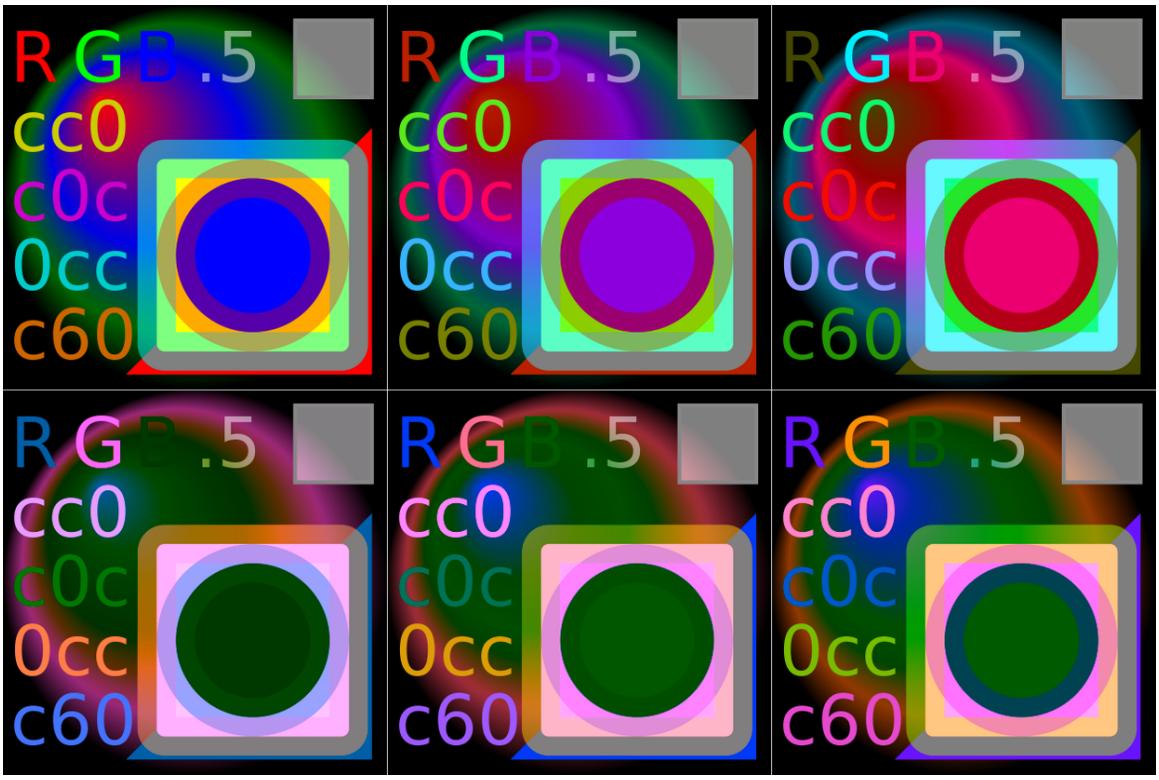


Abb. 331 Farbrotation

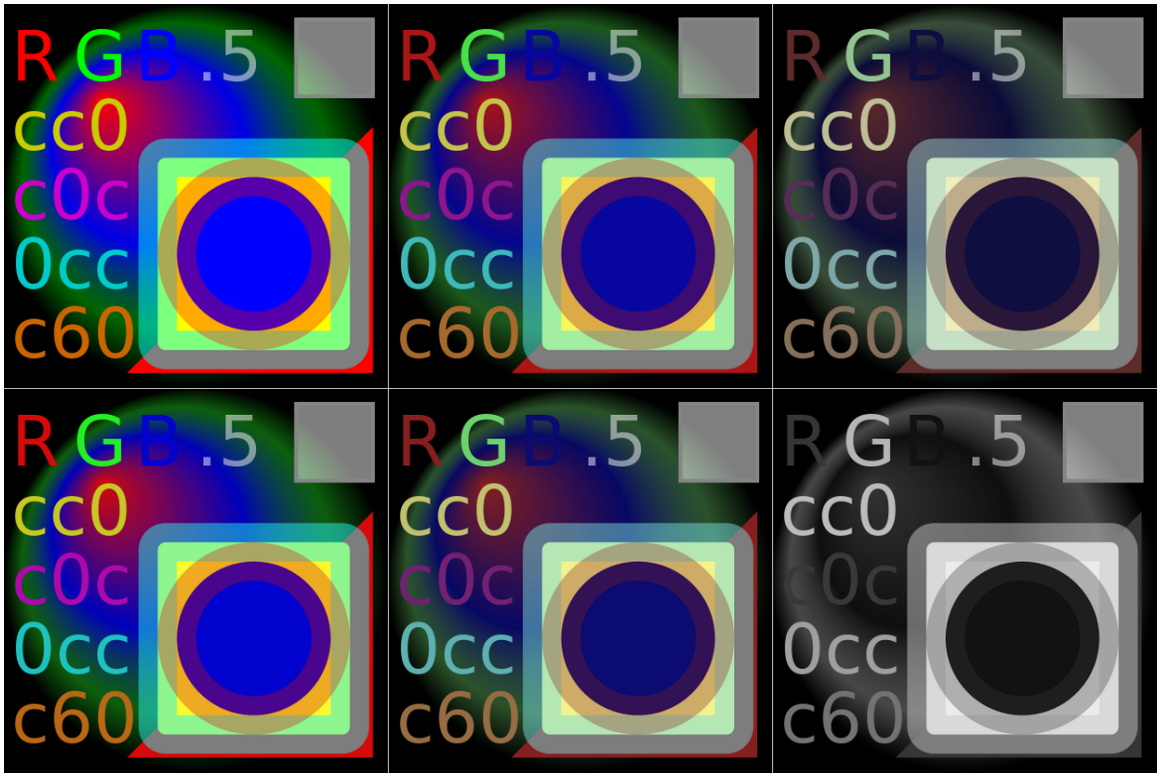


Abb. 332 Sättigung

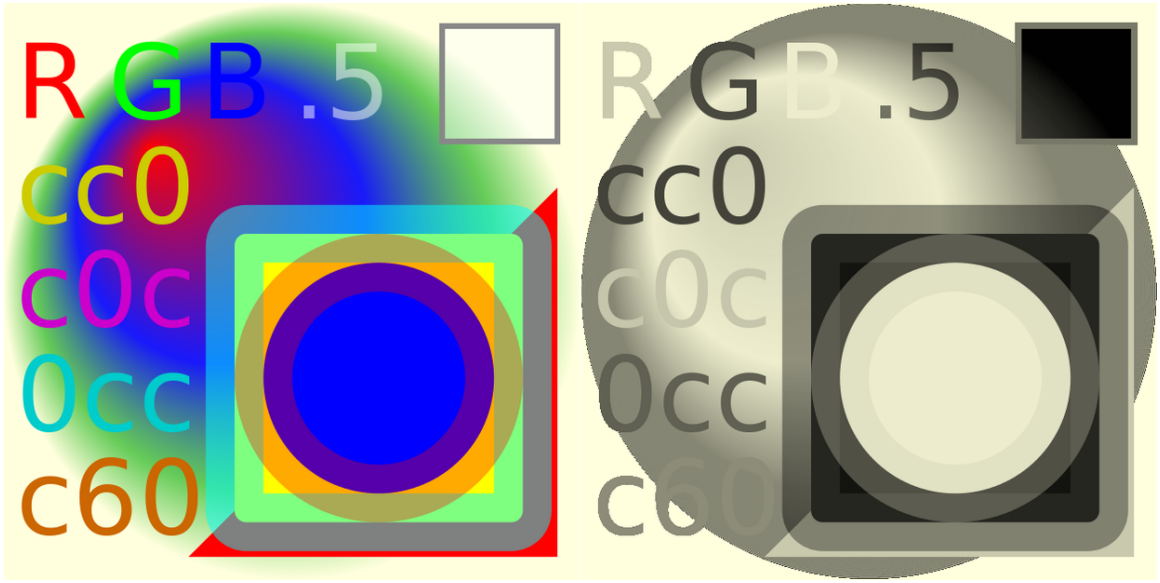


Abb. 333 Leuchtdichte

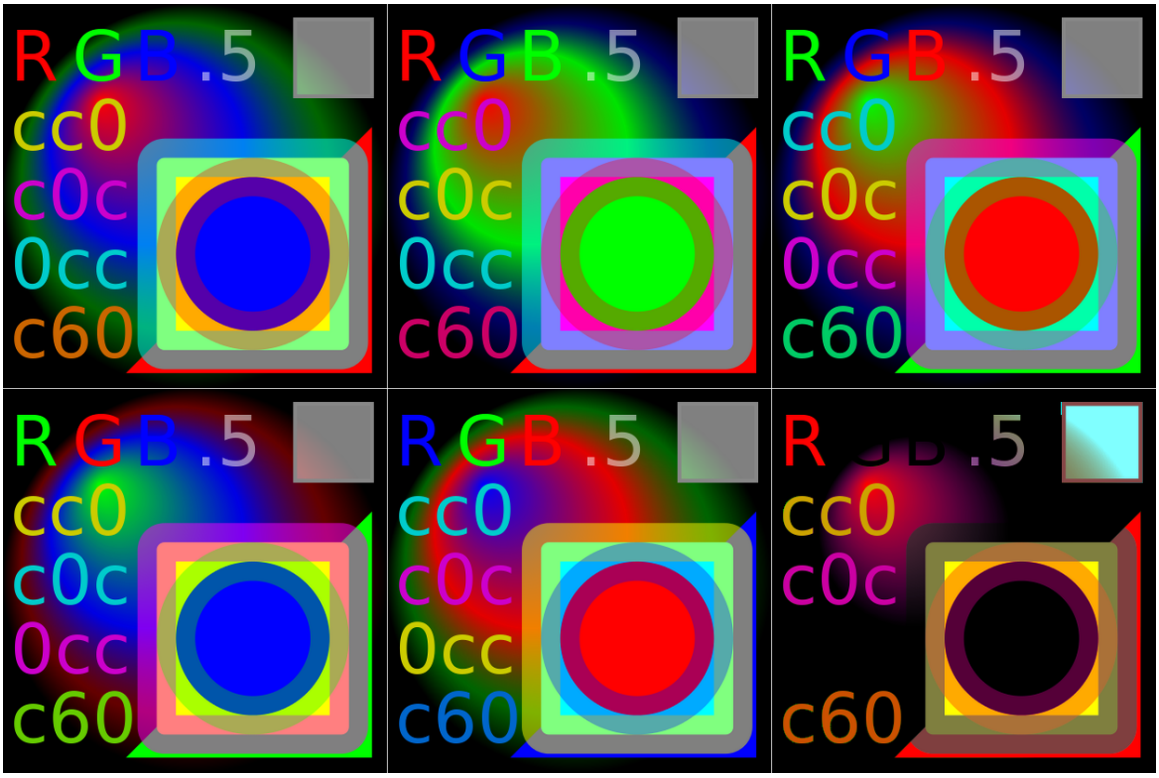


Abb. 334 Tauschen

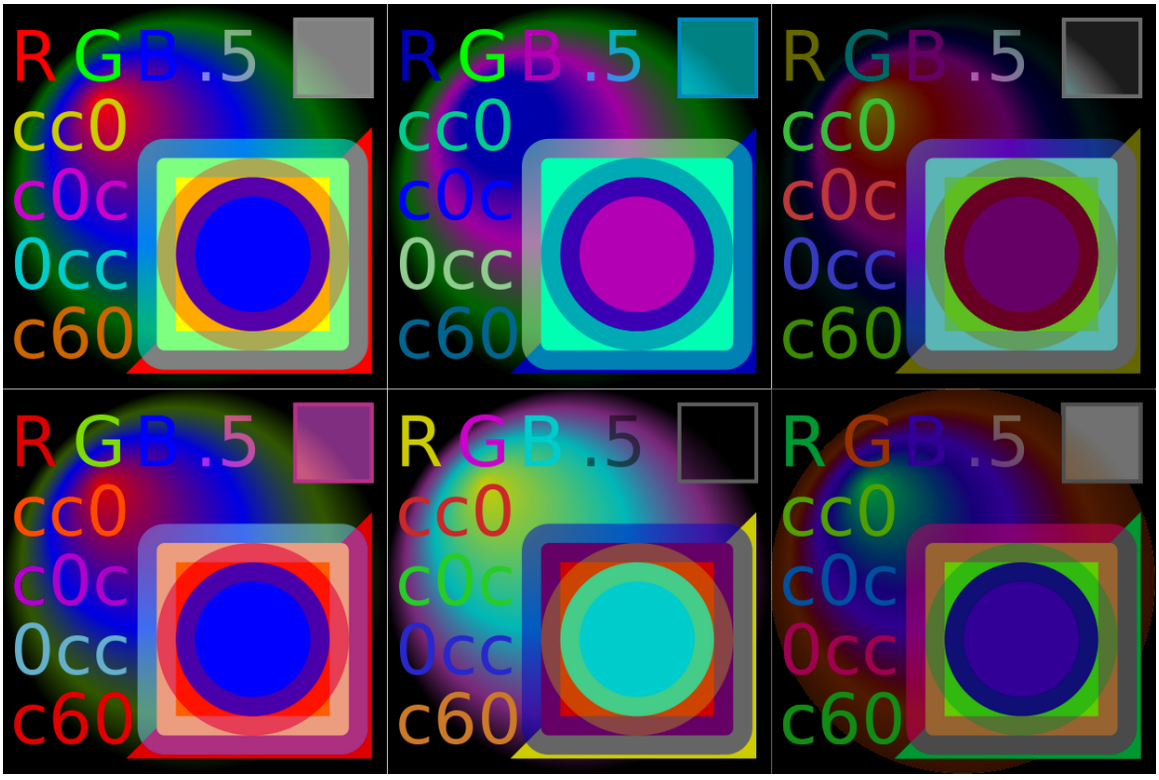


Abb. 335 mehr ...

Ein Filter, bei dem sich der Farbton per Animation einmal pro Minute komplett dreht:

```
<filter id="feCM" filterUnits="userSpaceOnUse"
x="0" y="0" width="500" height="500">
<feColorMatrix id="CM"
x="0" y="0" width="500" height="500"
type="hueRotate"
values="180">
<animate attributeName="values"
         values="0;360"
         dur="60s" repeatDur="indefinite"/>
</feColorMatrix>
</filter>
```

Farbmatrix vom Typ 'matrix' (abdunkeln und drehen im Unterraum rot und grün und halbdurchsichtig machen):

```
<feColorMatrix id="CM"
   x="0" y="0" width="500" height="500"
  type="matrix"
  values="
.7 .7 0 0 0
-.7 .7 0 0 0
0 0 1 0 0
0 0 0 .5 0" />
```

Einige Beispiele zur Transformation von Farbkanälen per Matrix. Links oben ist das ungefilterte Ausgangsbild zu sehen:

Farbrotation[28]

Es wird eine Farbrotation angewendet, Differenzwinkel jeweils 60 Grad.

Sättigung[29]

Es wird ein Sättigungswert angegeben, jeweils um 0.2 fallend.

Leuchtdichte[30]

Bei dem Filterergebnis rechts wird die Leuchtdichte in den Alphakanal konvertiert.

Tauschen[31]

Die Matrizen werden hier nur verwendet, um Farbkanäle miteinander zu vertauschen.

mehr...[32]

28 http://de.wikibooks.org/wiki/media%3ASVGfeColorMatrix01.svg
29 http://de.wikibooks.org/wiki/media%3ASVGfeColorMatrix02.svg
30 http://de.wikibooks.org/wiki/media%3ASVGfeColorMatrix03.svg
31 http://de.wikibooks.org/wiki/media%3ASVGfeColorMatrix04.svg
32 http://de.wikibooks.org/wiki/media%3ASVGfeColorMatrix05.svg

Weitere Beispiele...

20.16 Urfilter *feComponentTransfer* mit *feFuncR*, *feFuncG*, *feFuncB* und *feFuncA*

Squiggle (Batik)	-
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Pixelweise werden die Farbkanäle und der Alphakanal transformiert. Mit den Buchstaben R (rot), G (grün), B(blau), A (Alpha, Transparenz) für die Kanäle ergibt sich aus den Kanälen R,G,B,A, vor der Transformation durch die Transformation die Kanäle R',G',B',A' nach der Transformation:

R' = feFuncR(R)

G' = feFuncG(G)

B' = feFuncB(B)

A' = feFuncA(A)

Das erlaubt zum Beispiel Änderungen der Helligkeit, des Kontrastes und der Farbbalanz.

Während ***feComponentTransfer*** Ein- und Ausgabe und Unterregion festlegt, wird mit den Unterelementen ***feFuncR***, ***feFuncG***, ***feFuncB*** und ***feFuncA*** die Transformation festgelegt.

Wird eines der Unterelemente mehrmals im selben ***feComponentTransfer*** notiert, so wird nur das letzte angewendet. Ist ein Unterelement gar nicht angegeben, so wird für den entsprechenden Kanal die Identitätstransformation angenommen.

Sei im Folgenden C der Wert vor der Transformation eines Kanals, C' danach. Der Wert ist jeweils als Wert aus dem Intervall [0;1] anzunehmen, also 0 für keinen Anteil, 1 für vollen Anteil.

Folgende Attribute sind anwendbar auf all diese Unterelemente:

20.16.1 Attribut *type*

Mit dem Attribut *type* wird angegeben, welche Transformation ausgeführt werden soll. Es legt fest, ob die anderen Attributen eine Bedeutung haben oder welche Bedeutung die anderen Attribute haben.

Mögliche Werte sind:

identity

Identität, keine Änderung

table

Nach Tabelle

discrete

Diskrete Tabelle

linear

Affine Transformation

gamma

Exponentielle Funktion

Das Attribut ist animierbar.

20.16.2 Attribut *tableValues*

Mit dem Attribut wird für Typen 'table' und 'discrete' eine Tabelle zum Nachschlagen von Werten angegeben- Der Wert ist eine Liste von Zahlen.

Eine leere Liste entspricht einer Identitätstransformation (keine Änderung). Ist das Attribut nicht angegeben, so wird ebenfalls eine Identitätstransformation angenommen.

Das Attribut ist animierbar.

tableValues für *type*='table'

tableValues sei eine Liste von n+1 Werten v_0 bis v_n und *type* sei 'table'.

Für einen Wert C aus [0;1) wähle k, so dass k/n <= C < (k+1)/n

(oder k=floor(C*n); floor heißt abrunden auf die nächste kleinere ganze Zahl), dann ergibt sich:

$C' = v_k + (C - k/n)*n * (v_{k+1} - v_k)$

Für C = 1 ist $C' = v_n$.

tableValues für *type*='discrete'

tableValues sei eine Liste von n Werten v_0 bis v_{n-1} und *type* sei 'discrete'.

Für einen Wert C aus [0;1) wähle k, so dass k/n <= C < (k+1)/n

(oder k=floor(C*n); floor heißt abrunden auf die nächste kleinere ganze Zahl), dann ergibt sich:

$C' = v_k$

Für C = 1 ist $C' = v_{n-1}$.

20.16.3 Attribute *slope* und *intercept*

Die Attribute finden beim Typ 'linear' Anwendung. Der Wert von beiden ist jeweils eine Zahl.

slope ist die Steigung, *intercept* ist die Konstante einer affinen Funktion. Ist *slope* nicht angenommen, wird '1' dafür angenommen, ist *intercept* nicht angegeben, so wid '0' dafür angenommen.

Beide Attribute sind animierbar.

Es gilt im Falle von *type*='linear':

$C' = slope * C + intercept$

20.16.4 Attribute *amplitude*, *exponent* und *offset*

Die Attribute finden beim Typ 'gamma' Anwendung. Der Wert von ihnen ist jeweils eine Zahl.

amplitude ist der Vorfaktor, *exponent* der Exponent einer Exponentialfunktion, *offset* ist eine zusätzuliche additive Konstante.

Alle drei Attribute sind animierbar.

Es gilt im Falle von *type*='gamma':

$C' = amplitude * C^{exponent} + offset$

20.16.5 Beispiele *feComponentTransfer* mit *feFuncR*, *feFuncG*, *feFuncB* und *feFuncA*

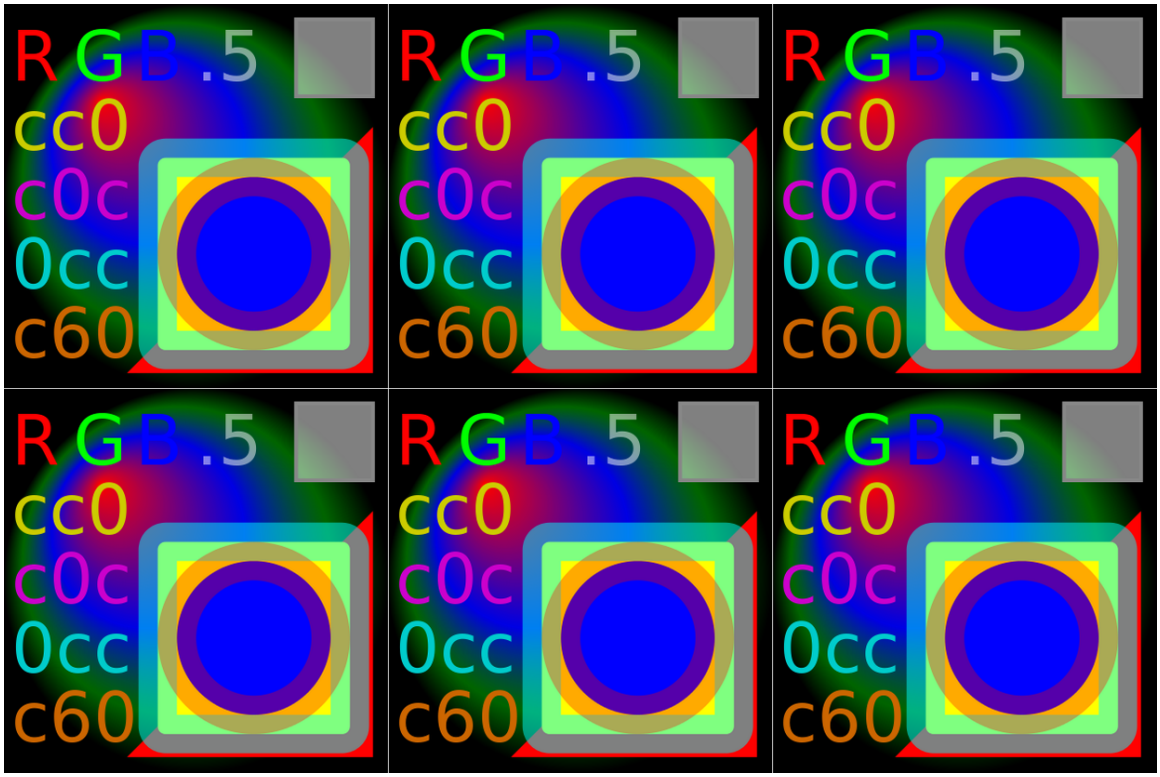


Abb. 336 linear

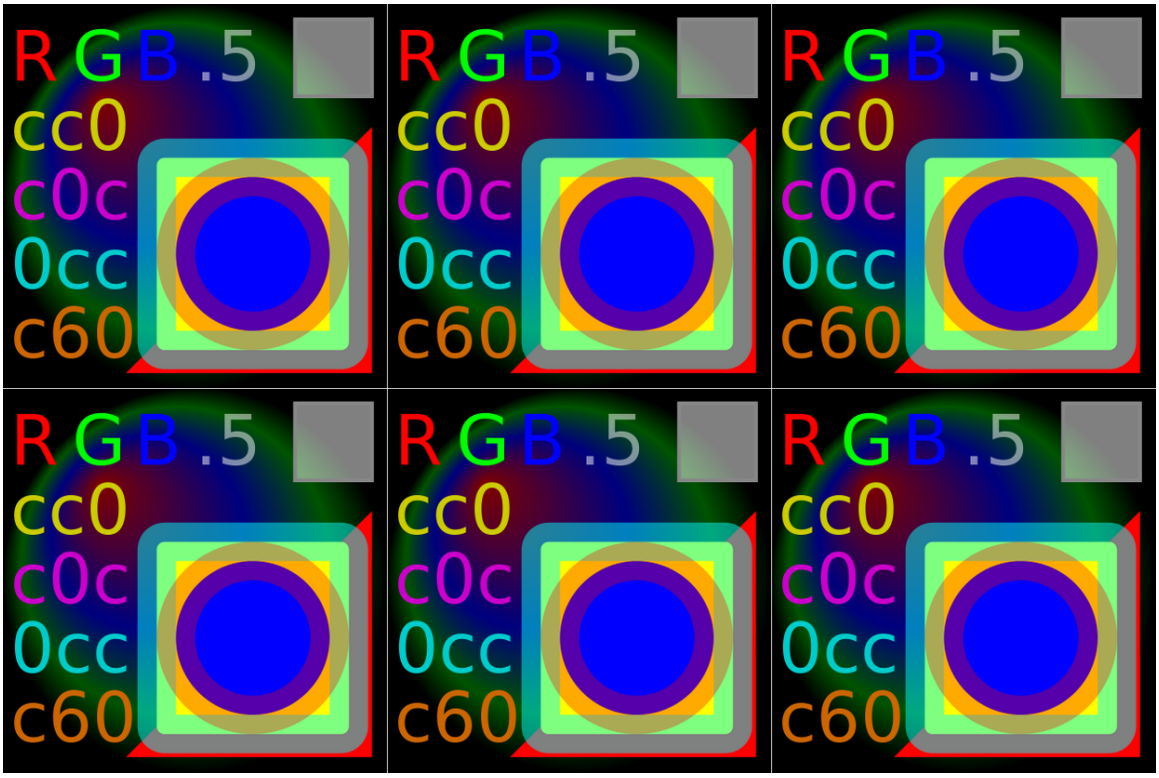


Abb. 337 gamma

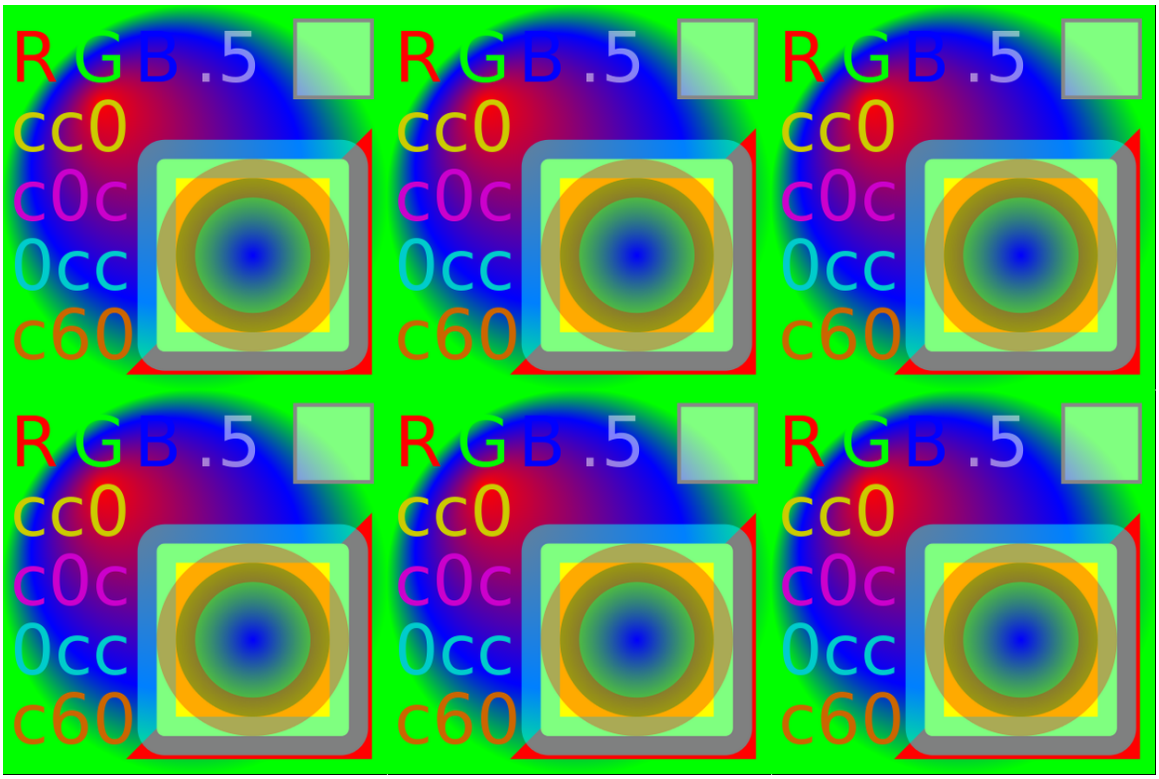


Abb. 338 diskret nach Tabelle

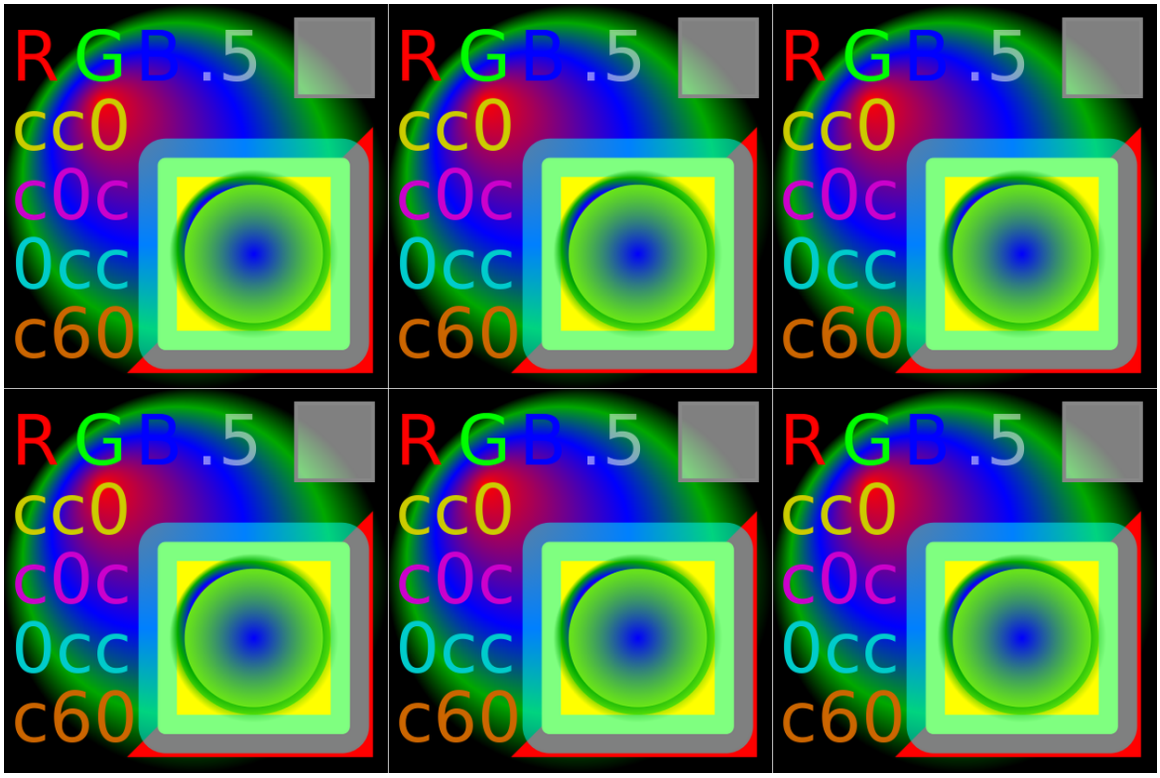


Abb. 339 nach Tabelle

Jeder Kanal wird mit einer jeweils anderen Methode modifiziert:

```
<feComponentTransfer x="0" y="0" width="500" height="500">
  <feFuncR type="linear" slope="0.5" intercept="0.1" />
  <feFuncG type="gamma" exponent="-0.1" amplitude="0.5" offset="0.5" />
  <feFuncB type="table" tableValues="0 .1 1 .5 .7 .2 1" />
  <feFuncA type="discrete" tableValues="0 .1 1 .5 .7 .2 1" />
</feComponentTransfer>
```

Einige Beispiele zur Transformation von Farbkanälen. Links oben ist das ungefilterte Ausgangsbild zu sehen:

linear[33]

Filter vom Typ linear.

gamma[34]

Filter vom Typ gamma.

33 http://de.wikibooks.org/wiki/media%3ASVGfeComponentTransfer01.svg
34 http://de.wikibooks.org/wiki/media%3ASVGfeComponentTransfer02.svg

diskret nach Tabelle[35]

Filter vom Typ discrete.

nach Tabelle[36]

Filter vom Typ table.

20.17 Urfilter *feComposite*

Squiggle (Batik)	1.7 (teilweise)
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Urfilter können zwei Eingaben miteinander kombiniert werden, wobei Kompostionsmethoden nach Thomas Porter und Tim Duff verwendet werden (over, in, atop, out, xor). Stattdessen kann auch eine komponentenweise arithmetische Operation verwendet werden.

20.17.1 Attribut *in2*

Mit dem Attribut wird die zweite Eingabe festgelegt. Der Attributwert ist der gleiche wie für *in*.

Das Attribut ist animierbar.

20.17.2 Attribut *operator*

Mit dem Attribut wird festgelegt, welcher Operation zur Kombination verwendet werden soll.

Mögliche Werte sind nach den Operationen gemäß Porter und Duff benannt: 'over', 'in', 'out', 'atop' und 'xor'.

Hinzu kommt noch der arithmetische Operator mit dem Wert 'arithmetic'.

Voreinstellung ist 'over'. Das entspricht dem normalen Übereinanderlegen.

Das Attribut ist animierbar.

Die Komposition erfolgt pixelweise. Sei A ein Farbwert des Pixels von *in* mit der Opazität a und B ein Farbwert des Pixels von *in2* mit Opazität b.

35 http://de.wikibooks.org/wiki/media%3ASVGfeComponentTransfer03.svg
36 http://de.wikibooks.org/wiki/media%3ASVGfeComponentTransfer04.svg

'over' entspricht dem üblichen Überlagern in SVG.

Das exakte Ergebnis für den Farbkanal ist C = a*A + (1-a)*b*B.

'in' wählt einen Anteil von A aus, wenn b nicht transparent ist.

Das exakte Ergebnis für den Farbkanal ist C = a*b*A

'out' wählt einen Anteil von A aus, wenn b transparent ist.

Das exakte Ergebnis für den Farbkanal ist C = a*(1-b)*A

'atop' fügt einen Anteil von A zu B hinzu, wenn b nicht transparent ist.

Das exakte Ergebnis für den Farbkanal ist C = a*b*A + (1-a)*b*B

'xor' ist ein anteiliges entweder oder von A und B.

Das exakte Ergebnis für den Farbkanal ist C = a*(1-b)*A + (1-a)*b*B

Um die jeweilige andere Kombination hinzugekommen, sind natürlich *in* und *in2* zu vertauschen, im Zweifelsfalle per Animation während der Darstellung.

20.17.3 Attribute *k1*, *k2*, *k3*, *k4*

Die Attribute sind nur für den Operator 'arithmetic' anwendbar und stellen die dafür notwendigen Parameter bereit.

Sofern einer davon nicht angegeben ist, wird für diesen '0' angenommen.

Die Attribute sind animierbar.

Ist *operator*='arithmetic' angegeben, so ist für jeden Kanal:

Ausgabe = *k1*in*in2 + k2*in + k3*in2 + k4*

Dabei ist *in* und *in2* die jeweilige Eingabe gemäß der gleichnamigen Attribute und die *ki* sind die Werte der jeweiligen Attribute.

Die Ausgabe wird auf den erlaubten Bereich eingeschränkt.

20.17.4 Beispiele *feComposite*

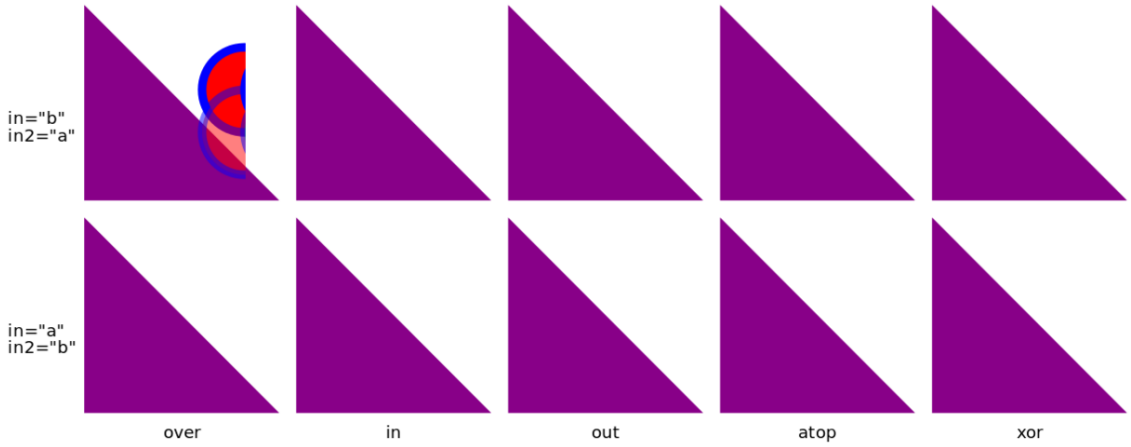


Abb. 340 Kompositionen, einfache Operatoren

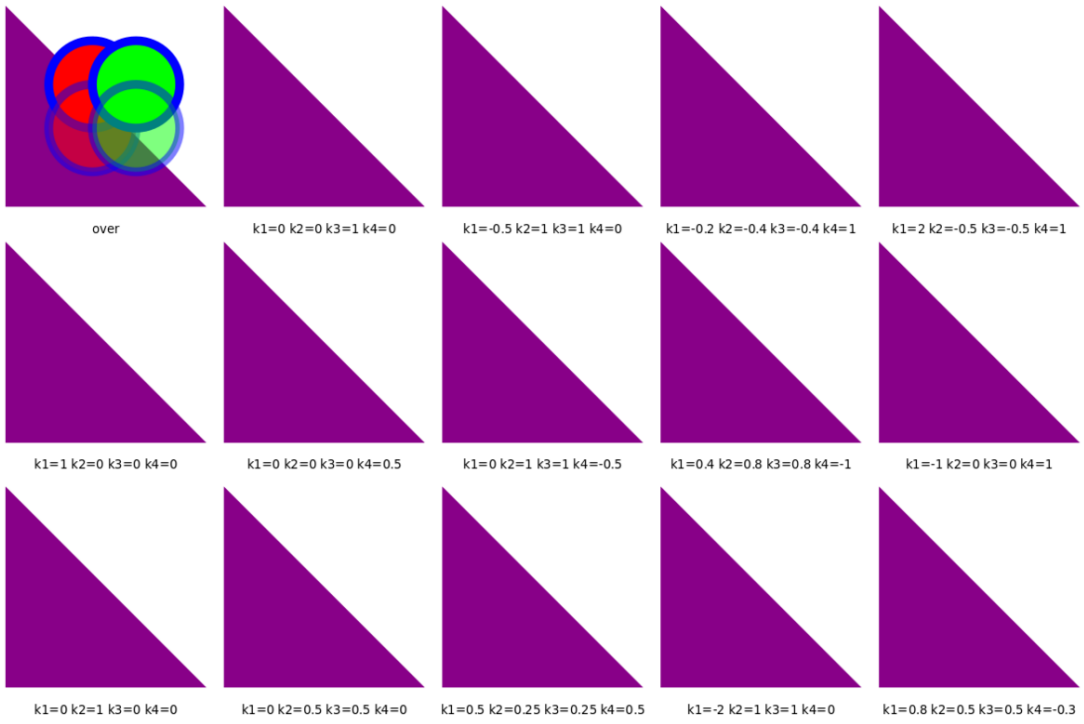


Abb. 341 Kompositionen, arithmetischer Operator

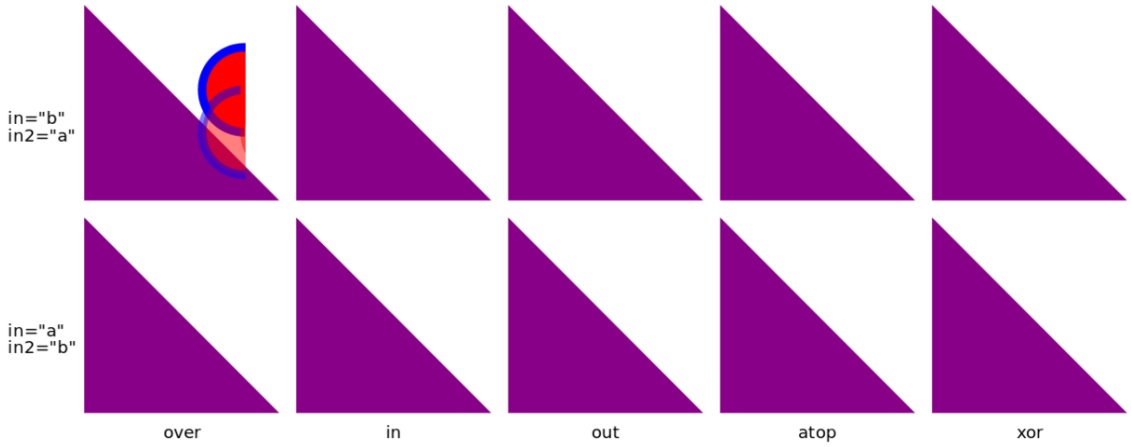


Abb. 342 Kompositionen, einfache Operatoren (2)

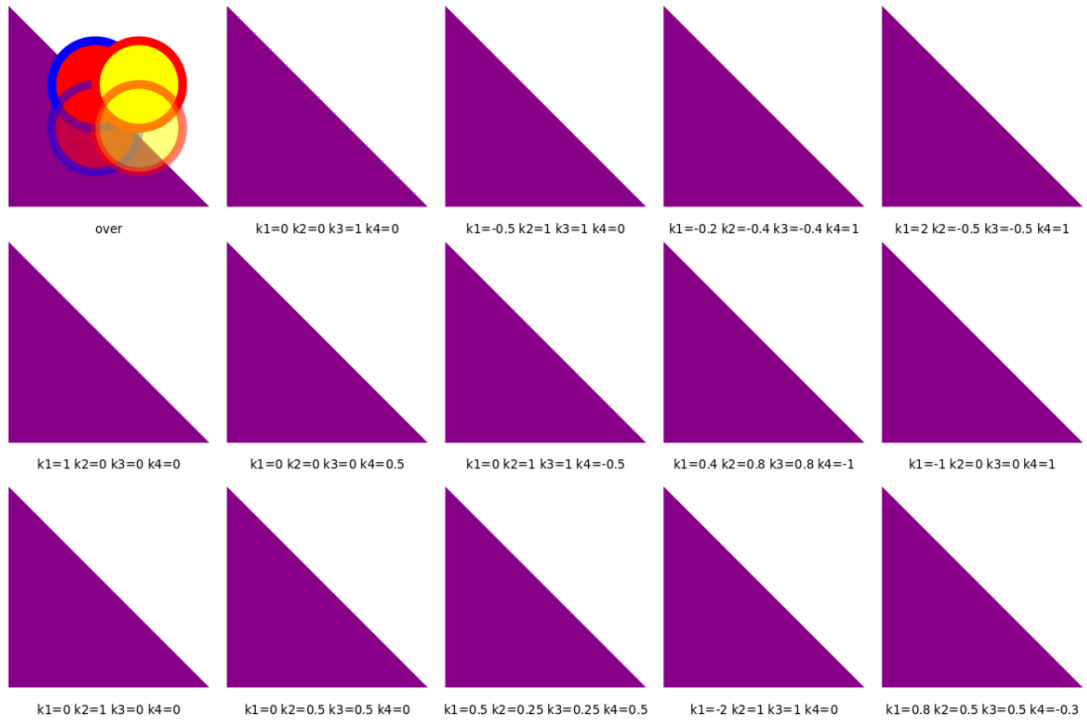


Abb. 343 Kompositionen, arithmetischer Operator (2)

Für die Eingaben werden per Animation die Operatoren nach Porter und Duff durchprobiert:

```
<feComposite in="i1" in2="i2" operator="over">
<animate attributeName="operator"
        values="in; atop; out; xor; over"
        begin="6s"
        dur="60s" />
<animate attributeName="in"
        values="i1;i2"
        begin="6s"
```

```
        dur="120s" />
<animate attributeName="in2"
        values="i2;i1"
        begin="6s"
        dur="120s" />
</feComposite>
```

Durch Animation werden verschiedene Kombinationen der Parameter des arithmetischen Operators durchprobiert:

```
<feComposite
    in="i1" in2="i2" operator="arithmetic"
    k1=".5" k2=".5" k3=".5" k4=".5">
  <animate attributeName="k1"
        from="0"
        to="1"
        begin="3s"
        dur="3s" />
  <animate attributeName="k2"
        from="0"
        to="1"
        begin="6s"
        dur="3s" />
  <animate attributeName="k3"
        from="0"
        to="1"
        begin="9s"
        dur="3s" />
  <animate attributeName="k4"
        from="0"
        to="1"
        begin="12s"
        dur="3s" />
</feComposite>
```

Kompositionen, einfache Operatoren[37]

Zehn verschiedene Möglichkeiten zur Komposition von zwei Bildern werden dargestellt. Das erste Bild a zeigt zwei rote Kreise mit blauem Rand, einer davon teiltranparent. Das zweite Bild b zeigt zwei grüne Kreise mit blauem Rand, einer davon teiltranparent. Die Kreise überlappen jeweils etwas.

Kompositionen, arithmetischer Operator[38]

Einige verschiedene Möglichkeiten zur Komposition mit dem Operator 'arithmetic' von zwei Bildern werden dargestellt. Zum Vergleich wird oben links auch noch der Operator 'over' dargestellt. Das erste Bild a zeigt zwei rote Kreise mit blauem Rand, einer davon teiltranparent. Das zweite Bild b zeigt zwei grüne Kreise mit blauem Rand, einer davon teiltranparent. Die Kreise überlappen jeweils etwas.

Kompositionen, einfache Operatoren (2)[39]

37 http://de.wikibooks.org/wiki/media%3ASVGfeComposite01.svg
38 http://de.wikibooks.org/wiki/media%3ASVGfeComposite02.svg
39 http://de.wikibooks.org/wiki/media%3ASVGfeComposite03.svg

Kompositionen, arithmetischer Operator (2)[40]

Wie die vorherigen Beispiele, hier nur für b mit gelben Kreisen mit rotem Rand, um zu sehen, was passiert, wenn a und b gemeinsame Farbkanäle haben.

20.18 Urfilter *feConvolveMatrix*

Squiggle (Batik)	1.7 (teilweise falsch oder gar nicht)
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (teilweise)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Urfilter ***feConvolveMatrix*** kann mittels einer Matrix eine Faltung vorgenommen werden. In der Faltung werden benachbarte Pixel miteinander kombiniert. Die Matrixkomponenten sind jeweils Gewichte, mit der die benachbarten Pixel aufaddiert werden. Die Position der Pixel korrespondiert mit der Position der Matrixkomponenten zueinander.

Weil die Faltung auf Pixelbasis stattfindet, empfiehlt sich die explizite Festlegung von *filterRes* oder *kernelUnitLength*, wenn das Ergebnis unabhängig von den aktuellen Anzeigebedingungen sein soll.

20.18.1 Attribut *order*

Das Attribut legt die Größe der Matrix fest. Der Wert ist eine positive ganze Zahl oder eine Liste von zwei positiven ganzen Zahlen. Ist eine Zahl angegeben, so gibt sie die Dimension der Matrix in x- und y-Richtung an. Sind zwei Zahlen angeben, gibt die erste (n) die Dimension in x-Richtung an, die zweite (m) in y-Richtung.

Ist das Attribut nicht angegeben, wird '3' angenommen. Aufgrund der Prozessorbelastung wird empfohlen, die Dimension relativ klein zu halten.

Das Attribut ist animierbar.

20.18.2 Attribut *kernelMatrix*

Das Attribut gibt die *kernelMatrix* als Liste von Zahlen an, bei order='n m' für eine n*m -Matrix sind dies also n*m Werte. Die Zeilen der Matrix werden nacheinander notiert, also erst alle Komponenten der ersten Zeile, dann der zweiten etc.

Das Attribut ist animierbar.

40 `http://de.wikibooks.org/wiki/media%3ASVGfeComposite04.svg`

20.18.3 Attribut *divisor*

Der *divisor* ist eine Zahl, durch welche das Ergebnis der Faltung geteilt wird. Voreinstellung ist die Summe über alle Matrixkomponenten, falls diese nicht gerade 0 ist, in dem Falle wird 1 angenommen. Der Wert 0 selbst ist ein Fehler und darf nicht angegeben werden. Die Voreinstellung ist eine Art Normierung, welche einen gewissen Ausgleich bietet und es leichter wird, Ergebnisse zu vermeiden, die außerhalb des erlaubten Bereiches für Farbwerte oder Opazitätswerte liegen.

Das Attribut ist animierbar.

20.18.4 Attribut *bias*

bias ist eine Zahl, welche zu jeder Komponenten des Ergebnisses der Faltung addiert wird, nachdem das Ergebnis durch den *divisor* geteilt wurde. Voreinstellung ist 0.

Das Attribut ist animierbar.

20.18.5 Attribute *targetX* und *targetY*

Die Attribute geben die Postion des Zielpixels der Faltung an, also anschaulich welche Matrixkomponente die Mitte darstellt. *targetX* also die Komonente in x-Richtung, *targetY* in y-Richtung.

Der jeweilige Wert liegt also eine nicht negative Zahl und ist kleiner als die Dimension in der jeweiligen Richtung. Links oben ist also für beide der Wert 0. Rechts unten ist jeweils die Dimension minus 1.

Die Voreinstellung ist jeweils die Mitte der Matrix, genauer die jeweilige Dimension geteilt durch 2, ist dies keine ganze Zahl, so wird auf die nächstkleinere Zahl gerundet.

Die Attribute sind animierbar.

20.18.6 Attribut *edgeMode*

An den Kanten der Filterregion tritt das Problem auf, dass mit der Matrix auch Pixel jenseits der Kante benötigt würden, die aber nicht verfügbar sind. Mit dem Attribut *edgeMode* wird festgelegt, wie mit solchen fehlenden Pixeln zu verfahren ist.

Mögliche Werte sind:

none

Für fehlende Pixel wird 0 angenommen

duplicate

Für fehlende Pixel wird jeweils der der letzte verfügbare Pixel verwendet, effektiv werden also die Kanten solange wiederholt, bis alle Pixel verfügbar sind

wrap

Für fehlende Pixel wird eine periodische Fortsetzung der Filterregion angenommen, es wird also mit der anderen Seite der Filterregion weitergemacht.

Voreinstellung ist 'duplicate'.

Das Attribut ist animierbar.

20.18.7 Attribut *kernelUnitLength*

Das Attribut gibt die Auflösung an, wievielen Pixeln als eine Matrixkomponente entspricht. Der Wert ist entweder eine positive Zahl oder eine Liste von zwei positiven Zahlen. Die erste Zahl gibt die Anzahl der Pixel pro Komponente in x-Richtung an, die zweite in y-Richtung. Ist nur eine Zahl angegeben, so gilt diese für beide Richtungen.

Durch Angabe des Attributes wird der Filter skalierbar. Um dies zu erreichen, muss wenigstens *kernelUnitLength* oder *filterRes* angegeben werden. In typischen Darstellungsprogramm ist die Effektivtät am höchsten, wenn ein Pixel der Matrix einem der Filterregion entspricht.

Angabe eines nicht positiven Wertes ist ein Fehler, der zum Abbruch der Darstellung des Dokumentes führt.

Das Attribut ist animierbar.

20.18.8 Attribut *preserveAlpha*

Mit dem Attribut wird angegeben, ob sich die Faltung auf alle Kanäle auswirken soll oder nicht Möglich sind die Werte 'true' und 'false'. Bei 'false' wird auch der Alpha-Kanal mitgefaltet, bei 'true' nicht.

Voreinstellung ist der Wert 'false'.

Das Attribut ist animierbar.

20.18.9 Beispiele *feConvolveMatrix*

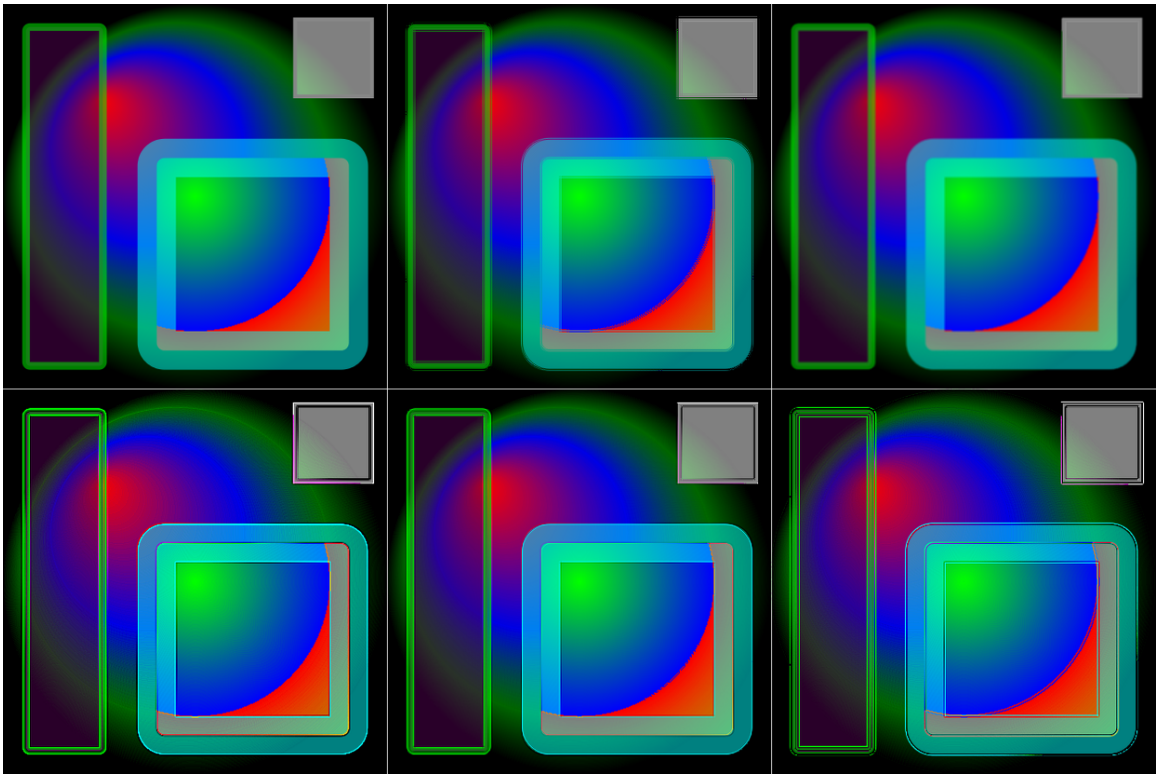


Abb. 344 Schärfen oder Weichzeichnen (1)

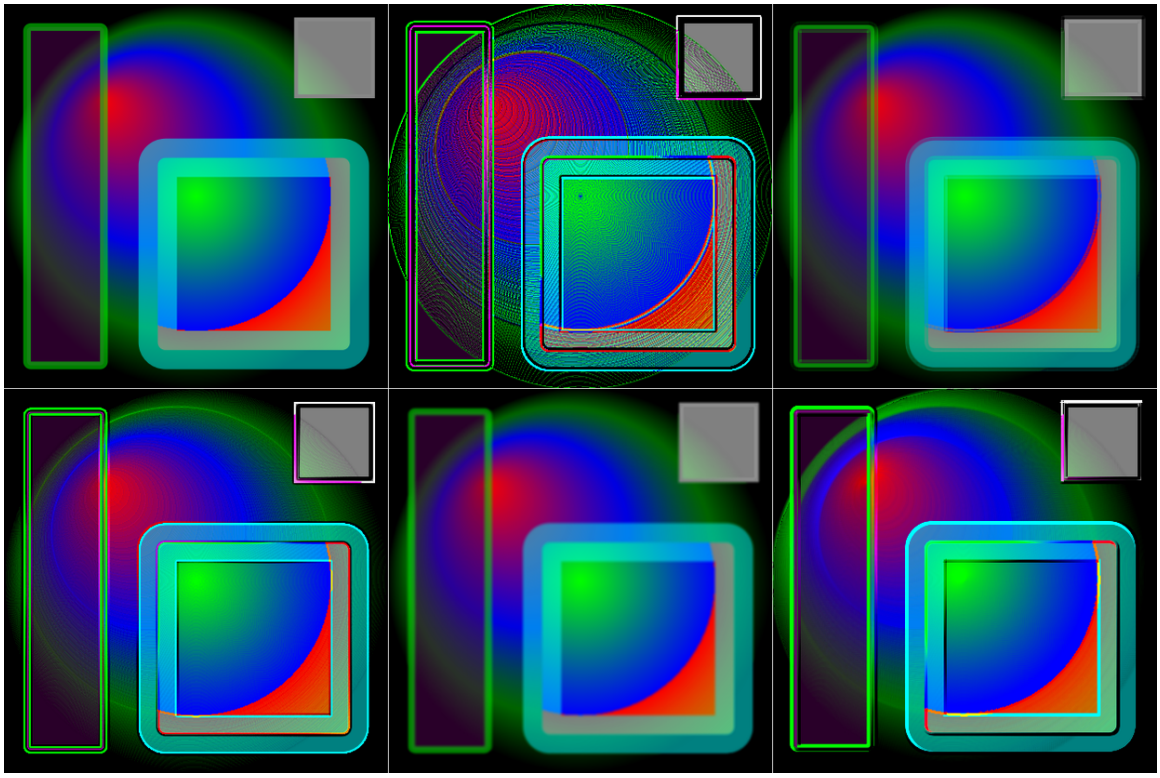


Abb. 345 Schärfen oder Weichzeichnen (2)

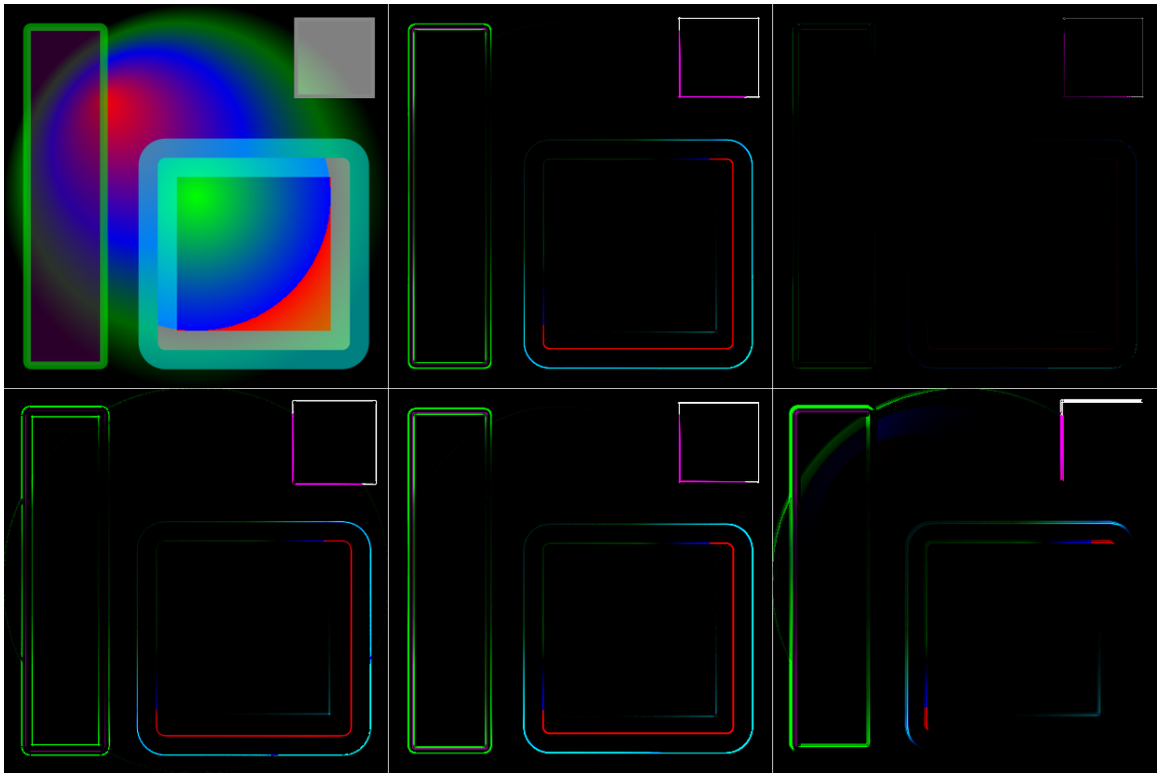


Abb. 346 Kanten betonen

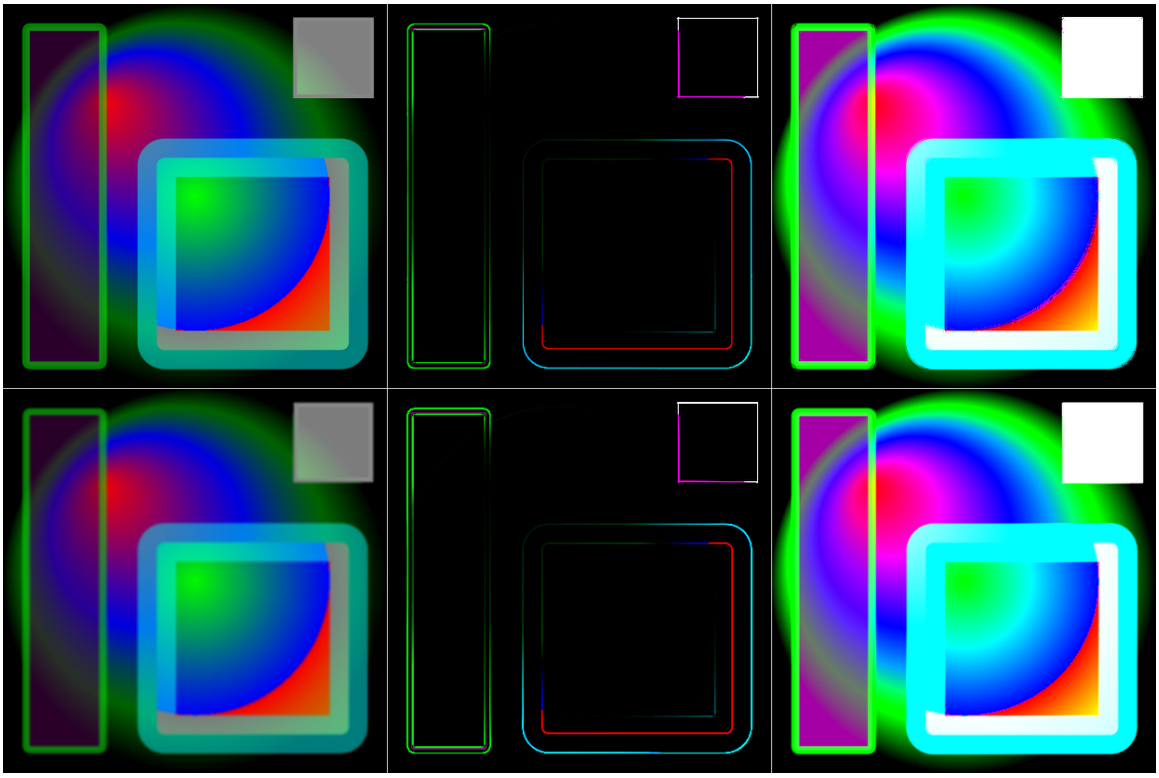


Abb. 347 Bias

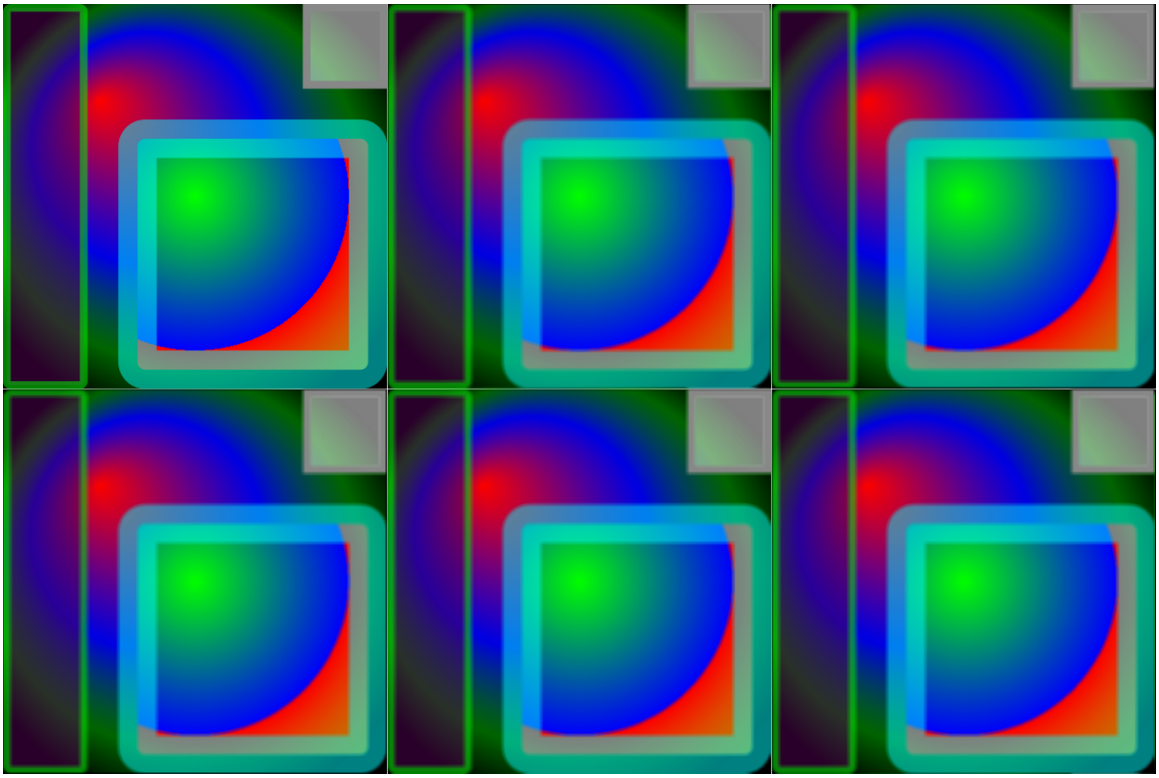


Abb. 348 *edgeMode*

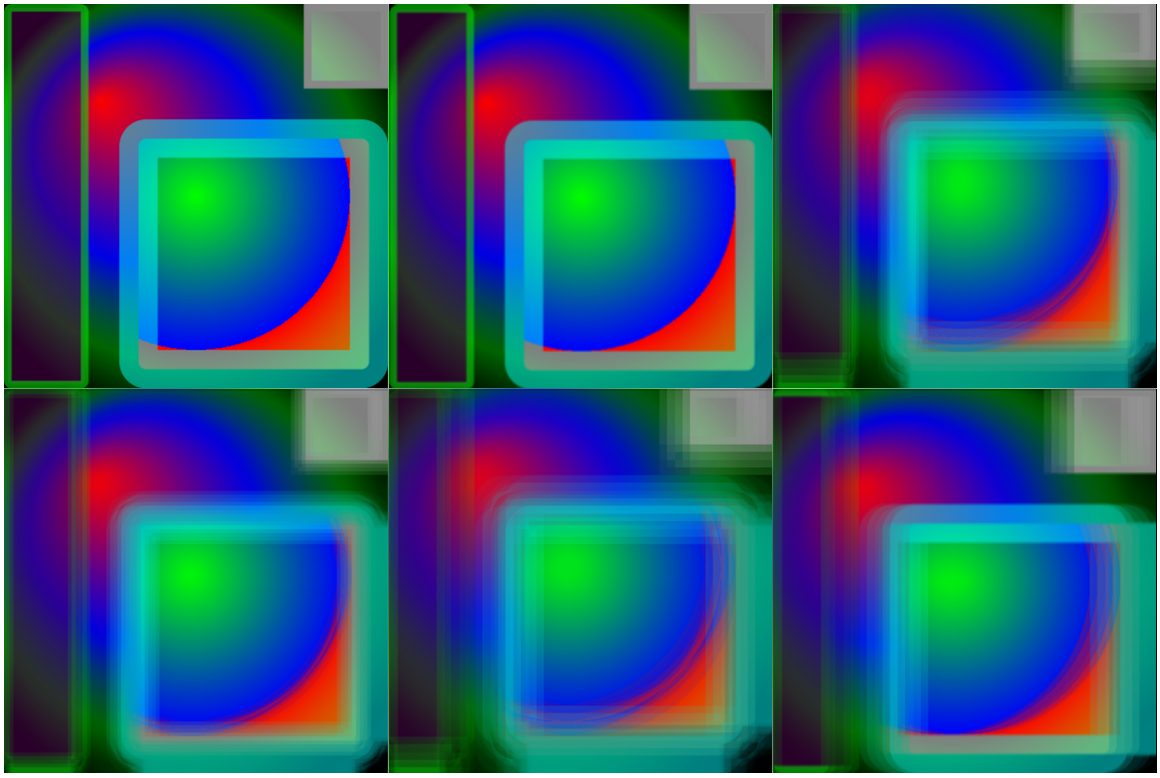


Abb. 349 *kernelUnitLength*

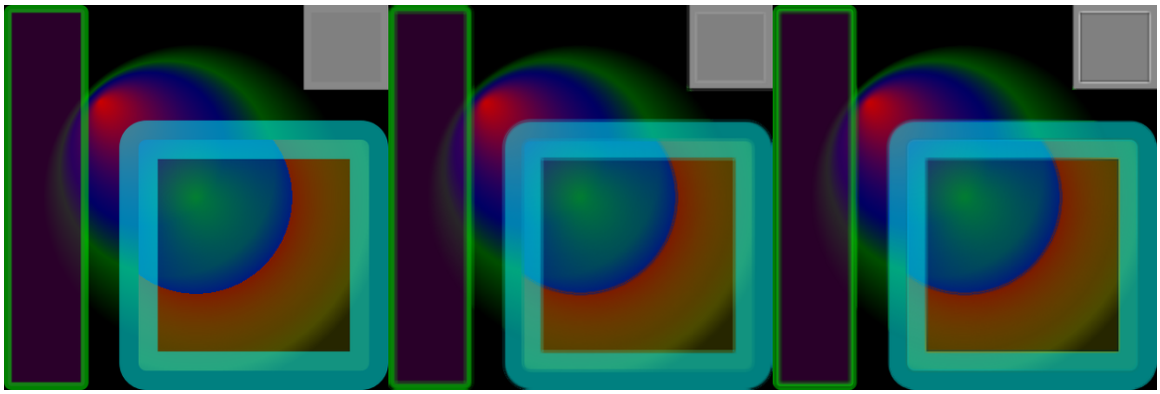


Abb. 350 *preserveAlpha*

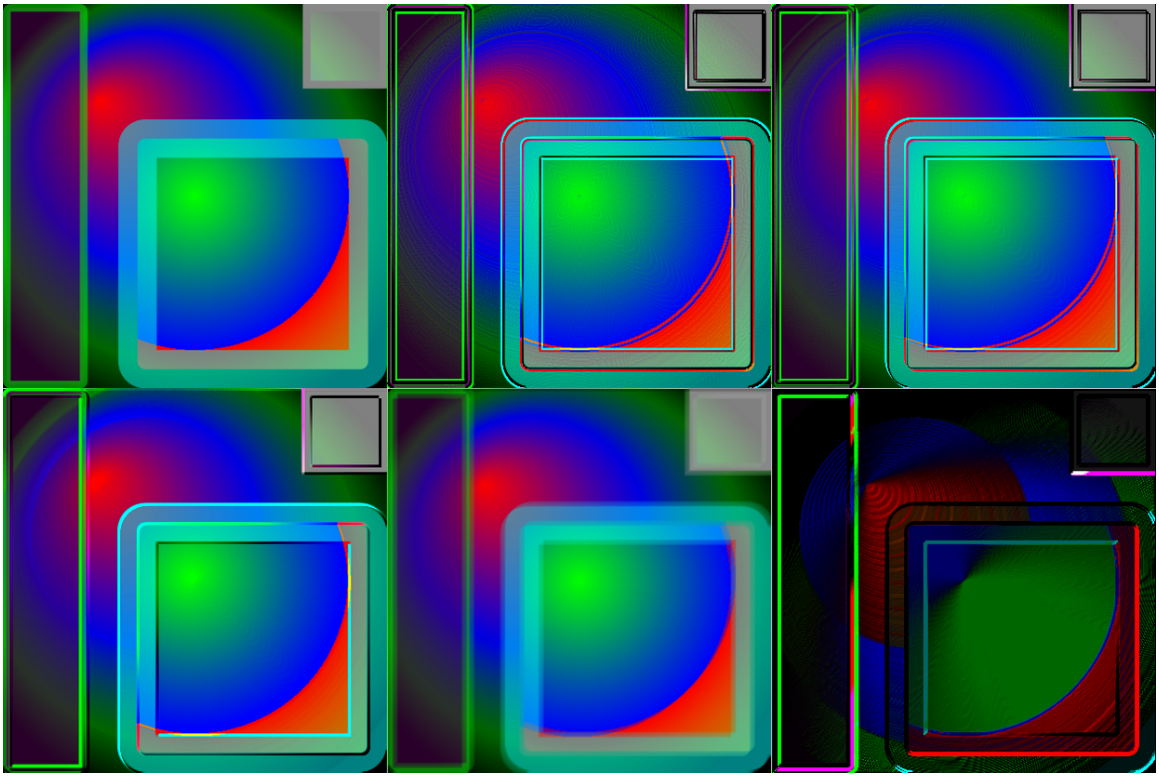


Abb. 351 mehr ...

Faltung des Eingangsbildes mit einer 8*8-Matrix:

```
<feConvolveMatrix in="SourceGraphic"
x="0" y="0" width="500" height="500"
order="8 8"
kernelMatrix="
10 0 0 1 0 0 0 24
0 0 1 2 0 0 0 0
0 1 2 3 1 0 0 0
1 2 3 5 2 1 0 10
0 1 2 3 1 0 0 0
0 0 1 2 0 0 0 2
0 0 0 1 0 0 0 4
25 0 1 2 0 0 0 5
"
divisor="110"
bias="0"
targetX="4"
targetY="4"
edgeMode="none"
kernelUnitLength="1 1"
preserveAlpha="true"
/>
```

Einige Beispiele zur Matrixfaltung. Links oben ist das ungefilterte Ausgangsbild zu sehen:

Schärfen oder Weichzeichnen (1)[41]

Matrizen der Ordnung 3 zum Schärfen oder Weichzeichnen. Schärfen ergibt sich vor allem durch das Abziehen benachbarter Pixel, wobei etwas in der Größe von einer Einheit übrigbleiben sollte. Mehr oder ein entsprechend kleiner Divisor hellen das Ergebnis auf, weniger oder ein entsprechend großer Divisor dunkeln ab. Ist ferner kein Divisor angegeben, so kehrt sich das Vorzeichen der Matrix um, wenn die Summe negativ wird.

Weichzeichnen ergibt sich durch das Mitteln über benachbarte Pixel. Anders als beim Gaußschen Weichzeichner tragen bei der Matrixfaltung aber nur die nächsten Nachbarn gemäß der Ordnung der Matrix bei, wobei der Autor die Gewichtung selbst bestimmen kann.

Schärfen oder Weichzeichnen (2)[42]

Matrizen der Ordnung 5 zum Schärfen oder Weichzeichnen.

Kanten betonen[43]

Matrizen der Ordnung 5 zum Betonen von Kanten. Eine Betonung von Kanten, also wo eine starke lokale Änderung der Farbe auftritt, ergibt sich durch Matrizen, deren Summe der Einzelkomponenten 0 ist oder nahezu 0 ist. Im letzteren Falle empfiehlt sich die explizite Angabe des Divisors. Dieser kann auch dazu dienen, das Ergebnis aufzuhellen (kleiner Divisor) oder abzudunkeln (großer Divisor).

Bias[44]

Beispiele mit Bias.

edgeMode[45]

Einfluss von *edgeMode*. Damit dies besser sichtbar wird, ist die Auflösung für die Filter kräftig heruntergesetzt worden, das Ergebnis besteht also nur aus wenigen Pixeln.

Links unten ist none, in der Mitte oben wrap, unten duplicate und rechts mit noch größeren Pixeln oben none, unten wrap.

kernelUnitLength[46]

Einfluss der *kernelUnitLength*. Verschiedene Auflösungen sind angegeben. Links unten wird die Auflösung animiert. Als praktische Anwendung einer Animation der Auflösung kommt etwa das Ein- oder Ausblenden von Objekten durch Verpixelung in Frage.

41 http://de.wikibooks.org/wiki/media%3ASVGfeConvolveMatrix01.svg
42 http://de.wikibooks.org/wiki/media%3ASVGfeConvolveMatrix02.svg
43 http://de.wikibooks.org/wiki/media%3ASVGfeConvolveMatrix03.svg
44 http://de.wikibooks.org/wiki/media%3ASVGfeConvolveMatrix04.svg
45 http://de.wikibooks.org/wiki/media%3ASVGfeConvolveMatrix05.svg
46 http://de.wikibooks.org/wiki/media%3ASVGfeConvolveMatrix06.svg

preserveAlpha[47]

Einfluss von *preserveAlpha*, in der Mitte false, rechts true.

mehr ...[48]

Einige weitere interessante Matrizen.

20.19 Urfilter *feDisplacementMap*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Abb. 352 Radialer Farbverlauf als Verschiebung

47 http://de.wikibooks.org/wiki/media%3ASVGfeConvolveMatrix07.svg
48 http://de.wikibooks.org/wiki/media%3ASVGfeConvolveMatrix08.svg

Abb. 353 Linearer Farbverlauf als Verschiebung

Abb. 354 Verschiebung durch Muster, täuscht Ben-Day-Technik vor

Mit diesem Urfilter wird eine lokale Verschiebung einer Eingabe erreicht. Die Pixel der Eingabe gemäß dem Attribut *in2* werden verwendet, um die Pixel der Eingabe gemäß dem Attribut *in* zu verschieben.

Sei P(x,y) die Eingabe gemäß *in* und P'(x,y) das Ergebnis des Urfilters. Seien ferner XC(x,y) und YC(x,y) jeweils die Komponentenwerte, festgelegt mittels der Attribute *xChannelSelector* und *yChannelSelector*. Der Wert des Attributes *scale* wird entsprechend benannt.

Dann ergibt sich:

P'(x,y) = P(x + *scale* * (XC(x,y) - 0.5), y + *scale* * (YC(x,y) - 0.5))

Die Eigenschaft *color-interpolation-filters* ist nur anwendbar auf die Eingabe gemäß *in2*, nicht für die Eingabe gemäß *in*.

20.19.1 Attribut *in2*

Mit dem Attribut wird die zweite Eingabe festgelegt. Der Attributwert ist der gleiche wie für *in*.

Das Attribut ist animierbar.

20.19.2 Attribut *scale*

Mit dem Attribut wird ein Skalierungsfaktor für die Verschiebung festgelegt. Der Faktor ist gemeint in Einheiten, die festgelegt sind durch *primitiveUnits* des zugehörigen Elementes *filter*.

Der Wert ist eine Zahl. Wird 0 angegeben oder das Attribut nicht notiert, ergibt sich keine Verschiebung.

Das Attribut ist animierbar.

20.19.3 Attribute *xChannelSelector* und *yChannelSelector*

Mit den Attributen wird angegeben, welche Kanäle für den Filtereffekt verwendeet werden. *xChannelSelector* gibt den Kanal an, welcher für die x-Richtung verwendet wird, *yChannelSelector* den für die y-Richtung.

Der Wert ist jeweils einer der Buchstaben 'R', 'G', 'B', 'A' für den entsprechenden Kanal, in der Reihenfolge also rot, grün, blau, alpha.

Die Attribute sind animierbar.

20.19.4 Beispiele *feDisplacementMap*

Zwei Fragmente im gleichen SVG-Dokument werden als Eingaben verwendet. Von der zweiten Eingabe werden die Kanäle R und B für die Verschiebung verwendet, R für die x-Richtung, B für die y-Richtung.

```
<filter id="f1" filterUnits="userSpaceOnUse"
  x="0" y="0" width="1" height="1">
<feImage id="fi1" xlink:href="#i1"
  x="0.2" y="0.2" width="0.4" height="0.4" result="i1" />
<feImage id="fi2" xlink:href="#i2"
  x="0.2" y="0.2" width="0.4" height="0.4" result="i2" />
<feDisplacementMap in="i1" in2="i2"
  scale="1"
  xChannelSelector="R"
  yChannelSelector="B"
  x="0" y="0" width="1" height="1" />
</filter>
```

Radialer Farbverlauf als Verschiebung[49]

Linearer Farbverlauf als Verschiebung[50]

Verschiebung durch Muster, täuscht Ben-Day-Technik vor[51]

49 http://de.wikibooks.org/wiki/media%3ASVGfeDisplacementMap01.svg
50 http://de.wikibooks.org/wiki/media%3ASVGfeDisplacementMap02.svg
51 http://de.wikibooks.org/wiki/media%3ASVGfeDisplacementMap03.svg

20.20 Urfilter *feMorphology*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Urfilter wird ein Verdicken oder Verdünnen der Eingabe veranlaßt. Hat die zu filternde Eingabe keine Struktur, so hat der Urfilter keinen Effekt. Bei einem Muster (***feTile***) werden periodische Randbedingungen verwendet.

20.20.1 Attribut *operator*

Mit dem Attribut wird festgelegt, ob verdickt oder verdünnt werden soll. Entsprechend sind zwei Werte möglich, 'erode' für verdünnen und 'dilate' für verdicken.

Voreinstellung ist 'erode'.

Das Attribut ist animierbar.

20.20.2 Attribut *radius*

Mit dem Attribut wird angegeben, um wie viel verdickt oder verdünnt werden soll, also die Ausdehnung oder der Radius des Effektes pro Pixel.

Der Wert ist entweder eine nicht negative Zahl oder eine Liste von exakt zwei nicht negativen Zahlen. Die Angabe einer negativen Zahl ist ein Fehler, welcher zum Abbruch der Darstellung des Dokumentes führt. Wird 0 angegeben, so hat der Filter keinen Effekt, Ist nur ein Wert angegeben, gilt er für x- und y-Richtung. Bei zweien ist der erste für die x- und der zweite für die y-Richtung.

Die Werte sind gemeint in Einheiten, die festgelegt sind durch *primitiveUnits* des zugehörigen Elementes *filter*.

Ist das Attribut nicht angegeben, wird 0 angenomen.

Das Attribut ist animierbar.

20.20.3 Beispiele *feMorphology*

Abb. 355 Verdünnen oder Verdicken

Verdünnung um die Radien 5 und 7:

```
<feMorphology in="SourceGraphic"
  x="0" y="0" width="500" height="500"
  operator="erode"
  radius="5 7" />
```

Verdünnen oder Verdicken[52]

feMorphology kann per Animation anhand zweier Ellipsen und der Buchstabenfolge 'SVG' ausprobiert werden.

Ausgangszustand ist ein Filter, welcher das Ausgangsbild verdünnt.

Aktivierung des grünen Knopfes bewirkt ein Umschalten auf Verdickung.

Aktivierung des roten Knopfes bewirkt ein Umschalten zurück auf Verdickung.

52 http://de.wikibooks.org/wiki/media%3ASVGfeMorphology01.svg

Aktivierung des gelben Knopfes bewirkt eine Animation des Attributes *radius*.

20.21 Urfilter *feTurbulence*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Der Urfilter erzeugt ein Bild mit Perlin-Turbulenz. Er kann verwendet werden, um Wolken zu erzeugen oder Marmorierungen. Der Effekt erstreckt sich über die gesamte Unterregion des Urfilters.

Es ist möglich, Bandbreitenbegrenztes Rauschen nur einer Oktave zu erzeugen (eine Oktave ist eine Frequenzverdopplung).

Mit dem Attribut *seed* kann ein Startwert der Zufallsverteilung festgelegt werden, um immer dasselbe Ergebnis zu erreichen.

Die erzeugten Farbmuster und Teiltransparenzen ergeben sich in dem durch *color-interpolation-filters* festgelegten Farbraum.

Die SVG-Definition des Urfilters gibt einen Quelltext für ein C-Programm an, mit dem der Effekt berechnet werden kann.

20.21.1 Attribut *baseFrequency*

Das Attribut gibt die Grundfrequenz(en) des Rauschens an. Der Wert ist entweder eine Zahl oder eine Liste von zwei Zahlen. Bei einer Zahl gilt diese als Grundfrequenz für das Rauschen in x- und y-Richtung. Bei zwei Werten gilt die erste für die x-Richtung, die zweite für die y-Richtung.

Die Angabe einer negativen Zahl ist ein Fehler, welcher zum Abbruch der Darstellung des Dokumentes führt.

Ist das Attribut nicht angegeben, wird 0 angenommen.

Das Attribut ist animierbar.

20.21.2 Attribut *numOctaves*

Das Attribut gibt die Anzahl der verwendeten Oktaven der Rauschfunktion an.

Der Wert ist eine (positive) ganze Zahl. Ist das Attribut nicht angegeben, wird 1 angenommen. Daß die ganze Zahl positiv sein muss, ergibt sich eher aus dem C-Quelltext, nicht aus der Definition des Attributes selbst.

Das Attribut ist animierbar (aufgrund der Bedingung, dass es eine ganze Zahl sein soll, ergibt sich implizit, dass nur eine diskrete Animation möglich ist, um Überraschungen zu vermeiden, sollte der Autor dies allerdings explizit angeben).

20.21.3 Attribut *seed*

Das Attribut ist der Startwert für den Zufallsgenerator. Der Wert ist eine Zahl. Ist das Attribut nicht angegeben, so wird 0 angenommen. Es ist also nicht ohne weitere Maßnahmen möglich, in einer einfachen SVG-Datei ein zufälliges Resulat zu erzielen, welches also bei jedem Aufruf anders wäre. Dazu müsste *seed* per Skript mit einem Wert aus einem Zufallsgenerator gesetzt werden.

Das Attribut ist animierbar.

20.21.4 Attribut *stitchTiles*

Mit dem Attribut kann festgelegt werden, ob die Grundfrequenz exakt wie angegeben verwendet wird oder and Höhe oder Breite der Filterunterregion angepasst werden soll.

Mögliche Werte sind 'stitch' und 'noStitch'.

Bei 'stitch' wird die Turbulenzfunktion wie angegeben verwendet. Dabei gibt es allerdings an den Rändern der Filterunterregion keine weichen Übergänge, was aber auch nicht immer relevant sein muss.

Bei 'noStitch' werden die Grundfrequenzen so angepasst, dass in die Höhe und die Breite der Filterunteregion ein ganzzahliges Vielfaches des Perlin-Musters der ersten Oktave passt. Es wird jeweils das dem angegebenen Wert nächstgelegene Vielfache verwendet.

Voreinstellung ist 'noStitch'.

Das Attribut ist animierbar.

20.21.5 Attribut *type*

Das Attribut gibt an, welcher Typ von turbulentem Rauschen verwendet wird. Möglich sind die Werte 'fractalNoise' für fraktales Rauschen und 'turbulence' für turbulentes Rauschen.

Voreinstellung ist 'turbulence'.

Das Attribut ist animierbar.

20.21.6 Beispiele *feTurbulence*

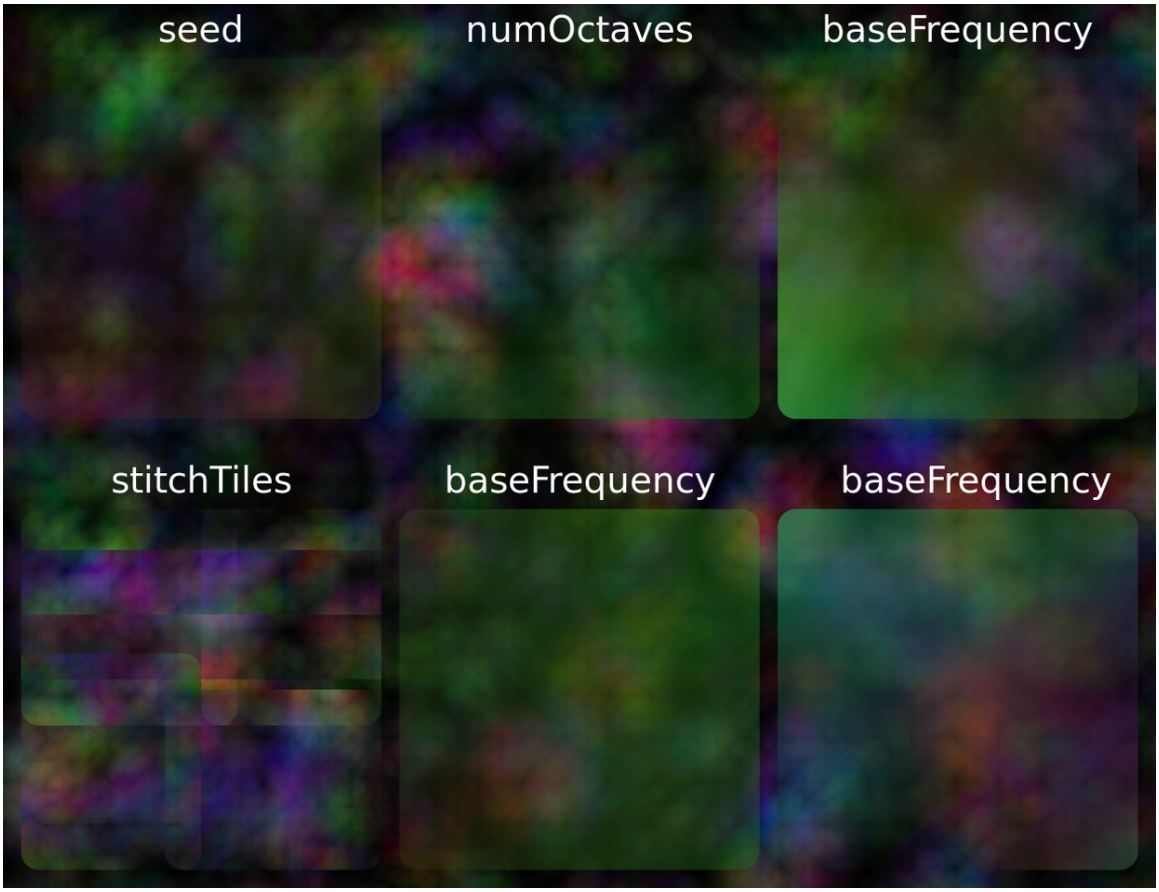


Abb. 356 Rauschfilter 'turbulence'

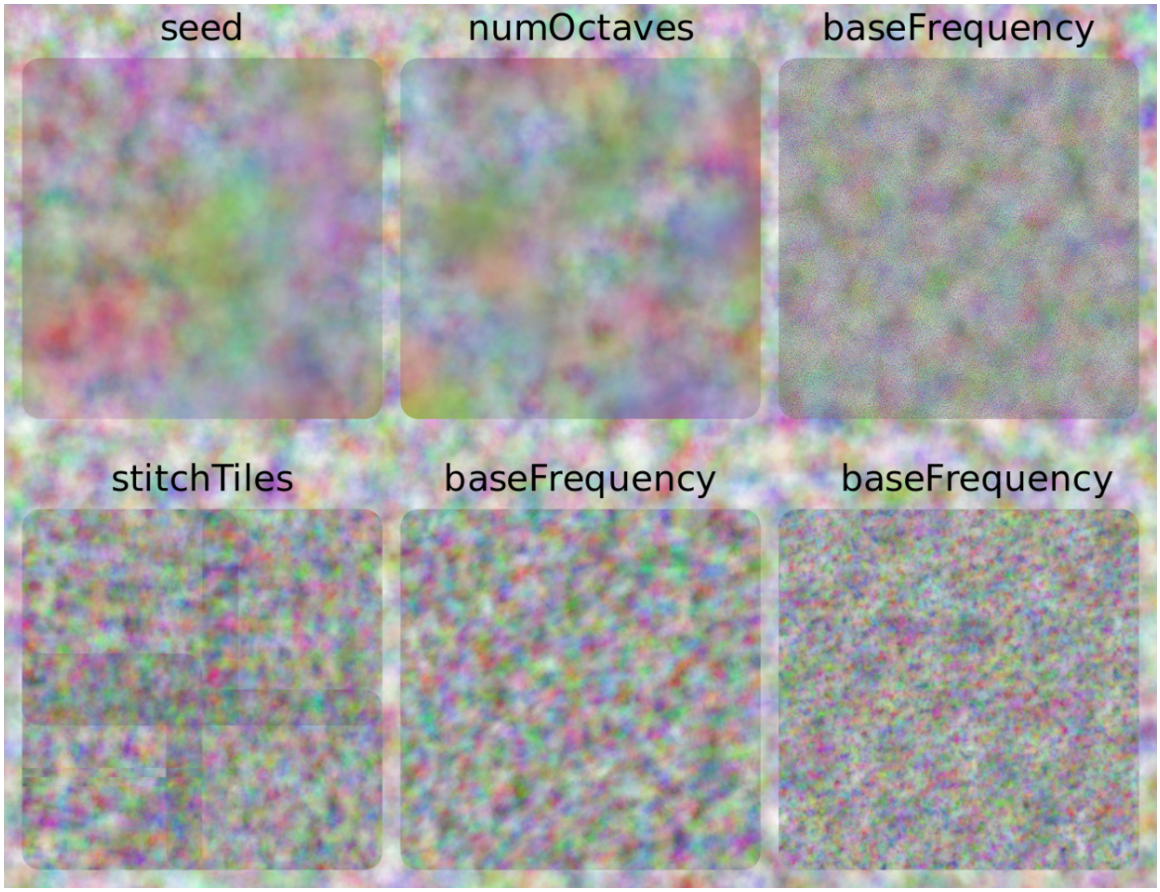


Abb. 357 Rauschfilter 'fractalNoise'

Abb. 358 Rauschfilter als Verschiebungskarte

Abb. 359 Rauschfilter als Texturvorlage

Der Filter sollte ausgiebig ausprobiert werden, hier werden einige Frequenzen per Animation durchprobiert:

```
<feTurbulence
x="0" y="0" width="500" height="500"
type="turbulence"
baseFrequency="0.001 0.002"
numOctaves="8"
seed="74.65"
stitchTiles="stitch">
<animate attributeName="baseFrequency"
        values="0.001 0.001;0.01 0.05;0.2 0.1; 0.03 0.04; 0.005 0.004"
        dur="90s" />
</feTurbulence>
```

Rauschfilter 'turbulence'[53]

Animation von Attributen von feTurbulence für turbulentes Rauschen. Jeweilige Fläche aktivieren oder anklickern, um die Animation auszulösen.

Rauschfilter 'fractalNoise'[54]

53 http://de.wikibooks.org/wiki/media%3ASVGfeTurbulence01.svg
54 http://de.wikibooks.org/wiki/media%3ASVGfeTurbulence02.svg

Animation von Attributen von feTurbulence für fraktales Rauschen. Jeweilige Fläche aktivieren oder anklickern, um die Animation auszulösen.

Rauschfilter als Verschiebungskarte[55]

Rauschfilter als Texturvorlage[56]

Ellipsen mit den Buchstaben 'SVG' drin werden mit Filtern etwas aufgepeppt.

20.22 Beleuchtung und Oberflächenstruktur

Mit einer ganzen Gruppe von Urfiltern kann eine Beleuchtung und eine reliefartige Oberflächenstruktur für die Filterregion vorgetäuscht werden. Zum einen gibt es drei Urfilter und eine Eigenschaft, welche Lichtquellen beschreiben. Dies sind die Elemente ***feDistantLight***, ***fePointLight***, ***feSpotLight*** und die Eigenschaft *lighting-color*. Und dann gibt es zwei Urfilter, welche die Art der Beleuchtung festlegen, dies sind ***feDiffuseLighting*** und ***feSpecularLighting***.

Eines der Elemente ***feDistantLight***, ***fePointLight*** oder ***feSpotLight*** ist jeweils Kindelement von ***feDiffuseLighting*** oder ***feSpecularLighting***. Für mehrere Lichtquellen sind dann jeweils mehrere Elemente ***feDiffuseLighting*** oder ***feSpecularLighting*** anzugeben und miteinander zu kombinieren.

Das verwendete Modell ist ein Modell nach Bui-Tuong Phong. Mit dem empirischen Modell können glatte Oberflächen wie Plastik ganz gut beschrieben werden. Es basiert allerdings nicht auf physikalisch haltbaren Grundlagen und reklektierende Aspekte der Darstellung werden nur phänomenologisch berücksichtigt, um mit einem einfachen Effekt eine Illusion einer beleuchteten Oberfläche hervorzurufen. Reale spiegelnde Oberfläche ergeben ein mehr oder weniger deutliches Abbild von gespiegelten Lichtquellen oder beleuchteten Objekten. Dies leistet das Modell nicht - woher sollte dem Darstellungsprogramm bekannt sein, was sich vor dem Darstellungsbereich befindet, was sich spiegeln könnte (sogenannte webcams sind jedenfalls nicht Bestandteil dieses Modells, um die Illusion zu verbessern).

Da SVG ja nicht dafür konzipiert ist, eine dreidimensionale Szenerie zu beschreiben, sondern zweidimensionale Graphiken, sind diese Urfilter nur als rudimentäre Erweiterung zu verstehen, um die Illusion von dreidimensionalen, reliefartigen Flächen aus den zweidimensionalen Graphiken zu erzeugen. Für diesen Zweck ist das Modell nach Phong recht gut einsetzbar.

Neben den in SVG bereits verwendeten Richtungen x und y kommt mit der Beleuchtung und Oberflächenstruktur noch eine weitere Richtung hinzu, dies ist die z-Richtung. Sie zeigt vom Darstellungsbereich zum Betrachter. Ein negativer z-Wert befindet sich also vom Betrachter aus gesehen hinter dem Darstellungsbereich, ein positiver davor.

55 `http://de.wikibooks.org/wiki/media%3ASVGfeTurbulence03.svg`
56 `http://de.wikibooks.org/wiki/media%3ASVGfeTurbulence04.svg`

20.23 Eigenschaft *lighting-color*, Beleuchtungsfarbe

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Die Eigenschaft legt die Farbe der Beleuchtung fest und ist anwendbar auf die Elemente ***feDiffuse-Lighting*** und ***feSpecularLighting***.

Mögliche Werte sind 'currentColor', 'inherit' oder ein Farbwert, wie bereits für die Eigenschaft *fill* erläutert. Der Initialwert ist weiß, also 'white' (#fff). Die Eigenschaft wird nicht vererbt und ist animierbar.

20.23.1 Beispiele *lighting-color*

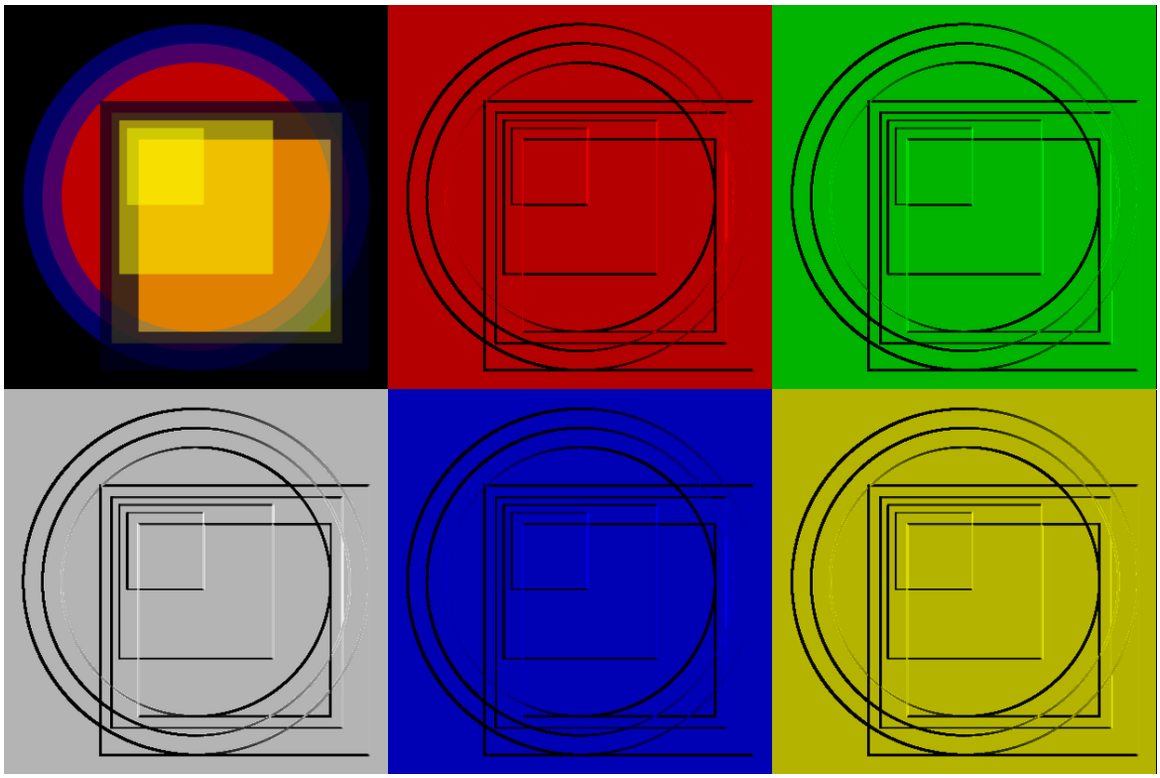


Abb. 360 Beleuchtungsfarbe (1)

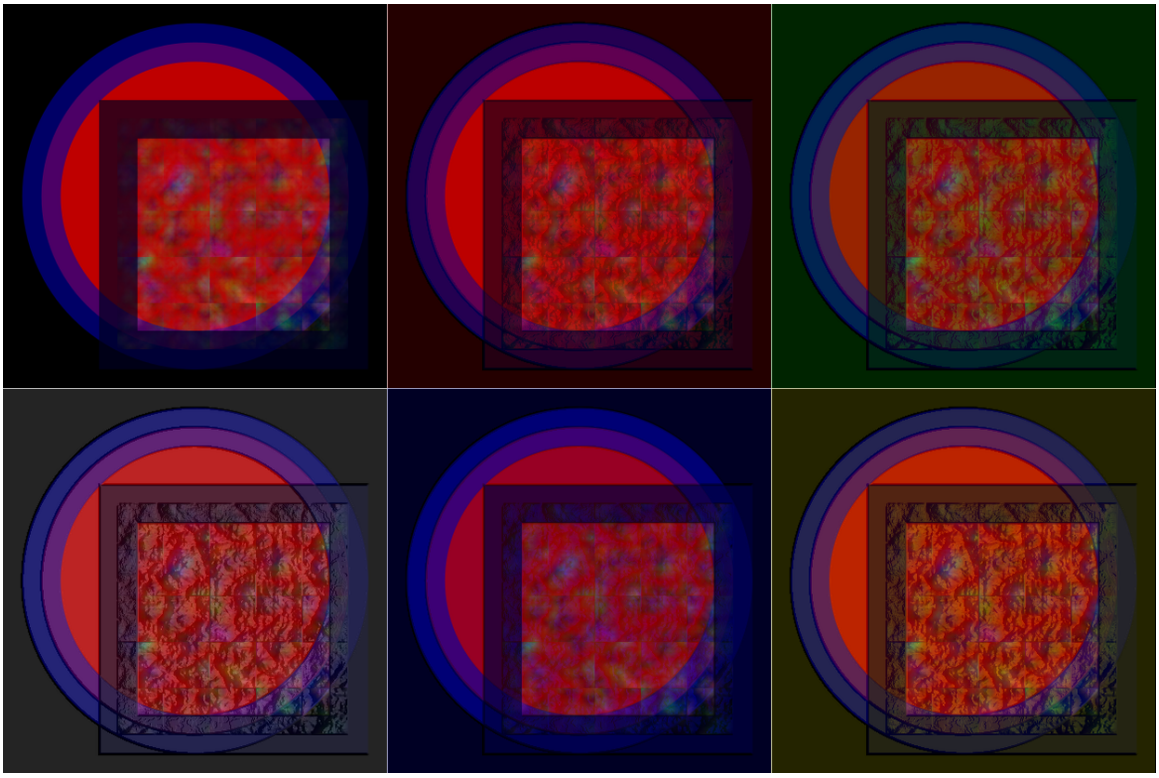


Abb. 361 Beleuchtungsfarbe (2)

Einige Beispiele zur Beleuchtungsfarbe. Links oben ist jeweils das ungefilterte Ausgangsbild zu sehen. Diffuses Licht kommt von rechts, pro Bild jeweils in anderer Farbe.

Beleuchtungsfarbe (1)[57]

Das beleuchtete Motiv besteht aus einem großen Kreis und übereinanderliegenden Rechtecken.

Beleuchtungsfarbe (2)[58]

Das beleuchtete Motiv besteht aus einem großen Kreis und zwei Rechtecken, eines davon mit einem turbulenten Muster. Per Komposition wird die diffuse Beleuchtung einer gleichmäßigen Beleuchtung überlagert.

57 http://de.wikibooks.org/wiki/media%3ASVGlighting-color01.svg
58 http://de.wikibooks.org/wiki/media%3ASVGlighting-color02.svg

20.24 Urfilter *feDistantLight*, entfernte Lichtquelle

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (teilweise falsche Filterregion)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Urfilter wird eine unendlich weit entfernte Lichtquelle angegeben.

20.24.1 Attribute *azimuth* und *elevation*

Um solch eine Lichtquelle zu charakterisieren, ist es nur notwendig, eine Richtung festzulegen, aus der das Licht kommt. Dazu dienen die Attribute *azimuth* und *elevation*.

Allerdings ist die Definition der Attribute in den aktuellen Versionen von SVG falsch beschrieben (dies ergäbe zwei Richtungen und ist so auch nirgends implementiert.

Der Wert ist jeweils eine Zahl, welche einen Winkel in Grad repräsentiert. Voreinstellung ist 0. Die Attribute sind animierbar.

azimuth ist ein Winkel von der x-Achse aus (Richtung y-Achse wie üblich in SVG) in der von der x- und y-Achse aufgepannten Ebene. Sei jene Richtung, welche sich ergibt, wenn die x-Achse um den angegebenen Winkel gedreht wird, die a-Achse. *elevation* ist ein Winkel von der a-Achse aus in Richtung der z-Achse in der von der a- und z-Achse aufgespannten Ebene. Allgemeiner ist *elevation* der Winkel zwischen der xy-Ebene und der Richtung zur Lichtquelle, auch Höhenwinkel genannt.

Die Richtung zur Lichtquelle ergibt sich, wenn die a-Achse um den angegebenen Winkel gedreht wird, die so entstehende Achse zeigt dann in Richtung der Lichtquelle.

Ist also insbesondere *elevation*='0', so liegt die Lichtquelle in der xy-Ebene. Ist *azimuth*='0', so liegt die Lichtquelle in Richtung der x-Achse. Das Licht kommt also im lokalen Koordinatensystem von rechts. Ist *azimuth*='90', so liegt die Lichtquelle in Richtung der y-Achse. Das Licht kommt also im lokalen Koordinatensystem von unten.

Ist *elevation*='90', liegt die Lichtquelle in Richtung der z-Achse, also senkrecht über der Ebene, jenseits des Betrachters, egal welchen Wert *azimuth* hat. Die Filterregion wird exakt von oben beleuchtet.

Sind zum Beispiel *azimuth* und *elevation* beide '45', so trifft das Licht also unter einem Winkel von 45 Grad von oberhalb der Ebene auf die Filterregion. Die Projektion der Richtung zum Licht auf die xy-Ebene ergibt die Richtung der Winkelhalbierenden von x- und y-Richtung.

20.24.2 Beispiele *feDistantLight*

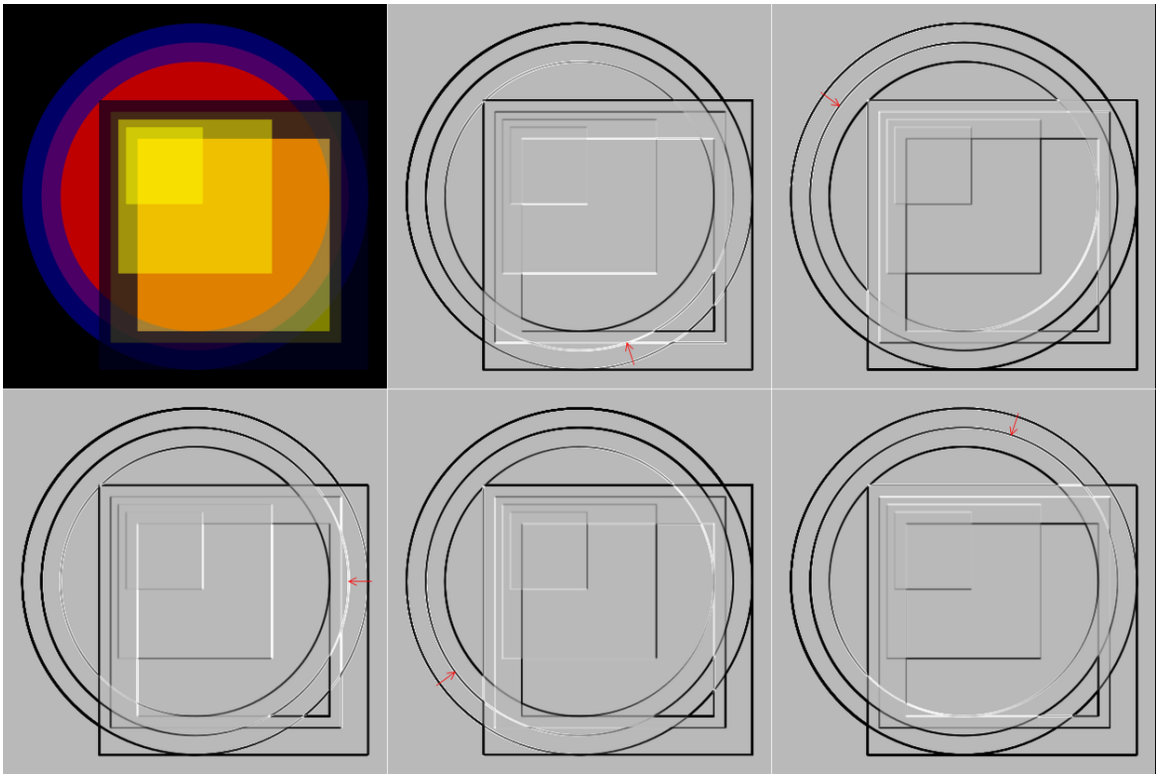


Abb. 362 Azimutaler Winkel

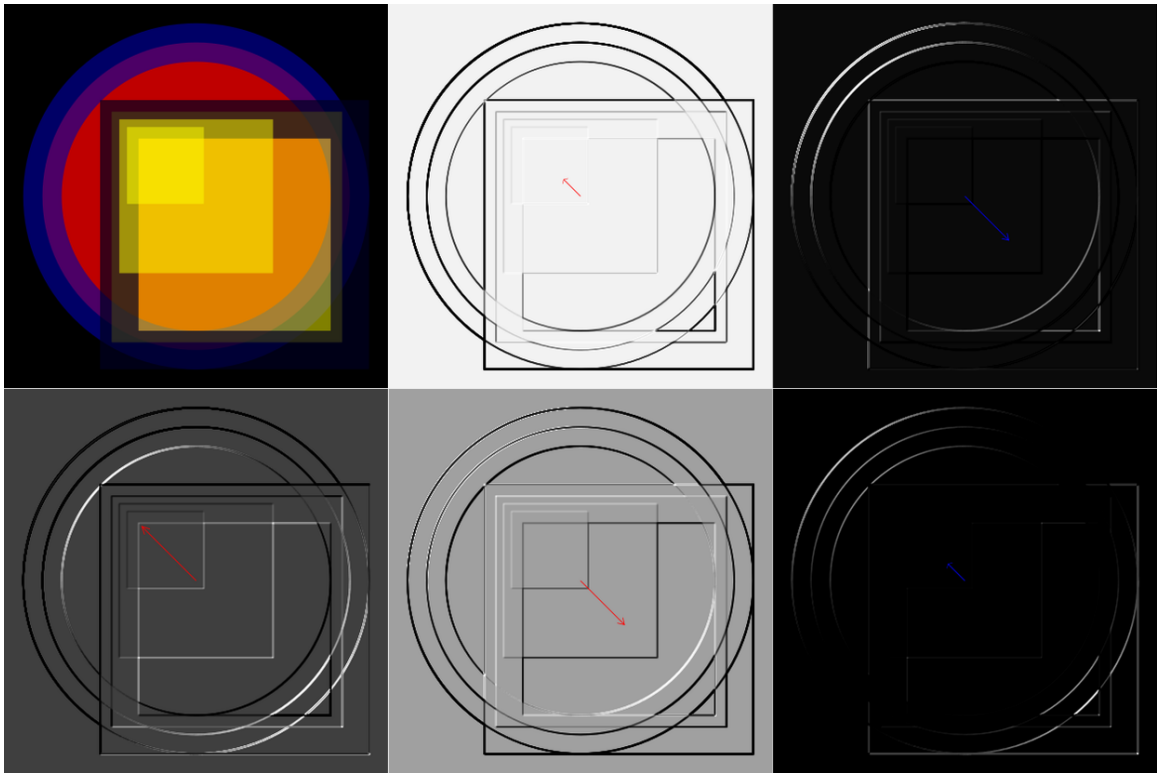


Abb. 363 Höhenwinkel

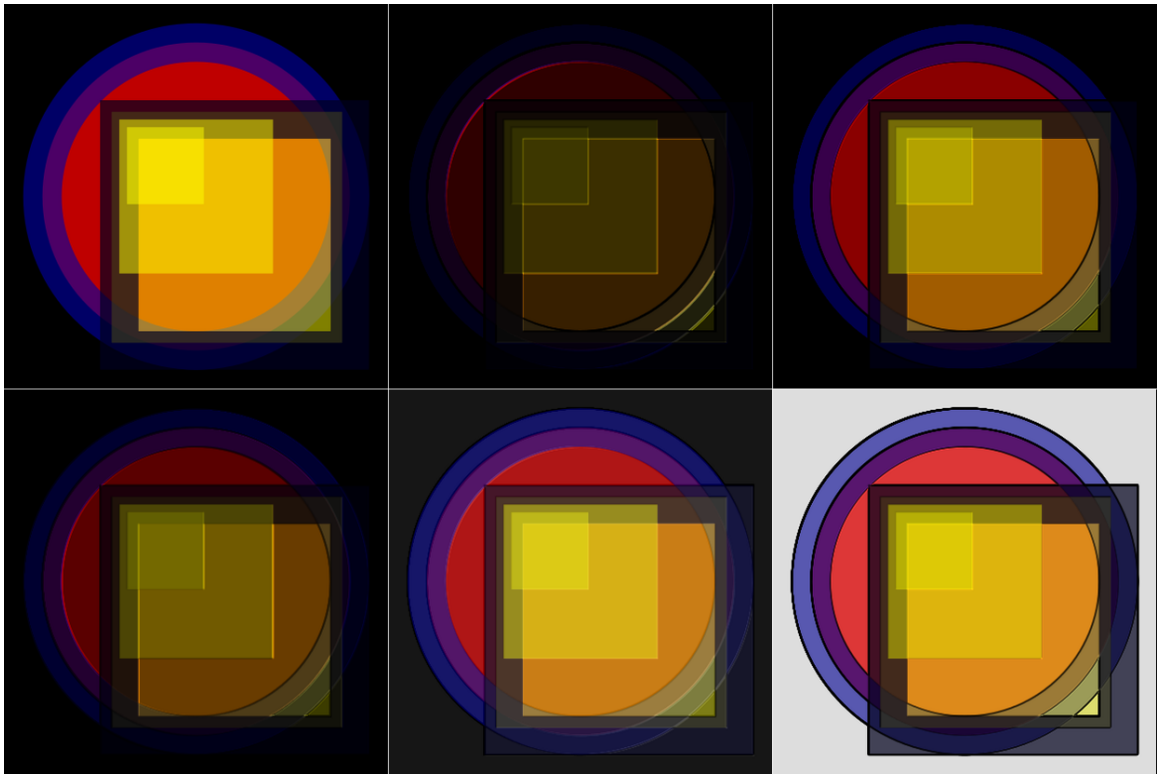


Abb. 364 Animation

Diffuse Beleuchtung von einer unendlich weit entfernten grünen Lichtquelle. Die Lichtquelle ändert ihre Richtung per Animation:

```
<filter id="DL"
    filterUnits="userSpaceOnUse"
    x="0" y="0" width="500" height="500">
  <feDiffuseLighting
      surfaceScale="10"
      diffuseConstant="1"
      lighting-color="#080">
    <feDistantLight azimuth="45" elevation="45">
      <animate attributeName="azimuth"
        values="0;360"
        dur="29.479s"
        repeatDur="indefinite" />
      <animate attributeName="elevation"
        values="0;90;0"
        dur="61.143s"
        repeatDur="indefinite" />
    </feDistantLight>
  </feDiffuseLighting>
</filter>
```

Einige Beispiele zur Beleuchtung mit unendlich weit entfernter Lichtquelle. Links oben ist das ungefilterte Ausgangsbild zu sehen. Das beleuchtete Motiv besteht aus einem großen Kreis und übereinanderliegenden Rechtecken.

Azimutaler Winkel[59]

Änderung des azimutalen Winkels (0, 72, 144, 216, 288 Grad) bei einer Lichtquelle 45 Grad über der Ebene. Die Richtung ist jeweils mit einem roten Pfeil angedeutet.

Höhenwinkel[60]

Änderung des Höhenwinkels, des Winkels zur Zeichenebene (0, 72, 144, 216, 288 Grad) bei einer Lichtquelle mit einem azimutalen Winkel von 45 Grad. Die Richtung ist jeweils mit einem roten oder blauen Pfeil angedeutet. Die Länge des Pfeiles ergibt sich aus der Projektion eines immer gleichlangen Pfeiles auf die Zeichenebene. Rote Pfeile kennzeichnen eine Lichtquelle in oder oberhalb der Zeichenebene, blau darunter.

Animation[61]

Die Position der Lichtquelle wird jeweils animiert. Ausgangsbild und Beleuchtung werden auch verschiedene Weise kombiniert.

59 http://de.wikibooks.org/wiki/media%3ASVGfeDistantLight01.svg
60 http://de.wikibooks.org/wiki/media%3ASVGfeDistantLight02.svg
61 http://de.wikibooks.org/wiki/media%3ASVGfeDistantLight03.svg

20.25 Urfilter *fePointLight*, Punktlichtquelle

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (teilweise falsch und falsche Filterregion)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Urfilter wird eine endlich weit entfernte Punktlichtquelle angegeben.

20.25.1 Attribute *x*, *y* und *z*

Mit den Attributen wird die Position der Lichtquelle angegeben. Der Wert ist jeweils eine Zahl, welche eine Koordinaten in den Einheiten darstellt, welche durch *primitiveUnits* des zugehörigen Elementes ***filter*** festgelegt sind. Also legt *x* die x-Koordinate fest, *y* die y-Koordinate und *z* die z-Koordinate.

Voreinstellung ist jeweils '0'. Die Attribute sind animierbar.

Zum Beispiel liegt *x*='0' *y*='0' *z*='1000' um 1000 Einheiten von der Zeichenebene aus in Richtung des Betrachters über dem lokalen Ursprung.

20.25.2 Beispiele *fePointLight*

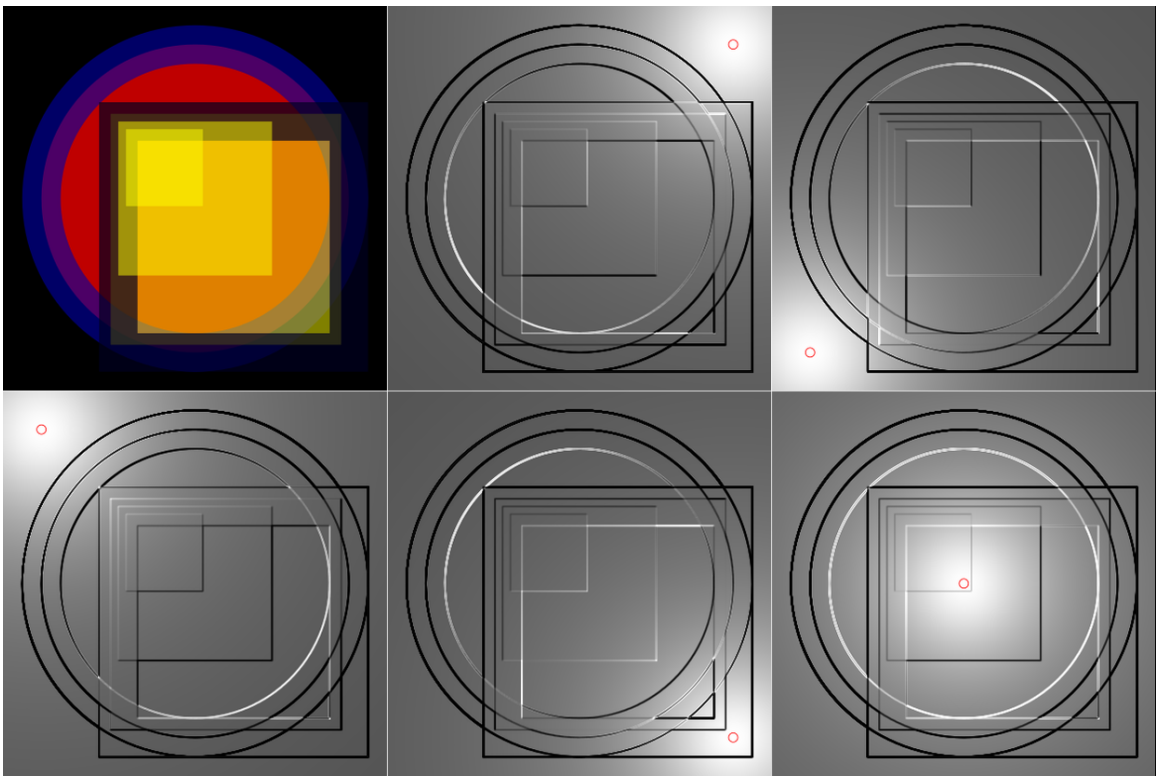


Abb. 365 x-y-Position

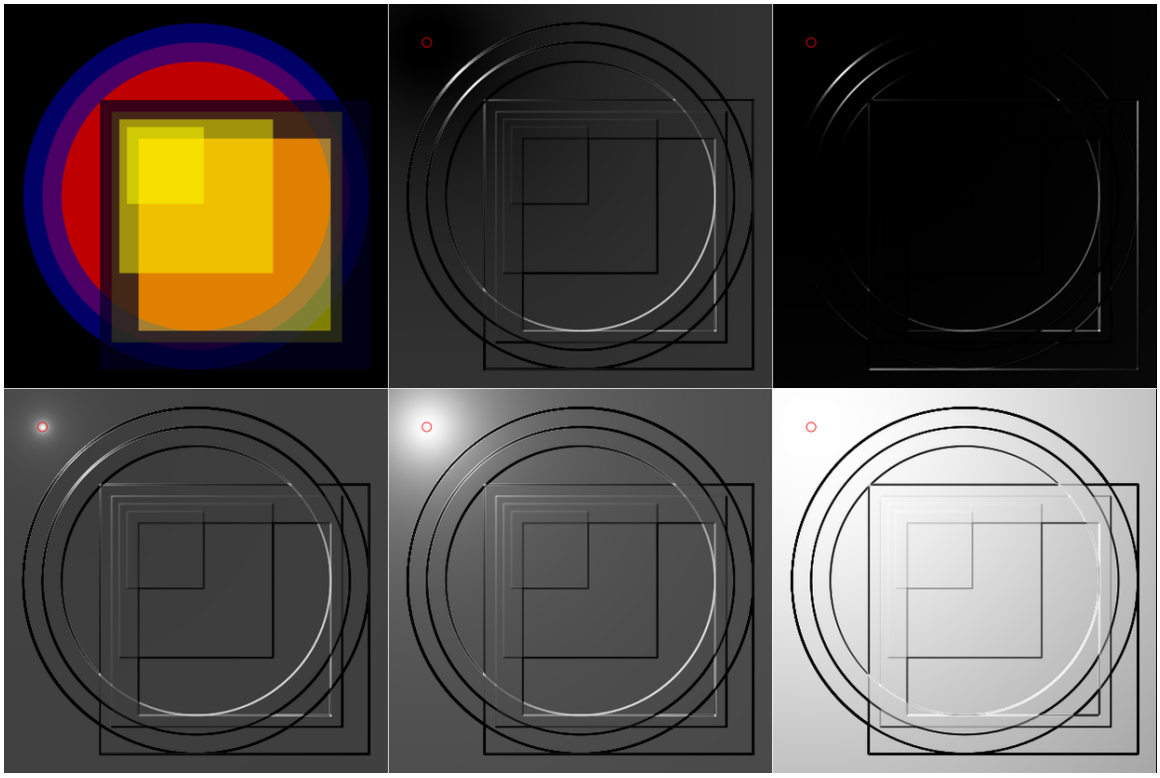


Abb. 366 z-Position

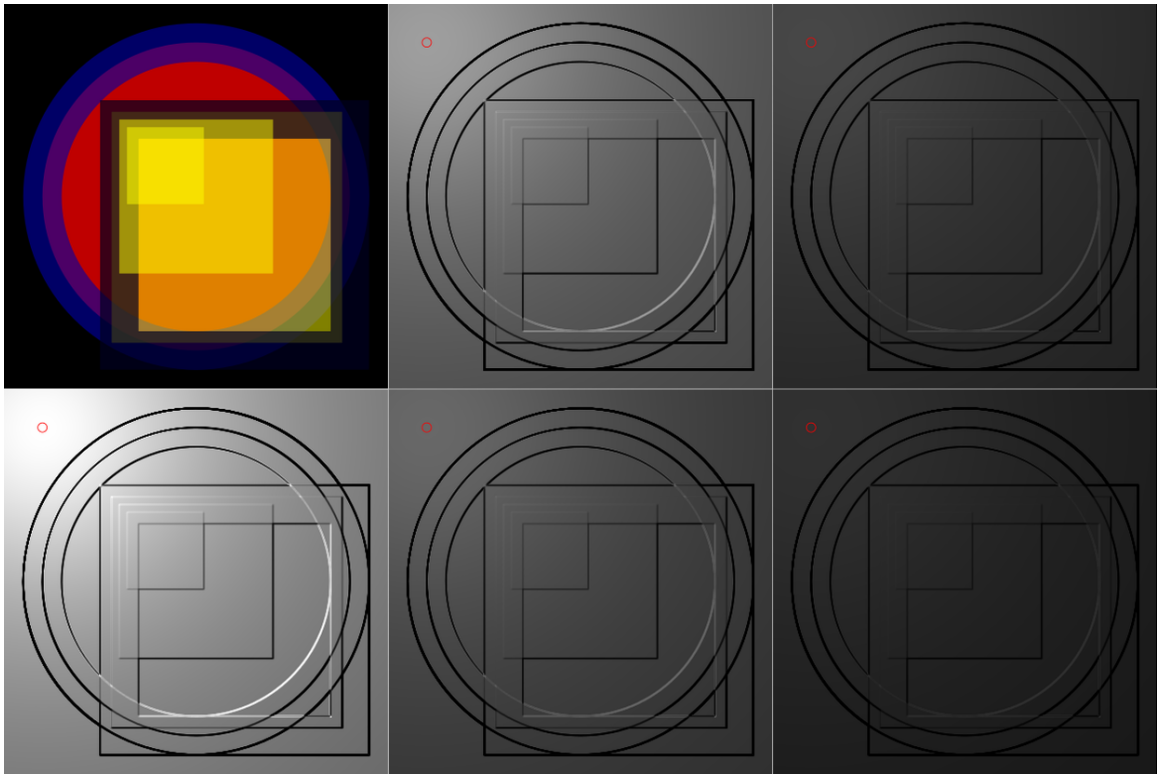


Abb. 367 z-Position und Intensität

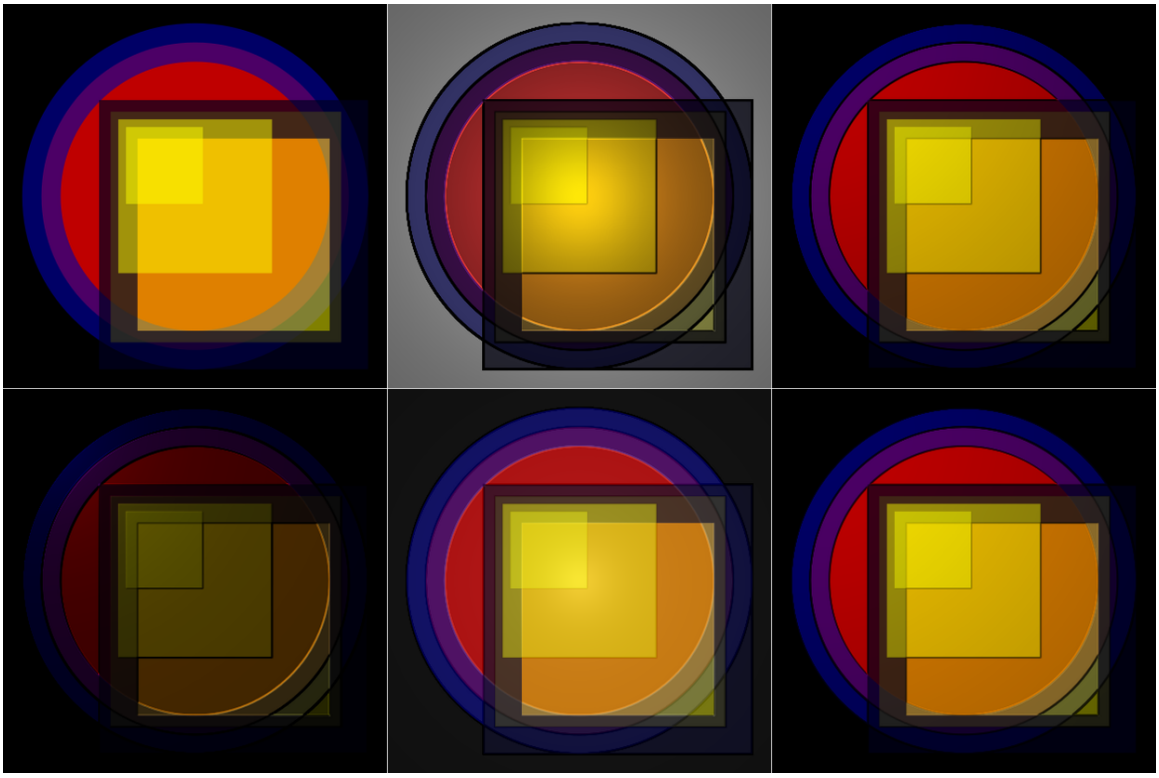


Abb. 368 Animation

Diffuse Beleuchtung von einer rosa Punktlichtquelle. Die Lichtquelle ändert ihre Position per Animation:

```
<filter id="DL" filterUnits="userSpaceOnUse"
    x="0" y="0" width="500" height="500">
  <feDiffuseLighting
      x="0" y="0" width="500" height="500"
      surfaceScale="10"
      diffuseConstant="1"
      kernelUnitLength="1"
      lighting-color="#f8f">
    <fePointLight x="500" y="250" z="250">
      <animate attributeName="x"
        values="0;100;500;100;0"
        dur="30s"
        repeatDur="indefinite" />
      <animate attributeName="y"
        values="0;500;400;-100;0"
        dur="31s"
        repeatDur="indefinite" />
      <animate attributeName="z"
        values="0;1000;500;-100;0"
        dur="37s"
        repeatDur="indefinite" />
    </fePointLight>
  </feDiffuseLighting>
</filter>
```

Einige Beispiele zur Beleuchtung mit punktförmiger Lichtquelle. Links oben ist das ungefilterte Ausgangsbild zu sehen. Das beleuchtete Motiv besteht aus einem großen Kreis und übereinanderliegenden Rechtecken.

x-y-Position[62]

Die Position oberhalb der Zeichenebene ist immer dieselbe, ansonsten ist die Position mit einem kleinen roten gestrichenen Kreis markiert.

z-Position[63]

Die Position z zur Zeichenebene ändert sich, oben -50 und -500 (also Lichtquelle unter der Zeichenebene), unten 5, 50 und 500 (also Lichtquelle oberhalb der Zeichenebene). Ansonsten ist die Position mit einem kleinen roten gestrichenen Kreis markiert.

Da ansonsten an Lichtquelle und Motiv nichts verändert wird, ist deutlich zu erkennen, dass die Darstellung nichts mit realen Lichtquellen zu tun hat, dort müsste eine Lichtquelle mit kleinem positiven z mehr Licht auf derselben Stelle der Oberfläche produzieren als bei größerem z. Um einen realistischeren Effekt zu erhalten, müsste die Intensität der Farbe der Lichtquelle entsprechend bei großen z quadratisch zum z verkleinert werden.

z-Position und Intensität[64]

Die Position z zur Zeichenebene ändert sich, z ist 200, 225, 250, 275 und 300. Zusätzlich wird die Farbe der Lichtquelle entsprechend so an den Abstand angepasst, als ob die Lichtquelle immer die gleiche Menge Licht abgeben täte. Ansonsten ist die Position mit einem kleinen roten gestrichenen Kreis markiert.

Animation[65]

Die Position der Lichtquelle wird jeweils animiert und mit der Ausgangsgraphik in einer Komposition kombiniert.

20.26 Urfilter *feSpotLight*, Scheinwerfer

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (falsch)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Urfilter wird ein Scheinwerfer als Lichtquelle angegeben.

62 http://de.wikibooks.org/wiki/media%3ASVGfePointLight01.svg
63 http://de.wikibooks.org/wiki/media%3ASVGfePointLight02.svg
64 http://de.wikibooks.org/wiki/media%3ASVGfePointLight03.svg
65 http://de.wikibooks.org/wiki/media%3ASVGfePointLight04.svg

20.26.1 Attribute *x*, *y* und *z*

Mit den Attributen wird die Position der Lichtquelle angegeben. Der Wert ist jeweils eine Zahl, welche eine Koordinaten in den Einheiten darstellt, welche durch *primitiveUnits* des zugehörigen Elementes ***filter*** festgelegt sind. Also legt *x* die x-Koordinate fest, *y* die y-Koordinate und *z* die z-Koordinate.

Voreinstellung ist jeweils '0'. Die Attribute sind animierbar.

Zum Beispiel liegt *x*='0' *y*='0' *z*='1000' um 1000 Einheiten von der Zeichenebene aus in Richtung des Betrachters über dem lokalen Ursprung.

20.26.2 Attribute *pointsAtX*, *pointsAtY* und *pointsAtZ*

Mit den Attributen wird angegeben, wohin der Scheinwerfer zeigt, also den Zielpunkt des Scheinwerfers.

Der Wert ist jeweils eine Zahl, welche eine Koordinaten in den Einheiten darstellt, welche durch *primitiveUnits* des zugehörigen Elementes ***filter*** festgelegt sind. Also legt *pointsAtX* die x-Koordinate fest, *pointsAtY* die y-Koordinate und *pointsAtZ* die z-Koordinate.

Voreinstellung ist jeweils '0'. Die Attribute sind animierbar.

Zum Beispiel liegt *pointsAtX*='0' *pointsAtY*='0' *pointsAtZ*='-100' um 100 Einheiten vom Betrachter aus gesehen hinter der Zeichenebene unter dem lokalen Ursprung.

20.26.3 Attribut *specularExponent*

Das Attribut bestimmt die Eigenschaften des Stahlprofiles oder der Winkelverteilung der Intensität des Scheinwerferlichtes. Der Wert ist eine Zahl. 0 bedeutet eine gleichmäßige Ausleuchtung.

Zahlen größer als 0 sorgen dafür, dass das Scheinwerferlicht in der mitte Intensiver ist als außen, je höher der Unterschied, desto dunkter wird es außen.

Zahlen kleiner als 0 sind eher nicht realistisch, um realen Scheinwerfer vorzutäuschen und führen im Außenbereich zu einer Aufhellung des Scheinwerferlichtes gegenüber der Mitte, sofern der jeweilige Kanal nicht schon den maximalen Wert erreicht hat. Zu große Farbwerte werden dann einfach auf den maximalen Wert eingeschränkt.

Voreinstellung ist 1.

Das Attribut ist animierbar.

20.26.4 Attribut *limitingConeAngle*

Das Scheinwerferlicht sorgt nur für eine Beleuchtung innerhalb eines Kegels um die Richtung herum, die von der Lichtquelle zum Zielpunkt des Scheinwerfers. Das Attribut gibt den Winkel zwischen

der Achse und dem Kegelmantel an. Darstellungsprogramme sollten auf den Rand allerdings einen nicht im Detail festgelegten Weichzeichner anwenden, um einen pixeligen Rand zu vermeiden.

Der Wert ist eine Zahl, die einen Winkel in Grad repräsentiert. Ist das Attribut nicht angegeben, wird keine Einschränkung durch einen Kegel angenommen. Der Bereich der Ausleuchtung bestimmt sich dann nur durch die anderen Attribute, zum Beispiel das Strahlprofil gemäß *specularExponent*.

Das Attribut ist animierbar.

20.26.5 Beispiele *feSpotLight*

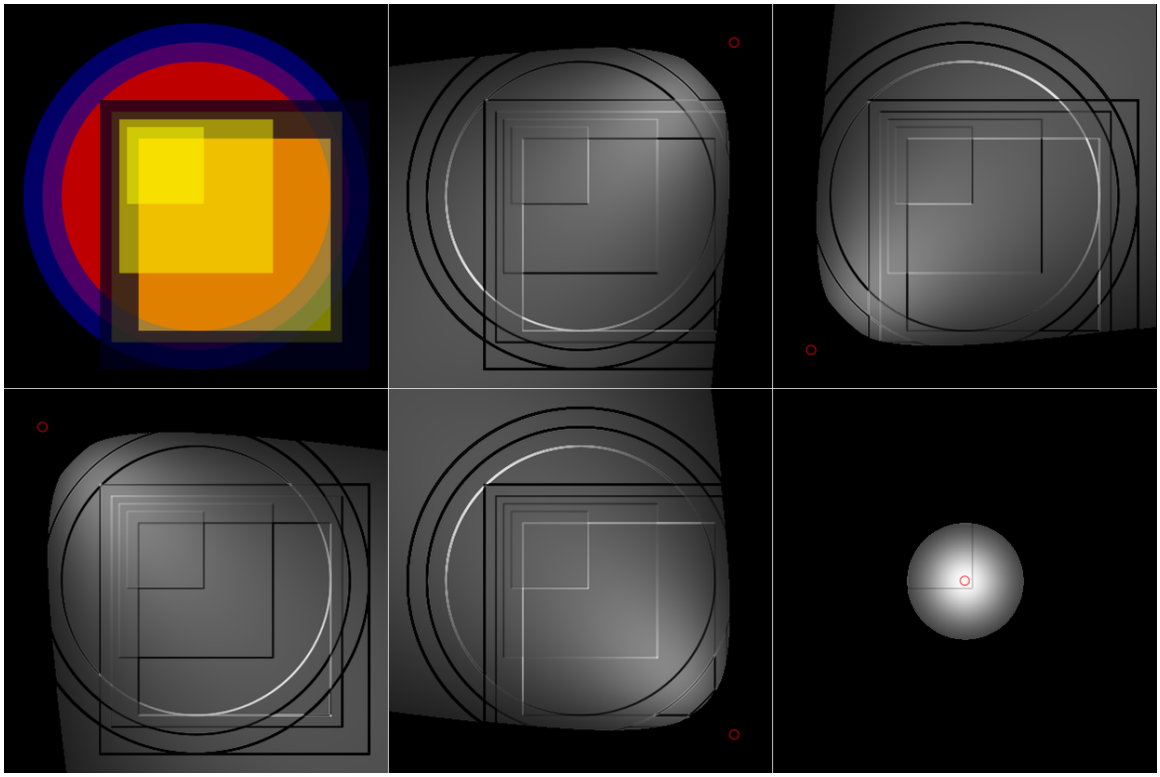


Abb. 369 x-y-Position

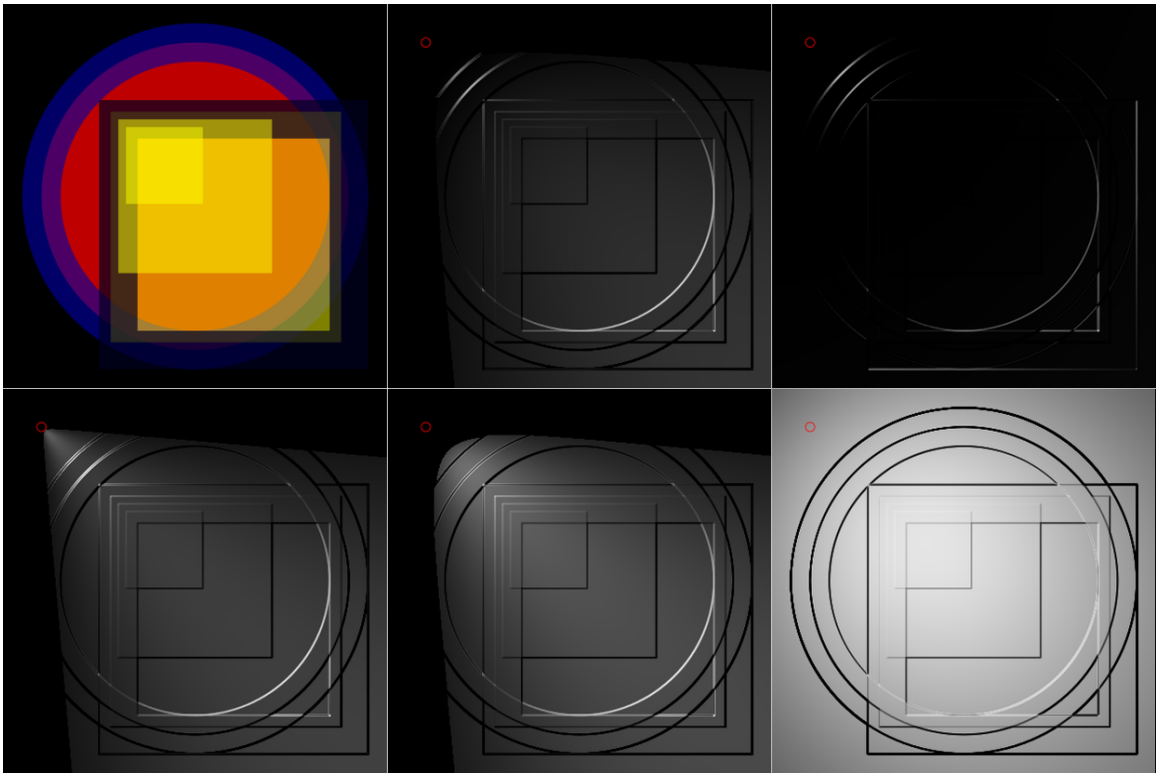


Abb. 370 z-Position

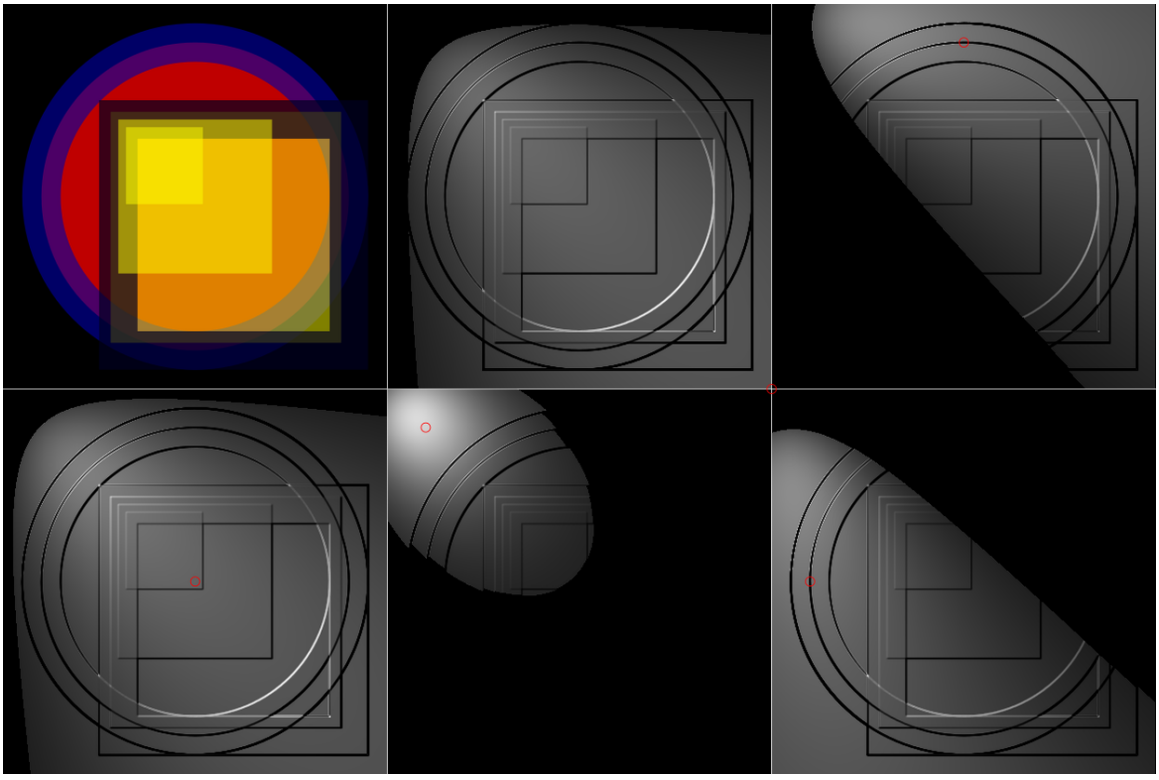


Abb. 371 Zielpunkt

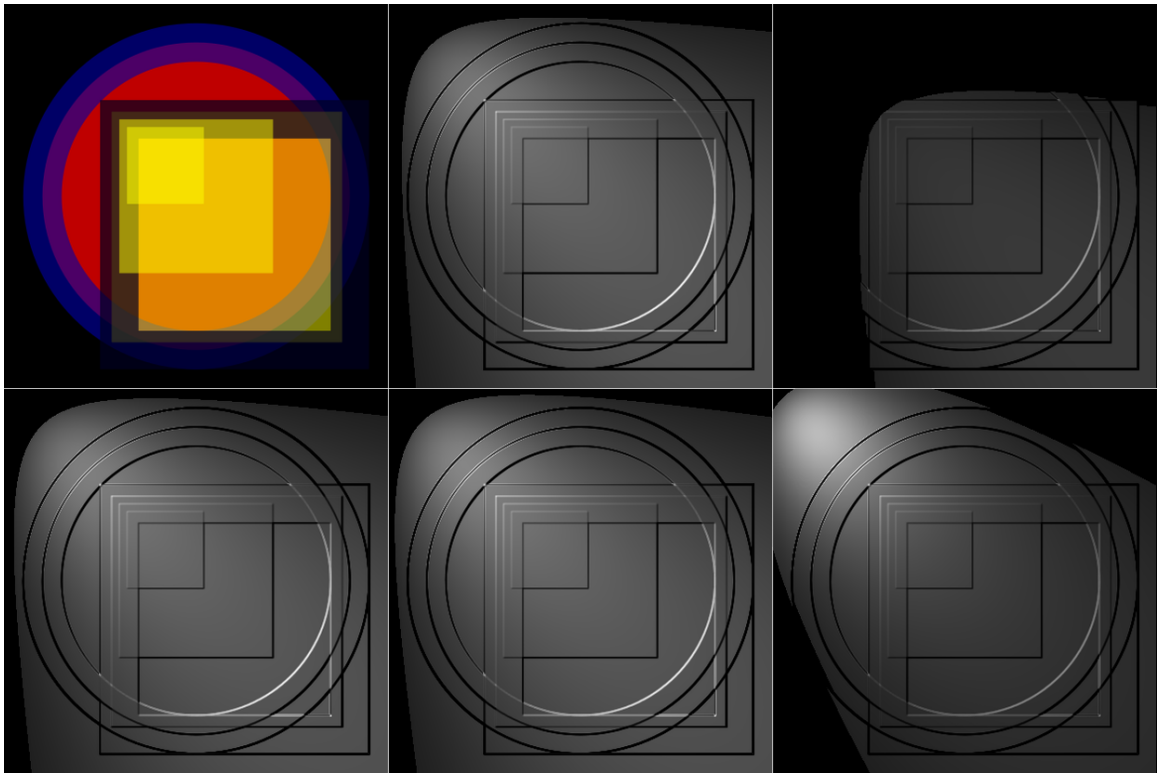


Abb. 372 z-Zielpunkt

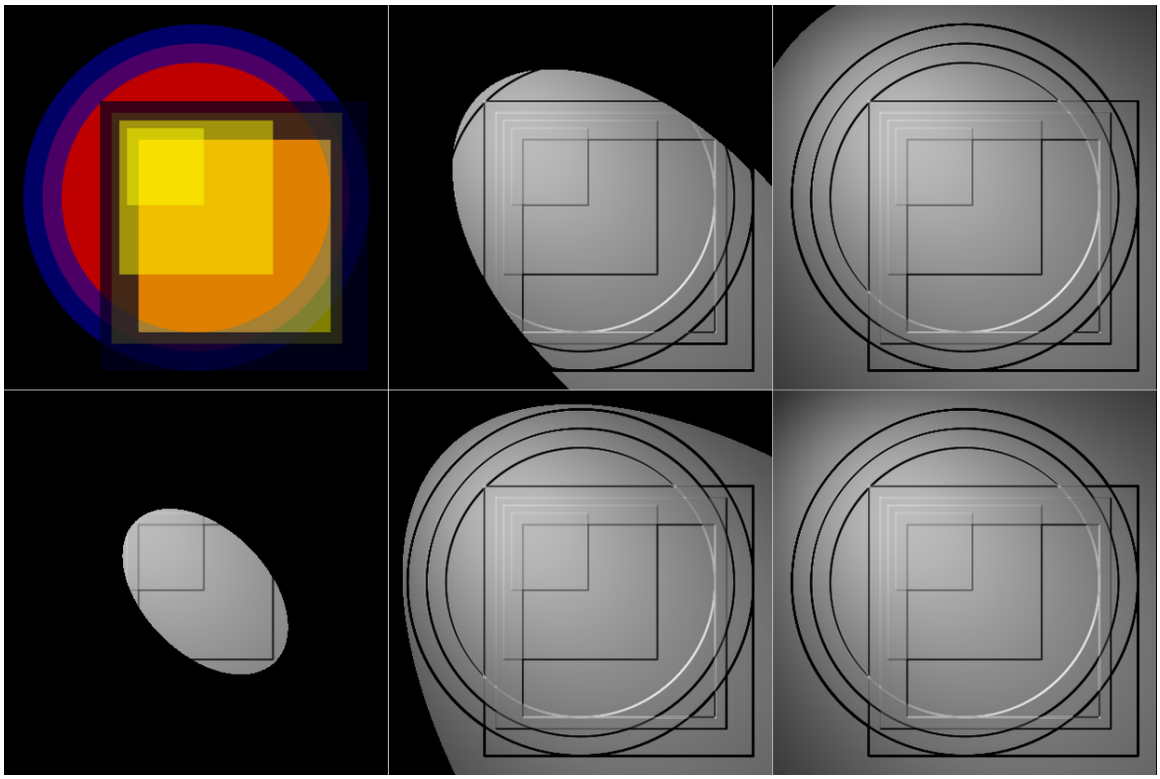


Abb. 373 Strahlkegel

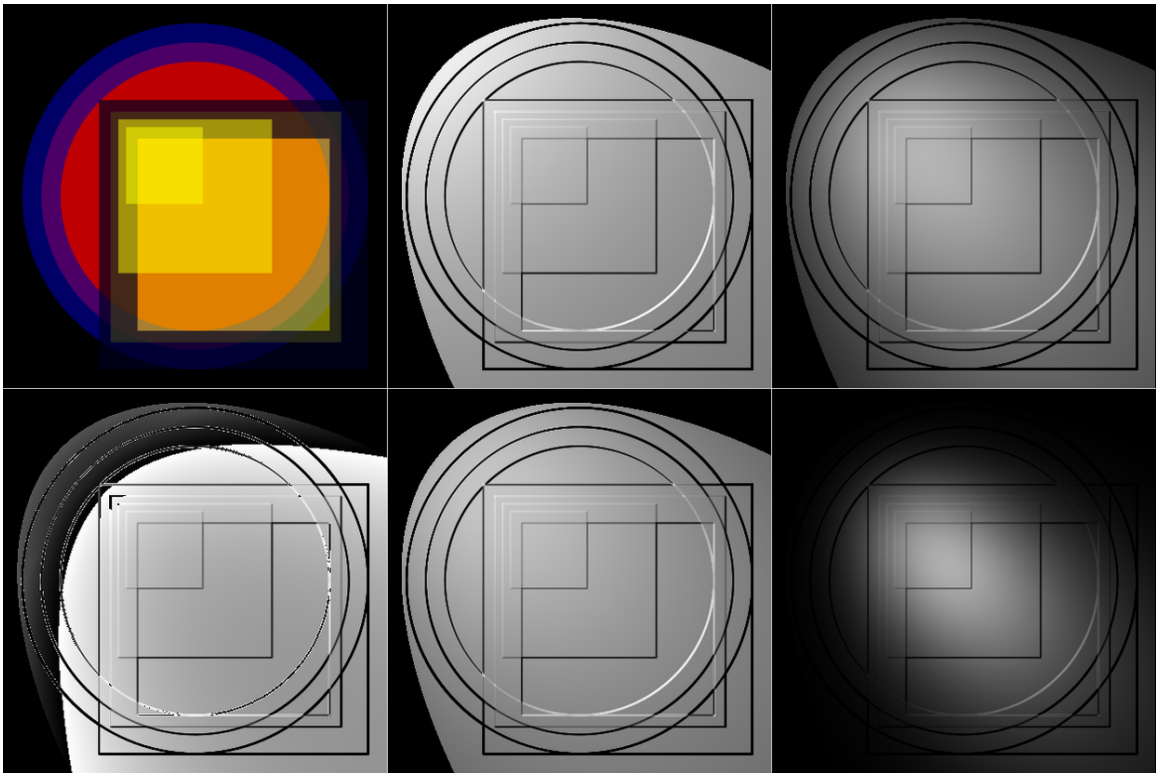


Abb. 374 Strahlprofil

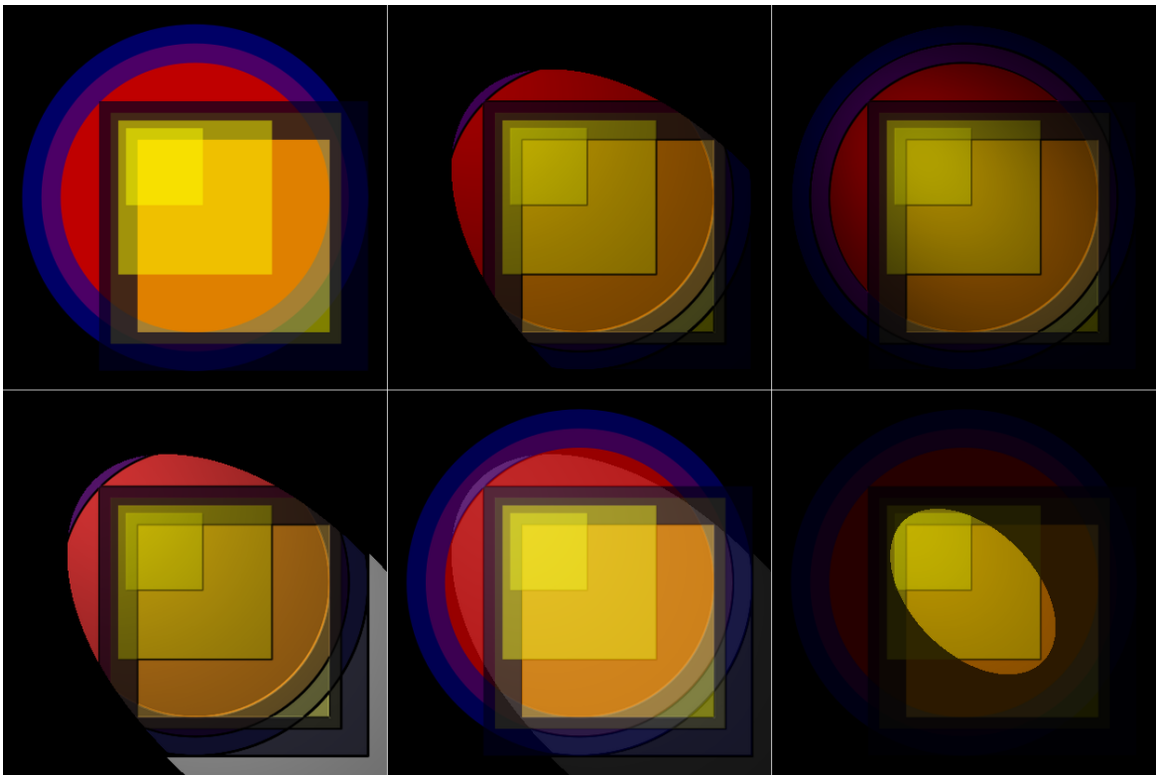


Abb. 375 Animation

Scheinwerfer mit animierten Strahlprofil:

```
<filter id="DL"
    filterUnits="userSpaceOnUse"
    x="0" y="0" width="500" height="500">
  <feDiffuseLighting
      surfaceScale="100"
      diffuseConstant="1"
      lighting-color="#444">
    <feSpotLight x="500" y="250" z="500"
        pointsAtX="400" pointsAtY="250" pointsAtZ="0"
        specularExponent="10" limitingConeAngle="30">
      <animate attributeName="specularExponent"
        values="0;10;100;1000;100;10;0;-1;-100"
        dur="60s" begin="5s" />
    </feSpotLight>
  </feDiffuseLighting>
</filter>
```

Einige Beispiele zur Beleuchtung mit Scheinwerfer. Links oben ist das ungefilterte Ausgangsbild zu sehen. Das beleuchtete Motiv besteht aus einem großen Kreis und übereinanderliegenden Rechtecken.

x-y-Position[66]

Die Position oberhalb der Zeichenebene und die Eigenschaften der Scheinwerfers sind immer dieselben, ansonsten ist die Position mit einem kleinen roten gestrichenen Kreis markiert. Der Scheinwerfer zeigt jeweils auf die Mitte des Motivs.

z-Position[67]

Die Eigenschaften der Scheinwerfers sind immer dieselben. Der Scheinwerfer zeigt jeweils auf die Mitte des Motivs. Die Position z zur Zeichenebene ändert sich, oben -50 und -500 (also Lichtquelle unter der Zeichenebene), unten 5, 50 und 500 (also Lichtquelle oberhalb der Zeichenebene). Ansonsten ist die Position mit einem kleinen roten gestrichenen Kreis markiert. Da ansonsten an Lichtquelle und Motiv nichts verändert wird, ist deutlich zu erkennen, dass die Darstellung nichts mit realen Lichtquellen zu tun hat, dort müsste eine Lichtquelle mit kleinem positiven z mehr Licht auf derselben Stelle der Oberfläche produzieren als bei größerem z. Um einen realistischeren Effekt zu erhalten, müsste die Intensität der Farbe der Lichtquelle entsprechend bei großen z quadratisch zum z verkleinert werden.

Zielpunkt[68]

Die Eigenschaften der Scheinwerfers sind immer dieselben. Die Position des Scheinwerfers ist die linke obere Ecke mit z=100. Auf welchen Punkt der Ebene der Scheinwerfer leuchtet, ist mit einem kleinen roten gestrichenen Kreis markiert.

z-Zielpunkt[69]

66 http://de.wikibooks.org/wiki/media%3ASVGfeSpotLight01.svg
67 http://de.wikibooks.org/wiki/media%3ASVGfeSpotLight02.svg
68 http://de.wikibooks.org/wiki/media%3ASVGfeSpotLight03.svg
69 http://de.wikibooks.org/wiki/media%3ASVGfeSpotLight04.svg

Die Eigenschaften der Scheinwerfers sind immer dieselben. Die Position des Scheinwerfers ist die linke obere Ecke mit z=100. Zwar zeigt der Scheinwerfer immer in Richtung der Mitte bezogen auf x und y, allerdings variiert z, oben 20 und 200, unten 0, -20, -200.

Strahlkegel[70]

Die Position des Scheinwerfers ist die linke obere Ecke mit z=300. Der Scheinwerfer zeigt immer in Richtung der Mitte. Der Scheinwerferkegelwinkel ändert sich, die Winkel sind 10, 20, 30, 40, 50 Grad.

Strahlprofil[71]

Der Scheinwerfer zeigt immer in Richtung der Mitte. Die Winkelverteilung des Scheinwerferlichtes ändert sich, das Attribut specularExponent hat entsprechend die Werte -2, 0, 1, 4, 16.

Animation[72]

Der Scheinwerfer wird jeweils animiert und per Animation nur das angezeigt, was im Scheinwerferlicht liegt.

20.27 Urfilter *feDiffuseLighting*, diffuse Beleuchtung

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (teilweise und falsche Filterregion)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Urfilter wird eine diffuse Beleuchtung gemäß Phong-Modell realisiert. Dabei ergibt sich das Relief der Oberfläche aus dem Alpha-Kanal.

Um mehrere Lichtquellen ***feDistantLight***, ***fePointLight*** und ***feSpotLight*** miteinander zu kombinieren, müssen mehrere Urfilter ***feDiffuseLighting*** mit jeweils einer Lichtquelle notiert werden und deren Ergebnisse können dann miteinander kombiniert werden.

Weil die Berechnung pixelweise stattfindet, ist das Ergebnis abhängig von der Auflösung. Es empfielt sich die explizite Festlegung von *filterRes* oder *kernelUnitLength*, wenn das Ergebnis unabhängig von den aktuellen Anzeigebedingungen sein soll.

In der Definition ist präzise in Formeln angegeben, wie sich der Filtereffekt berechnet.

70 http://de.wikibooks.org/wiki/media%3ASVGfeSpotLight05.svg
71 http://de.wikibooks.org/wiki/media%3ASVGfeSpotLight06.svg
72 http://de.wikibooks.org/wiki/media%3ASVGfeSpotLight07.svg

20.27.1 Attribut *kernelUnitLength*

Das Attribut gibt die Auflösung. Der Wert ist entweder eine positive Zahl oder eine Liste von zwei positiven Zahlen. Die erste Zahl gibt die Anzahl der Pixel pro Komponente in x-Richtung an, die zweite in y-Richtung. Ist nur eine Zahl angegeben, so gilt diese für beide Richtungen.

Durch Angabe des Attributes wird der Filter skalierbar. Um dies zu erreichen, muss wenigstens *kernelUnitLength* oder *filterRes* angegeben werden. In typischen Darstellungsprogramm ist die Effektivtät am höchsten, wenn ein Pixel der Matrix einem der Filterregion entspricht.

Angabe eines nicht positiven Wertes ist ein Fehler, der zum Abbruch der Darstellung des Dokumentes führt.

Das Attribut ist animierbar.

20.27.2 Attribut *surfaceScale*

Das Attribut gibt einen Skalierungsfaktor an, der bestimmt, welche z-Koordinate sich aus dem Alphakanal ergibt, also z = *surfaceScale* * Alphakanalwert.

Der Wert ist eine Zahl. Voreinstellung ist 1. Das Attribut ist animierbar.

20.27.3 Attribut *diffuseConstant*

Das Attribut gibt den Reflexionsfaktor für die diffuse Komponente der Reflexion nach dem Phong-Beleuchtungsmodell an. Der Wert ist eine nicht negative Zahl. Ein kleiner Wert ergibt also wenig diffus gestreutes Licht von der Oberfläche. Typisch sind Werte zwischen 0 und 1, da nicht weniger als nichts diffus gestreut werden kann und nicht mehr als alles. Andererseits ist das Modell aber ohnehin nicht besonders realistisch, so dass auch größere Werte nicht schaden. Gegebenenfalls wird einfach bei der Darstellung das Ergebnis der Farbkanäle auf den erlaubten Bereich eingeschränkt.

Voreinstellung ist 1. Das Attribut ist animierbar.

20.27.4 Beispiele *feDiffuseLighting*

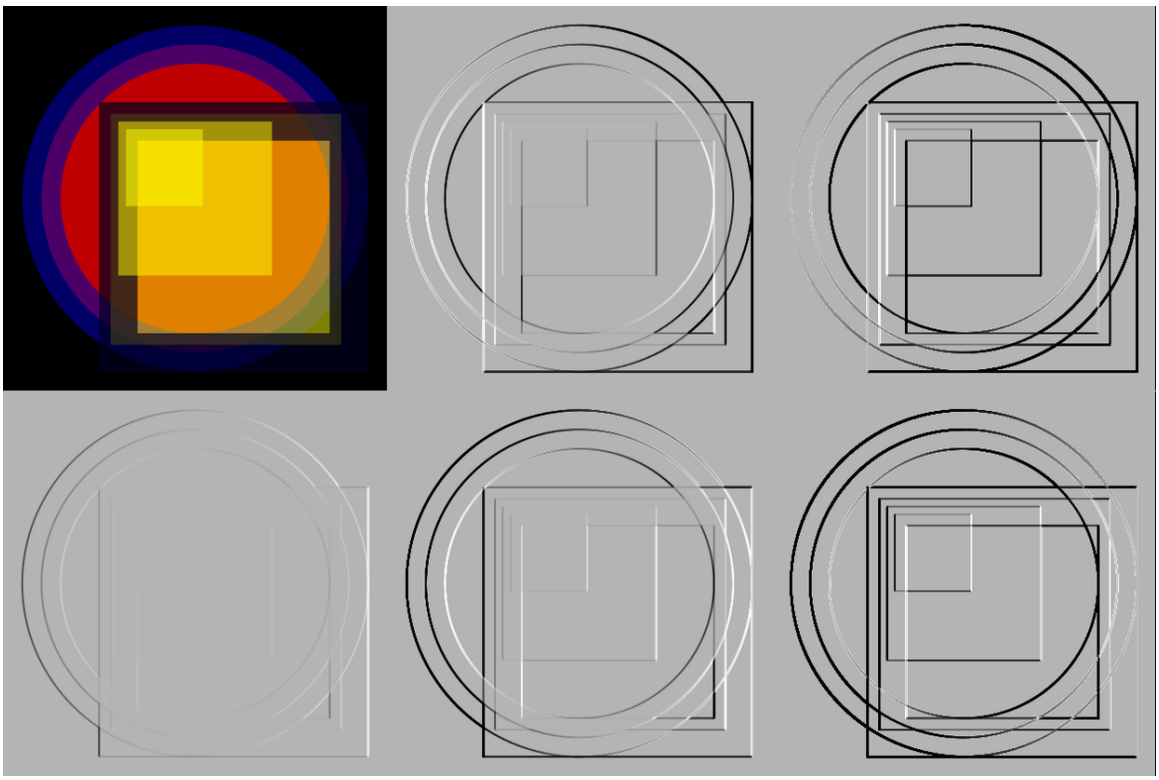


Abb. 376 Oberfläche (1)

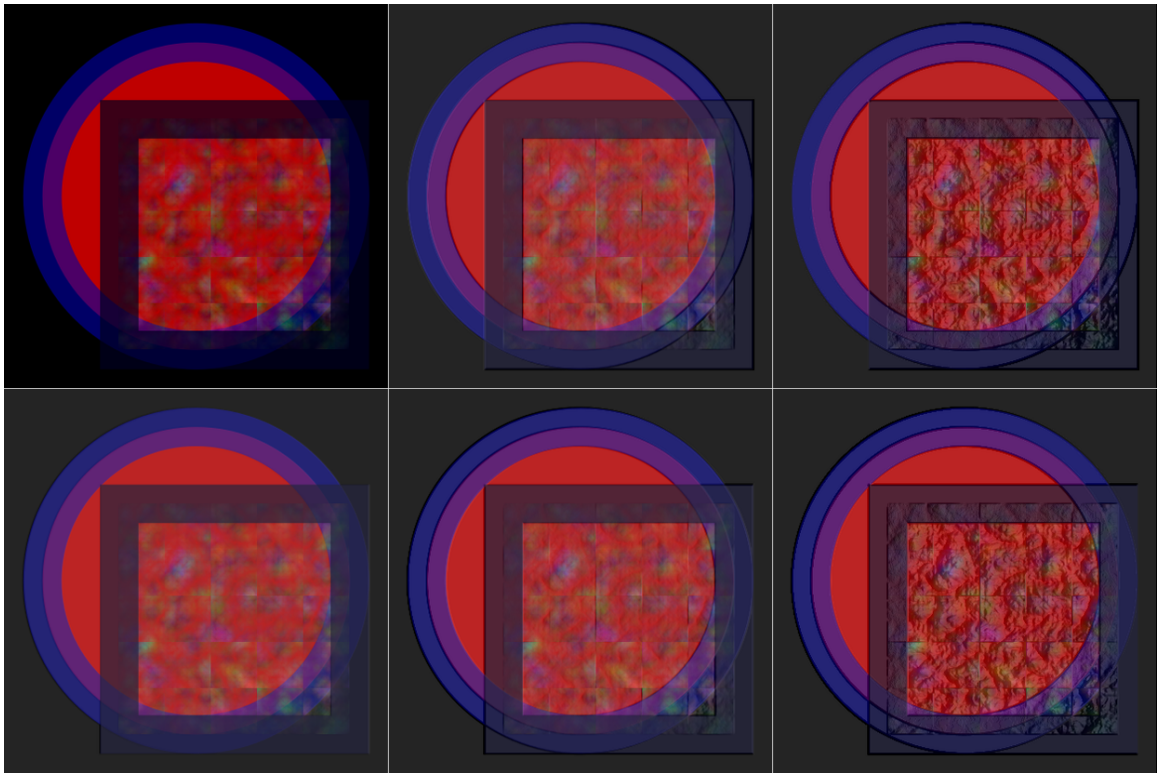


Abb. 377 Oberfläche (2)

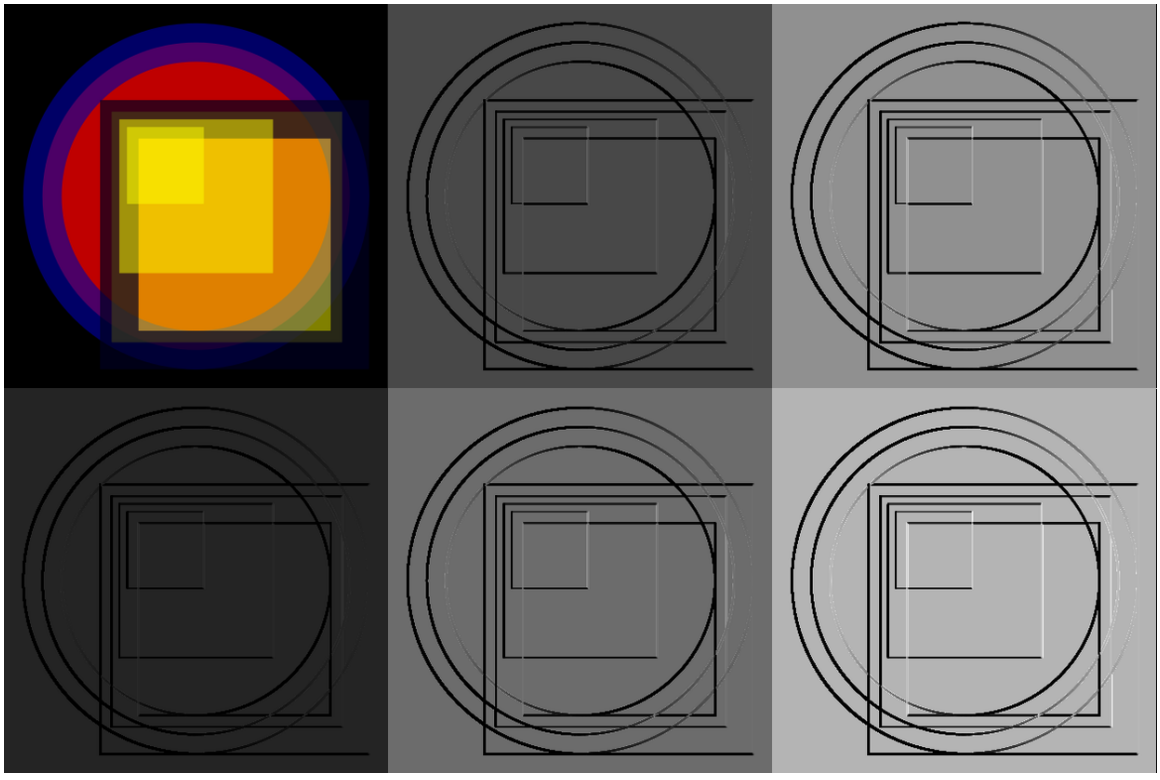


Abb. 378 diffuse Konstante (1)

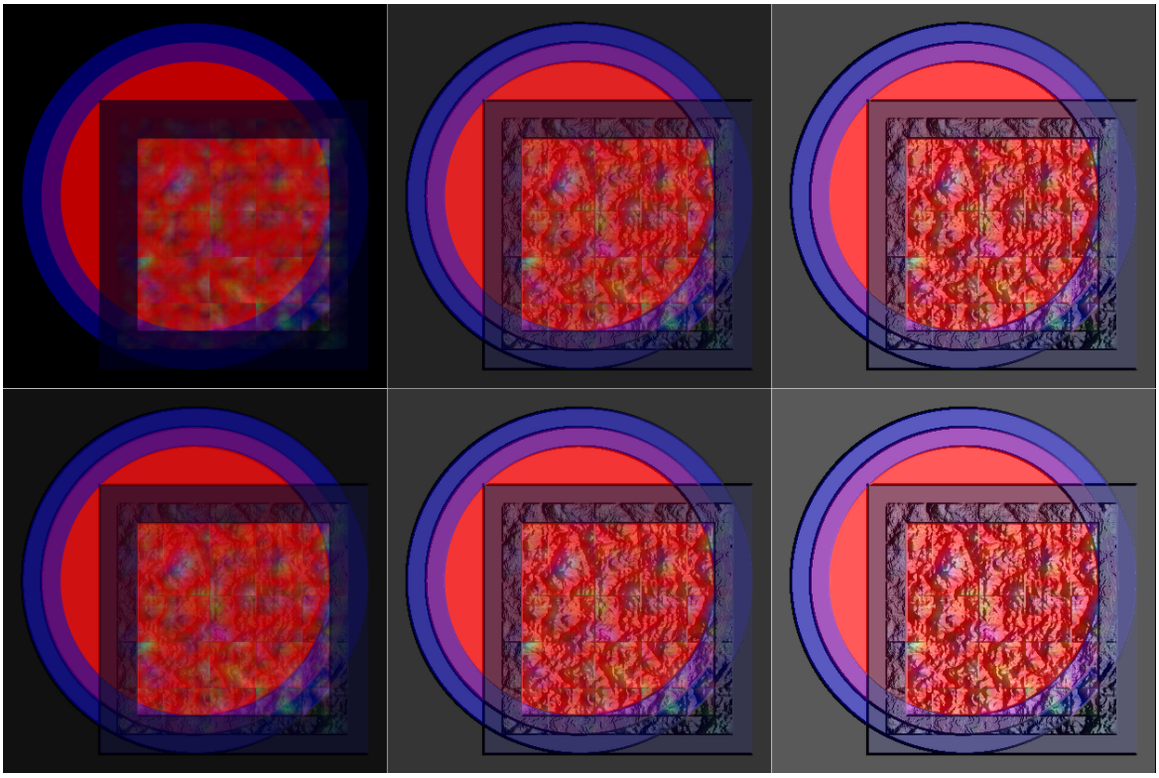


Abb. 379 diffuse Konstante (2)

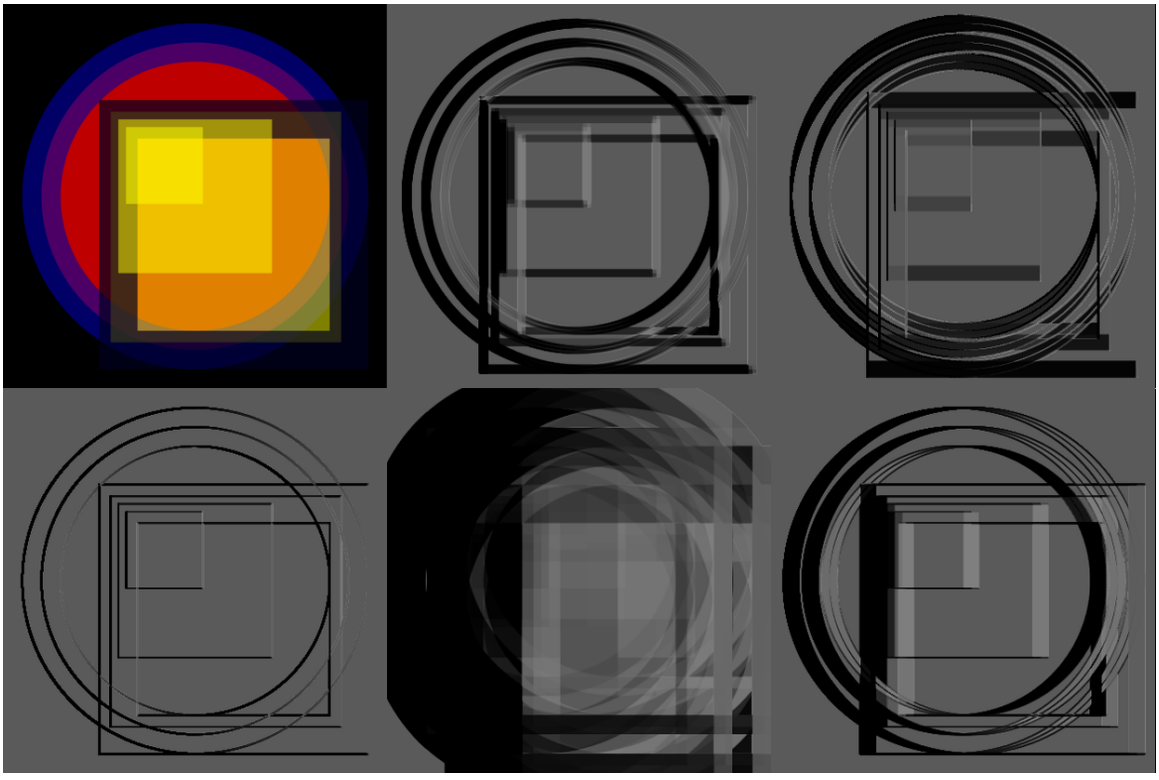


Abb. 380 *kernelUnitLength*

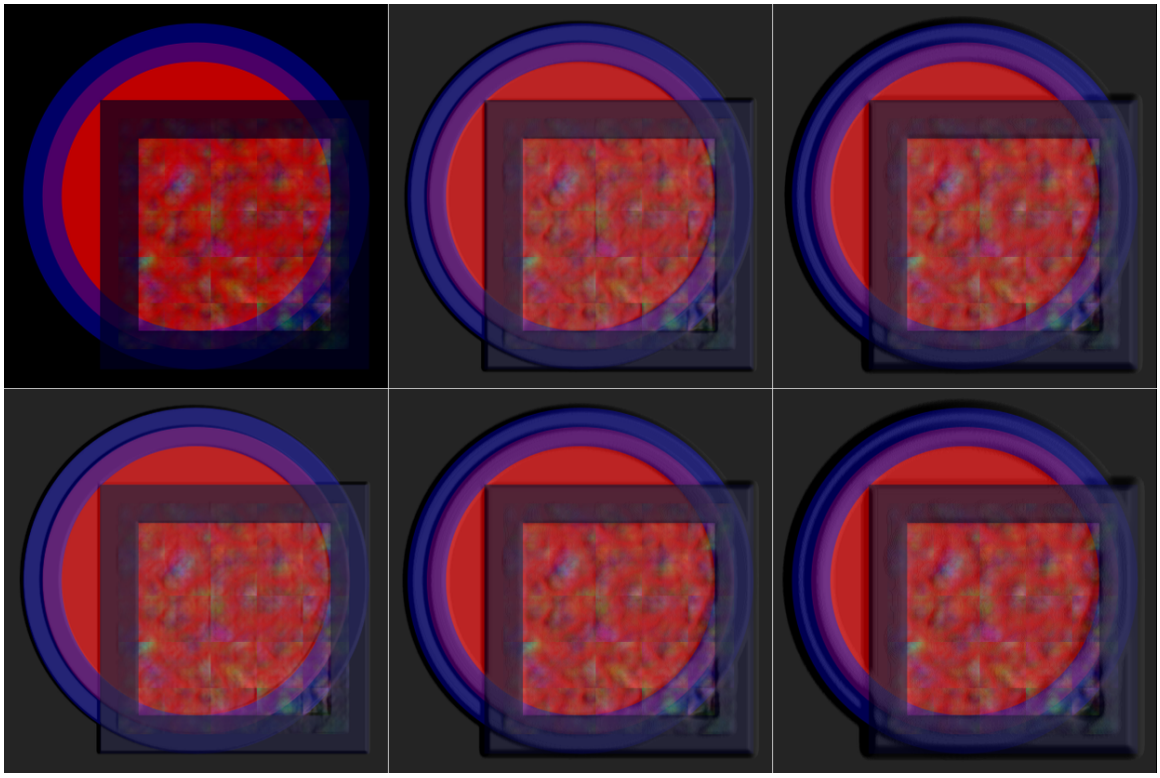


Abb. 381 Kanten ...

Diffuse Beleuchtung mit rotem Scheinwerfer:

```
<filter id="DL"
    filterUnits="userSpaceOnUse"
    x="0" y="0" width="500" height="500">
  <feDiffuseLighting
      surfaceScale="1"
      diffuseConstant="0.7"
      kernelUnitLength="1"
      lighting-color="#f00">
    <feSpotLight x="250" y="500" z="500"
        pointsAtX="200" pointsAtY="200" pointsAtZ="0"
        specularExponent="12" limitingConeAngle="60" />
  </feDiffuseLighting>
</filter>
```

Einige Beispiele zur diffusen Beleuchtung. Links oben ist das ungefilterte Ausgangsbild zu sehen.

Oberfläche (1)[73]

Es werden jeweils unterschiedliche Höhen für die Oberfläche angegeben, unten positiv, von links nach rechts steigend, in der Mitte und rechts darüber entsprechende negative Werte.

Diffuses Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und übereinanderliegenden Rechtecken.

73 http://de.wikibooks.org/wiki/media%3ASVGfeDiffuseLighting01.svg

Um Kanten besser sichtbar zu machen, kann es auch hilfreich sein, einen Weichzeichner auf den Alphakanal anzuwenden, welcher die Teiltransparenz etwas verschmiert, so dass die Kanten etwas breiter und mit mehr Werten auftreten, Beleuchtungseffekte an solchen Kanten also auch etwas ausgedehnter sind.

Oberfläche (2)[74]

Es werden jeweils unterschiedliche Höhen für die Oberfläche angegeben, unten positiv, von links nach rechts steigend, in der Mitte und rechts darüber entsprechende negative Werte.

Diffuses Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und zwei Rechtecken, eines davon mit einem turbulenten Muster. Per Komposition wird die diffuse Beleuchtung einer gleichmäßigen Beleuchtung überlagert.

diffuse Konstante (1)[75]

Einfluss der diffusen Konstante.

Diffuses Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und übereinanderliegenden Rechtecken.

diffuse Konstante (2)[76]

Einfluss der diffusen Konstante.

Diffuses Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und zwei Rechtecken, eines davon mit einem turbulenten Muster. Per Komposition wird die diffuse Beleuchtung einer gleichmäßigen Beleuchtung überlagert.

Statt die diffuse Konstante des Beleuchtungsurfilters zu ändern, könnte in diesem Beispiel auch der Anteil in der Komposition verändert werden.

kernelUnitLength[77]

Einfluss der kernelUnitLength.

Diffuses Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und übereinanderliegenden Rechtecken.

Kanten...[78]

Es werden jeweils unterschiedliche Höhen für die Oberfläche angegeben, kombiniert mit Weichzeichnern, welche die Kanten etwas verbreitern.

74 http://de.wikibooks.org/wiki/media%3ASVGfeDiffuseLighting02.svg
75 http://de.wikibooks.org/wiki/media%3ASVGfeDiffuseLighting03.svg
76 http://de.wikibooks.org/wiki/media%3ASVGfeDiffuseLighting04.svg
77 http://de.wikibooks.org/wiki/media%3ASVGfeDiffuseLighting05.svg
78 http://de.wikibooks.org/wiki/media%3ASVGfeDiffuseLighting06.svg

Diffuses Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und zwei Rechtecken, eines davon mit einem turbulenten Muster. Per Komposition wird die diffuse Beleuchtung einer gleichmäßigen Beleuchtung überlagert.

20.28 Urfilter *feSpecularLighting*, Glanzlichter

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3 (teilweise und falsche Filterregion)
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Urfilter werden Glanzlichter gemäß Phong-Modell realisiert. Ausgangspunkt dafür ist ähnlich wie für ***feDiffuseLighting*** der Alpha-Kanal.

Um mehrere Lichtquellen ***feDistantLight***, ***fePointLight*** und ***feSpotLight*** miteinander zu kombinieren, müssen mehrere Urfilter ***feSpecularLighting*** mit jeweils einer Lichtquelle notiert werden und deren Ergebnisse können dann miteinander kombiniert werden.

Oft werden ***feSpecularLighting*** und ***feDiffuseLighting*** innerhalb eines Filters miteinander kombiniert, welche typische Flächen sowohl teilweise spiegelnde Reflexion als auch diffuse Streuung aufweisen.

Weil die Berechnung pixelweise stattfindet, ist das Ergebnis abhängig von der Auflösung. Es empfielt sich die explizite Festlegung von *filterRes* oder *kernelUnitLength*, wenn das Ergebnis unabhängig von den aktuellen Anzeigebedingungen sein soll.

20.28.1 Attribut *kernelUnitLength*

Das Attribut gibt die Auflösung. Der Wert ist entweder eine positive Zahl oder eine Liste von zwei positiven Zahlen. Die erste Zahl gibt die Anzahl der Pixel pro Komponente in x-Richtung an, die zweite in y-Richtung. Ist nur eine Zahl angegeben, so gilt diese für beide Richtungen.

Durch Angabe des Attributes wird der Filter skalierbar. Um dies zu erreichen, muss wenigstens *kernelUnitLength* oder *filterRes* angegeben werden. In typischen Darstellungsprogramm ist die Effektivtät am höchsten, wenn ein Pixel der Matrix einem der Filterregion entspricht.

Angabe eines nicht positiven Wertes ist ein Fehler, der zum Abbruch der Darstellung des Dokumentes führt.

Das Attribut ist animierbar.

20.28.2 Attribut *surfaceScale*

Das Attribut gibt einen Skalierungsfaktor an, der bestimmt, welche z-Koordinate sich aus dem Alphakanal ergibt, also z = *surfaceScale* * Alphakanalwert.

Der Wert ist eine Zahl. Voreinstellung ist 1. Das Attribut ist animierbar. In der Definition ist präzise in Formeln angegeben, wie sich der Filtereffekt berechnet.

20.28.3 Attribut *specularConstant*

Das Attribut gibt den Reflexionsfaktor für die spiegelnde Komponente der Reflexion nach dem Phong-Beleuchtungsmodell an. Der Wert ist eine nicht negative Zahl. Ein kleiner Wert ergibt also wenig diffus gestreutes Licht von der Oberfläche. Typisch sind Werte zwischen 0 und 1, da nicht weniger als nichts diffus gestreut werden kann und nicht mehr als alles. Andererseits ist das Modell aber ohnehin nicht besonders realistisch, so dass auch größere Werte nicht schaden. Gegebenenfalls wird einfach bei der Darstellung das Ergebnis der Farbkanäle auf den erlaubten Bereich eingeschränkt.

Voreinstellung ist 1. Das Attribut ist animierbar.

20.28.4 Attribut *specularExponent*

Das Attribut gibt den Exponenten für den spiegelnden Anteil nach dem Phong-Beleuchtungsmodell an. Der Wert ist eine Zahl zwischen 1 und 128. Ist das Attribut nicht angegeben, wird 1 angenommen. Ein größerer Wert bedeutet jeweils ein stärkeres Glänzen der Reflexion, also ein ausgeprägteres Glanzlicht.

Das Attribut ist animierbar.

20.28.5 Beispiele *feSpecularLighting*

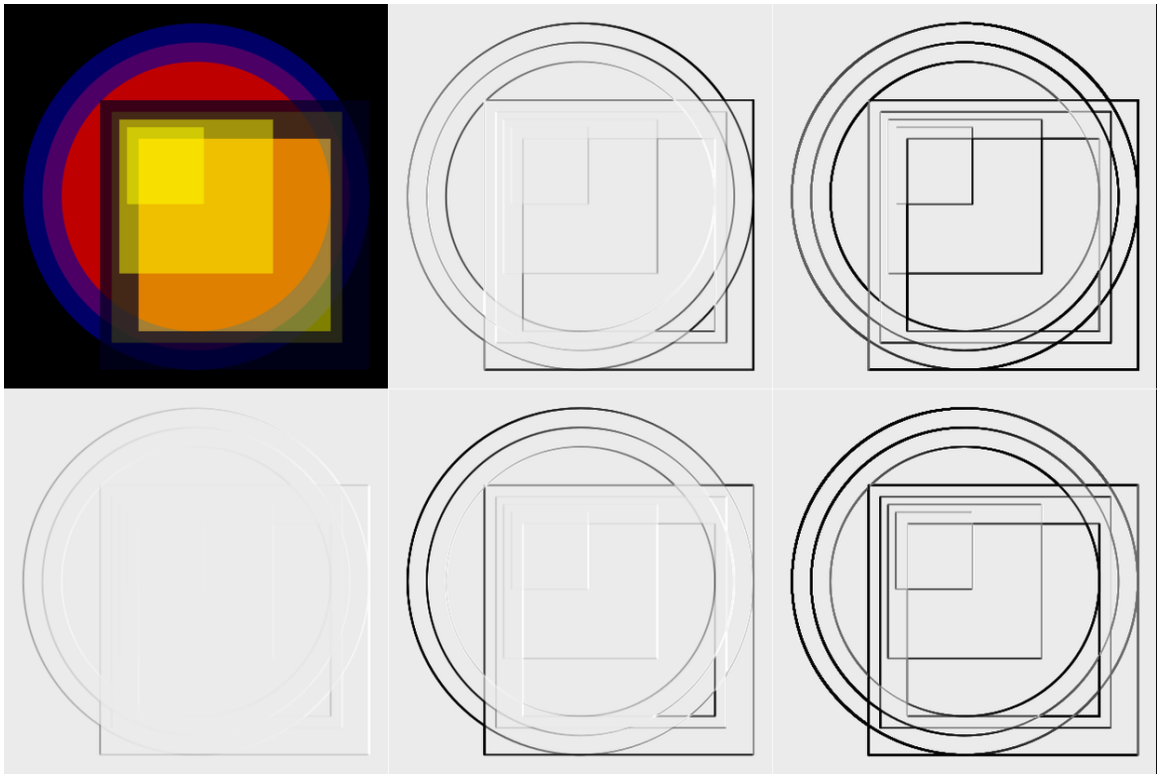


Abb. 382 Oberfläche (1)

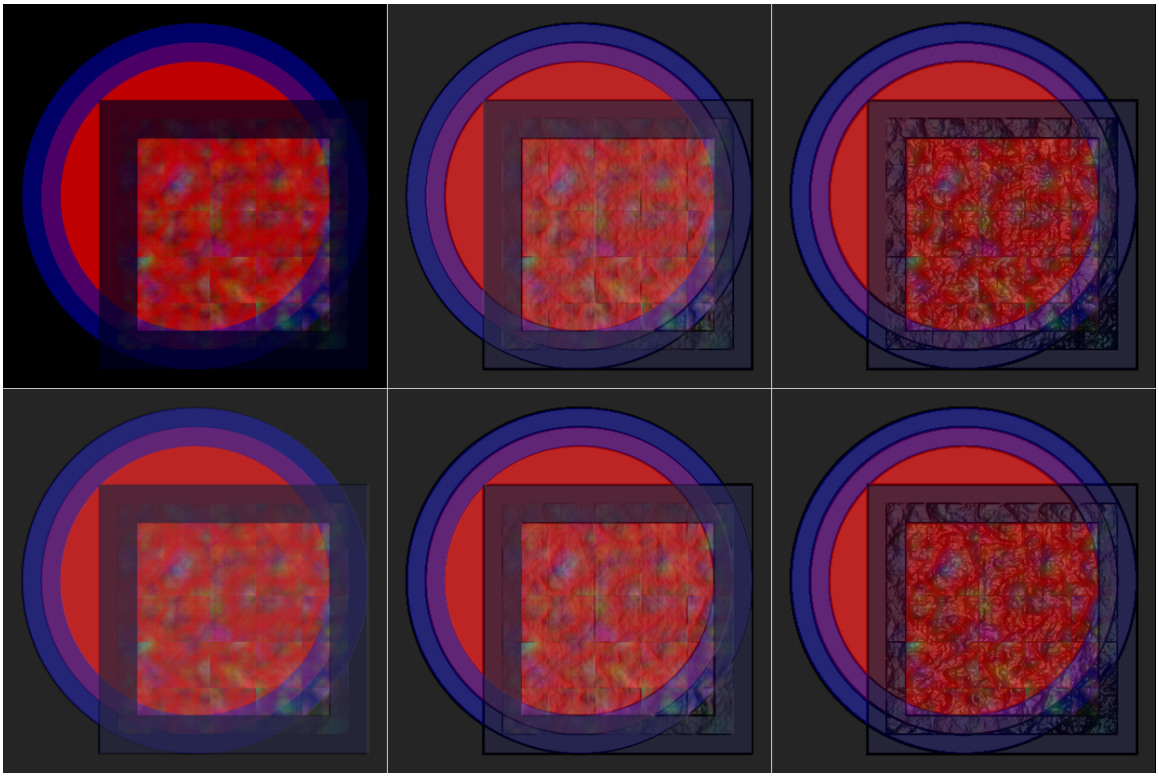


Abb. 383 Oberfläche (2)

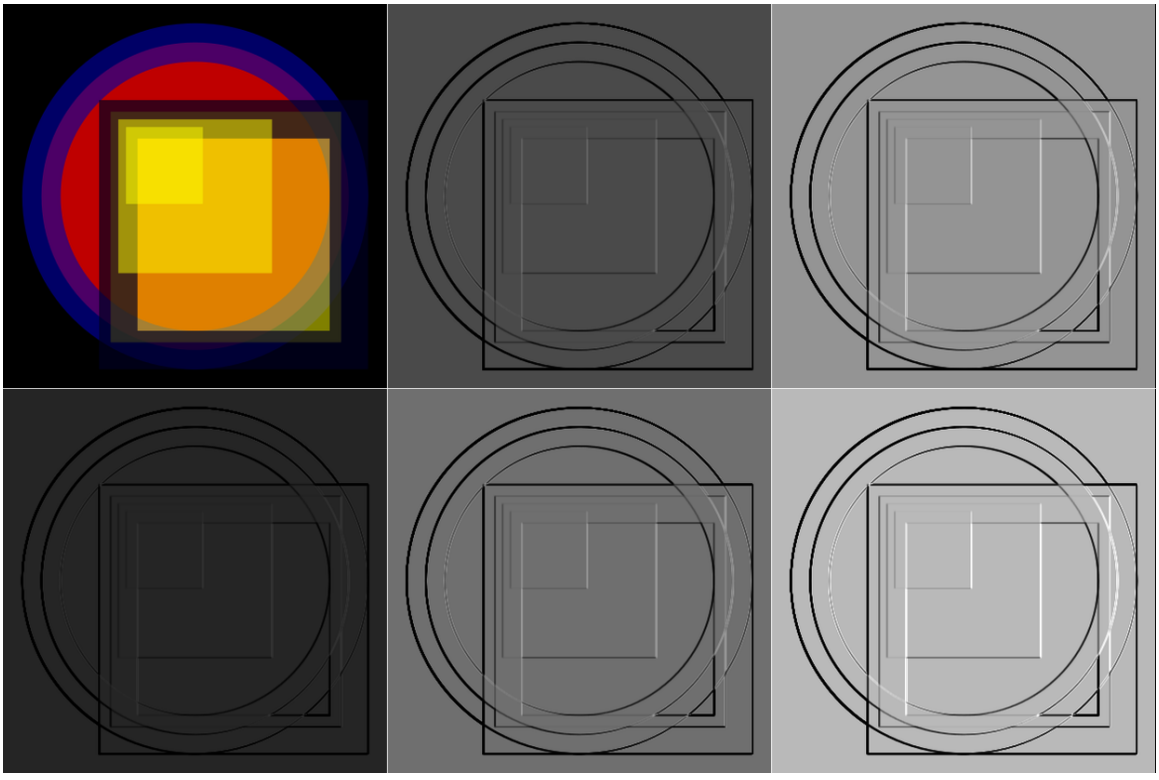


Abb. 384 Reflexionsstärke (1)

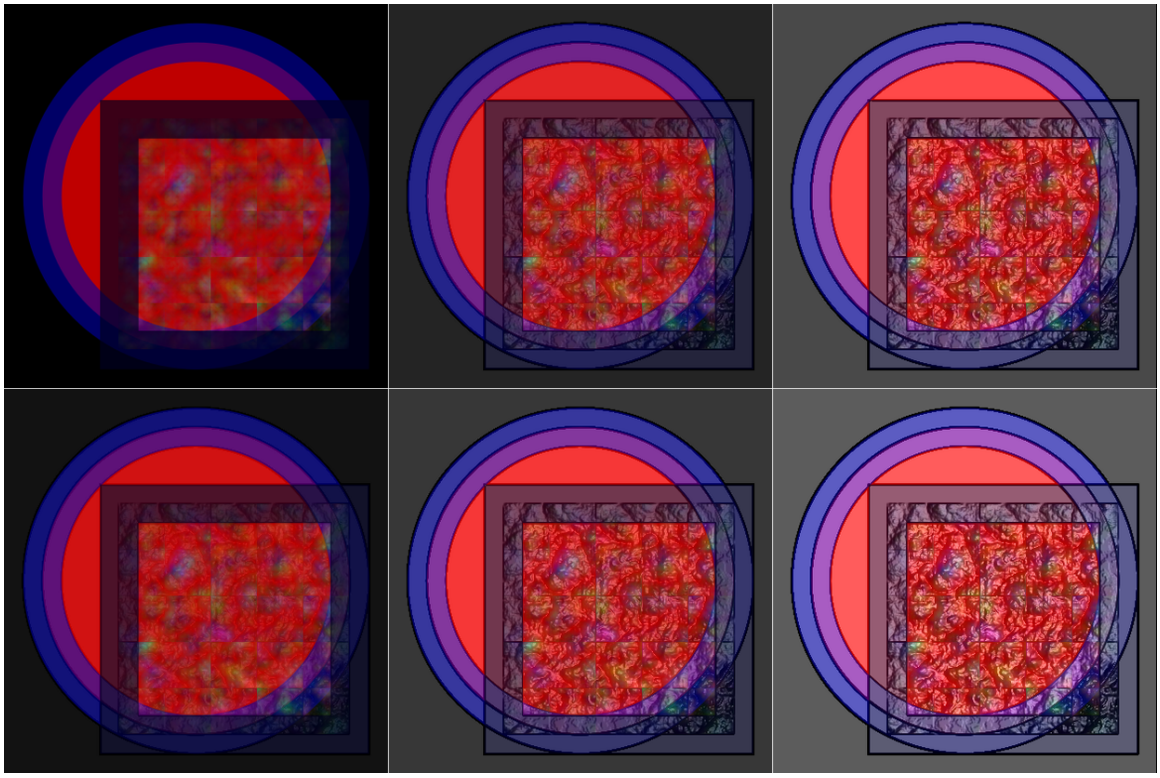


Abb. 385 Reflexionsstärke (2)

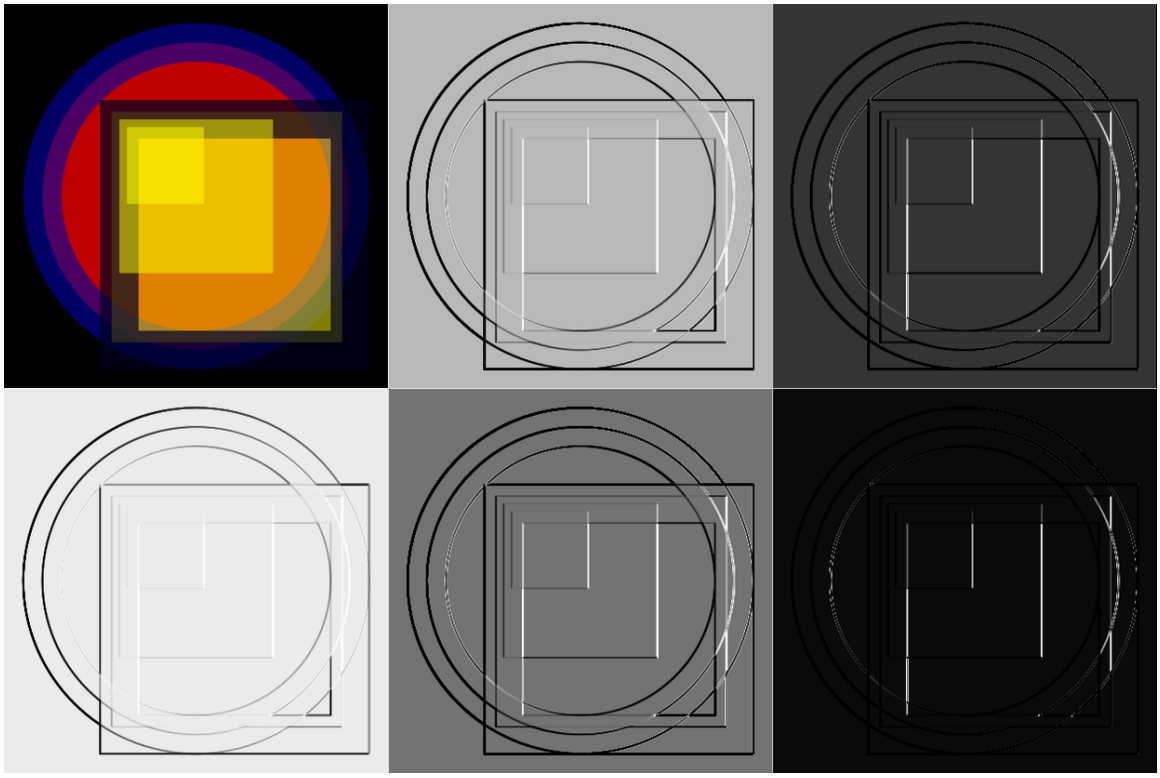


Abb. 386 Reflexionsexponent (1)

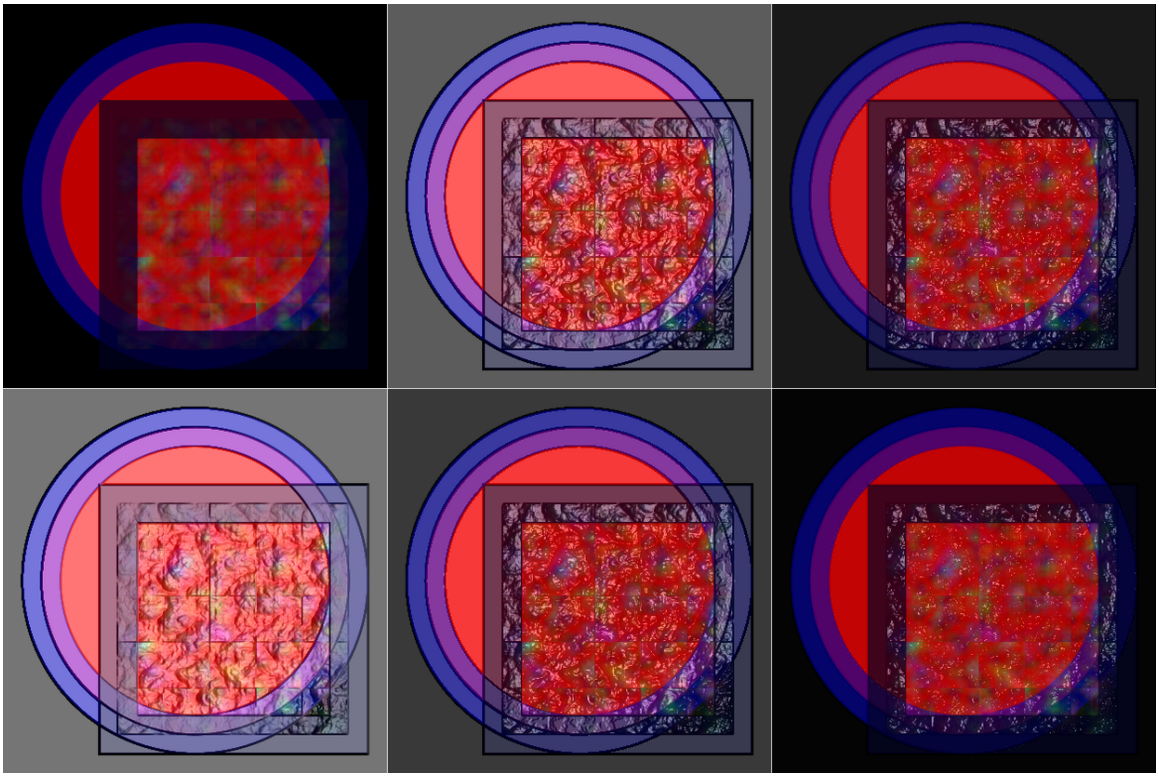


Abb. 387 Reflexionsexponent (2)

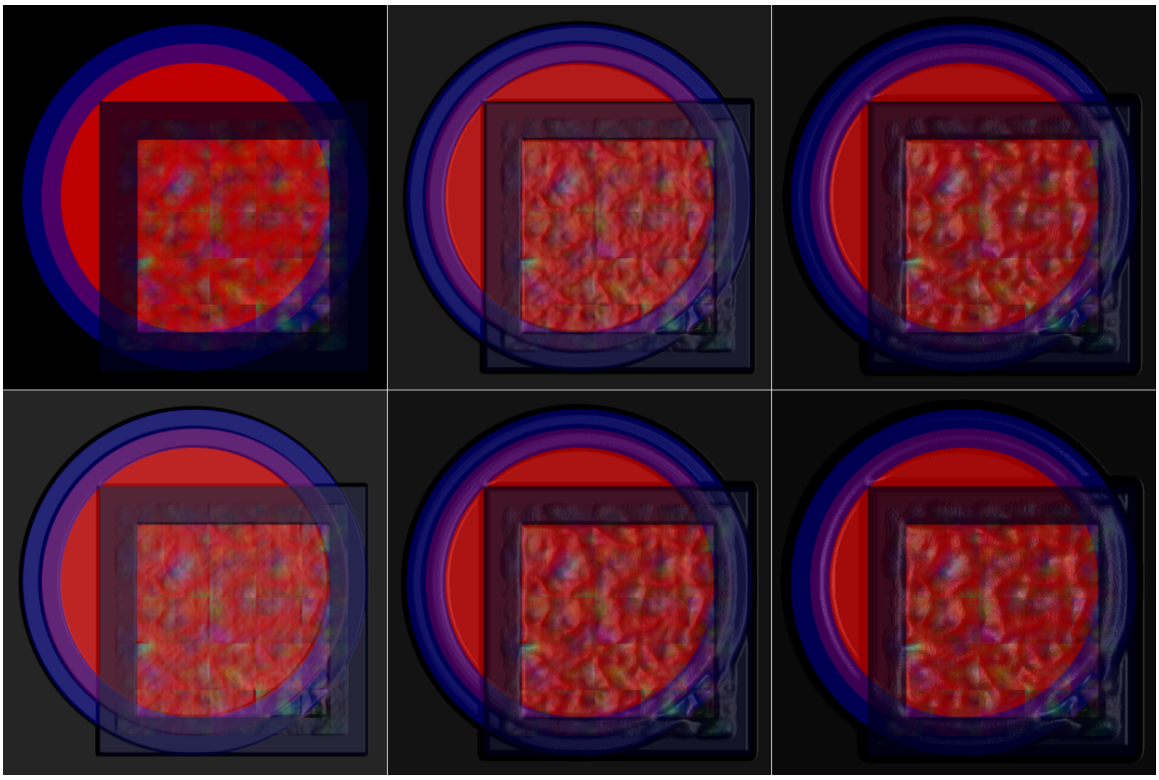


Abb. 388 mit Weichzeichner

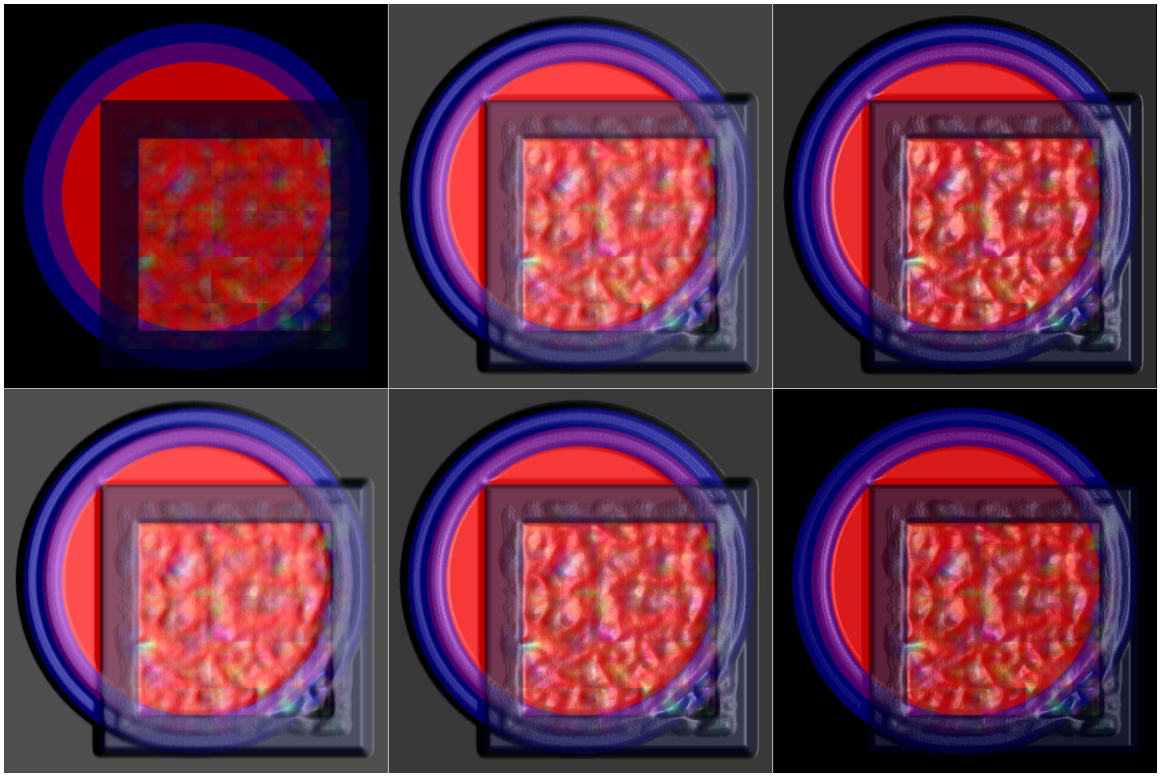


Abb. 389 diffuse und reflektierende Beleuchtung

Glanzlicht mit gelber Punktlichtquelle:

```
<feSpecularLighting id="fSL"
    x="0" y="0" width="500" height="500"
    surfaceScale="100"
    specularConstant="0.7"
    specularExponent="4"
    lighting-color="#ff1">
  <fePointLight x="500" y="250" z="120" />
</feSpecularLighting>
```

Einige Beispiele zur reflektierenden Beleuchtung. Links oben ist das ungefilterte Ausgangsbild zu sehen.

Oberfläche (1)[79]

Es werden jeweils unterschiedliche Höhen für die Oberfläche angegeben, unten positiv, von links nach rechts steigend, in der Mitte und rechts darüber entsprechende negative Werte.

Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und übereinanderliegenden Rechtecken.

Um Kanten besser sichtbar zu machen, kann es auch hilfreich sein, einen Weichzeichner auf den Alphakanal anzuwenden, welcher die Teiltransparenz etwas verschmiert, so dass die Kanten etwas

79 http://de.wikibooks.org/wiki/media%3ASVGfeSpecularLighting01.svg

breiter und mit mehr Werten auftreten, Beleuchtungseffekte an solchen Kanten also auch etwas ausgedehnter sind.

Oberfläche (2)[80]

Es werden jeweils unterschiedliche Höhen für die Oberfläche angegeben, unten positiv, von links nach rechts steigend, in der Mitte und rechts darüber entsprechende negative Werte.

Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und zwei Rechtecken, eines davon mit einem turbulenten Muster. Per Komposition wird die diffuse Beleuchtung einer gleichmäßigen Beleuchtung überlagert.

Reflexionsstärke (1)[81]

Einfluss der reflektierenden Konstante.

Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und übereinanderliegenden Rechtecken.

Reflexionsstärke (2)[82]

Einfluss der reflektierenden Konstante.

Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und zwei Rechtecken, eines davon mit einem turbulenten Muster. Per Komposition wird die diffuse Beleuchtung einer gleichmäßigen Beleuchtung überlagert.

Statt die reflektierende Konstante des Beleuchtungsurfilters zu ändern, könnte in diesem Beispiel auch der Anteil in der Komposition verändert werden.

Reflexionsexponent (1)[83]

Einfluss des reflektierenden Exponenten.

Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und übereinanderliegenden Rechtecken.

Reflexionsexponent (2)[84]

Einfluss des reflektierenden Exponenten.

Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und zwei Rechtecken, eines davon mit einem turbulenten Muster. Per Komposition wird die diffuse Beleuchtung einer gleichmäßigen Beleuchtung überlagert.

80 http://de.wikibooks.org/wiki/media%3ASVGfeSpecularLighting02.svg
81 http://de.wikibooks.org/wiki/media%3ASVGfeSpecularLighting03.svg
82 http://de.wikibooks.org/wiki/media%3ASVGfeSpecularLighting04.svg
83 http://de.wikibooks.org/wiki/media%3ASVGfeSpecularLighting05.svg
84 http://de.wikibooks.org/wiki/media%3ASVGfeSpecularLighting06.svg

kernelUnitLength[85]

Einfluss der *kernelUnitLength*.

Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und übereinanderliegenden Rechtecken.

mit Weichzeichner[86]

Es werden jeweils unterschiedliche Höhen für die Oberfläche und andere Exponenten für die Lichtquelle angegeben, kombiniert mit Weichzeichnern, welche die Kanten etwas verbreitern.

Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und zwei Rechtecken, eines davon mit einem turbulenten Muster. Per Komposition wird die reflektierende Beleuchtung einer gleichmäßigen Beleuchtung überlagert.

diffuse und reflektierende Beleuchtung[87]

Es werden jeweils verschiedene Anteile von diffuser und reflektierender Beleuchtung gemischt und kombiniert mit Weichzeichnern, welche die Kanten etwas verbreitern.

Licht kommt von rechts. Das beleuchtete Motiv besteht aus einem großen Kreis und zwei Rechtecken, eines davon mit einem turbulenten Muster. Per Komposition wird die diffuse und reflektierende Beleuchtung einer gleichmäßigen Beleuchtung überlagert.

Beim letzten Beispiel unten rechts wird zusätzlich die Reflexion mit dem weichgezeichneten Alphakanal maskiert, damit kein Licht auf den schwarzen Hintergrund wirkt.

20.29 Literatur

- SVG 1.1 Filter[88] (en)
- Arbeitsentwurf SVG 1.2 Filter-Modul[89] (en)
- T. Porter and T. Duff - Compositing Digital Images SIGGRAPH 84 (Computer Graphics Volume 18, Number 3 July 1984) pp 253-259 [90] (en, PDF)
- UNKNOWN TEMPLATE W
  Phong-Beleuchtungsmodell
- UNKNOWN TEMPLATE W
  Perlin-NoisePerlin-Rauschen
- Willhelm Burger: Gradientenbasierte Rauschfunktion und Perlin Noise (2008)[91]

85 http://de.wikibooks.org/wiki/media%3ASVGfeSpecularLighting07.svg
86 http://de.wikibooks.org/wiki/media%3ASVGfeSpecularLighting08.svg
87 http://de.wikibooks.org/wiki/media%3ASVGfeSpecularLighting09.svg
88 http://www.w3.org/TR/SVG11/filters.html
89 http://www.w3.org/TR/SVGFilter12/
90 http://keithp.com/~keithp/porterduff/p253-porter.pdf
91 http://staff.fh-hagenberg.at/burger/publications/reports/2008GradientNoise/Burger-GradientNoiseGerman-2008.pdf

21 Skripte

21.1 Skripte und Programme

Beim Zusammenspiel von SVG mit Skripten oder Programmen sind verschiedene Typen von Skripten oder Programme zu unterscheiden.

Zunächst gibt es jene Programme, welche ein SVG-Dokument präsentieren sollen, also graphisch darstellen, drucken, vorlesen etc. Dies sind die Darstellungsprogramme, von denen bislang in diesem Buch die Rede war. Diese Programme benötigen korrekte SVG-Dokumente als Eingabe und die korrekte Präsentation ist in den Empfehlungen zu SVG festgelegt.

Dann gibt es Skripte oder Programme, welche von einem Format in ein anderes konvertieren können. Diese benötigen korrekte Dokumente in einem Format als Eingabe und haben die Aufgabe, korrekte Dokumente in einem anderen Format wieder auszugeben. Bei einer Optimierung kann das Ein- und Ausgabeformat auch das gleiche sein. Die Interaktion mit dem Benutzer ist zumeist auf definierte Eingabeparameter begrenzt. Einige dieser Programme sind auch interaktiv und fragen während des Programmablaufes Parameter ab oder werden mit Steuerdateien beeinflusst.

Ferner gibt es Dienstprogramme (englisch: server).

Wird an diese eine Anfrage gestellt, kann eine Verarbeitung mit einer Ausgabe die Folge sein. Zum Beispiel liefern webserver bei Anfragen gemäß dem Protokoll HTTP (Hypertext-Transfer-Protokoll) Dokumente oder allgemeiner Ausgaben aus.

Von so einem webserver können wiederum Skripte abgefragt werden (PHP - Hypertext-Präprozessor; Perl etc), diese werden dann ausgeführt und produzieren eine Ausgabe.

Zurückgegeben wird also nicht das Skript selbst, sondern dessen Ausgabe.

Im Falle von SVG ist das in der Regel ein komplettes SVG-Dokument, welches ganz oder teilweise gemäß der Anfrage dynamisch erstellt wurde.

Solche Skripte oder Programme werden auch server-seitige Skripte genannt.

Die funktionieren unabhängig von den Möglichkeiten des Darstellungsprogrammes. Erforderlich ist nur eine Anfrage an das Dienstprogramm.

Das Skript erzeugt dann zumeist ein komplettes SVG-Dokument (oder ein Dokument in einem anderen Format).

Das Darstellungsprogramm interpretiert nur die Ausgabe, insofern ist eine klare zeitliche und inhaltliche Aufgabetrennung zwischen dem Skript und dem Darstellungsprogramm gegeben.

Entsprechend sind die Interaktionsmöglichkeiten des Nutzers mit dem Skript begrenzt. Eine Interaktion erfordert eine erneute Anfrage an das Dienstprogramm und einen erneuten Durchlauf des Skriptes.

Für die Darstellung ist es dafür allerdings belanglos, welche Skript- oder Programmiersprache verwendet wird. Das Skript oder Programm muss nur in der Lage sein, eine Textausgabe zu erzeugen, die ein korrektes SVG-Dokument ergibt. Die Funktion des Skriptes ist also unabhängig vom Darstellungsprogramm und von dessen Einstellungen und Möglichkeiten, was ein großer Vorteil für den Autor ist, der so die komplette Kontrolle über das Skript und dessen Ausführung behält. Sicherheits- oder technische Mängel des Skriptes stellen andererseits kein dramatisches Problem für den Nutzer der Ausgabe dar. Schlimmstenfalls kommt die Ausgabe nicht oder nicht korrekt zustande.

Bezogen auf dieses Buch sind die zuvor beschriebenen Programme jenseits des Themas dieses Buches und werden nicht weiter diskutiert, eben weil sie unabhängig von den Gegegebenheiten von SVG funktionieren und das SVG-Darstellungsprogramm unabhängig von ihnen funktioniert. Sie können unbedenklich von Autoren eingesetzt werden.

Zusätzlich gibt es in der Voll- und in der Basisversion von SVG 1.1 und in SVG tiny 1.2 auch die Möglichkeit, dass das Darstellungsprogramm selbst Skripte ausführt. Dies ist in SVG tiny 1.1 ausgeschlossen. Diese Skripte werden anwenderseitige Skripte bezeichnet und sind entweder direkt im SVG-Dokument notiert oder im SVG-Dokument wird das Skript referenziert. Nur um solche Skripte geht es im folgenden Abschnitt dieses Kapitels.

21.2 Im Darstellungsprogramm ausführbare Skriptsprachen

Neben der eigentlichen Darstellung des SVG-Inhaltes kann ein Darstellungsprogramm auch zusätzlich die Fähigkeit haben, Skriptsprachen zu interpretieren. Ob ein im Dokument befindliches Skript ausgeführt wird, hängt von der verwendeten Skriptsprache, vom Darstellungsprogramm und dessen Fähigkeiten ab und davon, was der Nutzer des Darstellungsprogrammes für Voreinstellungen vorgenommen hat. Weil auch nicht jedes Darstellungsprogramm jede geeignete Skriptsprache interpretiert und der Nutzer die Interpretation per Voreinstellung unterbinden oder erlauben kann, kann sich der Autor nie darauf verlassen, dass ein Skript wirklich wie geplant ausgeführt wird.

SVG selbst sieht keine Skriptsprache vor, welche von einem Darstellungsprogramm interpretiert werden muss. Allerdings definiert SVG ein Dokument-Objekt-Modell, in welchem die Skriptsprache agieren kann. Häufig implementiert sind unterschiedliche Umsetzungen des Standards ecmascript (von Ecma International standartisiertes JavaScript).

Allerdings werden anwenderseitige Skripte gegebenenfalls während der Darstellung des Dokumentes ausgeführt, was eine permanente Interaktion des Nutzers mit dem Dokument ermöglicht. SVG bietet dafür Schnittstellen, um Nutzerinteraktionen und Ereignisse auszuwerten und darauf zu reagieren.

Komplexere Skripte können sogar mit Dienstprogrammen über das internet interagieren (Stichwort: AJAX; englisch: Asynchronous JavaScript and XML).

Weil die Skripte im Darstellungsprogramm des Nutzers auf dessen Rechner laufen, trägt dieser auch das Risiko bei technischen Mängeln und bei Sicherheitsmängeln des Skriptes oder des Darstellungsprogrammes. Je nach Skriptsprache oder Umsetzung der Skriptsprache besteht auch die Möglichkeit, Informationen vom Rechner des Nutzers auszuwerten.

Durch die AJAX-Interaktionsmöglichkeit ergibt sich weiterhin das Potential, dass die Daten vom Nutzer unbemerkt zu anderen Rechnern weitergeleitet werden.

Zwar sind sich die Anbieter der Darstellungsprogramme dessen bewusst und setzen Konzepte um, um ein unbemerktes Versenden brisanter Informationen zu verhindern, auch werden die Sprachen so konstruiert, um solche zweifelhaften Anwendungen weitgehend auszuschließen. Allerdings hat die bisherige Historie der konkreten Umsetzung anwenderseitiger Skriptsprachen bislang gezeigt, dass immer wieder unbeabsichtigte Sicherheitslecks auftreten. Das Sicherheitsrisiko trägt also letztlich der Nutzer, welcher die Interpretation von Skripten erlaubt.

Das technische Risiko von anwenderseitigen Skripten liegt darin, dass schlecht oder fehlerhaft geschriebene Skripte oder eine eventuell beim Darstellungsprogramm ineffektive Umsetzung der Skriptinterpretation zu einer starken Belastung des Prozessors des Rechners des Nutzers führen kann - oder auch zum Einfrieren oder zum Absturz des Darstellungsprogrammes. Weil der Nutzer die Qualität der Skripte beim Aufruf unbekannter Dokumente im vorhinein nicht beurteilen kann, kann dies ein weiterer Grund sein, dass Nutzer die Interpretation von Skripten nicht erlauben, ebenso wie die gegebenenfalls bekannte starke Prozessorbelastung bei ineffektiver Umsetzung der Skriptinterpretation oder die Tendenz zu Abstürzen.

21.2.1 Zugänglichkeitsprobleme und Lösungsstrategie

Aus diesen Unwägbarkeiten, ob das Darstellungsprogramm das Skript interpretieren kann oder darf, ergibt sich für Dokumente mit Skripten ein Zugänglichkeitsproblem, dem sich der Autor des Dokumentes stellen muss, um Probleme zu vermeiden. Wäre Inhalt nur bei Skriptinterpretation verfügbar, so wäre dies eine Zugänglichkeitsbarriere, die zu vermeiden ist.

Die pauschale Lösung des Problems liegt darin, die komplette Information eines Dokumentes ohne die Notwendigkeit der Skriptinterpretation verfügbar zu machen. Zu diesem Zwecke erstellt der Autor zunächst das komplette Dokument mit der beabsichtigten Information ohne Skript. Dieses Dokument enthält dann per Definition die komplette Information, die mit dem Dokument vermittelt werden soll.

Wie dann anschließend mittels CSS eine alternative Ansicht des Dokumentes formuliert werden kann, so kann zusätzlich über anwenderseitige Skripte ein zusätzlicher alternativer Zugang zur Information des Dokumentes angeboten werden. Wie die Dekoration per CSS den Inhalt, die Information eines Dokumentes nicht ändert, so ändert auch die Ausführung eines Skriptes nicht den Inhalt oder die Information eines Dokumentes, ermöglicht nur einen anderen Zugang.

Mit diesem Ausgangspunkt kann ein Autor an sich durch Skripte keine Barrieren für Nutzer ohne Skriptinterpretation erstellen. Allenfalls handelt es sich um inhaltsleere Dokumente, bei denen Nut-

zern mit Skriptinterpretation Inhalt vorgetäuscht wird. Allerdings ergibt sich bei dieser Interpretation das Problem, dass Nutzer den vorgetäuschten Inhalt als abweichend vom wahren Inhalt und eventuell als nützlicher empfinden könnten, woraus sich dann doch wieder ein Zugänglichkeitsproblem zur Täuschung für jene ergibt, bei denen die Skripte nicht ausgeführt werden.

Neben der technischen Realisation der Skripte ist es also zentrale Aufgabe des Autors eines Dokumentes, welches anwenderseitige Skripte enthält, den Zugang zur Information unabhängig von der Skriptinterpretation bereitzustellen, um Missverständnisse und Barrieren zu vermeiden.

21.2.2 Themeneingrenzung

Weil SVG selbst keine Skriptsprache vorsieht, welche interpretiert werden muss, ist das Erlernen von Skriptsprachen auch nicht Thema dieses Buches. SVG stellt jedoch Elemente, Schnittstellen und Methoden bereit, um Skripte einzubetten, zu referenzieren, auf Ereignisse zu reagieren und über das Dokument-Objekt-Modell die Struktur oder die Dekoration eines Dokumentes zu manipulieren.

In den Beispielen wird ecmascript verwendet. Eine konkrete Umsetzung davon ist java-script. Wikibooks hat auch ein Buch names Javascript[1], welches das Verständnis der Beispiele erleichtern kann, ebenso wie die Referenzen zum Dokument-Objekt-Modell am Endes dieses Kapitels.

Anders als in HTML, wo der Skriptautor zum Beispiel mit java-script noch Inhalte direkt in das Dokument schreiben konnte, erfolgt in SVG die Manipulation immer über das DOM, wobei es einerseits für XML ein generischen DOM für den allgemeinen Zugriff auf Attribute und Elemente und deren Inhalt gibt als auch für SVG 1.1 und SVG tiny 1.2 spezielle Konstruktionen, die den besonderen Anforderungen des Formates Rechnung tragen.

21.2.3 Angaben zur verwendeten Skriptsprache

Die verwendete Skriptsprache kann explizit angegeben werden. Mit dem Attribut *contentScriptType* des Elementes ***svg*** kann dokumentweit notiert werden, welche Skriptsprache verwendet wird, wenn diese nicht lokal angegeben wird. Das Attribut ist bereits im Kapitel Dokumentstruktur im Abschnitt über das Element ***svg*** beschrieben worden. Der Wert von *contentScriptType* ist ein Medientyp (ehemals MIME-Typ). Typisch für Skripte innerhalb von SVG ist zum Beispiel 'application/ecmascript', was in SVG tiny 1.2 auch die Voreinstellung ist, wenn das Attribut nicht angegeben ist.

Zum Zeitpunkt, als SVG 1.1 spezifiziert wurde, hatte ecmascript noch keinen Medientyp. Statt 'application/ecmascript' wurde damals angenommen, dass der Medientyp 'text/ecmascript' heißen wird, was daher für SVG 1.1 die Voreinstellung ist. Es ist daher zum empfehlen, den Medientyp in SVG 1.1 immer explizit anzugeben, selbst wenn ecmascript verwendet wird.

Mit dem Attribut *type* kann ferner bei jedem Element, welches Skripte beinhalten kann, der Medientyp gesondert und gegebenenfalls auch abweichend von der globalen Angabe notiert werden. Der Wert ist auch jeweils ein Medientyp.

Beide Attribute sind nicht animierbar.

1 `http://de.wikibooks.org/wiki/Javascript`

Durch bedingte Verarbeitung mit dem Element ***switch*** und dem Attribut *requiredFormats* in SVG tiny 1.2 kann ferner die Darstellung davon abhängig gemacht werden, ob eine Interpretation der verwendete Skriptsprache gegeben ist oder nicht.

Ferner kann mit dem Attribut *requiredFeatures* festgestellt werden, ob überhaupt eine Skriptinterpretation durchgeführt wird.

Durch die bedingte Verarbeitung ist es so möglich, verschiedene Skriptsprachen und skriptfreie Alternativen anzubieten.

21.2.4 Element *script*

Squiggle (Batik)	1.7
Opera (Presto)	9
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	3
Konqueror (KSVG)	3.2
Safari (Webkit)	4
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Das Element ***script*** kann ausführbaren Inhalt enthalten oder ausführbaren Inhalt referenzieren.

Das Skript wird gegebenenfalls ausgeführt, sobald das Element geladen ist.

Weil Skripte nicht direkt dargestellter Inhalt sind, empfiehlt es sich, ***script***-Elemente im Element ***defs*** des Hauptelementes ***svg*** zu notieren.

Das Element entspricht dem gleichnamigen Element in XHTML. Man beachte, dass dieses anders als das gleichnamige Element in HTML zu interpretierenden Inhalt enthält. Wird das Skript als Elementinhalt notiert, so sind spezielle Zeichen von XML im Konfliktfalle folglich zu maskieren, insbesondere < als < und > als >. Alternativ kann der Inhalt als CDATA gekennzeichnet werden.

Neben dem bereits beschriebenen Attribut *type* stehen für ***script*** auch Attribute von XLink zur Verfügung. Insbesondere dient *href* dazu, ein externes Skript zu referenzieren. Sofern *href* von XLink angegeben ist, wird gegebenenfalls vorhandener Inhalt von ***script*** ignoriert und nicht ausgeführt, gegebenenfalls nur der das referenzierte externe Skript. Ist die angegeben IRI ungültig, wird kein Skript ausgeführt.

Kompilierte Skripte/Programme, wie etwa java-Programme (JAR-Archiv) sind immer zu referenzierten und nicht im Element zu notieren.

href ist für das Element ***script*** nicht animierbar.

Beispiel, Skript direkt eingebettet:

```
<script type="application/ecmascript">
 // Skriptquelltext &lt; &gt; etc
<script>
```

oder:

```
<script type="application/ecmascript">
<![CDATA[
 // Skriptquelltext < > etc
```

```
 ]]>
<script>
```

Oder es wird auf ein externes Dokument verwiesen:

```
<script type="application/ecmascript"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:href="scripts/meinScript.js"/>
```

Oder:

```
<script type="application/java-archive"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xlink:href="java/Beispiel.jar"/>
```

Sofern der Namensraum für XLink bereits bei einem Elternelement angegeben ist, ist dies natürlich nicht abernals notwendig.

21.2.5 Ereignisbehandlung in SVG 1.1

In SVG 1.1 basiert die Ereignisbehandlung in Verbindung mit Skripten einerseits auch DOM2-Ereignissen und auf speziellen Attributen. Die Attribute dienen als Empfänger von Ereignissen und damit als Schnittstelle zwischen dem Dokument und der Skriptsprache.

Die Attribute sind *onfocusin*, *onfocusout*, *onactivate*, *onclick*, *onmousedown*, *onmouseup*, *onmouseover*, *onmousemove*, *onmouseout* und *onload*. Die zugehörigen Ereignissnamen ergeben sich, wenn das vorangestellte 'on' entfernt wird. Zusätzlich sind Dokumentereignisse nutzbar. Dieses sind *onunload*, *onabort*, *onerror*, *onresize*, *onscroll*, *onzoom*. Die Attribute sind anwendbar auf graphische Elemente und Gruppierungselemente.

Eine genauere Beschreibung des jeweiligen Ereignisses ist im Kapitel Interaktivität zu finden.

Ferner gibt es Attribute bezogen auf Animationselemente. Dies sind: *onbegin*, *onend*, *onrepeat*. Die beziehen sich entsprechend auf den Beginn, das Ende, die Wiederholung einer Animation, ebenfalls wie für die entsprechenden Ereignisse im Kapitel Interaktivität beschrieben.

Die Attribute sind nicht animierbar.

Das jeweilige Attribut wird bei dem Element notiert, bei welchem das Ereignis registriert werden soll.

Der Wert eines Attributes ist ein Skript, welches beim Eintreten eines Ereignisses ausgeführt werden soll, oft nur ein Funktionsaufruf samt Parameter für eine Funktion, welche im Element ***script*** zu definieren ist, siehe Beispiele unten. Auch hier ist darauf zu achten, dass spezielle XML-Zeichen maskiert werden, insbesondere neben den Zeichen $<$ und $>$ auch das jeweilige Anführungszeichen, mit dem der Wert des Attributes eingeschlossen wird.

Beispiel:

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" version="1.1"
     viewBox="0 0 1000 1000"
     contentScriptType="application/ecmascript"
```

```
    xml:lang="de">
  <title>Ereignisbehandlung</title>
  <rect x="20" y="20" width="100" height="100"
        fill="green" stroke="blue" stroke-width="2"
        onmousedown="evt.target.width.baseVal.value += 10"
        onmouseup="evt.target.height.baseVal.value += 10"
        onmousemove="evt.target.height.baseVal.value = 100;
                     evt.target.width.baseVal.value = 100">
    <set attributeName="width" to="50" begin="mouseout" />
    <set attributeName="height" to="50" begin="mouseout" />
  </rect>
</svg>
```

Bei jedem Runterdrücken der Taste des Zeigergerätes wird bei Skriptinterpretation die Breite des Rechtecks um 10 erhöht, beim Loslassen wird die Höhe um 10 erhöht. Eine Bewegung des Zeigergerätes setzt die Größe zurück auf den Ausgangswert. Allerdings unterbinden die Animationen effektiv eine weitere sichtbare Änderung, sobald das Zeigergerät nicht mehr über dem Rechteck ist, weil die Animation höhere Priorität hat als die Manipulation des DOMs mit dem Skript.

21.2.6 Ereignisbehandlung in SVG tiny 1.2

Während in SVG 1.1 die möglichen Ereignisse wie beschrieben explizit in SVG festgelegt ist, wurde in SVG tiny 1.2 eine etwas elegantere Methode eingeführt. Diese ermöglicht es, XML-Ereignisse zu verwenden, statt fester Attribute.

Zusätzlich gibt es das Element ***handler*** (zu deutsch etwa: die Anwendung), welches ebenfalls das auszuführende Skript enthält. ***handler*** ist dann das Kindelement jenes Elementes, welches der Empfänger eines Ereignisses ist.

Eine Anwendung kann auch mehrfach verwendet werden. Dazu wird ein Empfänger notiert, dies ist das Element aus dem Namensraum von XML Events ***listener***. Dies enthält eine Angabe des zu registrierenden Ereignisses und eine Angabe der auszuführenden Anwendung.

Der Namensraum von XML Events ist:

```
http://www.w3.org/2001/xml-events
```

Typisch wird dazu im Hauptelement ***svg*** der Namensraum wie folgt notiert:

```
xmlns:ev="http://www.w3.org/2001/xml-events"
```

Elemente und Attribute aus diesem Namensraum haben dann das Präfix 'ev'.

Die Liste der verfügbaren Ereignisse ist im Kapitel über Interaktivität angeführt, sie deckt sich inhaltlich in größeren Teilen mit der Liste für SVG 1.1.

Die Ereignisse sind *DOMFocusIn*, *DOMFocusOut*, *DOMactivate*, *click*, *mousedown*, *mouseup*, *mouseover*, *mousemove*, *mouseout*, *mousewheel*, *textInput*, *keydown*, *keyup* und *load*.

Zusätzlich sind Dokumentereignisse nutzbar. Dieses sind *resize*, *scroll*, *SVGZoom*, *SVGRotate*.

Eine genauere Beschreibung des jeweiligen Ereignisses ist im Kapitel Interaktivität zu finden.

Ferner gibt es Attribute bezogen auf Animationselemente. Dies sind: *beginEvent*, *endEvent*, *repeatEvent*.

Dazu gibt es Ereignisse bezüglich des Ladens externer Ressourcen: *loadstart*, *progress*, *loadend*.

Ferner gibt es noch den *SVGTimer*.

21.2.7 Anwendung, Element *handler*

Squiggle (Batik)	-
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Das Element ***handler*** ähnelt dem Element ***script***. Entweder beinhaltet das Element selbst ein Skript oder referenziert es.

Anders als bei ***script*** wird das Skript allerdings gegebenenfalls nur ausgeführt, wenn das entsprechende Ereignis eingetreten ist, nicht bereits, wenn das Element geladen ist. Die Funktionalität entspricht also der Ereignisbehandlung in SVG 1.1 mit den entsprechenden Attributen.

Das bereits erwähnte Attribut *type* dient dazu, den Inhaltstype für die Skriptsprache anzugeben.

Attribut *href* von XLink

Mit dem Attribut *href* von XLink kann ein externes Skript referenziert werden. Ist dies Attribut angegeben, so wird der Inhalt des Elementes nicht interpretiert.

Für dieses Element ist *href* nicht animierbar.

Mit *href* sind auch die anderen Attribute von XLink verfügbar.

Attribut *event* von XML Events

Mit dem Attribut *event* von XML Events wird angegeben, welches Ereignis die Anwendung auslösen soll.

Das Attribut gehört zum Namensraum

```
http://www.w3.org/2001/xml-events
```

Dieser Namensraum muss dann auch korrekt angegeben werden.

Die Liste der möglichen Ereignisse wurde bereits angegeben.

Das Attribut ist nicht animierbar.

Beispiel zum Element *handler*

Folgendes Beispiel entspricht dem zur Ereignisbehandlung in SVG 1.1.

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:ev="http://www.w3.org/2001/xml-events"
     version="1.2" baseProfile="tiny"
```

```
    viewBox="0 0 1000 1000"
    xml:lang="de">

  <desc>Beispiel Element handler</desc>

  <rect id="Rechteck" x="20" y="20" width="100" height="100"
        fill="green" stroke="blue" stroke-width="2">
    <handler type="application/ecmascript" ev:event="mousedown">
      var Rechteck = evt.target;
      var width = Rechteck.getFloatTrait("width");
      Rechteck.setFloatTrait("width", width + 10);
    </handler>
    <handler type="application/ecmascript" ev:event="mouseup">
      var Rechteck = evt.target;
      var height = Rechteck.getFloatTrait("height");
      Rechteck.setFloatTrait("height", height + 10);
    </handler>
    <handler type="application/ecmascript" ev:event="mousemove">
      var Rechteck = evt.target;
      var width = Rechteck.getFloatTrait("width");
      var height = Rechteck.getFloatTrait("height");
      Rechteck.setFloatTrait("width", 100);
      Rechteck.setFloatTrait("height", 100);
    </handler>
     <set attributeName="width" to="50" begin="mouseout" />
     <set attributeName="height" to="50" begin="mouseout" />
  </rect>

</svg>
```

21.2.8 Empfänger, Element *listener* von XML Events

Squiggle (Batik)	-
Opera (Presto)	9.5
Firefox (Gecko; auch SeaMonkey, Iceape, Iceweasel etc)	-
Konqueror (KSVG)	-
Safari (Webkit)	-
Chrome (Webkit)	
Microsoft Internet Explorer (Trident)	
librsvg	

Mit dem Element ***listener*** von XML Events kann ein Empfänger von Ereignissen notiert werden, welcher dann die Ausführung von Skripten auslösen kann, wenn das angegeben Ereignis eintritt.

Gegenüber dem Element ***handler*** ist ***listener*** eine Ergänzung für kompliziertere Konstruktionen.

Das Element gehört zum Namensraum

```
http://www.w3.org/2001/xml-events
```

Dieser Namensraum muss dann auch korrekt angegeben werden.

Attribut *event*

Mit dem Attribut wird ein Ereignis angegeben, auf welches der Empfänger warten soll. Der Wert ist ein Ereignis wie bereits beschrieben.

Das Attribut ist nicht animierbar.

Attribut *observer*

Mit dem Attribut kann angegeben werden, welches Element als der Empfänger des zu erwartenden Ereignisses registriert wird. Der Wert ist ein Fragmentidentifizierer im aktuellen Dokument.

Ist das Attribut nicht angegeben, so ist das Elternelement vom ***listener*** der Empfänger.

Das Attribut ist nicht animierbar.

Attribut *target*

Das Attribut gibt den Fragmentidentifizierer des Zielelementes des Ereignisses an, wo also das Ereignis wirklich stattfindet. Der Wert ist ein Fragmentidentifizierer im aktuellen Dokument.

Ferner ist das Zielelement immer entweder das Element selbst, welches mit *observer* identifiziert wird oder ein Kindelement davon. Damit ist es also möglich, ein Kindelement innerhalb eines Gruppierungselementes anzusprechen, wobei das Gruppierungselement der Empfänger ist und das Kindelement das Zielelement des Ereignisses. Exakt das angegebene Element ist dann das Zielelement, nicht etwa ein Kindelement desselben.

Das Attribut ist nicht animierbar.

Beispiel:

```
<g id="g">
<ev:listener event="click" observer="g"
     target="t" handler="#Anwendung"
 xmlns:ev="http://www.w3.org/2001/xml-events"/>
<handler id="Anwendung" type="application/ecmascript">
 var t = evt.target;
 t.setFloatTrait("y", 80);
</handler>
<text x="20" y="20" id="t">Hallo <tspan id="s">Welt</tspan>!</text>
</g>
```

Nur wenn 'Hallo ' oder '!' angeklickert wird, wird das Ereignis ausgelöst, nicht wenn 'Welt' angeklickert wird, weil das Zielelement des Ereignisses dann nicht #t ist, sondern #s.

Attribut *handler*

Das Attribut referenziert eine Anwendung, einen ***handler***, welcher beim Eintritt des Ereignisses gegebenenfalls ausgeführt wird.

Der Wert ist eine IRI einer Anwendung.

Das Attribut ist nicht animierbar.

Attribut *phase*

Das Attribut setzt die Einfangphase für das Ereignis. Der einzige Wert ist 'default', was auch die Voreinstellung ist, daher muss das Attribut nicht explizit notiert werden.

Das Attribut ist nicht animierbar.

Attribut *propagate*

Das Attribut gibt an, ob Ereignisse nach dem Eintritt weitergeleitet werden sollen oder nicht, also ob Einfang und Blubbern eingeleitet werden sollen.

Der Wert des Attributes ist entweder 'stop' oder 'continue'.

Mit 'stop' wird das Ereignis nicht weitergeleitet, das Ereignis blubbert also nicht weiter zu den Elternelementen. Mit 'continue' wird das Ereignis gegebenenfalls weitergeleitet, sofern es blubbert und wird auch von den Elternelementen registiert.

Die Voreinstellung ist 'continue'.

Das Attribut ist nicht animierbar.

Attribut *defaultAction*

Das Attribut gibt an, ob nach der Skriptausführung die voreingestellte Aktion ausgeführt wird oder nicht. Bei einem Element ***a*** ist beispielsweise beim Anklickern die voreingestellte Aktion der Aufruf des Verweiszieles.

Der Wert des Attributes ist entweder 'cancel' oder 'perform'.

Mit 'cancel' wird die voreingestellte Aktion unterbunden. Mit 'perform' findet die voreingestellte Aktion nach der Skriptausführung statt.

Voreinstellung ist 'perform'.

Nicht alle Aktionen können unterbunden werden. Können sie nicht unterbunden werden, wird das Attribut ignoriert. Im Kapitel Interaktivität ist etwa angegeben, welche Ereignisse unterbrochen/unterbunden werden können.

Das Attribut ist nicht animierbar.

Beispiel für Element *listener*

Ist Skriptinterpretation für ecmascript verfügbar, so springt das jeweilige Quadrat ein Stück weiter nach links unten, wenn sich das Zeigergerät ein bestimmtes Ereignis auslöst. Beim grünen Quadrat ist es 'mouseover', beim gelben 'mousewheel', beim magenta 'mousemove'. Wird der untere oder der rechte Rand des Dokumentes nahezu erreicht, springt das Quadrat nach oben, beziehungsweise links.

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.2" baseProfile="tiny"
     xmlns:ev="http://www.w3.org/2001/xml-events"
     viewBox="0 0 1200 1000"
     xml:lang="de">

<title>Beispiel Elemente handler und listener</title>
<defs>
  <ev:listener event="mouseover" observer="rect1" handler="#Anwendung"/>
  <ev:listener event="mousewheel" observer="rect2" handler="#Anwendung"/>
  <ev:listener event="mousemove" observer="rect3" handler="#Anwendung"/>

  <handler id="Anwendung" type="application/ecmascript">
    var rect = evt.target;
    var x = rect.getFloatTrait("x");
```

```
    if (x &gt;= 1090) {
    x=x-1090;
    }
    var y = rect.getFloatTrait("y");
    if (y &gt;= 890) {
    y=y-890;
    }
    rect.setFloatTrait("x", (x+10));
    rect.setFloatTrait("y", (y+10));
  </handler>
</defs>

  <rect width="1200" height="1000" fill="#ddd"/>
  <rect id="rect1" x="10" y="10" width="100" height="100"
        fill="#080" stroke="blue" stroke-width="2"/>
  <rect id="rect2" x="215" y="10" width="100" height="100"
        fill="#ff0" stroke="blue" stroke-width="2"/>
  <rect id="rect3" x="420" y="10" width="100" height="100"
        fill="#a0f" stroke="blue" stroke-width="2"/>

</svg>
```

21.2.9 Weitere Beispiele

Aufgrund der bereits diskutierten Sicherheitsprobleme ist es bei Wikibooks unterbunden, Dokumente hochzuladen, die Skripte enthalten, daher sind die Beispiele nicht direkt ausführbar, sondern nur als Quelltext verfügbar. Nach einer entsprechenden Prüfung, kann der Nutzer diese in ein SVG-Dokument kopieren und ausprobieren, sofern bei der Prüfung keine Bedenken aufgekommen sind und ein Darstellungsprogramm zur Verfügung steht, bei der die verwendete Skriptsprache interpretiert wird.

drag 'n' drop

Bei aktivierter Skriptinterpretation der Sprache ecmascript können die Kreise verschoben werden. Durch Drücken der Taste des Zeigerelementes und Ziehen kann jeweils ein Kreis verschoben werden. Beim Loslassen verbleibt der Kreis an der letzten Position:

```
<?xml version="1.0"?>
 <svg version="1.1"
      width="600px" height="400px"
      xmlns="http://www.w3.org/2000/svg">
       <title>drag 'n' drop</title>
       <defs>
        <script type="application/ecmascript" >
          <![CDATA[
          function dnd(evt)
          {
            ziel = evt.target
            dx = ziel.cx.baseVal.value - evt.clientX;
            dy = ziel.cy.baseVal.value - evt.clientY;
            document.addEventListener("mousemove", mousemove_listener, true);
            document.addEventListener("mouseup", mouseup_listener, true);
          }

          function mouseup_listener(evt)
          {
            document.removeEventListener("mousemove", mousemove_listener, true);
            document.removeEventListener("mouseup", mouseup_listener, true);
          }
```

```
          function mousemove_listener(evt)
          {
            var id = ziel.ownerSVGElement.suspendRedraw(1000);
            ziel.cx.baseVal.value = evt.clientX + dx;
            ziel.cy.baseVal.value = evt.clientY + dy;
            ziel.ownerSVGElement.unsuspendRedraw(id);
          }

          ]]>
        </script>
        </defs>
          <polyline points="0,0 600,0 600,400 0,400 0,0"
           stroke="black" fill="DarkOrange"/>
          <circle id="circ" r="1cm" cx="5cm" cy="3cm"
           fill="SkyBlue" stroke="PaleTurquoise" stroke-width="5"
           onmousedown="dnd(evt)"/>
          <circle id="circ2" r="2cm" cx="3cm" cy="4cm"
           fill="Linen" stroke="SandyBrown" stroke-width="5" fill-opacity="0.5"
           onmousedown="dnd(evt)"/>
  </svg>
```

Skriptanwendung, um die Orientierung von Markierungen zu erläutern

Ein Beispiel mit ecmascript, um die Funktion von *orient*="auto" zu erläutern:

Bei aktivierter Skriptinterpretation folgt das Linienende mit einer pfeilartigen Markierung der Bewegung der Maus. Bei mousedown wird der mit einem roten Quadrat markierte Anfang der Linie auf die aktuelle Position gesetzt.

Die Spitze des Pfeiles zeigt immer in Richtung der Linie. Das Quadrat ist immer entlang der x- und y-Achse ausgerichtet.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg version="1.1" width="400" height="300"
     xmlns="http://www.w3.org/2000/svg"
     onmousemove="set(evt,'2')" onmousedown="set(evt,'1')"
     xml:lang="de">
<title>Skriptanwendung und Markierungsorientierung</title>
<defs>
<script type="application/ecmascript">

function set(evt,num)
{
var linie = document.getElementById("lin");
linie.setAttribute("x"+num, evt.clientX);
linie.setAttribute("y"+num, evt.clientY);
}

</script>

 <marker id="start" markerWidth="20" markerHeight="20" refX="5" refY="5">
  <rect fill="red"  width="10" height="10" />
 </marker>
 <marker id="end" markerWidth="10" markerHeight="10" refX="10" refY="5"
 orient="auto">
  <polygon points="0,0,10,5,0,10,0,0" fill="black"/>
 </marker>
</defs>
```

```
<line id="lin" x1="5" y1="50" x2="50" y2="100" stroke="green"
      marker-end="url(#end)" marker-start="url(#start)" />
</svg>
```

Vergleich deklarative Animation und Skript

Sofern die Interpretation von Skripten vom Typ emcmascript verfügbar ist, ergibt sich bei folgendem Beispiel ein Vergleich einer schrittweisen Änderung von Attributen und Eigenschaften mit deklarativer Animation und Skript.

Die violett gefüllten Objekte werden per Skript animiert, die grünen per deklarativer Animation.

Unmittelbar sichtbar ist, dass die deklarative Animation für diese Anwendung deutlich einfacher und präziser ist als das Skript. Eine kontinuierliche deklarative Animation ist noch einfacher zu notieren, wie an der Gruppe mit dem cyanen Strich zu sehen ist, während dies mit einem Skript nicht durchführbar ist. Die zeitlichen Abweichungen liegen daran, dass das Skript Verarbeitungszeit benötigt, somit also die Aktion insgesamt die veranschlagten 10s und die Verarbeitungszeit dauert. Die Verarbeitungszeit ist indes vom Darstellungsprogramm abhängig, lässt sich also nicht so einfach berücksichtigen. Bei der deklarativen Animation berücksichtigt das Darstellungsprogramm die Verarbeitungszeit hingegen automatisch.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg viewBox="-200 -200 400 400"
     xmlns="http://www.w3.org/2000/svg"
     onload="StartAnimation(evt)" version="1.1">
<title>Vergleich deklarative Animation und Skript</title>

<defs>
  <script type="application/ecmascript"><![CDATA[
    //Zeiten in Millisekunden
    var Zeit = 0;
    var Zeitschritt = 100;
    var Dauer = 10000;

    var Gruppe;

    function StartAnimation(evt) {
      Gruppe = evt.target.ownerDocument.getElementById("g");
      Skalieren();
    }
    function Skalieren() {
      // Gruppe schrittweise vergrößern, bis es zehnfach vergrößert ist
      Skalierung = (Zeit * 10.) / Dauer;
      Gruppe.setAttribute("transform", "scale(" + Skalierung + ")");
      // Durchsichtigkeit ändern
      Durchsichtigkeit = Zeit / Dauer;
      Gruppe.setAttribute("fill-opacity", Durchsichtigkeit);
      //Zeit vergrößern
      Zeit = Zeit + Zeitschritt;
      //nach maximaler Zeit beenden
      if (Zeit > Dauer)
        return;
      // Skalieren nochmal aufrufen, Zeitschritt später.
      setTimeout("Skalieren()", Zeitschritt)
    }
  ]]></script>
</defs>
```

```
  <rect x="-200" y="-200" width="400" height="400"
        fill="#eee" stroke="black"/>
  <g id="g" fill="#a0f" stroke="#ff0">
   <desc>Gegebenenfalls per Skript manipulierte Gruppe</desc>
   <circle r="9" cx="-10" cy="-10" />
   <rect x="1" y="-19" width="18" height="18" rx="1" />
  </g>

   <g fill="#062" stroke="#00a">
    <desc>Vergleich diskrete deklarative Animation</desc>
    <animate attributeName="fill-opacity" calcMode="discrete" fill="freeze"
      values="0;0.01" dur="0.1s" repeatDur="10s" accumulate="sum" />
    <animateTransform attributeName="transform" type="scale" fill="freeze"
      calcMode="discrete" values="0;0.1" dur="0.1s" repeatDur="10s"
 accumulate="sum"/>
   <circle r="9" cx="10" cy="10" />
   <rect x="-19" y="1" width="18" height="18" rx="1" />
  </g>

  <g fill="none" stroke="#0aa" stroke-width="0.2">
    <desc>Vergleich kontinuierliche deklarative Animation</desc>
    <animate attributeName="stroke-opacity" fill="freeze" values="0;1" dur="10s"
 />
    <animateTransform attributeName="transform" type="scale" fill="freeze"
      values="0;10" dur="10s"/>
   <circle r="9" cx="10" cy="10" />
   <rect x="-19" y="1" width="18" height="18" rx="1" />
  </g>

</svg>
```

switch und *script*

Einfaches Beispiel, welches switch, Merkmalszeichenkette (featureString) und ecma-script verwendet. Die Anzeige des Dokumentes hängt ab von einer feature-Zeichenkette und von ecma-script.

Erster Fall: Merkmalszeichenkette gibt wahr für script zurück und Skriptausführung ist aktiv - der Nutzer erhält eine Bitte, Skriptausführung abzuschalten, um mit dem Test fortzufahren.

Zweiter Fall: Merkmalszeichenkette gibt wahr für script zurück und Skriptausführung ist deaktiviert - der Nutzer bekommt eine Information über einen Fehler. Das Dokument enthält nur ecma-script. Falls das Darstellungsprogramm dies nicht interpretieren kann, sollte es falsch für die Merkmalszeichenkette und für dieses Dokument zurückgeben.

Dritter Fall: Merkmalszeichenkette gibt falsch für script zurück und Skriptausführung ist trotzdem aktiv - der Nutzer sieht eine Information über den Fehler - Skriptausführung sollte generell nicht verfügbar sein, wenn die Merkmalszeichenkette falsch zurückgibt.

Vierter Fall: Merkmalszeichenkette gibt falsch für script zurück und Skriptausführung ist deaktiviert - der Nutzer sieht einen Alternativtext. Zu beachten ist, dass die Merkmalszeichenkette eine Information darüber angibt, ob ein Merkmal unterstützt wird oder nicht, nicht ob es generell und irgendwelchen Umständen möglich ist, dass ein Merkmal unterstützt wird, in diesem Falle nur, ob Skriptausführung aktiviert ist. Falls die Merkmalszeichenketten irgendeinen Nutzen für Autoren haben sollen, muss der zurückgegebene Wert der Merkmalszeichenkette von den spezifischen aktuellen Bedingungen des entsprechenden Dokumentes und von den Voreinstellungen des Nutzers abhängen, nicht von einer generellen philosophischen Stellungnahme des Darstellungsprogrammes.

Das ausgeführte Skript in diesem Dokument ist eine Änderung des Präsentationsattributes display auf none beziehungsweise auf block, um dem Nutzer zusätzliche Informationen zur Skriptausführung zu geben.

Bei SVG tiny 1.2 kann zusätzlich noch eine bedingte Verarbeitung davon abhängig gemacht werden, ob *requiredFormats*="application/ecmascript" zutriff oder nicht, um herauszufinden, welche Skriptsprache verfügbar ist, wenn die Merkmalszeichenkette zu erkennen gibt, dass überhaupt eine Skriptinterpretation gegeben ist.

```
<?xml version="1.0" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg viewBox="0 0 1000 600"
    xmlns="http://www.w3.org/2000/svg" version="1.1"
    xml:lang="de"
    onload="display_bsp()">
<title> switch und script </title>
<defs>
  <script type="application/ecmascript">

    function display_bsp() {
      var bsp0 = document.getElementById("bsp0");
      bsp0.setAttributeNS(null,"display", 'none');
      var bsp1 = document.getElementById("bsp1");
      bsp1.setAttributeNS(null,"display", 'none');
      var bsp2 = document.getElementById("bsp2");
      bsp2.setAttributeNS(null,"display", 'none');
      var bsp3 = document.getElementById("bsp3");
      bsp3.setAttributeNS(null,"display", 'block');
      var bsp4 = document.getElementById("bsp4");
      bsp4.setAttributeNS(null,"display", 'block');
      var bsp5 = document.getElementById("bsp5");
      bsp5.setAttributeNS(null,"display", 'block');
    }
  </script>
</defs>
<switch>
   <g requiredFeatures="http://www.w3.org/TR/SVG11/feature#Script"
   fill="red" font-size="20" font-family="sans-serif">
      <text x="20" y="100" id="bsp0"
            display="block">Nur sichtbar, falls Merkmalszeichenkette für script wahr
 ist,</text>
      <text x="20" y="200" id="bsp1"
            display="block">aber kein ecmascript interpretiert wird!</text>
      <text x="20" y="300" id="bsp2"
            display="block">Allerdings enthält das Dokument keine andere
 Skriptsprache als ecmascript.</text>
      <text x="20" y="400" id="bsp3" fill="gray"
            display="none">Skriptinterpretation aktiv!</text>
      <text x="20" y="500" id="bsp4" fill="black"
            display="none">Bitte Skriptinterpretation deaktivieren und neuladen, um
 mit dem Test fortzufahren!</text>
   </g>
   <g font-size="30" font-family="sans-serif" fill="#440">
      <text x="20" y="70" font-size="50">Alternativtext</text>
      <text x="20" y="140" font-size="20">Sichtbar, wenn keine
 Skriptinterpretation verfügbar ist.</text>
      <text x="20" y="210" font-size="20" fill="#66f"
            display="block">Bei Bedarf Skriptinterpretation aktivieren und neuladen,
 um mit dem Test fortzufahren.</text>

      <text x="20" y="500" font-size="30" fill="red" display="none"
            id="bsp5">Skriptinterpretation trotzdem aktiv? Fehler!</text>
  </g>
```

```
</switch>
</svg>
```

21.3 Weiterführende Literatur

- java-script und DOM bei selfhtml[2]
- carto.net: Manipulating SVG Documents using the DOM and ECMAScript[3] (en)
- carto.net: Event Handling using the DOM and ECMAScript[4] (en)
- XML Events[5] (en)
- SVG tiny 1.2 Mikro-DOM[6] (en)
- Einbindung von ecmascript in SVG tiny 1.2[7] (en)
- Einbindung von Java in SVG tiny 1.2[8] (en)
- Einbindung von Perl in SVG tiny 1.2[9] (en)
- Einbindung von Python in SVG tiny 1.2[10] (en)
- SVG 1.1 DOM[11] (en)
- Einbindung von ecmascript in SVG 1.1[12] (en)
- Einbindung von Java in SVG 1.1[13] (en)

2 http://de.selfhtml.org/javascript/index.htm
3 http://www.carto.net/papers/svg/manipulating_svg_with_dom_ecmascript/
4 http://www.carto.net/papers/svg/eventhandling/
5 http://www.w3.org/TR/xml-events/
6 http://www.w3.org/TR/SVGTiny12/svgudom.html
7 http://www.w3.org/TR/SVGTiny12/ecmascript-binding.html
8 http://www.w3.org/TR/SVGTiny12/java-binding.html
9 http://www.w3.org/TR/SVGTiny12/perl-binding.html
10 http://www.w3.org/TR/SVGTiny12/python-binding.html
11 http://www.w3.org/TR/SVG11/svgdom.html
12 http://www.w3.org/TR/SVG11/ecmascript-binding.html
13 http://www.w3.org/TR/SVG11/java.html

22 SVG in (X)HTML einbetten

22.1 Verwendung von SVG im Internet

SVG ist als Format für die Verwendung im Internet konzipiert. Der einfachste Fall ist die Verwendung als eigenständige Dateien. Jedes Format, welches auf andere Dokumente verweisen kann (hyperlink), kann auch auf eigenständige SVG-Dateien verweisen. Sofern in dem jeweiligen Format das Pseudoprotokoll 'data:' verwendet werden kann, können zudem damit wie auch in SVG selbst externe Dateien direkt in der URI/IRI des Verweises notiert werden, somit auch SVG-Dateien.

Daneben gibt es zahlreiche weitere Möglichkeiten, wie SVG-Dokumente innerhalb von anderen Formaten als eigenständige Dateien eingebettet werden können oder auch als Dokumentfragmente in einem XML-Dokument mit anderen XML-Formaten gemischt werden können.

22.2 SVG als Fragment in XHTML

SVG kann als XML-Format zusammen mit anderen XML-Formaten gemeinsam in einem Dokument verwendet werden.

Da auch XHTML ein XML-Format ist, geht dies auch dort. Das X in XML und XHTML steht für eXtensible, weil es erweiterbar ist. Ein XHTML-Dokument kann also insbesondere mittels eines Fragmentes aus dem Namensraum SVG erweitert werden.

Beispiel - ein Dokument in der aktuellen XHTML-Version XHTML+RDFa 1.0 mit eingebettetem SVG:

```
<?xml version="1.0" encoding="utf-8"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      version="XHTML+RDFa 1.0"
      xml:lang="de">
<head>
<title>Freude schöner Götterfunken...</title>
</head>
  <body>
    <h1>Freude schöner Götterfunken...</h1>
     <div class="svg">
     <svg xmlns="http://www.w3.org/2000/svg"
          xmlns:xlink="http://www.w3.org/1999/xlink"
          version="1.1" width="810" height="540">
       <title>Sterne als Götterfunken</title>
       <defs>
             <g id="s">
                   <g id="c">
                     <path id="t" d="M0,0v1h0.5z" transform="translate(0,-1)rotate(18)"/>
                         <use xlink:href="#t" transform="scale(-1,1)"/>
                    </g>
                        <g id="a">
                          <use xlink:href="#c" transform="rotate(72)"/>
```

```
                    <use xlink:href="#c" transform="rotate(144)"/>
                  </g>
                  <use xlink:href="#a" transform="scale(-1,1)"/>
                </g>
              </defs>
      <rect fill="#039" width="810" height="540"/>
              <g fill="#fc0" transform="scale(30)translate(13.5,9)">
              <use xlink:href="#s" y="-6"/>
              <use xlink:href="#s" y="6"/>
              <g id="l">
                <use xlink:href="#s" x="-6"/>
                <use xlink:href="#s" transform="rotate(150)translate(0,6)rotate(66)"/>
                <use xlink:href="#s" transform="rotate(120)translate(0,6)rotate(24)"/>
                <use xlink:href="#s" transform="rotate(60)translate(0,6)rotate(12)"/>
                <use xlink:href="#s" transform="rotate(30)translate(0,6)rotate(42)"/>
              </g>
              <use xlink:href="#l" transform="scale(-1,1)"/>
            </g>
    </svg>
    </div>
    <p>
    Ein Text, welcher anderen XHTML-Inhalt repräsentiert.
    </p>
  </body>
</html>
```

Die Idee ist ebenso einfach wie brillant - durch Angabe der Namensräume für XHTML, XLink und SVG werden die jeweiligen Elemente, Kindelemente und Attribute den jeweiligen Namensräumen zugeordnet, so dass das Darstellungsprogramm diese eindeutig identifizieren und korrekt darstellen kann. Bei der obigen XHTML-Version kann sogar die Semantik von Inhalten durch Verwendung eigener Namensräume genauer festgelegt werden. Dazu werden die gleichen Attribute von RDFa verwendet wie in SVG tiny 1.2, die beiden Formatversionen harmonieren also bereits hervorragend, um inhaltlich qualitativ hochwertige Dokumente zu erstellen.

Für den Autor ist die Angelegenheit denkbar einfach, das SVG-Fragment wird in das XHTML-Dokument an eine passende Stelle kopiert und fertig, schon funktioniert es in allen Darstellungsprogrammen, die SVG und XHTML interpretieren. Zu beachten ist beim Kopieren von Fragmenten allerdings immer, dass die Zuordnung zu den Namensräumen erhalten bleiben muss, im Zweifelsfalle sind entsprechende Angabe also zusätzlich zu kopieren oder anzupassen. Bei obigem Beispiel stehen alle für das SVG-Fragment relevanten Angaben allerdings bereits im ***svg***-Element und die für XHTML im Element ***html***.

In XHTML sollten die direkten Kindelemente von ***body*** immer Blockelemente sein, weswegen hier noch ein ***div*** um das SVG-Fragment gelegt ist. Durch die Verwendung von *class* ergibt sich ferner die einfache Möglichkeit der Positionierung und Skalierung mit CSS.

Ein Problem ergibt sich allenfalls, weil der immer noch recht häufig verwendete internet-explorer von microsoft bis Version 8 weder XHTML noch SVG interpretieren kann, also mit obiger Lösung auch nichts anfangen kann. Ohne weitere Tricks kann dieses Programm auch das XHTML nicht als HTML interpretieren und das SVG darin an ein plugin durchreichen.

Formal korrekt ist ein solches Dokument immer als application/xhtml+xml oder application/xml auszuliefern. Ohne Angabe eines doctype mag der microsoft-internet-explorer im zweiten Falle das Dokument sogar als XML interpretieren, weiß dann aber nicht, was die Elemente in den Namensräumen von XHTML und SVG zu bedeuten haben, kann sie also auch nicht sinnvoll darstellen.

Es ist teilweise auch üblich, XHTML wie HTML als text/html auszuliefern, das ist dann allerdings eben fehlerhaftes HTML und kein XHTML mehr. Wenn es aber geschickt genug gemacht wird, fallen die Fehler nicht auf, weil der HTML-Markierungssuppen-parser mit kleineren Fehlern ganz gut zurechtkommt und sie vor dem Betrachter verbirgt. Allerdings ist HTML4 nicht erweiterbar, so dass darin kein SVG vorkommen kann. Das derzeit (2008/2012) in Arbeit befindliche HTML5 sieht zwar das Einbetten von MathML und SVG in HTML5 vor, liegt aber erst in einem Arbeitsentwurf vor und ist noch keine Spezifikation. Das ändert allerdings nichts daran, dass Programme, die SVG nicht interpretieren, auch innerhalb von HTML5 nicht von alleine beginnen werden, dies zu tun.

22.3 Einbetten einer SVG-Datei

Damit ein Darstellungsprogramm in (X)HTML selbst entscheiden kann, ob es das SVG selbst interpretiert oder ob wohlmöglich ein plugin verwendet werden kann, empfiehlt sich die Verwendung des Elementes ***object***. Als Inhalt des Elementes kann ein Ersatzbild oder eine Textalternative angegeben werden, für jene Programme, die kein SVG interpretieren. Mit einem direkten Verweis auf die SVG-Datei darin erhält der Betrachter ferner die Chance, sich das SVG mit einem externen Programm anzusehen.

```
<object type="image/svg+xml" data="test.svg" width="400" height="300">
 <p><img src="test.png" width="400" height="300" alt="Textäquivalent zum PNG"
 /></p>
 <p>Offenbar ist kein eingebetteter Betrachter für SVG verfügbar,
    mit einem externen Programm versuchen:
 <a href="test.svg" type="image/svg+xml">Test</a></p>
</object>
```

Wenn beim microsoft-internet-explorer indes ein bestimmtes sogenanntes service-pack2 installiert ist und der Nutzer das SVG-plugin von adobe installiert hat, ist hingegen bekannt, dass der microsoft-internet-explorer die Anzeige des SVG im ***object*** sabotieren wird. Insofern kann es auch sinnvoll sein, den direkten Verweis auf die SVG-Datei außerhalb des ***object*** anzuordnen, denn dann funktioniert das adobe-plugin ohne Einbettung auch im microsoft-internet-explorer.

Es ist natürlich auch möglich, ein SVG als "eigene web-Seite" mit dem ***iframe***-Element einzubetten. Der Inhalt des ***iframe*** kann analog zum ***object*** verwendet werden.

```
<iframe src="test.svg" width="400" height="300" name="imap">
Alternativtext oder -bild
</iframe>
```

Wie im Standard festgelegt, sollte ein SVG auch mit dem Element ***img*** referenziert unterstützt werden, bei älteren Darstellungsprogramm-Versionen kann es damit allerdings Probleme geben. Bei aktuellen funktioniert aus Sicherheitsgründen dann aber kein ecma-script im SVG.

```
<img src="test.svg" width="400" height="300" alt="Textäquivalent zur
 SVG-Testdatei" />
```

Eine weitere Möglichkeit ergibt sich bei Verwendung von CSS als dekoratives Hintergrundbild:

```
body {background: blue url(deko.svg)}
```

Ein Darstellungsprogramm, welches kein SVG interpretiert, wird einfach blauen Hintergrund darstellen, eines mit SVG-Interpretation (als CSS-Hintergrundbild) wird hingegen das SVG auf dem blauen

Hintergrund darstellen. Leider gibt es in CSS keinen Mechanismus, mit dem man ein Ersatzbild in einem anderen Format angeben kann. Allenfalls ließe sich etwas ähnliches durch Verschachtelung von Elementen mit Hintergrundbildern in verschiedenen Formaten erreichen.

22.4 WebSVG: Versuch der Konversion zu flash

Es gibt einige Leute, die sich um das Problem des microsoft-internet-explorer bemühen, obgleich microsoft bereits in der Version 9 selbst eine Interpretation von SVG integriert hat. Die Bemühungen gehen in die Richtung, das SVG für ältere Versionen in ein anderes Format zu konvertieren, um es darstellbar zu machen.

Eine Konversionsversuch gibt es mit WebSVG. Damit wird eine Konversion zu flash versucht.

Ein prinzipielles Problem der Formatkonversion ist neben dem Aufwand das Problem, dass nicht alles, was in SVG möglich ist, sich gut in jedes andere Format konvertieren lässt. Zudem beruht die Idee auf der Hypothese, dass das andere Format eine bessere Verfügbarkeit hat als SVG selbst. Das mag für Formate wie PNG, JPEG/JFIF oder (X)HTML (Textalternative!) zutreffen, in Zukunft vermutlich aber nicht mehr für das bei WebSVG bevorzugte flash.

Zum Testen von WebSVG kann das Paket einfach unter http://code.google.com/p/svgweb/[1] geladen werden.

Auf dem server wird davon das Verzeichnis '/src' benötigt. Das Beispiel von oben sieht mit WebSVG als 'HTML5-Dokument' nach den Angaben der Entwickler von WebSVG so aus (es handelt sich um kein gültiges HTML4-Dokument! Selbst ohne das SVG ist es kein gültiges HTML4-Dokument, nach den aktuellen Entwürfen und Intentionen ist es auch für HTML5 als problematisch anzusehen; wegen philosophischer Probleme der W3C-Arbeitsgruppe für HTML5 ist es auch gar nicht möglich anzugeben, dass es sich um ein 'HTML5-Dokument' handelt):

```
<!DOCTYPE html>
<html>
<head>
<title>Freude schöner Götterfunken...</title>
</head>
  <body>
    <head>
    <meta name="svg.render.forceflash" content="true">
    <script src="./src/svg.js"  data-path="./src/"  data-debug="true"></script>
  </head>
<body>

    <h1>Freude schöner Götterfunken...</h1>
        <script type="image/svg+xml">
     <svg xmlns="http://www.w3.org/2000/svg"
          xmlns:xlink="http://www.w3.org/1999/xlink"
          version="1.1" width="810" height="540">
       <title>Sterne als Götterfunken</title>
       <defs>
             <g id="s">
                   <g id="c">
                     <path id="t" d="M0,0v1h0.5z" transform="translate(0,-1)rotate(18)"/>
                         <use xlink:href="#t" transform="scale(-1,1)"/>
                    </g>
```

1 http://code.google.com/p/svgweb/

```
                    <g id="a">
                      <use xlink:href="#c" transform="rotate(72)"/>
                      <use xlink:href="#c" transform="rotate(144)"/>
                    </g>
                    <use xlink:href="#a" transform="scale(-1,1)"/>
                  </g>
                </defs>
        <rect fill="#039" width="810" height="540"/>
                <g fill="#fc0" transform="scale(30)translate(13.5,9)">
                <use xlink:href="#s" y="-6"/>
                <use xlink:href="#s" y="6"/>
                <g id="l">
                  <use xlink:href="#s" x="-6"/>
                  <use xlink:href="#s" transform="rotate(150)translate(0,6)rotate(66)"/>
                  <use xlink:href="#s" transform="rotate(120)translate(0,6)rotate(24)"/>
                  <use xlink:href="#s" transform="rotate(60)translate(0,6)rotate(12)"/>
                  <use xlink:href="#s" transform="rotate(30)translate(0,6)rotate(42)"/>
                </g>
                <use xlink:href="#l" transform="scale(-1,1)"/>
               </g>
     </svg>
         </script>

     <p>
     Ein Text, welcher anderen XHTML-Inhalt repräsentiert.
     </p>
   </body>
</html>
```

Achtung!: WebSVG funktioniert nicht lokal wegen der Konversion zu flash. Die Dateien müssen also auf einem server liegen, weitere Hinweise gibt es in der WebSVG-Dokumentation.

Unser Beispiel hier verwendet immer flash für die Darstellung, auch wenn das Darstellungsprogramm das SVG selbst besser interpretieren kann oder kar kein plugin für flash verfügbar oder aktiviert hat. So ist die Verwendung von WebSVG also sicher nicht sinnvoll!

Zu beachten ist dabei, dass dies als ein 'HTML5-Dokument' angesehen wird, was es genaugenommen gar nicht gibt. Gemäßt Arbeitsentwurf zu HTML5 ist es nicht notwendig, SVG in einem ***script***-Element zu verstecken (siehe weiter unten), noch ist es an sich notwendig, das mittels Skripten zu interpretieren, das kann gemäß HTML5 auch so funktionieren. Da HTML5 derzeit (2008/2012) nur ein Arbeitsentwurf ist, unterliegt diese Version auch steten Änderungen, eignet sich also nicht für stabile Dokumente.

Diese Transformation in ein flash-Dokument kann zudem bestenfalls funktionieren, wenn beim Betrachter java-script aktiviert ist und die Interpretation von flash aktiviert ist. Es stellt also keine Methode dar, bei der der Autor gut abschätzen kann, was beim Betrachter wirklich angezeigt wird. Auch gibt es Teilbereiche von SVG, die mittels flash nicht nachvollziehbar sind, eignet sich also bestenfalls für eine Teilmenge von SVG-Dokumenten. Die Interpretation von SVG als Skriptsprache innerhalb des ***script***-Elementes ist jedenfalls eine sehr gewagte Interpretation von SVG und HTML, die mindestens semantisch fragwürdig ist, hier aber immerhin andeutet, dass das SVG mehr dekorativen und keinen inhaltlich relevanten Charakter hat, weswegen eine Interpretation auch nur optional zu sein braucht.

Für inhaltlich relevante Graphik ist diese Konstruktion keinesfalls geeignet, auch nicht, wenn der Autor zuverlässig vorhersagen möchte, was sich daraus bei einem Nutzer für eine Anzeige ergibt. Die Konstruktion ist eher als experimentelle Spielerei anzusehen, welche eine SVG-Interpretation im Darstellungsprogramm selbst nicht ersetzen kann.

22.5 SVG in HTML5

Und so ist die Verwendung von SVG in HTML5 ohne Anwendung fragwürdiger Tricks in der HTML-Variante geplant:

```
<!doctype html>
<html>
<head>
<title>SVG eingebettet in text/html</title>
</head>
<body>
<p>
 Ein grüner Kreis:
 <svg> <circle r="50" cx="50" cy="50" fill="green"/> </svg>
</p>
</body>
</html>
```

Dies ist für Autoren eine auf den ersten Blick recht einfache Konstruktion. Allerdings werden Angaben zu Namensräumen innerhalb von HTML5 (genauer der HTML-Variante davon) nicht interpretiert. Aufgrund der Komplexität eines HTML5-Markierungssuppen-parsers und der fehlenden Zuordnung zu den Namensräumen unterliegt die Verwendung von SVG und MathML in HTML5 jedoch gewissen Einschränkungen. Elemente mit gleichem Namen in HTML5 und SVG (oder MathML) können problematisch sein, auch die Verwendung von Elementen aus anderen Namensräumen im SVG-Fragment.

Mögliche Fehler im SVG-Fragment können ferner vom HTML5-parser vertuscht werden. Sollte der Autor das Fragment später kopieren wollen, um es in anderen (XML-)Dokumenten zu verwenden, können die vertuschten Fehler zu Problemen führen. Allerdings ist für HTML5-parser, welche auch solches 'HTML5-SVG' interpretieren, vorgesehen, dass diese eine Ausgabe anbieten müssen, welche das korrigierte SVG zur Weiterverwendung darstellt. Diese korrigierte Ausgabe sollte dann weiterverwendet werden, nicht der originale, möglicherweise fehlerhafte Quelltext des HTML5-Originaldokumentes.

Zudem ist es auch bei HTML5 keine Voraussetzung, dass ein HTML5-parser SVG oder MathML interpretieren können muss. Es ist lediglich formal erstmals für eine HTML-Version überhaupt möglich, MathML oder SVG darin einzubetten.

Aufgrund der in HTML5 reichlich vorhandenen Altlasten ist die Kombination in der HTML-Variante problematisch und die Sprache selbst ist nur schwierig allgemein durch den Autor erweiterbar. Allerdings gibt es auch zu HTML5 eine XML-Variante, bei der die direkte Kombination mit Elementen aus verschiedenen Namensräumen problemlos funktioniert, weil dort die Angabe von Namensräumen problemlos möglich ist.

22.6 Als Java-Anwendung

Die Apache-Organisation bietet mit Batik/Squiggle eine Java-Anwendung an, mit der ebenfalls SVG dargestellt werden kann oder eine Konversion von SVG in Pixelgraphik erfolgen kann, sofern das inhaltlich möglich ist. Batik/Squiggle interpretiert bereits größere Teilbereiche von SVG 1.1, hat aber ebenso wie andere Programme teils noch größere Lücken oder Fehler in der Interpretation.

Es gibt auch andere Anbieter, die SVG-Darstellungsprogramme auf Basis von Java anbieten, insbesondere auch für Mobiltelephone.

22.7 Konfiguration eines web-servers anhand von Apache

Ein prinzipielles Problem von Dateiformaten ist, wie dem Darstellungsprogramm vermittelt wird, um welches Format es sich handelt.

Das Problem ist größtenteils historisch bedingt und resultiert letztlich daraus, dass Dateisysteme meist nicht über die Möglichkeit verfügen, standardisiert vielfältige Metainformationen einer jeden Datei zuzuordnen.

In den Anfängen der Computer gab es auch nur wenige Formate und kleine Speichermedien, so dass es dem Nutzer zumutbar war, die Zuordnung zu kennen. Teils haben auch die Programme versucht, das Format anhand der ersten paar Bytes einer Datei zu raten.

Ein anderer Ansatz bestand darin, eine Abkürzung für das Format am Ende des Dateinamen zu notieren, bevorzugt mit einem Punkt separiert, auch Dateiendung genannt. Teils werden mehrere Endungen angehängt. Mit der Zeit gibt es allerdings sehr viele Formate und keine Eindeutigkeit der Abkürzungen mehr, auch weil es nie eine zentrale Registrierung für solche Abkürzungen gab. Auch heute noch basiert die Formaterkennung vieler Programme auf dieser unzuverlässigen und fehleranfälligen Methode.

Mit dem Aufkommen des internets und von emails hat sich das Problem verschärft, weil nun auch Dateien zwischen verschiedenen Rechnern ausgestauscht wurden. Das hat dazu geführt, dass für ausgereifte angesehene Formate eindeutigen Kennzeichnungen registriert wurden, ursprünglich MIME-Typen genannt, heute allgemeiner Inhaltstypen. Ist einem email-Programm oder einem web-server also das Format einer Datei und der Inhaltstyp bekannt, kann diese Information dem Empfänger der Datei als Metainformation vermittelt werden, bevor die Datei gesendet wird, eventuell kann auch ausgehandelt werden, welches Format der Empfänger interpretieren kann, wenn mehrere Formate verfügbar sind - oder bei emails, welches Programm zur Darstellung eines Anhangs geeignet sein könnte. So wird nur Metainformation über das Format ausgetauscht und nicht notwendig gleich große Dateien, um die Interpretierbarkeit festzustellen.

Jedenfalls verbleibt das Problem der Zuordnung des Formates auf dem Rechner mit dem web-server oder dem email-Programm.

Im Falle des web-servers Apache erfolgt die Zuordnung zumeist ebenfalls über Dateiendungen, denen in Listen Kennzeichnungen zugeordnet werden. Damit kann das Programm die entsprechen Metainformation senden, die dann für das Programm relevant ist, welches die Datei angefordert hat.

Bei SVG ist es nun so, dass die Registrierung der Kennzeichnung seit einigen Jahren läuft und wohl auch in näherer Zukunft (Stand 2010) endgültig zum Abschluss kommt. Die Kennzeichnung image/svg+xml ist allerdings bereits allgemein gebräuchlich. Als Dateiendung wird '.svg' empfohlen. Zusätzlich gibt es die Möglichkeit, SVG mittels gzip zu komprimieren, für solche Dateien wird die Endung '.svgz' empfohlen. Zusätzlich zur Kennzeichnung des Formates ist dann als Metainformation allerdings noch anzugeben, dass eine solche Kompression vorliegt.

Bei dynamisch zum Beispiel per PHP erzeugten SVG-Ausgaben kann es allerdings sinnvoll sein, für solche Skripte spezielle Endungen einzuführen, um ebenfalls zu gewährleisten, dass automatisch

korrekte Metainformationen gesendet werden. Andernfalls sind die Metainformationen bei jeder einzelnen Datei manuell per PHP zu senden.

Bei aktuellen Versionen vom Apache sollten die Zuordnung des Typs für die Endungen '.svg' und '.svgz' bereits korrekt vorliegen.

Nun kann es sein, dass eine ältere Version vom Apache verwendet werden muss oder die Konfiguration nicht geändert werden kann, darf oder soll.

In solchen Fällen kann dem Apache die Zurordnung mit einer Datei namens '.htaccess' definiert werden. Diese Datei wird in dem Verzeichnis untergebracht, für welches die Zuordnung einschließlich aller Unterverzeichnisse gelten soll.

Entsprechend obigen Angaben ergibt sich dann folgender Inhalt:

```
AddType image/svg+xml svg svgz
AddEncoding gzip svgz
```

Sollen auch PHP-Skripte spezielle Endungen, etwa '.psvg' und '.psvgz' erhalten, so ergibt sich folgender Inhalt:

```
AddType application/x-httpd-php psvg psvgz
AddType image/svg+xml svg svgz
AddEncoding gzip svgz psvgz
```

Der Inhaltstyp ist dann allerdings in jedem Skript vor jeglicher anderer Ausgabe zu senden:

```
<?php
# sonstiger PHP-Kram ohne jegliche Ausgabe ...
header("Content-type: image/svg+xml");
# sonstiger PHP-Kram samt SVG-Ausgabe ...
?>
```

Dies ist auch notwendig, wenn die normale Endung für PHP '.php' verwendet wird.

Sollen und können die Änderungen in den Konfigurationsdateien durchgeführt werden, so sind dies typisch die Dateien httpd.conf oder apache2.conf je nach Version des Apache. Typisch sind diese in folgenden Verzeichnissen zu finden: /etc/httpd/conf/ or /etc/apache2/

Die jeweilige Anleitung zur Version des Apachen ist zu beachten. Die genannten Ergänzungen sollten auch nur vorgenommen werden, wenn sie wirklich nicht bereits vorhanden sind. Quelle der Weisheit für Inhaltstypen ist unter Unix/Linux zumeist eine bestimmte Datei, meistens ist dies /etc/mime.types genannt. Dort sollte 'image/svg+xml svg svgz' bereits notiert sein. Die Datei wird gegebenenfalls auch von anderen Programmen verwendet und wird kann auch bei Aktualisierungen des Betriebssystems automatisch ergänzt werden.

22.8 Literatur

Websiteentwicklung: XHTML[2]

Batik[3]

http://httpd.apache.org/docs/[4]

mit PHP, ASP, ASP.NET, Perl, JSP[5] (en)

kaioa.com: How to Configure Apache to Serve SVG/SVGZ the Right Way[6] (en)

2 http://de.wikibooks.org/wiki/Websiteentwicklung%3A%20XHTML
3 http://xmlgraphics.apache.org/batik/
4 http://httpd.apache.org/docs/
5 http://www.svgopen.org/2002/papers/hauser_wenz__server/
6 http://kaioa.com/node/45

23 Referenz

23.1 Elemente

23.1.1 A

a[1]

altGlyph[2]

altGlyphDef[3]

altGlyphItem[4]

animate[5]

animateColor[6]

animateMotion[7]

animateTransform[8]

animation[9]

audio[10]

23.1.2 C

circle[11]

clipPath[12]

color-profile[13]

---

1 Kapitel 15.5 auf Seite 519
2 Kapitel 19.1 auf Seite 587
3 Kapitel 19.1 auf Seite 587
4 Kapitel 19.1 auf Seite 587
5 Kapitel 14.14.1 auf Seite 508
6 Kapitel 14.14.3 auf Seite 508
7 Kapitel 14.14.5 auf Seite 508
8 Kapitel 14.14.4 auf Seite 508
9 Kapitel 17.2 auf Seite 550
10 Kapitel 17.2 auf Seite 550
11 Kapitel 6.7.3 auf Seite 139
12 Kapitel 13.4.3 auf Seite 401
13 Kapitel 10.2 auf Seite 310

cursor[14]

23.1.3 D

definition-src[15]

defs[16]

desc[17]

discard[18]

23.1.4 E

ellipse[19]

ev:listener[20]

23.1.5 F

feBlend[21]

feColorMatrix[22]

feComponentTransfer[23]

feComposite[24]

feConvolveMatrix[25]

feDiffuseLighting[26]

feDisplacementMap[27]

feDistantLight[28]

feFlood[29]

14 Kapitel 16.9 auf Seite 547
15 Kapitel 19.1 auf Seite 587
16 Kapitel 3.10 auf Seite 70
17 Kapitel 4.5 auf Seite 92
18 Kapitel 3.10 auf Seite 70
19 Kapitel 6.7.3 auf Seite 139
20 Kapitel 21.2.9 auf Seite 743
21 Kapitel 20.16.2 auf Seite 663
22 Kapitel 20.16.2 auf Seite 663
23 Kapitel 20.16.2 auf Seite 663
24 Kapitel 20.16.2 auf Seite 663
25 Kapitel 20.16.2 auf Seite 663
26 Kapitel 20.16.2 auf Seite 663
27 Kapitel 20.16.2 auf Seite 663
28 Kapitel 20.16.2 auf Seite 663
29 Kapitel 20.16.2 auf Seite 663

feFuncA[30]

feFuncB[31]

feFuncG[32]

feFuncR[33]

feGaussianBlur[34]

feImage[35]

feMerge[36]

feMergeNode[37]

feMorphology[38]

feOffset[39]

fePointLight[40]

feSpecularLighting[41]

feSpotLight[42]

feTile[43]

feTurbulence[44]

filter[45]

font[46]

font-face[47]

font-face-format[48]

font-face-name[49]

30 Kapitel 20.16.2 auf Seite 663
31 Kapitel 20.16.2 auf Seite 663
32 Kapitel 20.16.2 auf Seite 663
33 Kapitel 20.16.2 auf Seite 663
34 Kapitel 20.16.2 auf Seite 663
35 Kapitel 20.16.2 auf Seite 663
36 Kapitel 20.16.2 auf Seite 663
37 Kapitel 20.16.2 auf Seite 663
38 Kapitel 20.16.2 auf Seite 663
39 Kapitel 20.16.2 auf Seite 663
40 Kapitel 20.16.2 auf Seite 663
41 Kapitel 20.16.2 auf Seite 663
42 Kapitel 20.16.2 auf Seite 663
43 Kapitel 20.16.2 auf Seite 663
44 Kapitel 20.16.2 auf Seite 663
45 Kapitel 20.16.2 auf Seite 663
46 Kapitel 19.1 auf Seite 587
47 Kapitel 19.1 auf Seite 587
48 Kapitel 19.1 auf Seite 587
49 Kapitel 19.1 auf Seite 587

font-face-src[50]

font-face-uri[51]

foreignObject[52]

23.1.6 G

g[53]

glyph[54]

glyphRef[55]

23.1.7 H

handler[56]

hkern[57]

23.1.8 I

image[58]

23.1.9 L

line[59]

linearGradient[60]

23.1.10 M

marker[61]

mask[62]

50 Kapitel 19.1 auf Seite 587
51 Kapitel 19.1 auf Seite 587
52 Kapitel 18.2.4 auf Seite 579
53 Kapitel 3.10 auf Seite 70
54 Kapitel 19.1 auf Seite 587
55 Kapitel 19.1 auf Seite 587
56 Kapitel 21.2.9 auf Seite 743
57 Kapitel 19.1 auf Seite 587
58 Kapitel 3.10 auf Seite 70
59 Kapitel 6.7.3 auf Seite 139
60 Kapitel 11.2.2 auf Seite 326
61 Kapitel 12.6 auf Seite 385
62 Kapitel 13.4.3 auf Seite 401

metadata[63]

missing-glyph[64]

mpath[65]

23.1.11 P

path[66]

pattern[67]

polygon[68]

polyline[69]

prefetch[70]

23.1.12 R

radialGradient[71]

rect[72]

23.1.13 S

script[73]

set[74]

solidColor[75]

stop[76]

style[77]

svg[78]

63 Kapitel 4.5 auf Seite 92
64 Kapitel 19.1 auf Seite 587
65 Kapitel 14.14.2 auf Seite 508
66 Kapitel 7 auf Seite 143
67 Kapitel 11.2.2 auf Seite 326
68 Kapitel 6.7.3 auf Seite 139
69 Kapitel 6.7.3 auf Seite 139
70 Kapitel 3.10 auf Seite 70
71 Kapitel 11.2.2 auf Seite 326
72 Kapitel 6.7.3 auf Seite 139
73 Kapitel 21.2.9 auf Seite 743
74 Kapitel 14.14.2 auf Seite 508
75 Kapitel 10.3 auf Seite 312
76 Kapitel 11.2.2 auf Seite 326
77 Kapitel 9.6.4 auf Seite 251
78 Kapitel 3.10 auf Seite 70

switch[79]

symbol[80]

23.1.14 T

tbreak[81]

text[82]

textArea[83]

textPath[84]

title[85]

tref[86]

tspan[87]

23.1.15 U

use[88]

23.1.16 V

video[89]

view[90]

hkern[91]

79 Kapitel 4.5 auf Seite 92
80 Kapitel 3.10 auf Seite 70
81 Kapitel 8.1 auf Seite 193
82 Kapitel 8.1 auf Seite 193
83 Kapitel 8.1 auf Seite 193
84 Kapitel 8.1 auf Seite 193
85 Kapitel 4.5 auf Seite 92
86 Kapitel 8.1 auf Seite 193
87 Kapitel 8.1 auf Seite 193
88 Kapitel 3.10 auf Seite 70
89 Kapitel 17.2 auf Seite 550
90 Kapitel 15.5 auf Seite 519
91 Kapitel 19.1 auf Seite 587

23.2 Attribute

23.2.1 A

about[92]

accent-height[93]

accumulate[94]

additive[95]

alphabetic[96]

amplitude[97]

arabic-form[98]

ascent[99]

attributeName[100]

attributeType[101]

azimuth[102]

23.2.2 B

bandwidth[103]

baseFrequency[104]

baseProfile[105]

bbox[106]

begin[107] **(allgemein), begin**[108] **(discard)**

92 Kapitel 4.5 auf Seite 92
93 Kapitel 19.1 auf Seite 587
94 Kapitel 14.14.2 auf Seite 508
95 Kapitel 14.14.2 auf Seite 508
96 Kapitel 19.1 auf Seite 587
97 Kapitel 20.16.2 auf Seite 663
98 Kapitel 19.1 auf Seite 587
99 Kapitel 19.1 auf Seite 587
100 Kapitel 14.2.2 auf Seite 416
101 Kapitel 14.2.3 auf Seite 418
102 Kapitel 20.16.2 auf Seite 663
103 Kapitel 3.10 auf Seite 70
104 Kapitel 20.16.2 auf Seite 663
105 Kapitel 3.10 auf Seite 70
106 Kapitel 19.1 auf Seite 587
107 Kapitel 14.4.1 auf Seite 428
108 Kapitel 3.10 auf Seite 70

bias[109]

by[110]**, by**[111] **(from-by)**

23.2.3 C

calcMode[112]

cap-height[113]

class[114]

clipPathUnits[115]

content[116]

contentScriptType[117]

contentStyleType[118]

cx[119] **(circle), cx**[120] **(ellipse), cx**[121] **(radialGradient)**

cy[122] **(circle), cy**[123] **(ellipse), cy**[124] **(radialGradient)**

23.2.4 D

d[125] **(glyph), d**[126] **(path)**

datatype[127]

descent[128]

diffuseConstant[129]

109 Kapitel 20.16.2 auf Seite 663
110 Kapitel 14.14.2 auf Seite 508
111 Kapitel 14.14.2 auf Seite 508
112 Kapitel 14.12 auf Seite 504
113 Kapitel 19.1 auf Seite 587
114 Kapitel 3.10 auf Seite 70
115 Kapitel 13.4.3 auf Seite 401
116 Kapitel 4.5 auf Seite 92
117 Kapitel 3.10 auf Seite 70
118 Kapitel 3.10 auf Seite 70
119 Kapitel 6.7.3 auf Seite 139
120 Kapitel 6.7.3 auf Seite 139
121 Kapitel 11.2.2 auf Seite 326
122 Kapitel 6.7.3 auf Seite 139
123 Kapitel 6.7.3 auf Seite 139
124 Kapitel 11.2.2 auf Seite 326
125 Kapitel 19.1 auf Seite 587
126 Kapitel 7.4 auf Seite 187
127 Kapitel 4.5 auf Seite 92
128 Kapitel 19.1 auf Seite 587
129 Kapitel 20.16.2 auf Seite 663

divisor[130]

dur[131]

dx[132] **(altGlyph), dx**[133] **(feOffset), dx**[134] **(glyphRef), dx**[135] **(text), dx**[136] **(tref), dx**[137] **(tspan)**

dy[138] **(altGlyph), dy**[139] **(feOffset), dy**[140] **(glyphRef), dy**[141] **(text), dy**[142] **(tref), dy**[143] **(tspan)**

23.2.5 E

edgeMode[144]

editable[145]

elevation[146]

end[147]

ev:defaultAction[148]

ev:event[149] **(handler), ev:event**[150] **(listener)**

ev:handler[151]

ev:observer[152]

ev:phase[153]

ev:propagate[154]

ev:target[155]

130 Kapitel 20.16.2 auf Seite 663
131 Kapitel 14.3 auf Seite 422
132 Kapitel 19.1 auf Seite 587
133 Kapitel 20.16.2 auf Seite 663
134 Kapitel 19.1 auf Seite 587
135 Kapitel 8.1 auf Seite 193
136 Kapitel 8.1 auf Seite 193
137 Kapitel 8.1 auf Seite 193
138 Kapitel 19.1 auf Seite 587
139 Kapitel 20.16.2 auf Seite 663
140 Kapitel 19.1 auf Seite 587
141 Kapitel 8.1 auf Seite 193
142 Kapitel 8.1 auf Seite 193
143 Kapitel 8.1 auf Seite 193
144 Kapitel 20.16.2 auf Seite 663
145 Kapitel 16.5 auf Seite 532
146 Kapitel 20.16.2 auf Seite 663
147 Kapitel 14.4.2 auf Seite 430
148 Kapitel 21.2.9 auf Seite 743
149 Kapitel 21.2.9 auf Seite 743
150 Kapitel 21.2.9 auf Seite 743
151 Kapitel 21.2.9 auf Seite 743
152 Kapitel 21.2.9 auf Seite 743
153 Kapitel 21.2.9 auf Seite 743
154 Kapitel 21.2.9 auf Seite 743
155 Kapitel 21.2.9 auf Seite 743

exponent[156]

externalResourcesRequired[157]

23.2.6 F

fill[158]

filterRes[159]

filterUnits[160]

focusable[161]

focusHighlight[162]

font-family[163]

font-size[164]

font-stretch[165]

font-style[166]

font-variant[167]

font-weight[168]

format[169] **(altGlyph), format**[170] **(glyphRef)**

from[171] **(-by), from**[172] **(-to)**

fx[173]

fy[174]

156 Kapitel 20.16.2 auf Seite 663
157 Kapitel 3.10 auf Seite 70
158 Kapitel 14.5 auf Seite 469
159 Kapitel 20.16.2 auf Seite 663
160 Kapitel 20.16.2 auf Seite 663
161 Kapitel 16.3 auf Seite 530
162 Kapitel 16.4 auf Seite 531
163 Kapitel 19.1 auf Seite 587
164 Kapitel 19.1 auf Seite 587
165 Kapitel 19.1 auf Seite 587
166 Kapitel 19.1 auf Seite 587
167 Kapitel 19.1 auf Seite 587
168 Kapitel 19.1 auf Seite 587
169 Kapitel 19.1 auf Seite 587
170 Kapitel 19.1 auf Seite 587
171 Kapitel 14.14.2 auf Seite 508
172 Kapitel 14.14.2 auf Seite 508
173 Kapitel 11.2.2 auf Seite 326
174 Kapitel 11.2.2 auf Seite 326

23.2.7 G

g1[175]

g2[176]

glyph-name[177]

glyphRef[178] **(altGlyph), glyphRef**[179] **(glyphRef)**

gradientTransform[180] **(linearGradient), gradientTransform**[181] **(radialGradient)**

gradientUnits[182] **(linearGradient), gradientUnits**[183] **(radialGradient)**

23.2.8 H

hanging[184]

height[185] **(animation), height**[186] **(filter), height**[187] **(foreignObject), height**[188] **(image), height**[189] **(rect), height**[190] **(svg), height**[191] **(textArea), height**[192] **(Urfilter), height**[193] **(use), height**[194] **(video)**

horiz-adv-x[195] **(font), horiz-adv-x**[196] **(glyph)**

horiz-origin-x[197]

horiz-origin-y[198]

175 Kapitel 19.1 auf Seite 587
176 Kapitel 19.1 auf Seite 587
177 Kapitel 19.1 auf Seite 587
178 Kapitel 19.1 auf Seite 587
179 Kapitel 19.1 auf Seite 587
180 Kapitel 11.2.2 auf Seite 326
181 Kapitel 11.2.2 auf Seite 326
182 Kapitel 11.2.2 auf Seite 326
183 Kapitel 11.2.2 auf Seite 326
184 Kapitel 19.1 auf Seite 587
185 Kapitel 17.2 auf Seite 550
186 Kapitel 20.16.2 auf Seite 663
187 Kapitel 18.2.4 auf Seite 579
188 Kapitel 3.10 auf Seite 70
189 Kapitel 6.7.3 auf Seite 139
190 Kapitel 3.10 auf Seite 70
191 Kapitel 8.1 auf Seite 193
192 Kapitel 20.16.2 auf Seite 663
193 Kapitel 3.10 auf Seite 70
194 Kapitel 17.2 auf Seite 550
195 Kapitel 19.1 auf Seite 587
196 Kapitel 19.1 auf Seite 587
197 Kapitel 19.1 auf Seite 587
198 Kapitel 19.1 auf Seite 587

23.2.9 I

id[199]

ideographic[200]

in[201]

in2[202] **(feBlend), in2**[203] **(feComposite), in2**[204] **(feDisplacementMap)**

initialVisibility[205] **(animation), initialVisibility**[206] **(video)**

intercept[207]

23.2.10 K

k[208]

k1[209]

k2[210]

k3[211]

k4[212]

kernelMatrix[213]

kernelUnitLength[214] **(feConvolveMatrix), kernelUnitLength**[215] **(feDiffuseLighting), kernelUnitLength**[216] **(feSpecularLighting)**

keyPoints[217]

keySplines[218]

199 Kapitel 3.10 auf Seite 70
200 Kapitel 19.1 auf Seite 587
201 Kapitel 20.16.2 auf Seite 663
202 Kapitel 20.16.2 auf Seite 663
203 Kapitel 20.16.2 auf Seite 663
204 Kapitel 20.16.2 auf Seite 663
205 Kapitel 17.2 auf Seite 550
206 Kapitel 17.2 auf Seite 550
207 Kapitel 20.16.2 auf Seite 663
208 Kapitel 19.1 auf Seite 587
209 Kapitel 20.16.2 auf Seite 663
210 Kapitel 20.16.2 auf Seite 663
211 Kapitel 20.16.2 auf Seite 663
212 Kapitel 20.16.2 auf Seite 663
213 Kapitel 20.16.2 auf Seite 663
214 Kapitel 20.16.2 auf Seite 663
215 Kapitel 20.16.2 auf Seite 663
216 Kapitel 20.16.2 auf Seite 663
217 Kapitel 14.14.2 auf Seite 508
218 Kapitel 14.14.2 auf Seite 508

keyTimes[219]

23.2.11 L

lang[220]

lengthAdjust[221] **(text), lengthAdjust**[222] **(tref), lengthAdjust**[223] **(tspan), lengthAdjust**[224] **(textPath)**

limitingConeAngle[225]

local[226]

23.2.12 M

markerHeight[227]

markerUnits[228]

markerWidth[229]

maskContentUnits[230]

maskUnits[231]

mathematical[232]

max[233]

media[234]

mediaCharacterEncoding[235]

mediaContentEncodings[236]

mediaSize[237]

---

219 Kapitel 14.11 auf Seite 499
220 Kapitel 19.1 auf Seite 587
221 Kapitel 8.1 auf Seite 193
222 Kapitel 8.1 auf Seite 193
223 Kapitel 8.1 auf Seite 193
224 Kapitel 8.1 auf Seite 193
225 Kapitel 20.16.2 auf Seite 663
226 Kapitel 10.3 auf Seite 312
227 Kapitel 12.6 auf Seite 385
228 Kapitel 12.6 auf Seite 385
229 Kapitel 12.6 auf Seite 385
230 Kapitel 13.4.3 auf Seite 401
231 Kapitel 13.4.3 auf Seite 401
232 Kapitel 19.1 auf Seite 587
233 Kapitel 14.14.2 auf Seite 508
234 Kapitel 9.6.4 auf Seite 251
235 Kapitel 3.10 auf Seite 70
236 Kapitel 3.10 auf Seite 70
237 Kapitel 3.10 auf Seite 70

mediaTime[238]

method[239]

min[240]

mode[241]

23.2.13 N

name[242]

nav-down[243]

nav-down-left[244]

nav-down-right[245]

nav-left[246]

nav-next[247]

nav-prev[248]

nav-right[249]

nav-up[250]

nav-up-left[251]

nav-up-right[252]

numOctaves[253]

23.2.14 O

offset[254] **(stop), offset**[255] **(Filter)**

238 Kapitel 3.10 auf Seite 70
239 Kapitel 8.1 auf Seite 193
240 Kapitel 14.14.2 auf Seite 508
241 Kapitel 20.16.2 auf Seite 663
242 Kapitel 10.3 auf Seite 312
243 Kapitel 16.6.1 auf Seite 535
244 Kapitel 16.6.1 auf Seite 535
245 Kapitel 16.6.1 auf Seite 535
246 Kapitel 16.6.1 auf Seite 535
247 Kapitel 16.6.1 auf Seite 535
248 Kapitel 16.6.1 auf Seite 535
249 Kapitel 16.6.1 auf Seite 535
250 Kapitel 16.6.1 auf Seite 535
251 Kapitel 16.6.1 auf Seite 535
252 Kapitel 16.6.1 auf Seite 535
253 Kapitel 20.16.2 auf Seite 663
254 Kapitel 11.2.2 auf Seite 326
255 Kapitel 20.16.2 auf Seite 663

onabort[256]

onactivate[257]

onbegin[258]

onclick[259]

onend[260]

onerror[261]

onfocusin[262]

onfocusout[263]

onload[264]

onmousedown[265]

onmousemove[266]

onmouseout[267]

onmouseover[268]

onmouseup[269]

onrepeat[270]

onresize[271]

onscroll[272]

onunload[273]

onzoom[274]

operator[275] **(feComposite), operator**[276] **(feMorphology)**

---

256 Kapitel 16.2.2 auf Seite 527
257 Kapitel 16.2.2 auf Seite 527
258 Kapitel 16.2.2 auf Seite 527
259 Kapitel 16.2.2 auf Seite 527
260 Kapitel 16.2.2 auf Seite 527
261 Kapitel 16.2.2 auf Seite 527
262 Kapitel 16.2.2 auf Seite 527
263 Kapitel 16.2.2 auf Seite 527
264 Kapitel 16.2.2 auf Seite 527
265 Kapitel 16.2.2 auf Seite 527
266 Kapitel 16.2.2 auf Seite 527
267 Kapitel 16.2.2 auf Seite 527
268 Kapitel 16.2.2 auf Seite 527
269 Kapitel 16.2.2 auf Seite 527
270 Kapitel 16.2.2 auf Seite 527
271 Kapitel 16.2.2 auf Seite 527
272 Kapitel 16.2.2 auf Seite 527
273 Kapitel 16.2.2 auf Seite 527
274 Kapitel 16.2.2 auf Seite 527
275 Kapitel 20.16.2 auf Seite 663
276 Kapitel 20.16.2 auf Seite 663

order[277]

orient[278]

orientation[279]

origin[280]

overlay[281]

overline-position[282]

overline-thickness[283]

23.2.15 P

panose-1[284]

path[285]

pathLength[286]

patternContentUnits[287]

patternTransform[288]

patternUnits[289]

playbackOrder[290]

points[291] **(polygon), points**[292] **(polyline)**

pointsAtX[293]

pointsAtY[294]

pointsAtZ[295]

277 Kapitel 20.16.2 auf Seite 663
278 Kapitel 12.6 auf Seite 385
279 Kapitel 19.1 auf Seite 587
280 Kapitel 14.14.2 auf Seite 508
281 Kapitel 17.2 auf Seite 550
282 Kapitel 19.1 auf Seite 587
283 Kapitel 19.1 auf Seite 587
284 Kapitel 19.1 auf Seite 587
285 Kapitel 14.14.2 auf Seite 508
286 Kapitel 7.4 auf Seite 187
287 Kapitel 11.2.2 auf Seite 326
288 Kapitel 11.2.2 auf Seite 326
289 Kapitel 11.2.2 auf Seite 326
290 Kapitel 3.10 auf Seite 70
291 Kapitel 6.7.3 auf Seite 139
292 Kapitel 6.7.3 auf Seite 139
293 Kapitel 20.16.2 auf Seite 663
294 Kapitel 20.16.2 auf Seite 663
295 Kapitel 20.16.2 auf Seite 663

preserveAlpha[296]

preserveAspectRatio[297] **(animation), preserveAspectRatio**[298] **(image), preserveAspectRatio**[299] **(marker), preserveAspectRatio**[300] **(svg), preserveAspectRatio**[301] **(symbol), preserveAspectRatio**[302] **(video)**

primitiveUnits[303]

property[304]

23.2.16 R

r[305] **(circle), r**[306] **(radialGradient)**

radius[307]

refX[308]

refY[309]

rel[310]

rendering-intent[311]

repeatCount[312]

repeatDur[313]

requiredExtensions[314]

requiredFeatures[315]

requiredFonts[316]

requiredFormats[317]

296 Kapitel 20.16.2 auf Seite 663
297 Kapitel 17.2 auf Seite 550
298 Kapitel 3.10 auf Seite 70
299 Kapitel 12.6 auf Seite 385
300 Kapitel 3.10 auf Seite 70
301 Kapitel 3.10 auf Seite 70
302 Kapitel 17.2 auf Seite 550
303 Kapitel 20.16.2 auf Seite 663
304 Kapitel 4.5 auf Seite 92
305 Kapitel 6.7.3 auf Seite 139
306 Kapitel 11.2.2 auf Seite 326
307 Kapitel 20.16.2 auf Seite 663
308 Kapitel 12.6 auf Seite 385
309 Kapitel 12.6 auf Seite 385
310 Kapitel 4.5 auf Seite 92
311 Kapitel 10.3 auf Seite 312
312 Kapitel 14.14.2 auf Seite 508
313 Kapitel 14.14.2 auf Seite 508
314 Kapitel 4.5 auf Seite 92
315 Kapitel 4.5 auf Seite 92
316 Kapitel 4.5 auf Seite 92
317 Kapitel 4.5 auf Seite 92

resource[318]

restart[319]

result[320]

role[321]

rotate[322] **(altGlyph), rotate**[323] **(animateMotion), rotate**[324] **(text), rotate**[325] **(tref), rotate**[326] **(tspan)**

rx[327] **(rect), rx**[328] **(ellipse)**

ry[329] **(rect), ry**[330] **(ellipse)**

23.2.17 S

scale[331]

seed[332]

slope[333] **(font-face), slope**[334] **(feComponentTransfer)**

snapshottime[335]

spacing[336]

specularConstant[337]

specularExponent[338] **(feSpotLight), specularExponent**[339] **(feSpecularLighting)**

spreadMethod[340] **(linearGradient), spreadMethod**[341] **(radialGradient)**

318 Kapitel 4.5 auf Seite 92
319 Kapitel 14.14.2 auf Seite 508
320 Kapitel 20.16.2 auf Seite 663
321 Kapitel 4.5 auf Seite 92
322 Kapitel 19.1 auf Seite 587
323 Kapitel 14.14.2 auf Seite 508
324 Kapitel 8.1 auf Seite 193
325 Kapitel 8.1 auf Seite 193
326 Kapitel 8.1 auf Seite 193
327 Kapitel 6.7.3 auf Seite 139
328 Kapitel 6.7.3 auf Seite 139
329 Kapitel 6.7.3 auf Seite 139
330 Kapitel 6.7.3 auf Seite 139
331 Kapitel 20.16.2 auf Seite 663
332 Kapitel 20.16.2 auf Seite 663
333 Kapitel 19.1 auf Seite 587
334 Kapitel 20.16.2 auf Seite 663
335 Kapitel 3.10 auf Seite 70
336 Kapitel 8.1 auf Seite 193
337 Kapitel 20.16.2 auf Seite 663
338 Kapitel 20.16.2 auf Seite 663
339 Kapitel 20.16.2 auf Seite 663
340 Kapitel 11.2.2 auf Seite 326
341 Kapitel 11.2.2 auf Seite 326

startOffset[342]

stdDeviation[343]

stemh[344]

stemv[345]

stitchTiles[346]

strikethrough-position[347]

strikethrough-thickness[348]

string[349]

style[350]

surfaceScale[351] **(feDiffuseLighting), surfaceScale**[352] **(feSpecularLighting)**

syncBehavior[353]

syncBehaviorDefault[354]

syncMaster[355]

syncTolerance[356]

syncToleranceDefault[357]

systemLanguage[358]

23.2.18 T

tableValues[359]

target[360]

342 Kapitel 8.1 auf Seite 193
343 Kapitel 20.16.2 auf Seite 663
344 Kapitel 19.1 auf Seite 587
345 Kapitel 19.1 auf Seite 587
346 Kapitel 20.16.2 auf Seite 663
347 Kapitel 19.1 auf Seite 587
348 Kapitel 19.1 auf Seite 587
349 Kapitel 19.1 auf Seite 587
350 Kapitel 9.6.4 auf Seite 251
351 Kapitel 20.16.2 auf Seite 663
352 Kapitel 20.16.2 auf Seite 663
353 Kapitel 17.2 auf Seite 550
354 Kapitel 17.2 auf Seite 550
355 Kapitel 17.2 auf Seite 550
356 Kapitel 17.2 auf Seite 550
357 Kapitel 17.2 auf Seite 550
358 Kapitel 4.5 auf Seite 92
359 Kapitel 20.16.2 auf Seite 663
360 Kapitel 15.5 auf Seite 519

targetX[361]

targetY[362]

textLength[363] **(text), textLength**[364] **(tref), textLength**[365] **(tspan), textLength**[366] **(textPath)**

timelineBegin[367]

title[368]

to[369]**, to**[370] **(from-to),**

transform[371]

transformBehavior[372]

type[373] **(audio), type**[374] **(feColorMatrix), type**[375] **(feCompontentTransfer), type**[376] **(feTurbulence), type**[377] **(image), type**[378] **(script), type**[379] **(style) type**[380] **(video)**

typeof[381]

23.2.19 U

u1[382]

u2[383]

underline-position[384]

underline-thickness[385]

361 Kapitel 20.16.2 auf Seite 663
362 Kapitel 20.16.2 auf Seite 663
363 Kapitel 8.1 auf Seite 193
364 Kapitel 8.1 auf Seite 193
365 Kapitel 8.1 auf Seite 193
366 Kapitel 8.1 auf Seite 193
367 Kapitel 3.10 auf Seite 70
368 Kapitel 9.6.4 auf Seite 251
369 Kapitel 14.14.2 auf Seite 508
370 Kapitel 14.14.2 auf Seite 508
371 Kapitel 5.1 auf Seite 99
372 Kapitel 17.2 auf Seite 550
373 Kapitel 17.2 auf Seite 550
374 Kapitel 20.16.2 auf Seite 663
375 Kapitel 20.16.2 auf Seite 663
376 Kapitel 20.16.2 auf Seite 663
377 Kapitel 3.10 auf Seite 70
378 Kapitel 21.2.3 auf Seite 732
379 Kapitel 9.6.4 auf Seite 251
380 Kapitel 17.2 auf Seite 550
381 Kapitel 4.5 auf Seite 92
382 Kapitel 19.1 auf Seite 587
383 Kapitel 19.1 auf Seite 587
384 Kapitel 19.1 auf Seite 587
385 Kapitel 19.1 auf Seite 587

unicode[386]

unicode-range[387]

units-per-em[388]

23.2.20 V

v-alphabetic[389]

v-hanging[390]

v-ideographic[391]

v-mathematical[392]

values[393] **(Animation) values**[394] **(Filter)**

version[395]

vert-adv-y[396] **(font), vert-adv-y**[397] **(glyph)**

vert-origin-x[398] **(font), vert-origin-x**[399] **(glyph)**

vert-origin-y[400] **(font), vert-origin-y**[401] **(glyph)**

viewBox[402] **(image), viewBox**[403] **(marker), viewBox**[404] **(svg), viewBox**[405] **(symbol)**

viewTarget[406]

386 Kapitel 19.1 auf Seite 587
387 Kapitel 19.1 auf Seite 587
388 Kapitel 19.1 auf Seite 587
389 Kapitel 19.1 auf Seite 587
390 Kapitel 19.1 auf Seite 587
391 Kapitel 19.1 auf Seite 587
392 Kapitel 19.1 auf Seite 587
393 Kapitel 14.14.2 auf Seite 508
394 Kapitel 20.16.2 auf Seite 663
395 Kapitel 3.10 auf Seite 70
396 Kapitel 19.1 auf Seite 587
397 Kapitel 19.1 auf Seite 587
398 Kapitel 19.1 auf Seite 587
399 Kapitel 19.1 auf Seite 587
400 Kapitel 19.1 auf Seite 587
401 Kapitel 19.1 auf Seite 587
402 Kapitel 3.10 auf Seite 70
403 Kapitel 12.6 auf Seite 385
404 Kapitel 3.10 auf Seite 70
405 Kapitel 3.10 auf Seite 70
406 Kapitel 15.5 auf Seite 519

23.2.21 W

width[407] (animation), width[408] (filter), width[409] (foreignObject), width[410] (image), width[411] (rect), width[412] (svg), width[413] (textArea), width[414] (Urfilter), width[415] (use), width[416] (video)

widths[417]

23.2.22 X

x[418] (altGlyph), x[419] (animation), x[420] (cursor), x[421] (fePointLight), x[422] (feSpotLight), x[423] (filter), x[424] (foreignObject), x[425] (glyphRef), x[426] (image), x[427] (rect), x[428] (svg), x[429] (text), x[430] (textArea), x[431] (tref), x[432] (tspan), x[433] (Urfilter), x[434] (use), x[435] (video)

x-height[436]

x1[437] (line), x1[438] (linearGradient)

x2[439] (line), x2[440] (linearGradient)

xChannelSelector[441]

---

407 Kapitel 17.2 auf Seite 550
408 Kapitel 20.16.2 auf Seite 663
409 Kapitel 18.2.4 auf Seite 579
410 Kapitel 3.10 auf Seite 70
411 Kapitel 6.7.3 auf Seite 139
412 Kapitel 3.10 auf Seite 70
413 Kapitel 8.1 auf Seite 193
414 Kapitel 20.16.2 auf Seite 663
415 Kapitel 3.10 auf Seite 70
416 Kapitel 17.2 auf Seite 550
417 Kapitel 19.1 auf Seite 587
418 Kapitel 19.1 auf Seite 587
419 Kapitel 17.2 auf Seite 550
420 Kapitel 16.9 auf Seite 547
421 Kapitel 20.16.2 auf Seite 663
422 Kapitel 20.16.2 auf Seite 663
423 Kapitel 20.16.2 auf Seite 663
424 Kapitel 18.2.4 auf Seite 579
425 Kapitel 19.1 auf Seite 587
426 Kapitel 3.10 auf Seite 70
427 Kapitel 6.7.3 auf Seite 139
428 Kapitel 3.10 auf Seite 70
429 Kapitel 8.1 auf Seite 193
430 Kapitel 8.1 auf Seite 193
431 Kapitel 8.1 auf Seite 193
432 Kapitel 8.1 auf Seite 193
433 Kapitel 20.16.2 auf Seite 663
434 Kapitel 3.10 auf Seite 70
435 Kapitel 17.2 auf Seite 550
436 Kapitel 19.1 auf Seite 587
437 Kapitel 6.7.3 auf Seite 139
438 Kapitel 11.2.2 auf Seite 326
439 Kapitel 6.7.3 auf Seite 139
440 Kapitel 11.2.2 auf Seite 326
441 Kapitel 20.16.2 auf Seite 663

xlink:actuate[442]

xlink:arcrole[443]

xlink:href[444] **(allgemein), xlink:href**[445] **(a), xlink:href**[446] **(altGlyph), xlink:href**[447] **(animation), xlink:href**[448] **(audio), xlink:href**[449] **(color-profile), xlink:href**[450] **(cursor), xlink:href**[451] **(discard), xlink:href**[452] **(feImage), xlink:href**[453] **(filter), xlink:href**[454] **(font-face-uri), xlink:href**[455] **(foreignObject), xlink:href**[456] **(glyphRef), xlink:href**[457] **(handler), xlink:href**[458] **(image), xlink:href**[459] **(prefetch), xlink:href**[460] **(use), xlink:href**[461] **(video)**

xlink:role[462]

xlink:show[463]

xlink:title[464]

xlink:type[465]

xml:base[466]

xml:id[467]**, xml:id**[468]

xml:lang[469]

xml:space[470]

442 Kapitel 15.5 auf Seite 519
443 Kapitel 15.5 auf Seite 519
444 Kapitel 15.5 auf Seite 519
445 Kapitel 15.5 auf Seite 519
446 Kapitel 19.1 auf Seite 587
447 Kapitel 17.2 auf Seite 550
448 Kapitel 17.2 auf Seite 550
449 Kapitel 10.3 auf Seite 312
450 Kapitel 16.9 auf Seite 547
451 Kapitel 3.10 auf Seite 70
452 Kapitel 20.16.2 auf Seite 663
453 Kapitel 20.16.2 auf Seite 663
454 Kapitel 19.1 auf Seite 587
455 Kapitel 18.2.4 auf Seite 579
456 Kapitel 19.1 auf Seite 587
457 Kapitel 21.2.9 auf Seite 743
458 Kapitel 3.10 auf Seite 70
459 Kapitel 3.10 auf Seite 70
460 Kapitel 3.10 auf Seite 70
461 Kapitel 17.2 auf Seite 550
462 Kapitel 15.5 auf Seite 519
463 Kapitel 15.5 auf Seite 519
464 Kapitel 15.5 auf Seite 519
465 Kapitel 15.5 auf Seite 519
466 Kapitel 1.19 auf Seite 19
467 Kapitel 1.19 auf Seite 19
468 Kapitel 3.10 auf Seite 70
469 Kapitel 1.19 auf Seite 19
470 Kapitel 1.19 auf Seite 19

23.2.23 Y

y[471] (altGlyph), y[472] (animation), y[473] (cursor), y[474] (fePointLight), y[475] (feSpotLight), y[476] (filter), y[477] (foreignObject), y[478] (glyphRef), y[479] (image), y[480] (rect), y[481] (svg), y[482] (text), y[483] (textArea), y[484] (tref), y[485] (tspan), y[486] (Urfilter), y[487] (use), y[488] (video)

y1[489] (line), y1[490] (linearGradient)

y2[491] (line), y2[492] (linearGradient)

yChannelSelector[493]

23.2.24 Z

z[494] (fePointLight), z[495] (feSpotLight)

zoomAndPan[496]

23.3 Eigenschaften

23.3.1 A

alignment-baseline[497]

471 Kapitel 19.1 auf Seite 587
472 Kapitel 17.2 auf Seite 550
473 Kapitel 16.9 auf Seite 547
474 Kapitel 20.16.2 auf Seite 663
475 Kapitel 20.16.2 auf Seite 663
476 Kapitel 20.16.2 auf Seite 663
477 Kapitel 18.2.4 auf Seite 579
478 Kapitel 19.1 auf Seite 587
479 Kapitel 3.10 auf Seite 70
480 Kapitel 6.7.3 auf Seite 139
481 Kapitel 3.10 auf Seite 70
482 Kapitel 8.1 auf Seite 193
483 Kapitel 8.1 auf Seite 193
484 Kapitel 8.1 auf Seite 193
485 Kapitel 8.1 auf Seite 193
486 Kapitel 20.16.2 auf Seite 663
487 Kapitel 3.10 auf Seite 70
488 Kapitel 17.2 auf Seite 550
489 Kapitel 6.7.3 auf Seite 139
490 Kapitel 11.2.2 auf Seite 326
491 Kapitel 6.7.3 auf Seite 139
492 Kapitel 11.2.2 auf Seite 326
493 Kapitel 20.16.2 auf Seite 663
494 Kapitel 20.16.2 auf Seite 663
495 Kapitel 20.16.2 auf Seite 663
496 Kapitel 3.10 auf Seite 70
497 Kapitel 9.6.4 auf Seite 251

audio-level[498]

23.3.2 B

baseline-shift[499]

buffered-rendering[500]

23.3.3 C

color[501]

color-interpolation[502]

color-interpolation-filters[503], **color-interpolation-filters**[504]

color-rendering[505]

clip[506]

clip-path[507]

clip-rule[508]

color-profile[509]

cursor[510]

23.3.4 D

direction[511]

display[512]

display-align[513]

498 Kapitel 17.2 auf Seite 550
499 Kapitel 9.6.4 auf Seite 251
500 Kapitel 9.6.4 auf Seite 251
501 Kapitel 9.6.4 auf Seite 251
502 Kapitel 9.6.4 auf Seite 251
503 Kapitel 20.16.2 auf Seite 663
504 Kapitel 9.6.4 auf Seite 251
505 Kapitel 9.6.4 auf Seite 251
506 Kapitel 13.4.3 auf Seite 401
507 Kapitel 13.4.3 auf Seite 401
508 Kapitel 13.4.3 auf Seite 401
509 Kapitel 10.3 auf Seite 312
510 Kapitel 9.6.4 auf Seite 251
511 Kapitel 9.7.3 auf Seite 266
512 Kapitel 9.6.4 auf Seite 251
513 Kapitel 8.1 auf Seite 193

dominant-baseline[514]

23.3.5 E

enable-background[515]

23.3.6 F

fill[516]

fill-opacity[517]

fill-rule[518]

filter[519]

flood-color[520]

flood-opacity[521]

font[522]

font-family[523]

font-size[524]

font-size-adjust[525]

font-stretch[526]

font-style[527]

font-variant[528]

font-weight[529]

514 Kapitel 9.6.4 auf Seite 251
515 Kapitel 20.16.2 auf Seite 663
516 Kapitel 9.6.4 auf Seite 251
517 Kapitel 9.6.4 auf Seite 251
518 Kapitel 9.6.4 auf Seite 251
519 Kapitel 20.16.2 auf Seite 663
520 Kapitel 20.16.2 auf Seite 663
521 Kapitel 20.16.2 auf Seite 663
522 Kapitel 9.6.4 auf Seite 251
523 Kapitel 9.6.4 auf Seite 251
524 Kapitel 9.6.4 auf Seite 251
525 Kapitel 9.6.4 auf Seite 251
526 Kapitel 9.6.4 auf Seite 251
527 Kapitel 9.6.4 auf Seite 251
528 Kapitel 9.6.4 auf Seite 251
529 Kapitel 9.6.4 auf Seite 251

23.3.7 G

glyph-orientation-horizontal[530]

glyph-orientation-vertical[531]

23.3.8 I

image-rendering[532]

23.3.9 K

kerning[533]

23.3.10 L

letter-spacing[534]

line-increment[535]

lighting-color[536]

23.3.11 M

marker[537]

marker-end[538]

marker-mid[539]

marker-start[540]

mask[541]

530 Kapitel 9.7.3 auf Seite 270
531 Kapitel 9.7.3 auf Seite 270
532 Kapitel 9.6.4 auf Seite 251
533 Kapitel 9.6.4 auf Seite 251
534 Kapitel 9.6.4 auf Seite 251
535 Kapitel 8.1 auf Seite 193
536 Kapitel 20.16.2 auf Seite 663
537 Kapitel 12.3 auf Seite 368
538 Kapitel 12.3 auf Seite 368
539 Kapitel 12.3 auf Seite 368
540 Kapitel 12.3 auf Seite 368
541 Kapitel 13.4.3 auf Seite 401

23.3.12 O

opacity[542]

overflow[543]

23.3.13 P

pointer-events[544]

23.3.14 S

shape-rendering[545]

solid-color[546]

solid-opacity[547]

stop-color[548]

stop-opacity[549]

stroke[550]

stroke-dasharray[551]

stroke-dashoffset[552]

stroke-linecap[553]

stroke-linejoin[554]

stroke-miterlimit[555]

stroke-opacity[556]

stroke-width[557]

542 Kapitel 9.6.4 auf Seite 251
543 Kapitel 9.6.4 auf Seite 251
544 Kapitel 9.6.4 auf Seite 251
545 Kapitel 9.6.4 auf Seite 251
546 Kapitel 10.3 auf Seite 312
547 Kapitel 10.3 auf Seite 312
548 Kapitel 11.2.2 auf Seite 326
549 Kapitel 11.2.2 auf Seite 326
550 Kapitel 9.6.4 auf Seite 251
551 Kapitel 9.6.4 auf Seite 251
552 Kapitel 9.6.4 auf Seite 251
553 Kapitel 9.6.4 auf Seite 251
554 Kapitel 9.6.4 auf Seite 251
555 Kapitel 9.6.4 auf Seite 251
556 Kapitel 9.6.4 auf Seite 251
557 Kapitel 9.6.4 auf Seite 251

23.3.15 T

text-align[558]

text-anchor[559]

text-decoration[560]

text-rendering[561]

23.3.16 U

unicode-bidi[562]

23.3.17 V

vector-effekt[563]

viewport-fill[564]

viewport-fill-opacity[565]

visibility[566]

23.3.18 W

word-spacing[567]

writing-mode[568]

558 Kapitel 8.1 auf Seite 193
559 Kapitel 9.6.4 auf Seite 251
560 Kapitel 9.7.6 auf Seite 285
561 Kapitel 9.6.4 auf Seite 251
562 Kapitel 9.7.3 auf Seite 265
563 Kapitel 9.6.4 auf Seite 251
564 Kapitel 9.6.4 auf Seite 251
565 Kapitel 9.6.4 auf Seite 251
566 Kapitel 9.6.4 auf Seite 251
567 Kapitel 9.6.4 auf Seite 251
568 Kapitel 9.7.3 auf Seite 268

24 Autoren

Edits	User
2	Aka[1]
2	Anton[2]
1	Aspiriniks[3]
1	Avatar[4]
30	Bastie[5]
1	BotMultichill[6]
2	CarsracBot[7]
1	Cepheiden[8]
3	Christian.ego[9]
3	Complex[10]
1	D[11]
2	DerHexer[12]
3	Diba[13]
137	Dirk Huenniger[14]
409	Doktorchen[15]
1	DooFi[16]
1	Dundak[17]
1	Eˆ(nix)[18]
1	ElRaki[19]
1	Eskimbot[20]
308	FischX[21]

1 http://de.wikibooks.org/w/index.php?title=Benutzer:Aka
2 http://de.wikibooks.org/w/index.php?title=Benutzer:Anton
3 http://de.wikibooks.org/w/index.php?title=Benutzer:Aspiriniks
4 http://de.wikibooks.org/w/index.php?title=Benutzer:Avatar
5 http://de.wikibooks.org/w/index.php?title=Benutzer:Bastie
6 http://de.wikibooks.org/w/index.php?title=Benutzer:BotMultichill
7 http://de.wikibooks.org/w/index.php?title=Benutzer:CarsracBot
8 http://de.wikibooks.org/w/index.php?title=Benutzer:Cepheiden
9 http://de.wikibooks.org/w/index.php?title=Benutzer:Christian.ego
10 http://de.wikibooks.org/w/index.php?title=Benutzer:Complex
11 http://de.wikibooks.org/w/index.php?title=Benutzer:D
12 http://de.wikibooks.org/w/index.php?title=Benutzer:DerHexer
13 http://de.wikibooks.org/w/index.php?title=Benutzer:Diba
14 http://de.wikibooks.org/w/index.php?title=Benutzer:Dirk_Huenniger
15 http://de.wikibooks.org/w/index.php?title=Benutzer:Doktorchen
16 http://de.wikibooks.org/w/index.php?title=Benutzer:DooFi
17 http://de.wikibooks.org/w/index.php?title=Benutzer:Dundak
18 http://de.wikibooks.org/w/index.php?title=Benutzer:E%5E%28nix%29
19 http://de.wikibooks.org/w/index.php?title=Benutzer:ElRaki
20 http://de.wikibooks.org/w/index.php?title=Benutzer:Eskimbot
21 http://de.wikibooks.org/w/index.php?title=Benutzer:FischX

2 Fleshgrinder[22]
1 Flying sheep[23]
3 Gnushi[24]
1 HAL Neuntausend[25]
4 Haddock[26]
4 Haeber[27]
1 Hagezussa[28]
3 Heuler06[29]
3 IP-Sichter[30]
4 Igelball[31]
2 JakobVoss[32]
1 JohannWalter[33]
1 JuergenL[34]
4 Juesch[35]
2 Juetho[36]
1 Kaisersoft[37]
4 Klaus Eifert[38]
2 Kopoltra[39]
3 Krawi[40]
1 Kristjan[41]
1 LC[42]
1 LaaknorBot[43]
1 Liquidat[44]
6 LivingShadow[45]
2 Mamue[46]

22 http://de.wikibooks.org/w/index.php?title=Benutzer:Fleshgrinder
23 http://de.wikibooks.org/w/index.php?title=Benutzer:Flying_sheep
24 http://de.wikibooks.org/w/index.php?title=Benutzer:Gnushi
25 http://de.wikibooks.org/w/index.php?title=Benutzer:HAL_Neuntausend
26 http://de.wikibooks.org/w/index.php?title=Benutzer:Haddock
27 http://de.wikibooks.org/w/index.php?title=Benutzer:Haeber
28 http://de.wikibooks.org/w/index.php?title=Benutzer:Hagezussa
29 http://de.wikibooks.org/w/index.php?title=Benutzer:Heuler06
30 http://de.wikibooks.org/w/index.php?title=Benutzer:IP-Sichter
31 http://de.wikibooks.org/w/index.php?title=Benutzer:Igelball
32 http://de.wikibooks.org/w/index.php?title=Benutzer:JakobVoss
33 http://de.wikibooks.org/w/index.php?title=Benutzer:JohannWalter
34 http://de.wikibooks.org/w/index.php?title=Benutzer:JuergenL
35 http://de.wikibooks.org/w/index.php?title=Benutzer:Juesch
36 http://de.wikibooks.org/w/index.php?title=Benutzer:Juetho
37 http://de.wikibooks.org/w/index.php?title=Benutzer:Kaisersoft
38 http://de.wikibooks.org/w/index.php?title=Benutzer:Klaus_Eifert
39 http://de.wikibooks.org/w/index.php?title=Benutzer:Kopoltra
40 http://de.wikibooks.org/w/index.php?title=Benutzer:Krawi
41 http://de.wikibooks.org/w/index.php?title=Benutzer:Kristjan
42 http://de.wikibooks.org/w/index.php?title=Benutzer:LC
43 http://de.wikibooks.org/w/index.php?title=Benutzer:LaaknorBot
44 http://de.wikibooks.org/w/index.php?title=Benutzer:Liquidat
45 http://de.wikibooks.org/w/index.php?title=Benutzer:LivingShadow
46 http://de.wikibooks.org/w/index.php?title=Benutzer:Mamue

1 Marc-André Aßbrock[47]
1 MarkusHagenlocher[48]
1 Matthäus Wander[49]
3 MichaelFrey[50]
1 Mjchael[51]
1 Mnh[52]
1 Morgenstund[53]
21 NEOatNHNG[54]
1 Neitram[55]
3 Nick-zug[56]
1 Ot[57]
3 Peter200[58]
1 Philipendula[59]
1 Php4you[60]
31 Phrood[61]
1 Pkn[62]
1 Plaicy[63]
24 Prog[64]
1 PsY.cHo[65]
1 Qhx[66]
4 Reseka[67]
1 Roo1812[68]
1 RufusLynx[69]
1 S1[70]
1 STBR[71]

47 http://de.wikibooks.org/w/index.php?title=Benutzer:Marc-Andr%C3%A9_A%C3%9Fbrock
48 http://de.wikibooks.org/w/index.php?title=Benutzer:MarkusHagenlocher
49 http://de.wikibooks.org/w/index.php?title=Benutzer:Matth%C3%A4us_Wander
50 http://de.wikibooks.org/w/index.php?title=Benutzer:MichaelFrey
51 http://de.wikibooks.org/w/index.php?title=Benutzer:Mjchael
52 http://de.wikibooks.org/w/index.php?title=Benutzer:Mnh
53 http://de.wikibooks.org/w/index.php?title=Benutzer:Morgenstund
54 http://de.wikibooks.org/w/index.php?title=Benutzer:NEOatNHNG
55 http://de.wikibooks.org/w/index.php?title=Benutzer:Neitram
56 http://de.wikibooks.org/w/index.php?title=Benutzer:Nick-zug
57 http://de.wikibooks.org/w/index.php?title=Benutzer:Ot
58 http://de.wikibooks.org/w/index.php?title=Benutzer:Peter200
59 http://de.wikibooks.org/w/index.php?title=Benutzer:Philipendula
60 http://de.wikibooks.org/w/index.php?title=Benutzer:Php4you
61 http://de.wikibooks.org/w/index.php?title=Benutzer:Phrood
62 http://de.wikibooks.org/w/index.php?title=Benutzer:Pkn
63 http://de.wikibooks.org/w/index.php?title=Benutzer:Plaicy
64 http://de.wikibooks.org/w/index.php?title=Benutzer:Prog
65 http://de.wikibooks.org/w/index.php?title=Benutzer:PsY.cHo
66 http://de.wikibooks.org/w/index.php?title=Benutzer:Qhx
67 http://de.wikibooks.org/w/index.php?title=Benutzer:Reseka
68 http://de.wikibooks.org/w/index.php?title=Benutzer:Roo1812
69 http://de.wikibooks.org/w/index.php?title=Benutzer:RufusLynx
70 http://de.wikibooks.org/w/index.php?title=Benutzer:S1
71 http://de.wikibooks.org/w/index.php?title=Benutzer:STBR

6 Sarang[72]
1 Sepp[73]
1 Sinn[74]
4 Stefan Kühn[75]
1 Supaari[76]
1 Svglogin1[77]
14 SvonHalenbach[78]
2 ThePacker[79]
1 Tohma[80]
1 Tsor[81]
16 Udm[82]
1 Vanis[83]
1 VolkovBot[84]
1 W!B:[85]
1 Wikimi-dhiann[86]
1 Wikinina[87]
3 Wolfgang H.[88]
7 YourEyesOnly[89]
3 Zumbo[90]
10 Zupftom[91]

72 http://de.wikibooks.org/w/index.php?title=Benutzer:Sarang
73 http://de.wikibooks.org/w/index.php?title=Benutzer:Sepp
74 http://de.wikibooks.org/w/index.php?title=Benutzer:Sinn
75 http://de.wikibooks.org/w/index.php?title=Benutzer:Stefan_K%C3%BChn
76 http://de.wikibooks.org/w/index.php?title=Benutzer:Supaari
77 http://de.wikibooks.org/w/index.php?title=Benutzer:Svglogin1
78 http://de.wikibooks.org/w/index.php?title=Benutzer:SvonHalenbach
79 http://de.wikibooks.org/w/index.php?title=Benutzer:ThePacker
80 http://de.wikibooks.org/w/index.php?title=Benutzer:Tohma
81 http://de.wikibooks.org/w/index.php?title=Benutzer:Tsor
82 http://de.wikibooks.org/w/index.php?title=Benutzer:Udm
83 http://de.wikibooks.org/w/index.php?title=Benutzer:Vanis
84 http://de.wikibooks.org/w/index.php?title=Benutzer:VolkovBot
85 http://de.wikibooks.org/w/index.php?title=Benutzer:W%21B:
86 http://de.wikibooks.org/w/index.php?title=Benutzer:Wikimi-dhiann
87 http://de.wikibooks.org/w/index.php?title=Benutzer:Wikinina
88 http://de.wikibooks.org/w/index.php?title=Benutzer:Wolfgang_H.
89 http://de.wikibooks.org/w/index.php?title=Benutzer:YourEyesOnly
90 http://de.wikibooks.org/w/index.php?title=Benutzer:Zumbo
91 http://de.wikibooks.org/w/index.php?title=Benutzer:Zupftom

Abbildungsverzeichnis

- GFDL: Gnu Free Documentation License. http://www.gnu.org/licenses/fdl.html
- cc-by-sa-3.0: Creative Commons Attribution ShareAlike 3.0 License. http://creativecommons.org/licenses/by-sa/3.0/
- cc-by-sa-2.5: Creative Commons Attribution ShareAlike 2.5 License. http://creativecommons.org/licenses/by-sa/2.5/
- cc-by-sa-2.0: Creative Commons Attribution ShareAlike 2.0 License. http://creativecommons.org/licenses/by-sa/2.0/
- cc-by-sa-1.0: Creative Commons Attribution ShareAlike 1.0 License. http://creativecommons.org/licenses/by-sa/1.0/
- cc-by-2.0: Creative Commons Attribution 2.0 License. http://creativecommons.org/licenses/by/2.0/
- cc-by-2.0: Creative Commons Attribution 2.0 License. http://creativecommons.org/licenses/by/2.0/deed.en
- cc-by-2.5: Creative Commons Attribution 2.5 License. http://creativecommons.org/licenses/by/2.5/deed.en
- cc-by-3.0: Creative Commons Attribution 3.0 License. http://creativecommons.org/licenses/by/3.0/deed.en
- GPL: GNU General Public License. http://www.gnu.org/licenses/gpl-2.0.txt
- LGPL: GNU Lesser General Public License. http://www.gnu.org/licenses/lgpl.html
- PD: This image is in the public domain.
- ATTR: The copyright holder of this file allows anyone to use it for any purpose, provided that the copyright holder is properly attributed. Redistribution, derivative work, commercial use, and all other use is permitted.
- EURO: This is the common (reverse) face of a euro coin. The copyright on the design of the common face of the euro coins belongs to the European Commission. Authorised is reproduction in a format without relief (drawings, paintings, films) provided they are not detrimental to the image of the euro.
- LFK: Lizenz Freie Kunst. http://artlibre.org/licence/lal/de
- CFR: Copyright free use.

- EPL: Eclipse Public License. `http://www.eclipse.org/org/documents/epl-v10.php`

Copies of the GPL, the LGPL as well as a GFDL are included in chapter Licenses[92]. Please note that images in the public domain do not require attribution. You may click on the image numbers in the following table to open the webpage of the images in your webbrower.

92 Kapitel 25 auf Seite 803

1		PD
2	Althepal[93]	PD
3		PD
4	Suricata	GFDL
5		PD
6	Suricata	GFDL
7		PD
8	Suricata	GFDL
9	P. Jaworski (PioM[94]),POLAND/Poznań; , inkscape devolpers	GFDL
10	This image was created by me, Booyabazooka[95]	GFDL
11	Bastie[96] http://www.Bastie.de	GFDL
12		PD
13	Dr. Olaf Hoffmann[97]	PD
14	Dr. Olaf Hoffmann[98]	PD
15	Dr. Olaf Hoffmann[99]	cc-by-sa-3.0
16	Dr. Olaf Hoffmann[100]	PD
17	Dr. Olaf Hoffmann[101]	PD
18	Dr. Olaf Hoffmann[102]	cc-by-sa-3.0
19	FischX	PD
20	FischX[103]	PD
21	User:Marc Mongenet[104] Credits: • User:-xfi-[105] • User:Zscout370[106]	PD
22	Dr. Olaf Hoffmann[107]	PD
23	Dr. Olaf Hoffmann[108]	PD
24	FischX[109]	GFDL
25	Dr. Olaf Hoffmann[110]	PD
26	Dr. Olaf Hoffmann[111]	cc-by-sa-3.0
27	Dr. Olaf Hoffmann[112]	cc-by-sa-3.0

93 http://de.wikibooks.org/wiki/%3Aen%3AUser%3AAlthepal
94 http://de.wikibooks.org/wiki/%3Apl%3AWikipedysta%3APiom
95 http://de.wikibooks.org/wiki/User%3ABooyabazooka
96 http://de.wikibooks.org/wiki/Benutzer%3ABastie
97 http://de.wikibooks.org/wiki/User%3ADoktorchen
98 http://de.wikibooks.org/wiki/User%3ADoktorchen
99 http://de.wikibooks.org/wiki/User%3ADoktorchen
100 http://de.wikibooks.org/wiki/User%3ADoktorchen
101 http://de.wikibooks.org/wiki/User%3ADoktorchen
102 http://de.wikibooks.org/wiki/User%3ADoktorchen
103 http://de.wikibooks.org/wiki/User%3AFischX
104 http://de.wikibooks.org/wiki/User%3AMarc%20Mongenet
105 http://de.wikibooks.org/wiki/User%3A-xfi-
106 http://de.wikibooks.org/wiki/User%3AZscout370
107 http://de.wikibooks.org/wiki/User%3ADoktorchen
108 http://de.wikibooks.org/wiki/User%3ADoktorchen
109 http://de.wikibooks.org/wiki/User%3AFischX
110 http://de.wikibooks.org/wiki/User%3ADoktorchen
111 http://de.wikibooks.org/wiki/User%3ADoktorchen
112 http://de.wikibooks.org/wiki/User%3ADoktorchen

28	Dr. Olaf Hoffmann[113]	cc-by-sa-3.0
29	Dr. Olaf Hoffmann[114]	PD
30	Dr. Olaf Hoffmann[115]	PD
31	Dr. Olaf Hoffmann[116]	PD
32	Dr. Olaf Hoffmann[117]	PD
33	Dr. Olaf Hoffmann[118]	PD
34	Dr. Olaf Hoffmann[119]	PD
35	Dr. Olaf Hoffmann[120]	PD
36	Dr. Olaf Hoffmann[121]	PD
37	Dr. Olaf Hoffmann[122]	PD
38	Dr. Olaf Hoffmann[123]	PD
39	Dr. Olaf Hoffmann[124]	PD
40	Dr. Olaf Hoffmann[125]	PD
41	Dr. Olaf Hoffmann[126]	PD
42	Dr. Olaf Hoffmann[127]	PD
43	Dr. Olaf Hoffmann[128]	PD
44	Dr. Olaf Hoffmann[129]	PD
45	Dr. Olaf Hoffmann[130]	PD
46	Dr. Olaf Hoffmann[131]	PD
47	Bastie[132] http://www.bastie.de/[133], korrigiert von ttbya[134]	GFDL
48	Dr. Olaf Hoffmann[135]	PD
49	Bastie[136]	GFDL
50	Dr. Olaf Hoffmann[137]	cc-by-sa-3.0
51	Dr. Olaf Hoffmann[138]	PD
52	Bastie[139]	GFDL

113 http://de.wikibooks.org/wiki/User%3ADoktorchen
114 http://de.wikibooks.org/wiki/User%3ADoktorchen
115 http://de.wikibooks.org/wiki/User%3ADoktorchen
116 http://de.wikibooks.org/wiki/User%3ADoktorchen
117 http://de.wikibooks.org/wiki/User%3ADoktorchen
118 http://de.wikibooks.org/wiki/User%3ADoktorchen
119 http://de.wikibooks.org/wiki/User%3ADoktorchen
120 http://de.wikibooks.org/wiki/User%3ADoktorchen
121 http://de.wikibooks.org/wiki/User%3ADoktorchen
122 http://de.wikibooks.org/wiki/User%3ADoktorchen
123 http://de.wikibooks.org/wiki/User%3ADoktorchen
124 http://de.wikibooks.org/wiki/User%3ADoktorchen
125 http://de.wikibooks.org/wiki/User%3ADoktorchen
126 http://de.wikibooks.org/wiki/User%3ADoktorchen
127 http://de.wikibooks.org/wiki/User%3ADoktorchen
128 http://de.wikibooks.org/wiki/User%3ADoktorchen
129 http://de.wikibooks.org/wiki/User%3ADoktorchen
130 http://de.wikibooks.org/wiki/User%3ADoktorchen
131 http://de.wikibooks.org/wiki/User%3ADoktorchen
132 http://de.wikibooks.org/wiki/Benutzer%3ABastie
133 http://www.bastie.de/
134 http://de.wikibooks.org/wiki/User%3Attbya
135 http://de.wikibooks.org/wiki/User%3ADoktorchen
136 http://de.wikibooks.org/wiki/Benutzer%3ABastie
137 http://de.wikibooks.org/wiki/User%3ADoktorchen
138 http://de.wikibooks.org/wiki/User%3ADoktorchen
139 http://de.wikibooks.org/wiki/Benutzer%3ABastie

53	Dr. Olaf Hoffmann[140]	cc-by-sa-3.0
54	Bastie[141] http://www.bastie.de/[142]	GFDL
55	Dr. Olaf Hoffmann[143]	cc-by-sa-3.0
56	Dr. Olaf Hoffmann[144]	PD
57	Dr. Olaf Hoffmann[145]	PD
58	Dr. Olaf Hoffmann[146]	PD
59	Dr. Olaf Hoffmann[147]	PD
60	Dr. Olaf Hoffmann[148]	PD
61	Dr. Olaf Hoffmann[149]	PD
62	Philip Tregoning	PD
63	Philip Tregoning	PD
64	Dr. Olaf Hoffmann[150]	PD
65	Dr. Olaf Hoffmann[151]	PD
66	Dr. Olaf Hoffmann[152]	cc-by-sa-3.0
67	Dr. Olaf Hoffmann[153]	PD
68	Dr. Olaf Hoffmann[154]	PD
69	Dr. Olaf Hoffmann[155]	PD
70	Dr. Olaf Hoffmann[156]	PD
71	Dr. Olaf Hoffmann[157]	PD
72	Dr. Olaf Hoffmann[158]	PD
73	Dr. Olaf Hoffmann[159]	PD
74	Dr. Olaf Hoffmann[160]	PD
75	Dr. Olaf Hoffmann[161]	PD
76	Dr. Olaf Hoffmann[162]	PD
77	Dr. Olaf Hoffmann[163]	PD
78	Dr. Olaf Hoffmann[164]	PD

140 http://de.wikibooks.org/wiki/User%3ADoktorchen
141 http://de.wikibooks.org/wiki/Benutzer%3ABastie
142 http://www.bastie.de/
143 http://de.wikibooks.org/wiki/User%3ADoktorchen
144 http://de.wikibooks.org/wiki/User%3ADoktorchen
145 http://de.wikibooks.org/wiki/User%3ADoktorchen
146 http://de.wikibooks.org/wiki/User%3ADoktorchen
147 http://de.wikibooks.org/wiki/User%3ADoktorchen
148 http://de.wikibooks.org/wiki/User%3ADoktorchen
149 http://de.wikibooks.org/wiki/User%3ADoktorchen
150 http://de.wikibooks.org/wiki/User%3ADoktorchen
151 http://de.wikibooks.org/wiki/User%3ADoktorchen
152 http://de.wikibooks.org/wiki/User%3ADoktorchen
153 http://de.wikibooks.org/wiki/User%3ADoktorchen
154 http://de.wikibooks.org/wiki/User%3ADoktorchen
155 http://de.wikibooks.org/wiki/User%3ADoktorchen
156 http://de.wikibooks.org/wiki/User%3ADoktorchen
157 http://de.wikibooks.org/wiki/User%3ADoktorchen
158 http://de.wikibooks.org/wiki/User%3ADoktorchen
159 http://de.wikibooks.org/wiki/User%3ADoktorchen
160 http://de.wikibooks.org/wiki/User%3ADoktorchen
161 http://de.wikibooks.org/wiki/User%3ADoktorchen
162 http://de.wikibooks.org/wiki/User%3ADoktorchen
163 http://de.wikibooks.org/wiki/User%3ADoktorchen
164 http://de.wikibooks.org/wiki/User%3ADoktorchen

79	Dr. Olaf Hoffmann[165]	PD
80	Dr. Olaf Hoffmann[166]	PD
81	Dr. Olaf Hoffmann[167]	PD
82	Dr. Olaf Hoffmann[168]	cc-by-sa-3.0
83	Dr. Olaf Hoffmann[169]	PD
84	Dr. Olaf Hoffmann[170]	PD
85		PD
86		PD
87		PD
88		PD
89	Bastie[171] http://www.Bastie.de	GFDL
90		PD
91		PD
92		PD
93		PD
94		PD
95		PD
96		PD
97		PD
98	Bastie[172] http://www.Bastie.de	GFDL
99		PD
100		PD
101		PD
102		PD
103		PD
104		PD
105		PD
106		PD
107	Bastie[173] http://www.Bastie.de	GFDL
108		PD
109		PD
110		PD
111		PD
112		PD
113		PD
114		PD
115		PD
116		PD
117		PD
118		PD
119		PD

165 http://de.wikibooks.org/wiki/User%3ADoktorchen
166 http://de.wikibooks.org/wiki/User%3ADoktorchen
167 http://de.wikibooks.org/wiki/User%3ADoktorchen
168 http://de.wikibooks.org/wiki/User%3ADoktorchen
169 http://de.wikibooks.org/wiki/User%3ADoktorchen
170 http://de.wikibooks.org/wiki/User%3ADoktorchen
171 http://de.wikibooks.org/wiki/Benutzer%3ABastie
172 http://de.wikibooks.org/wiki/Benutzer%3ABastie
173 http://de.wikibooks.org/wiki/Benutzer%3ABastie

120		PD
121		PD
122		PD
123		PD
124		PD
125		PD
126		PD
127		PD
128		PD
129		PD
130		PD
131		PD
132		PD
133		PD
134		PD
135		PD
136		PD
137		PD
138		PD
139		PD
140		PD
141		PD
142		PD
143		PD
144		PD
145		PD
146		PD
147		PD
148		PD
149		PD
150		PD
151		PD
152		PD
153		PD
154	FischX[174]	GFDL
155		PD
156	FischX[175]	PD
157		PD
158	Chris Chittleborough[176]	GFDL
159		PD
160		PD
161		PD
162	Chris Chittleborough[177]	GFDL
163		PD
164		PD
165		PD

174 http://de.wikibooks.org/wiki/User%3AFischX
175 http://de.wikibooks.org/wiki/User%3AFischX
176 http://de.wikibooks.org/wiki/User%3AChris%20Chittleborough
177 http://de.wikibooks.org/wiki/User%3AChris%20Chittleborough

166		PD
167		PD
168		PD
169		PD
170		PD
171		PD
172		PD
173		PD
174		PD
175		PD
176		PD
177	FischX[178]	PD
178		PD
179		PD
180		PD
181	Dr. Olaf Hoffmann[179]	PD
182	Dr. Olaf Hoffmann[180]	PD
183	Dr. Olaf Hoffmann[181]	PD
184	Dr. Olaf Hoffmann[182]	PD
185	Dr. Olaf Hoffmann[183]	PD
186	Dr. Olaf Hoffmann[184]	PD
187	Dr. Olaf Hoffmann[185]	PD
188	Dr. Olaf Hoffmann[186]	cc-by-sa-3.0
189		PD
190		PD
191		PD
192		PD
193		PD
194		PD
195		PD
196		PD
197		PD
198		PD
199		PD
200		PD
201		PD
202		PD
203		PD
204		PD
205		PD
206		PD

178 http://de.wikibooks.org/wiki/User%3AFischX
179 http://de.wikibooks.org/wiki/User%3ADoktorchen
180 http://de.wikibooks.org/wiki/User%3ADoktorchen
181 http://de.wikibooks.org/wiki/User%3ADoktorchen
182 http://de.wikibooks.org/wiki/User%3ADoktorchen
183 http://de.wikibooks.org/wiki/User%3ADoktorchen
184 http://de.wikibooks.org/wiki/User%3ADoktorchen
185 http://de.wikibooks.org/wiki/User%3ADoktorchen
186 http://de.wikibooks.org/wiki/User%3ADoktorchen

207		PD
208		PD
209	• UNKNOWN TEMPLATE F Flag_of_the_United_States.svg	PD
210	FischX[187]	PD
211	FischX[188]	PD
212	User:Dbenbenn[189] and User:Pumbaa80[190]	GFDL
213	FischX[191]	PD
214		PD
215		PD
216		PD
217		PD
218		PD
219		PD
220		PD
221		PD
222		PD
223		PD
224		PD
225		PD
226		PD
227		PD
228		PD
229		PD
230		PD
231		PD
232		PD
233		PD
234		PD
235		PD
236		PD
237		PD
238		PD
239		PD
240		PD
241		PD
242		PD
243		PD
244		PD
245		PD
246		PD
247		PD
248		PD

187 http://de.wikibooks.org/wiki/User%3AFischX
188 http://de.wikibooks.org/wiki/User%3AFischX
189 http://de.wikibooks.org/wiki/User%3ADbenbenn
190 http://de.wikibooks.org/wiki/User%3APumbaa80
191 http://de.wikibooks.org/wiki/User%3AFischX

249	Dr. Olaf Hoffmann[192]	cc-by-sa-3.0
250		PD
251		PD
252		PD
253		PD
254		PD
255		PD
256		PD
257		PD
258		PD
259		PD
260		PD
261	Dr. Olaf Hoffmann[193]	cc-by-sa-3.0
262		PD
263		PD
264		PD
265		PD
266		PD
267		PD
268		PD
269		PD
270	Dr. Olaf Hoffmann[194]	cc-by-sa-3.0
271		PD
272	FischX[195]	GFDL
273		PD
274		PD
275		PD
276		PD
277		PD
278		PD
279		PD
280		PD
281		PD
282		PD
283		PD
284		PD
285		PD
286		PD
287		PD
288		PD
289		PD
290		PD
291		PD
292		PD
293		PD
294		PD

192 http://de.wikibooks.org/wiki/User%3ADoktorchen
193 http://de.wikibooks.org/wiki/User%3ADoktorchen
194 http://de.wikibooks.org/wiki/User%3ADoktorchen
195 http://de.wikibooks.org/wiki/User%3AFischX

295		PD
296		PD
297		PD
298	Dr. Olaf Hoffmann[196] (SVG, Hintergrund); Joachim Ringelnatz (Gedicht, siehe metadata im Dokument)	PD
299	Dr. Olaf Hoffmann[197]	PD
300		PD
301		PD
302	Arne Nordmann (norro[198])	GFDL
303	Хрюша[199]	cc-by-sa-3.0
304		PD
305		PD
306		PD
307		PD
308		PD
309		PD
310		PD
311		PD
312		PD
313		PD
314		PD
315		PD
316		PD
317		PD
318		PD
319		PD
320		PD
321		PD
322		PD
323		PD
324		PD
325		PD
326		PD
327		PD
328		PD
329		PD
330		PD
331		PD
332		PD
333		PD
334		PD
335		PD
336		PD
337		PD
338		PD
339		PD

196 http://de.wikibooks.org/wiki/User%3ADoktorchen
197 http://de.wikibooks.org/wiki/User%3ADoktorchen
198 http://de.wikibooks.org/wiki/User%3ANorro
199 http://de.wikibooks.org/wiki/User%3AChriusha

340		PD
341		PD
342		PD
343		PD
344		PD
345		PD
346		PD
347		PD
348		PD
349		PD
350		PD
351		PD
352		PD
353		PD
354		PD
355		PD
356		PD
357		PD
358		PD
359		PD
360		PD
361		PD
362		PD
363		PD
364		PD
365		PD
366		PD
367		PD
368		PD
369		PD
370		PD
371		PD
372		PD
373		PD
374		PD
375		PD
376		PD
377		PD
378		PD
379		PD
380		PD
381		PD
382		PD
383		PD
384		PD
385		PD
386		PD
387		PD
388		PD
389		PD

25 Licenses

25.1 GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow. TERMS AND CONDITIONS 0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion. 1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work. 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary. 3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures. 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee. 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

* a) The work must carry prominent notices stating that you modified it, and giving a relevant date. * b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices". * c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it. * d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate. 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

* a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange. * b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge. * c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b. * d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements. * e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying. 7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

* a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or * b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or * c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or * d) Limiting the use for publicity purposes of names of licensors or authors of the material; or * e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or * f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way. 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10. 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so. 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it. 11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law. 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program. 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such. 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of

acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version. 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, your program’s commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

25.2 GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed. 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "freeïn the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference. 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Versionöf the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A SSecondary Sectionïs a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The Ïnvariant Sectionsäre certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Textsäre certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent"copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparentïs called Öpaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page"means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page"means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The "publisher"means any person or entity that distributes copies of the Document to the public.

A section Ëntitled XYZ"means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as Äcknowledgements", "Dedications", Ëndorsements", or "History".) To "Preserve the Titleöf such a section when you modify the Document means that it remains a section Ëntitled XYZäccording to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License. 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies. 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document’s license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document. 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

* A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. * B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. * C. State on the Title page the name of the publisher of the Modified Version, as the publisher. * D. Preserve all the copyright notices of the Document. * E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. * F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. * G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice. * H. Include an unaltered copy of this License. * I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "Historyïn the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. * J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "Historyßection. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. * K. For any section Entitled Äcknowledgementsör "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. * L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. * M. Delete any section Entitled Ëndorsements". Such a section may not be included in the Modified Version. * N. Do not retitle any existing section to be Entitled Ëndorsementsör to conflict in title with any Invariant Section. * O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled Ëndorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version. 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "Historyïn the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled Äcknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled Ëndorsements". 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document. 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ägggregateïf the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate. 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled Äcknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title. 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it. 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ör any later versionäpplies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Document. 11. RELICENSING

"Massive Multiauthor Collaboration Site"(or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration"(or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA"means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

Ïncorporate"means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is ëligible for relicensingïf it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing. ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright (C) YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with . . . Texts."line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

25.3 GNU Lesser General Public License

GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below. 0. Additional Definitions.

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work. 1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL. 2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

* a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or * b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

* a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the object code with a copy of the GNU GPL and this license document.

4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

* a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License. * b) Accompany the Combined Work with a copy of the GNU GPL and this license document. * c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document. * d) Do one of the following: o 0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source. o 1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user’s computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version. * e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

* a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License. * b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy’s public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.