XML » SVG Presenter

Strukturierte Multimedia-Präsentation im Web

Diese Diplomarbeit wurde erstellt
im Wintersemester 2002/2003,
Fachbereich Digitale Medien,
Studiengang Medieninformatik
an der Fachhochschule Furtwangen.

Autor: Till Voswinckel

Referent: Prof. Dr. Günter Hentrich
Koreferent: Prof. Nikolaus Hottong

Danksagung

Vor Beginn möchte ich mich natürlich erst einmal bei all jenen bedanken, die – direkt oder indirekt – zum Entstehen dieser Diplomarbeit beigetragen haben und mich auf verschiedenste Weise in deren Verlauf unterstützt haben:

Zunächst einmal danke ich Henner Henkel von der Hamburger Firma *surver:net* für die Inspiration und die konstruktiven Gespräche zu Beginn meiner Arbeit – von Seiten der Fachhochschule Furtwangen bedanke ich mich auch ganz herzlich bei Herrn Prof. Hentrich die Motivation wie auch Herrn Prof. Hottong für seine spontane, freundliche Bereitschaft, diese Arbeit mit zu betreuen.

Insbesondere gilt mein Dank hingegen Slavica Došenovic und natürlich meinen Eltern, ohne deren Geduld und Unterstützung diese Arbeit nicht zustande gekommen wäre.

Eidesstattliche Erklärung

Ich, Till Voswinckel, erklären hiermit an Eides statt,
dass ich die vorliegende Diplomarbeit selbständig
und ohne unzulässige fremde Hilfe angefertigt habe.

Alle verwendeten Quellen und Hilfsmittel sind angegeben.

Friedrichshafen, den 27. Februar 2003

Inhaltsverzeichnis

Anhang und Verzeichnisse

1 Einführung

1.1 Multimedia-Formate und Präsentation

1.1.1 Unüberschaubare Vielfalt

> *In ihrer täglichen Arbeit benötigen Entwickler und Multimedia-Experten häufig ein detailliertes Verständnis der populärsten Techniken. Der größte Anteil der heute zu findenden Literatur ist allerdings entweder zu umfassend oder […] in einer sehr engen Sichtweise beschrieben.*
>
> [Stei99:113]

Über diese lapidare Feststellung des „Multimedia-Experten" [Bach02:4] Ralf Steinmetz „stolperte" ich förmlich noch während der vorbereitenden Recherche zu dieser Diplomarbeit – trifft seine Aussage doch nicht nur den Kern der Medieninformatik selbst, nämlich die Ergründung und Hinterfragung multimedialer und „kommunikativer" Formate, sondern stellt überdies nahezu eine präzise Beschreibung der Situation dar, der ich, und mit mir sicherlich die meisten meiner Kommilitonen, fast täglich gegenüberstehe: Dem „kontinuierlichen Kampf" [Hues97:140][1] mit sowohl den technischen, als auch ästhetischen Möglichkeiten und Beschränkungen zahlloser Medienformate, die gerade aufgrund ihrer Neuartigkeit bedauerlicherweise in der Literatur mitunter nur dürftig kommentiert sind:

> *Diesem Problem begegnen wir ganz besonders in der Informationstechnologie, in der wir fast täglich mit neuen Standards, Programmiersprachen oder sonstigen Willkürlichkeiten der Software-Hersteller rechnen müssen. Datei-Formate überfluten uns derart, dass wir mit den Kenntnissen darüber fast nicht mehr hinterher kommen. „Kannst Du mir mal den Unterschied zwischen einem JPEG, einem PNG und einem GIF erklären?" Diese Frage wurde mir bereits mehrfach gestellt.*
>
> [Fibi01:1]

Insbesondere interessant erscheint jedoch nun unter dem Blickwinkel der Medieninformatik nicht ausschließlich die Frage, wie sich technische Aspekte einzelner, speziell Grafik- bzw. präsentationsorientierter Formate auf das Design praktischer Implementierungen konkret anwenden lassen, sondern ebenso, wie die Möglichkeiten des jeweiligen Formates, sowie dessen Dateistruktur und u.U. das Anwendungsprinzip eines eventuell assoziierten Autorenwergzeugs Ästhetik, Struktur und sogar Inhalt des Ergebnisses direkt beeinflussen können: So steht ja mittlerweile außer Frage, dass etwa die Ästhetik derzeitiger Web-Anwendungen sehr deutlich durch die Formatbeschränkungen der zugrunde liegenden HTML-Sprache geprägt ist.

Aufgrund einer nunmehr schier „unübersichtlichen Vielfalt" [Fibi01] ebenfalls Web-orientierter Formate, die sich speziell auf die *Präsentationsaspekte* des zuvor designtechnisch „restriktiven"[2] HTML-Webs konzentrieren, stellt sich hingegen die interessante Frage, ob Gestaltungsprinzipien und „Design-Frameworks" denn überhaupt noch mit den immens gesteigerten, formattechnischen Möglichkeiten mithalten können:

> *The problem is that technological advances […] have created an overwhelming space of design choices… Without a deep understanding of how to make these design choices, designers will be forced to search this growing design space in an ad hoc manner…*
>
> [Mack88:179]

---

1 „Der Entwurf ansprechender Web-Seiten war ein ständiger Kampf gegen die wenigen Gestaltungsmöglichkeiten von HTML…" [Hues97] p.140

2 „The [XHTML Basic Specification] was [...] the most restrictive language for coding web-pages to date, since it includes only a few necessary elements, that the major industrial actors could agree upon", aus: Eric Kafé: „Web Design." *MegaDoc*, Kopenhagen 2001

1.1.2 Fortschritt und Design

Dieses auch von Lori Scarlatos im Rahmen eines ACM-Panels[1] diskutierte Problem bislang „fehlender, auf diese neuartigen Möglichkeiten zugeschnittener Design-Prinzipien“ [Scar97:215] stellt in meinen Augen jedoch zugleich eine interessante Herausforderung und Chance dar, die in diesem Bereich noch „offensichtlich klaffende Lücke“ [Mack88] mit „Leben zu füllen“. Insbesondere die neuen Grafik- und Multimediaformate des WWW eröffnen in dieser Hinsicht nicht nur der Verbreitung, sondern eben und speziell auch der Gestaltung multimedialer Präsentationen an dieser Stelle neue Möglichkeiten:

> *The blending of technology and art has always turned me on. I think that's why the Web originally seduced me, and why I've stuck around so long. Many Web designers and developers share this attraction. The two sides of our brains can have an affair, and that makes for a fun and rarely boring relationship… for instance, Scalable Vector Graphics (SVG) is a perfect example of technology and design meeting on a level playing field.*
>
> [Holz01]

1.1.3 Präsentationsformate: Marktsituation und Design

Gerade angesichts der mittlerweile zahlreichen, teils sogar konkurrierenden Formate, die eine derartige „Ehe von Design und Technologie“ [Holz01] insbesondere im Rahmen Web-basierter Präsentationen möglich machen, ist jedoch im Gegensatz zu der soeben diskutierten *Vielfalt* und den damit einhergehenden Möglichkeiten am „Markt“ derzeit ein paradoxes, da genau entgegengesetztes Phänomen zu beobachten, nämlich einer „Formattechnischen Monokultur“: So führen konzeptionell interessante Formate wie SVG trotz allgemeiner Standardisierung [SVG01] eine derzeit eher „traurige Randexistenz“ und können nur in „akademischen Nischen“ bescheidene Erfolge feiern,[2] während die „breite Masse“ [Rutl99] hingegen nahezu ausschließlich auf quasi-monopolistische technisch jedoch eher „problematische“ Software wie etwa Microsofts *PowerPoint* [s.2.3] zurückgreift.

Ähnlich der im Rahmen einer Diplomarbeit von Stefan Dilger und Sebastian Iffländer [vgl. IfDi02] gewonnenen Erkenntnis, dass trotz der Existenz überzeugender Technologien (im angesprochenen Falle etwa einer 3D-Engine) im professionellen Bereich[3] bislang „recht veraltete Funktionaliät“ zum Einsatz kommt,[4] so kann im Rahmen dieser Diplomarbeit auch für den Bereich geschäftsorientierter Business-Präsentationen beobachtet werden, dass die wie selbstverständlich angewandten „Standard“-Softwareprodukte[5] derzeit weit noch weit hinter dem eigentlich Möglichen zurückbleiben.[6]

Obgleich insbesondere das professionelle bzw. Business-Marktsegment ja zweifelsohne über die entsprechenden finanziellen Kapazitäten verfügt, dominieren durch die Anwendung in weiten Bereichen „unbefriedigender“ Software-Produkte [s.2.3, 4.5.4] dennoch weithin „unprofessionell wirkende Ergebnisse“ [vgl. Godi01] das Erscheinungsbild geschäftlicher Präsentationen. Die „eigenwillige“ Ästhetik[7] insbesondere PowerPoint-basierter Gestaltungen verleitete selbst Alan Cooper zu der folgenden, „legendären“ Feststellung:

> *The word "design" [seems to be] toxic in the world of business.* [Ande01]

---

1 vgl. [Scar97]

2 Anm: Im Falle von SVG ist dies etwa der bereich der Internet-Kartographie, während *Flash* (noch) sämtliche andere Bereiche abdeckt.

3 Anm: Im Rahmen der angesprochenen Diplomarbeit [IfDi02] ist dies etwa der Bereich der 3D- Prozessvisualisierung.

4 vgl. [IfDi02] p.14

5 Anm: „Standard“ hier im Sinne von normativen *de-Facto*-Standards, nicht offiziell verabschiedeten Normen, etwa von Seiten der ISO oder des W3C. [vgl. NeWi00]

6 s. hierzu insbesondere 2.3.1-3

7 vgl. [Godi01] p.3

Aufgrund dieser Erkenntnis erscheint es daher im Rahmen dieser Diplomarbeit sinnvoll, bereits vorhandene bzw. ggf. noch in der Entwicklung befindliche, zukunftsweisende Multimedia-Formate auf ihre Eignung hinsichtlich *geschäftsorientierter Präsentationen* zu untersuchen, um diese eventuell als mögliche Alternativen zu den derzeitig unbefriedigenden Ansätzen zu positionieren und auch für die „PowerPoint-geplagte“ [Stew01] Business-Welt nutzbar zu machen.

1.2 Definitionen

1.2.1 Formate und Dateien

Bei der Betrachtung dieser Formate schein hingegen zunächst die Überlegung angebracht, was denn nun ein solches Format überhaupt darstellt, und welche Komponenten und Kriterien dementsprechend beleuchtet werden müssen: Zunächst einmal stellt ein (Multimedia-)Format ja nichts anderes dar als einen geschlossenen (Datei-)*Container*, ggf. auch in Gestalt eines (offenen) *Streams* [s.3.5.1], welcher die verschiedenen Inhalte und visuellen Elemente in Form einer vordefinierten Struktur ablegt und somit die Speicherung sowie Analyse („Parsing“ bzw. Rekonstruktion) einer Präsentation (oder simpler: einer einzelnen Grafik) ermöglicht. Die maßgeblichen Kriterien zur Konzeption eines solchen Formats werden somit in erster Linie durch die im Rahmen des zugrunde liegenden Grafikmodells mögliche, zumeist visuelle *Funktionalität* determiniert – die entsprechende Speicherstruktur sollte jedoch zugleich möglichst „einfach“ und „sauber“ aufgebaut sein. Insbesondere die Trennung semantischer bzw. textlicher Inhalte von ihrer eigentlichen, visuellen Darstellung spielt in diesem Zusammenhang speziell bei höheren („high-level“-) Formaten eine wichtige Rolle: Auf diese Weise ließen sich etwa im Rahmen eines entsprechend einfach und intuitiv ausgelegten Formats die gesamten Präsentationsdaten relativ komfortabel „über einen einfachen Text-Editor“ [NSW02:219] erstellen und manipulieren.

Im Hinblick auf die Erstellung bzw. Bearbeitung der eigentlichen Präsentation kommen natürlich an dieser Stelle eventuell auf dem jeweiligen Format aufbauende Bearbeitungswerkzeuge, oder auch „Authoring-Tools“ ins Spiel: Bei möglichst vollständiger Ausnutzung der durch das zugrunde liegende Format bereitgestellten Funktionalität sollten diese natürlich sowohl möglichst komfortabel („usable“) und „mächtig“ sein – besonders interessant ist hingegen im Rahmen dieser Diplomarbeit die Frage nach der *Qualität* der Ergebnisse: „Erzwingt“ das entsprechende Framework ästhetisch, funktional wie auch inhaltlich überzeugende Präsentationen, oder bringt es aufgrund eines problematischen Anwendungsprinzips eher zweifelhafte Resultate hervor? Nicht zuletzt spielt freilich auch die *Positionierung* des entsprechenden Formats eine Rolle: Hierbei stellen naturgemäß *offene*, frei verfügbare Formatspezifikationen speziell aufgrund flexiblerer Präsentations-Erstellung[1] erheblich vorteilhaftere Ansätze dar als „proprietäre Speicherformate“, da letztere einerseits in der Regel fest an kommerzielle (und mitunter sehr teuere) Produkte gebunden sind, andererseits jedoch zumeist lediglich ein „wenig strukturiertes Speicherabbild“ [Bryn99] des jeweiligen Programms darstellen und somit aus formattechnischer Sicht nur begrenzt interessant erscheinen.

Schließlich ist nicht zuletzt die Erkenntnis, dass ein Format weit mehr darstellt als „abgespeicherte Programmdaten,“ Bestandteil dieser Betrachtungen: Neben funktionellen Möglichkeiten, konzeptionellem und technischem Aufbau sowie „architektonischen“ Grundprinzipien der Formate sind natürlich auch *markttechnische Betrachtungen* von Belang, die auch die tatsächliche Verbreitung und einen entsprechenden „Erfolg“ des Formats speziell im Rahmen des WWW berücksichtigen: Insbesondere angesichts der zentralen Rolle der HTML-Browser für das Internet spielt in diesem Zusammenhang natürlich auch die Integration des Formats in eine entsprechende Web-Umgebung eine nicht zu vernachlässigende Rolle.

[1] Anm: Auf diesem Wege können Präsentationen nicht nur durch ein festgelegtes Autoren-Tool, sondern auf vielen verschiedenen Wegen (Alternativ-Programme, Filter-Schnittstelle, über eine Programmiersprache) erzeugt werden.

1.2.2 Präsentationsbegriff

Diese tendenzielle „Beschränkung“ grenzt natürlich auch die Auswahl der im Hinblick auf „Web-fähige“ Präsentationen geeigneten Formate drastisch ein: So ist zweifellos eine maximale „Multimedialität“ geschäftlicher Präsentationen „grundsätzlich Interessant“ [Jaco02b:42] – zugleich ist der Spielraum für derartige „Effekte“ jedoch „recht begrenzt“:[1] So lässt allein der enge, zeitliche Rahmen [Ziel02], die in diesem Zusammenhang zumeist ebenso eingeschränkte, gestalterische und „multimediale Kompetenz“ [vgl. Scha90, Bult01] der angepeilten „Business-Zielgruppe“, sowie nicht zuletzt das Bandbreiten- und Portability-Problem des Internet selbst nur wenig Platz für aufwendige „Spielereien“ [Jaco02a:41] im Rahmen geschäftsorientierter Vorträge. Nicht zuletzt deswegen werden insbesondere *Sound*-technische Aspekte, im Zusammenhang mit Geschäftspräsentationen „ohnehin mit Vorsicht zu genießen“ [Godi01:7], im Rahmen dieser Diplomarbeit nur am Rande zur Sprache kommen.

An dieser Stelle stellt sich daher die grundsätzliche Frage, was denn eine Präsentation, bzw. ein (insbesondere „Web-fähiges“) Präsentationsformat darstellt: Angesichts der Tatsache, dass eine *Präsentation* ja (ausgehend von deren semantischer Bedeutung)[2] im Prinzip „alles ist, was vorgezeigt, dargestellt wird“ und somit letztendlich „sichtbar“ ist, sollte vor der konkreten Formulierung praktischer Anforderungen an eine solche Präsentation zunächst einmal geklärt werden, welche *Art* von Präsentation bzw. welche inhaltliche Struktur somit in ein Format abgebildet werden soll. Da auf Basis des allgemeinen Präsentations-Verständnisses [vgl. Fisc95][3] ja im Prinzip „alles nur Erdenkliche in einer nicht näher definierten Form“ im Rahmen einer solchen Präsentation dargestellt werden kann, bedarf es an dieser Stelle freilich einer etwas konkreteren Kategorisierung von Inhalt und Darstellungsform: So hat sich neben der klassischen *Multimedia-Präsentation* [vgl. Stei99:11], die beliebige „diskrete“ und „kontinuierliche“ Elemente in der Regel grafisch aufwändig vereint [s.2.1], und des in der Form primär text-orientierter Hypertextdokumente organisierten „HTML-Webs“, welches ursprünglich der strukturierten Präsentation wissenschaftlicher Informationen dienen sollte [s.3.1], insbesondere die allgemein als *Business-Präsentation* bezeichnete, „dramaturgisch strukturierte Argumentationsfolge“ als allgemeines Synonym für den Begriff *Präsentation* durchgesetzt, obgleich diese im Prinzip ja lediglich die „visuelle Komponente“ eines „echten“ Vortrages darstellt [s.2.3]. Trotz der allgemeinen Dominanz der letztgenannten Ausprägung [vgl. Mane99, Stew01] ist es insbesondere Aufgabe dieser Diplomarbeit, die „fragwürdige“ [Park01] Darstellungsform zumeist linear aneinander gereihter „Slides“ kritisch zu beleuchten und bessere Lösungsmodelle zu entwickeln.

Aufgrund dessen stellt die „Business-Präsentation“ zwar die im Hinblick auf das bereits diskutierte „interessante“ Marktpotential den Hauptgegenstand meiner Untersuchungen da – gerade *weil* diese Ausprägungsform jedoch konzeptionell „problematisch“ erscheint [s.2.3], gilt es jedoch, die anderen Komponenten nicht zu vernachlässigen: So stellt (wie bereits diskutiert) ja insbesondere das WWW ein für die Zukunft interessantes Träger- und Verteilermedium multimedialer Präsentationen dar, bringt jedoch zugleich weitere Möglichkeiten, aber auch Probleme mit ins Spiel: Der offensichtliche Konflikt zwischen Präsentations*erstellern* (welcher möglichst dramaturgisch und linear strukturierte „Filme“ als Überzeugungsmittel [vgl. This00:161] im Internet darbieten will), und dem Internet-*Nutzer*, der dem Medium in der Regel *aktiv* spezielle Informationen zu „entlocken“ trachtet, ist an dieser Stelle nur ein Beispiel für die durchaus problematische Anwendung „konventioneller“ Präsentationen im Web.

Daher sollen zunächst alle relevanten Herangehensweisen beleuchtet werden, um so gegebenenfalls Gemeinsamkeiten und Schnittmengen sowohl bei den Formateigenschaften, als auch hinsichtlich der unterschiedlichen Anforderungen von „Ersteller“ bzw. „Nutzer“ der Präsentationen herauszufinden.

---

[1] vgl. [Jaco02b, Niel00]

[2] Anm: Aus dem lateinischen: *Praesentare*

[3] “Presentation is users' display of resources…” [Fisc95]

1.3 Anforderungen und untersuchungsrelevante Aspekte

1.3.1 Web-Gemäß

Zunächst gilt es bei der Betrachtung „web-gemäßer" Präsentationen bei aller dramaturgischen Argumentationsführung natürlich, die entsprechenden Präsentationsinhalte mit Rücksicht auf die speziellen *Usability*-Erfordernisse des *World Wide Web* aufzubereiten: Insbesondere die *Durchsuchbarkeit* der enthaltenen Information, eine Abbildungsmöglichkeit semantischer, logischer und hierarchischer Zusammenhänge der Präsentationsstruktur [BJS94][1] und –Dramaturgie, sowie die so genannte „*navigational intelligence*", also die interaktive Präsentations*steuerung* sind in diesem Zusammenhang von Interesse.

1.3.2 Multimedial

Im Hinblick auf die „emotionale" Relevanz einer Präsentation [vgl. Godi01] ist natürlich speziell die *multimediale Funktionalität* des zugrunde liegenden Formats interessant: So gilt es bei Betrachtung der fraglichen Formate insbesondere deren mediale „Dimensionen" zu erkunden, und etwa auf Interaktivität, Animationsfähigkeit und „Scriptability" zu untersuchen. Auf diese Weise kann unter anderem geklärt werden, ob das Format als „wirkliches Multimedia-Format" zu klassifizieren ist oder aufgrund etwaiger Ein-Dimensionalität (z.B. im Falle von „konventionellem" HTML oder PDF) für anspruchsvollere Präsentationslösungen doch eher ungeeignet erscheint. Darüber hinaus spielen natürlich auch die Darstellungsmöglichkeiten des Formats eine Rolle, etwa die Frage, welche „Elemente" (Rasterformat, Polygone, Bézierkurven…) durch das entsprechende Format abgebildet werden können. Auch die Anzeige von Bilddaten[2] sowie eine Einbettung von Schriftarten („Font-Embedding") erscheinen in diesem Zusammenhang relevant, weisen jedoch bereits auf ein weiteres, wichtiges Kriterium hin: Der „impliziten" Ästhetik des Formats.

1.3.3 Ästhetisch

Obgleich ein unzweifelhaft „weiches" Kriterium, so erscheint die Frage, ob Anwendungsprinzip, Formatstruktur oder durch das Format bereitgestellte Möglichkeiten der entsprechenden Präsentation eine bestimmte Bildästhetik aufprägen, in meinen Augen nicht nur angebracht, sondern im Hinblick auf die „Qualität" der erzielten Ergebnisse sogar notwendig: So kann aufgrund der entsprechenden Eigenschaften, Grundprinzipien oder Funktionalitäten einzelner Formate durchaus eine Verbindung zur Ästhetik der jeweils konkreten Resultate hergestellt werden.

Insbesondere die Forschungsarbeit von Kristiina Karvonen [Karv00] macht an dieser Stelle nicht nur klar, dass die „empfundene Schönheit" maßgeblichen Anteil an der emotionalen Wirkung einer Präsentation hat, sondern nennt überdies jenseits der (sicherlich maßgeblichen) Usability-Debatte, die jedoch „jeglichen Bezug auf ästhetische Traditionen und Grundsätze vermissen lassen",[3] grundsätzliche Kriterien zur Erzielung einer „konstruktiven", da *simplistischen* Ästhetik im Rahmen Web-basierter Präsentationen:

> *There could be a second kind of simplicity in terms of 'design' – clear, and 'clean', but in a stylistic and beautiful way.*
>
> [Karv00:89]

Die eigentliche Frage ist jedoch, inwieweit entsprechende Präsentations-Formate selbst oder ggf. darauf aufsetzende „Grafik-Frameworks" diese hinlänglich bekannten [vgl. SmFa90][4] ästhetischen Prinzipien im Sin-

---

[1] "The 'hyper' in hypermedia is taken to suggest the notion of complex information structuring…" [BJS94]

[2] Anm: So stellt sich im Rahmen des Anzeigesystems etwa die Frage, ob die Formen „geglättet" (d.h. mit Anti-Aliasing) dargestellt werden etc.

[3] "No real reference in UI Design is actually made to the tradition of aesthetics. The aesthetic principles are, for the most, made up ad hoc, without any justification from existing theories of the aesthetic that have been around and available for years." [Karvo00] p.86

[4] "Some of the principles certainly are the old standards: clarity, simplicity, appropriateness of form…" [SmFa96] p.40

ne einer Design-orientierten Präsentationserstellung verfolgen bzw. durch entsprechend bereitgestellte Möglichkeiten [s.5.4.2] oder (im Falle eines zugrunde liegenden Authoring-Tools) ein entsprechendes Anwendungsprinzip den Nutzer, auch und gerade bei eingeschränkter „gestalterischer Kompetenz" [vgl. Pirn01] in die aus ästhetischer Sicht „richtige Richtung" zu lenken vermögen.

Da das Gros der derzeit marktführenden Präsentationsansätze im Gegensatz hierzu jedoch „schlechtes Design" angeblich geradezu „erzwingen" [vgl. Norv99, Niel00], ist bei diesem unerwünschten, da gegenteiligen Fall etwa mit Goldrand verzierter PowerPoint-Präsentationen oder „überbordender" Flash-Animationen [vgl. Lipm00] natürlich ebenso eine genauere Betrachtung notwendig, ob und inwieweit denn das Grundprinzip der jeweiligen Ansätze für diese „Missstände direkt verantwortlich sind" [Godi01:3].

1.3.4 Zugänglich

Ebenso wie das „weiche", eher emotional gefärbte Kriterium der Ästhetik gilt es bei der Betrachtung Präsentationsgeeigneter Formate natürlich ebenso die entsprechend zur Verfügung stehenden (bzw. einfach zu implementierenden) *Schnittstellen* hinsichtlich des Format-Zugriffs („access interfaces") zu untersuchen: Insbesondere angesichts der Tatsache, dass diese Diplomarbeit im Studiengang Medieninformatik erstellt wird, erscheint ein Blick „unter die Motorhaube" der einzelnen Multimedia-Formate durchaus angebracht. Grundsätzlich sollen dahingehende Untersuchungen im Rahmen dieser Diplomarbeit also auch die Frage beantworten, inwieweit die den Formaten zugrunde liegende Dateistruktur bzw. Formatarchitektur den Zugriff auf die Präsentationsdaten bzw. auch deren Konvertierung ermöglicht.

Neben der Betrachtung der generellen „Black-Box"-Schnittstellen, also der Verfügbarkeit allgemeiner (ggf. grafischer) Bearbeitungs-Tools, Dateifilter und Konvertermodule ist natürlich insbesondere „aus Informatiker-Sicht" interessant, ob in einer „hohen Programmiersprache" wie etwa Java direkt („prozedural") auf das entsprechende Format zugegriffen werden kann. Dies könnte dann beispielsweise im Rahmen einer serverbetriebenen Präsentationslösung nutzbar gemacht werden: Existieren nutzbare Module oder gar Access-APIs? Wie sieht die Schnittstelle zur „Selber-Entwicklung" einer derartigen Komponente aus?

1.3.5 Einfach

Nicht nur im Hinblick auf den zuletzt betrachteten Aspekt spielt somit auch ein strukturell logischer und einfacher Aufbau der zu betrachtenden Formate eine nicht unbedeutende Rolle. Somit ist mit „Schlichtheit" im Rahmen dieser Diplomarbeit in der Regel nicht ausschließlich *Usability*, d.h. die einfache Benutzbarkeit eines entsprechenden Authoring-Programms gemeint, sondern insbesondere der Begriff der „*conceptional simplicity*", also eines möglichst „einfachen" Aufbaus der internen Struktur. Ein anschauliches Beispiel hierfür ist die sehr einfache, überschaubare und editierbare Syntax und Struktur des (überdies nicht an ein bestimmtes Authoring-Tool gebundenen) HTML-Standards [vgl. Cagl02:12ff]. Die zusätzlich aus struktureller Sicht interessante Trennung relevanter, inhaltlicher Daten von rein visuellen Aspekten der Präsentation [vgl. Cail97] gilt es an dieser Stelle natürlich ebenso zu betrachten: Da sich der Nutzer bei einem derartigen Ansatz lediglich mit den „eigentlichen Inhalten" auseinandersetzen muss, statt bei einer primär visuell orientierten Lösung „unnötige Zeit mit Gestaltungsfragen zu verschwenden" [vgl. Park01, Wine88], spielt in diesem Zusammenhang natürlich auch die Einfachheit bzw. Effizienz bei der Präsentationserstellung eine Rolle.

1.4 Zielsetzung

An Hand der soeben formulierten Anforderungen galt es im Rahmen der vorliegenden Diplomarbeit, in einer umfassenden Untersuchung der für die Erstellung „Web-gemässer Präsentationen" relevanter Multimedia-Formate und Präsentationslösungen die *Eignung* der jeweiligen Ansätze hinsichtlich der soeben definierten Kriterien zu ermitteln. Ziel dieser Betrachtungen stellt neben der reinen „Eignungsfeststellung" natürlich auch die Beantwortung der Frage nach einem möglichen, „optimalen" Präsentationsformat dar: Exis-

tiert das „ideale Framework“ [ChHu01:483] für multimediale Web-Präsentationen etwa bereits? – oder ist es (im Anschluss an diese Betrachtungen) möglich oder gar notwendig, bereits bestehende Ansätze dahingehend zu erweitern, um letztendlich auf dieser Basis einen zufriedenstellenden Prototypen zu entwickeln?

1.5 Gliederung

In den folgenden Kapiteln werde ich mich daher primär um Beantwortung dieser Fragen bemühen: Kapitel 2 gibt hier zunächst einen Überblick über die Entwicklung allgemeiner Multimedia-Ansätze, –Modelle und Standards. Nach einer entwicklungsgeschichtlichen Betrachtung insbesondere der derzeit wohl dominierenden PowerPoint-Software sollen danach „kritische Eigenschaften“ dieses Ansatzes beleuchtet werden, um schließlich anhand der Untersuchung des zugrunde liegenden Formats, eventueller Zugriffsschnittstellen und insbesondere der Web-Fähigkeit PowerPoint-basierter Präsentationen die Eignung des Microsoft-Marktführers hinsichtlich der soeben formulierten Kriterien zu überprüfen.

Im 3. Kapitel werden dagegen insbesondere Kriterien *Web-basierter* Präsentation betrachtet: Zunächst soll die Entwicklung, „Erweiterung“ und Präsentationsbezogene Eignung des HTML-Formates untersucht werden. Unter Einbeziehung multimedialer Komponenten, Erweiterungen und Zusatztechnologien soll danach ein kurzer Überblick über die bereits erfolgten Bemühungen gegeben werden, das Web als Plattform für „Multimediale Präsentationen nutzbar“ zu machen.

Insbesondere die in diesem Kapitel deutlich werdende Bandbreiten-Problematik des Internet lenkt unsere Aufmerksamkeit im 4. Kapitel dann auf die Entwicklung platzsparender, da *vektorbasierter* Formate: Ausgehend von den grundsätzlichen Eigenschaften vektorieller Grafik soll zunächst die „holprige“ Entwicklung Web-basierter Polygonformate skizziert werden, welche in einer kritischen Analyse des hier marktbeherrschenden *Flash*-Formats schließt.

Als textbasierte und somit zunehmend Web-orientierte Alternative werde ich dann im 5. Kapitel speziell XML-basierter Vektorformate betrachten, insbesondere die Entwicklung und die Eigenschaften des W3C-Vektorstandards SVG, sowie des „Multimedia-Präsentationsformats“ SMIL. Auf dieser Basis werde ich sodann im 6. Kapitel konzeptionelle Kriterien und die eigentliche Umsetzung des daraufhin im Rahmen dieser Diplomarbeit entwickelten Prototypen *XML » SVG Presenter* skizzieren, um die Arbeit dann im 7. Kapitel mit einer kurzen Zusammenfassung samt Ausblick zu abzuschließen.

2 Multimediale Präsentations-Systeme

Zum tieferen Verständnis des Grundkonzepts der „Multimedia-Präsentation“ und der darauf aufbauenden, unterschiedlichen Herangehensweisen ist neben der bereits im 1. Kapitel gegebenen, theoretischen Definition an dieser Stelle insbesondere eine entwicklungsgeschichtliche Betrachtung der verschiedenen Ansätze und damit verbundenen Formate von Interesse. Ohne zu weit den im Rahmen dieses Kapitels erarbeiteten Erkenntnissen vorgreifen zu wollen, sei zur Erläuterung dieser Vorgehensweise jedoch an dieser Stelle bereits erwähnt, dass frühzeitig veröffentlichte Realisierungen und hernach konkurrierende Entwicklungen erst zur theoretischen Formulierung der heute als Grundlage multimedialer Systeme bekannten Multimedia-Modelle geführt haben und nicht, wie man meinen möchte, Kommunikationstheoretische Veröffentlichungen zunächst die Basis darauf aufbauender, praktischer Entwicklungen bildeten. Dieses Reaktionsprinzip begründet sich auf der noch im Anfangsstadium „multimedialer Gehversuche“ offensichtlichen Trägheit akademischer Multimedia-Forschung[1] im Vergleich zur insbesondere zu Beginn der 80er Jahre stark boomenden, kommerziell motivierten Softwareindustrie. Die intensive Kooperation kommerzieller und universitärer Vertreter im Rahmen standardisierender Konsortien wie etwa des W3C sowie die intensiven Bemühungen zahlreicher Forschungsinstitute (hier seien insbesondere das Genfer CERN-Institut sowie das Bostoner MIT erwähnt) haben diesen „industriellen Vorsprung“ im Multimedia-Bereich jedoch längst kompensiert.

2.1 Entwicklungshistorische Betrachtung

2.1.1 Card-based Paradigma: HyperCard

Mit Einführung des *Lisa*-Systems und der anschließenden Entwicklungsarbeit an der grafisch orientierten Macintosh-Benutzerschnittstelle Anfang der 80er Jahre war es jedoch insbesondere das Kalifornische Softwareunternehmen *Apple*, das die frühzeitige Entwicklung Multimedialer Präsentationssysteme vorantrieb.

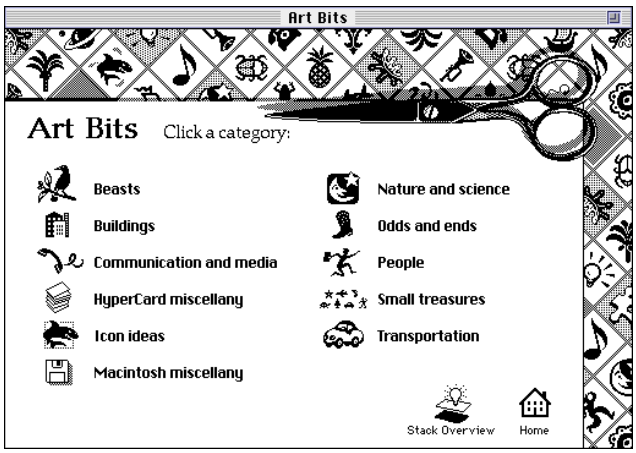


Abb. 2.1.1.1: HyperCard-Anwendung

Nachdem die Apple-Mitarbeiter, zusammen mit den Wissenschaftlern des Xerox PARC-Forschungslabors ihr revolutionäres GUI-Konzept im Rahmen des Apple-eigenen *Finder*-Betriebssystems populär gemacht hatten [vgl. Fors98], konnte das Apple-Entwicklerteam um Bill Atkinson überdies mit *HyperCard* das „erste für größere Anwenderkreise nutzbare“ [Rieh01:29] Multimedia-Autorensystem und zugleich ein revolutionäres Präsentationskonzept einem größeren Publikum zugänglich machen.

Dieses 1987 in Verbindung mit dem Apple Macintosh-Betriebssystem kostenlos ausgelieferte Programm [vgl. Blum98] stellt nicht nur Autoren- und Präsentationssystem zugleich dar, es führte zugleich das Konzept von *Hypermedia* als Komplex miteinander vernetzter, im [nach Stei99] engeren Sinne multimedialer Dokumente ein,[2] wenngleich der Begriff „Hypermedia“ [vgl. GrRa94, Hent98] hierfür erst später größere Verbreitung erfuhr [s.2.2]. Überdies „erfanden“ die HyperCard-Entwickler mit der Bereitstellung der Bear-

[1] Dies begründet sich insbesondere auf der anfangs mangelnden Motivation bzw. eines aus akademischer Sicht fehlenden Bedürfnisses nach Multimedialen Inhalten (Im Ggs. Zur kommerziellen Softwareindustrie).

[2] vgl. [Wild99] p.11

beitungs- und Erstellungskomponente zugleich ein völlig neues Authoring-Paradigma, das sich der (später ebenso in WML integrierten)[1] Metapher eines Kartenstapels (daher "card based paradigma")[2] bedient. Im Gegensatz zu [Mori99], die an dieser Stelle die Einzigartigkeit des HyperCard-Ansatzes anerkennt, fasst [Bole98] das HyperCard'sche Karten-Paradigma mit „ähnlich konzipierten" Anwendungs-Metaphern zu einer Kathegorie der sogenannten „*Frame*-basierten Autorensysteme"[3] zusammen:

> *Frame-basierte Autorensysteme lassen sich dadurch charakterisieren, dass die zu präsentierenden Objekte auf Flächen angeordnet werden, die als Frames, Karten, Seiten, Fenster oder auch Dias bezeichnet werden und die im Prinzip einen Bildschirm repräsentieren, wie ihn ein Benutzer während der Präsentation für einen bestimmten Zeitraum zu sehen bekommt.*
>
> [Bole98]

Da Apple darüber hinaus die einfach zu erlernende,[4] aber zugleich erstaunlich mächtige Skriptsprache *HyperTalk* in den Funktionsumfang des Programms mit aufnimmt, erfreut sich die Anwendung, nicht zuletzt aufgrund der kostenlosen, großflächigen Verbreitung in Verbindung mit dem Macintosh-Betriebssystem, sowohl in kommerziellen und akademischen Kreisen als auch bei Consumer-Anwendern größter Beliebtheit. Dennoch gelingt es dem Authoring-Framework, zumindest im Rahmen der Anfangs verfügbaren Schwarz-Weiß-Version einer breit gefächerten, nur selten professionell versierten Anwenderschaft zu größtenteils überraschend hochqualitativen Ergebnissen zu verhelfen. Dies mag nicht nur an HyperCards vorbildlich konzipiertem Bedienkonzept, sondern unter anderem auch an den optisch überzeugenden „Art Bits"-Grafiken liegen, die dem HyperCard-Programm beilagen [s.Abb.2.1.1] und sowohl zur übrigens lizenzfreien Verwendung als auch ästhetischen Nachahmung anregten.

Im Rahmen dieser Diplomarbeit, die sich ja primär auf die Untersuchung multimedialer *Formate* konzentrieren soll, rückt derweil eine weitere Eigenschaft der HyperCard-Architektur in den Mittelpunkt unseres Interesses: Die Trennung von Ausführungs- und Format-Schicht des Programms. So werden in den von HyperCard erzeugten *Stack*-Dateien stets nur die eigentlichen Präsentationsdaten abgespeichert, da die interpretierende, darstellende und editierende Runtime-Logik des Programms bereits im Rahmen der in der Regel vorinstallierten HyperCard-Anwendung selber vorliegt. Im Gegensatz zu der im Gefolge HyperCards veröffentlichten Authoring-Konkurrenz, die oftmals die ausführende Schicht in das eigentliche Präsentationsformat integriert,[5] liegt bei HyperCard somit ein „echtes" und zudem *transparentes*[6] Format vor. Aufgrund seiner weiten Verbreitung und überzeugenden Architektur, so stellt etwa der „klügste Kopf des Internet" [Depe02:22], Jakob Nielsen, lapidar fest, hätte HyperCard daher leicht zum universalen Standard, wenn nicht gar dem alles beherrschenden Internet-Format schlechthin erwachsen können:

> *If Apple had played its Cards better, HyperCard might have grown into the universal status now owned by the Web […] HyperCard even had a UI design standard – something we are still missing for the Web.*
>
> [Niel99:71]

2.1.2 Timeline-Paradigma: Director

Wie Nielsen allerdings bereits andeutet, kam es dennoch anders: Neben zahlreichen, in unmittelbarer Nachfolge veröffentlichten Konkurrenzprodukten wie etwa *ToolBook* oder *SuperCard*, die auf demselben „Frame"-basieren Authoring-Paradigma beruhten [Mich89, BrDw94], drängte mit Macromedias *Director* ein weiteres Multimedia-Autorentool auf den Markt, das eine gänzlich andere Herangehensweise verfolgte:

---

[1] vgl. [Rieh01] p.29
[2] vgl. [Mori01] p.16ff.
[3] vgl. [Bole99]
[4] vgl. [Rieh01] p.29
[5] Dies liegt oftmals freilich bereits an der mangelnden Verbreitung der ausführenden Runtime-Engine. Beispiel hierfür ist etwa das stets ausführbare Anwendungen exportierende Programm *Director*.
[6] Transparent unter dem Aspekt, dass stets die Möglichkeit einer Weiterverarbeitung besteht.

Ausgangspunkt der zunächst unter dem Namen „Macromind VideoWorks II“ bekannten Software ist im Gegensatz zum konzeptionell interaktiven *HyperCard*-Ansatz noch in der 1989 erschienenen Version „VideoWorks I“ die rein lineare, nicht-interaktiv voranschreitende *2D-Animation* [vgl. HoBo01]. Ursprünglich (worauf bereits der Name hinweist) ausschließlich für die Filmindustrie von Interesse, änderte Macromind-Firmengründer Marc Canter nach dem überwältigenden Erfolg der HyperCard-Software radikal Funktionalität und Anwendungsprinzip des Programms und brachte es schließlich 1990 unter dem Namen *Director 1.0* auf den Markt. Obgleich der potentiellen Anwenderschaft nun primär als „Authoring-Tool zur Erstellung interaktiver Multimedia-präsentationen“ [vgl. Robe95] angedient, wird allein bei Betrachtung der Benutzerschnittstelle die Animations-basierte Herkunft der Software mehr als offensichtlich.

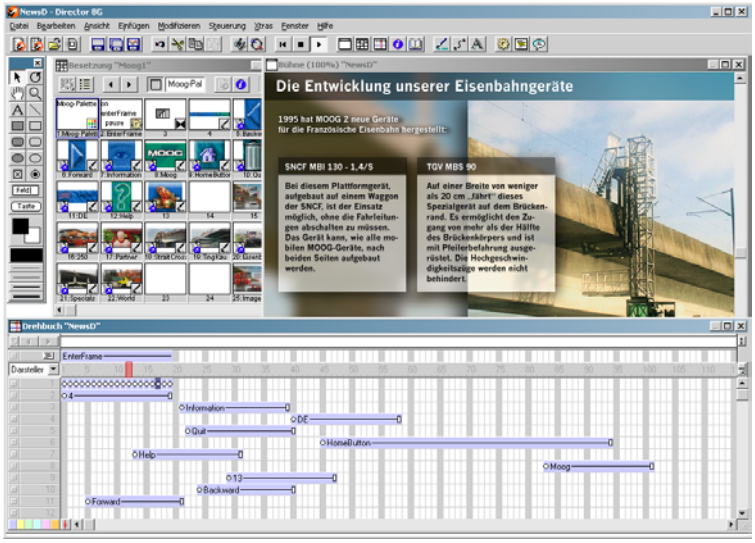


Abb. 2.1.2.1: Director-GUI (rechts)

Im Gegensatz zum rein Karten-basierten HyperCard bildet aufgrund der „filmischen Orientierung“ des Programms nun die *Zeitleiste* das Hauptnavigationselement des Director-Programms. Da dieses „Timeline-basierte“ Authoring-Paradigma [vgl. Bole98] aufgrund der *Zeitkontinuität* [vgl. Stei99:10], d.h. der grundsätzlichen „Unendlichkeit“ zeitlicher Einheiten reichlich problematisch erscheint, behilft sich das Director zugrunde liegende Editing-Modell der bereits aufgrund der Animations-Allegorie nahe liegenden „Frame“-Metapher. (An dieser Stelle wird übrigens auch die irreführende Terminologie in [Bole98] deutlich, wo der hierzu komplementäre Card-Ansatz als „Frame“-Paradigma bezeichnet wird, s.o.). Die kontinuierliche Zeitleiste wird somit diskretisiert, d.h. in einzelne „Bilder“ von jeweils festgelegter, in der Regel freilich nur kurzer Dauer eingeteilt. Aufgrund dieser Vereinfachung können nun auch die räumlichen Elemente bei Selektion einzelner „Frames“, ähnlich wie bereits in den Karten-basierten Autorensystemen direkt-manipulativ editiert werden.[1]

Dass dieses Bearbeitungsprinzip hingegen Interaktionen an sich aufgrund seines grundsätzlichen Prinzips eher *erschwert*, macht allein die Tatsache deutlich, dass die Animation stets kontinuierlich fortschreitet und etwaige interaktive Elemente innerhalb *eines* Frames dem Benutzer wiederum nur Bruchstücke einer Sekunde zur „Verfügung“ stehen, falls diese sich nicht über mehrere „Frames“ der Zeitleiste erstrecken [s. Abb. 2.1.1] Die einzige „Lösung“, diesen Zeitfortschritt anzuhalten und die Präsentation somit wieder in die weitaus vertrautere Karten-Umgebung zurückzuführen, besteht freilich in der Realisierung einer „Stop“-Anweisung, für die man sich bereits an dieser Stelle der integrierten Skriptsprache *Lingo* bedienen muss. Da dies meiner Einschätzung nach nicht nur etwas umständlich, sondern insbesondere auch auf ungeübte Benutzer unnötig anspruchsvoll wirkt [vgl. Bole98],[2] kann die Director-Anwendung zwar im Hinblick auf reine Animationsfunktionalität und lineare, nur rudimentär interaktive Präsentationen durchaus überzeugen, erscheint jedoch auch aufgrund der etwas verqueren „Theater“-Metaphorik[3] für die Erzeugung primär interaktiver Multimedia-Präsentationen in meinen Augen weniger geeignet als das überdies dem Hypertext-Konzept näher stehende Karten-Paradigma.

---

[1] „Die räumliche Anordnung der Objekte auf dem Bildschirm geschieht wie bei den frame-basierten Autorensystemen direkt-manipulativ bzw. über Menüs.“ [Bole98]

[2] „Viele Autorensysteme versuchen, Probleme dadurch zu umgehen, daß sie eine spezielle Programmiersprache in der Verbindungsphase zur Verfügung stellen. Dadurch wird allerdings eine wesentliche Zielsetzung von Autorensystemen, nämlich die Unterstützung von Nicht-Programmierern bei der Entwicklung interaktiver multimedialer Anwendungen, nicht erreicht.“ [ibid]

[3] Darsteller, Bühne, Drehbuch, etc. vgl. [Abb. 2.1.1]

2.1.3 Marktentwicklung

Und dennoch begründet sich das später zum gravierenden Markt*nachteil* erwachsende Problem HyperCards und seiner Derivate [Mich89, BrDw94] ironischerweise gerade in der Schwäche des dem *Card*-Paradigma zugrunde liegenden Hypermedia-Modells. Da dieses, wie bereits [Hard99, HORB99] ausführen, nur unzureichende Funktionalität hinsichtlich der Zeitbasiertheit gerade audiovisueller Medien zur Verfügung stellen kann, verwundert es nicht, dass insbesondere im Rahmen der Einbettung „echter" Multimedia-Komponenten wie etwa animierter Video-Sequenzen[1] in die bestehenden Autorensysteme, das bereits aus konzeptioneller Sicht eher auf zeitbasierte Funktionalität ausgelegte Timeline-Paradigma Directors im Vergleich zum HyperCard-Pendant, welches aufgrund des frühen Veröffentlichungsdatums ursprünglich überhaupt nicht auf Video-Embedding eingerichtet war, seine Vorteile im Zuge der „Multimedia-Revolution" weit besser ausspielen konnte.

Aufgrund dessen schmolz der Marktanteil des Apple-Autorensystems im Verlaufe der 90er Jahre, trotz einiger beeindruckender, auf HyperCard-Basis veröffentlichter Multimedia-Anwendungen [Dana97], daher zusehends dahin. Die darüber hinaus schwindende Unterstützung von Seiten des Mutterkonzerns[2] und der zentrale Zukauf des Vektor-Animationsprogramms *Flash*[3] [Star01:6] durch das mittlerweile unter neuem Namen firmierenden Konkurrenzunternehmen *Macromedia* sicherte schließlich die bis heute andauernde Marktführerschaft des *Director*-Autorenwerkzeugs und des damit verbundenen Timeline-Paradigmas, welches im übrigen darüber hinaus auch in Macromedias mittlerweile ebenfalls marktbeherrschendes Flash-System integriert wurde [s.4.5.1]

Mit dieser Entwicklung einher ging darüber hinaus freilich eine zusehende *Professionalisierung* des multimedialen Authoring-Prozesses: Neben dem „äußerst schwierigen" [Bult01] Produktionsvorgang professionell wirkenden Videomaterials[4] beschränken überdies Preisniveau[5] sowie die bereits erwähnte Komplexität des Director-Anwendungsprinzips die Verfügbarkeit des Autorensystems für semiprofessionelle und *Home*-Anwender, die somit auf kostengünstigere oder frei verfügbare Authoring-Lösungen angewiesen sind. Mit dem „Untergang" HyperCards schuf darüber hinaus die ausgesprochene *Intransparenz* des Director-Formates [vgl. Baja02] ein allgemeines Bedürfnis weiter Anwenderkreise nach universell verwendbaren Multimedia-Standards, welches nicht nur von Seiten der Softwareindustrie, sondern schließlich auch akademischen Kreisen erkannt wurde.

Darüber hinaus hatte die öffentlich ausgetragene Diskussion (s.o.) über das sowohl aus Autoren- als auch Präsentationssicht vorgeblich „optimale" Authoring-Paradigma [vgl. Bole98, Mori01] das Interesse der Kommunikationswissenschaft an der Formulierung zeitgemäßer Multimedia-Modelle geweckt. Aufgrund der Tatsache, dass noch theoretische Konzeption als auch Umsetzung des im Web dominierenden Dokumentenformates HTML [s.3.1], ursprünglich zum Austausch wissenschaftlicher Dokumente konzipiert,[6] der Federführung universitärer Institutionen[7] unterlegen hatte, durch die eingangs beschriebene „Multimedia-Revolution" das Heft des (multimedialen) Handelns jedoch augenscheinlich in die Hand kommerzieller Unternehmen gefallen war, sah sich die „scientific community" Anfang der 90er Jahre schließlich genötigt, endlich theoretische Grundlagen für allgemein gültige Standards zu formulieren, die auch den Bedürfnissen und Eigenschaften multimedialer Anwendungen entsprechen sollten.

---

[1] vgl. [Stei99] pp.10ff.

[2] vgl. hierzu Jim Stephensons abschließenden Kommentar zu dessen „HyperCard Heaven"-Projekt vom 17. November 2001: http://members.aol.com/hcheaven/inactive.html [3.2.03]

[3] s. 4.5

[4] vgl. [Bult01] p. 182

[5] Das Unternehmen selbst beziffert den „Einstiegspreis" des Produkts auf rund 1.200 US-Dollar.

[6] Dieser Umstand machte freilich zugleich den wissenschaftlichen Nutzen dieser Technologie auch der akademischen Welt äußerst transparent.

[7] Anm.: Sowohl HTTP als auch HTML entstanden ja zunächst am Genfer Kernforschunszentrum CERN.

2.2 Multimedia-Standards

2.2.1 Hypertext-Systeme

Ausgangspunkt dieser Überlegungen bildete freilich das bereits 1945 im Rahmen der Entwicklungsarbeit an Vannevar Bushs *Memex*-System entstandene Konzept assoziativ verbundener Textdokumente [Bush45], wenn auch der hierfür geläufige Begriff „Hypertext" erst in den 60er Jahren durch [Nels88] geprägt wurde. Der auf dieser Basis bereits frühzeitig formulierte Begriff „Hyper*media*" erweiterte dieses noch rein textliche Konzept schließlich um die Integration „multimedialer Daten (z.B. Ton und Bild" [Rieh01:23]. Dieser zunächst sehr einfache Multimedia-Begriff schloss mit zunehmender Digitalisierung auch animierten Videomaterials[1] Anfang der 80er Jahre auch audiovisuelle Komponenten mit ein, sodass laut mit Andrew Lippmanns „Aspen Movie Map" das Bostoner Massachusetts Institute of Technology (MIT) die „erste Hypermedia-Anwendung der Geschichte",[2] [vgl. Negr95:86ff.] bzw. die erste verlinkte Multimedia-Anwenung vorstellen konnte. Da insbesondere der hochkomplexe Authoring-Ansatz dieser Anwendung, wie auch der noch folgenden universitären Multimedia-Prototypen[3] jegliche *Usability*-Aspekte jedoch völlig unberücksichtigt ließ, bleib diesen Konzeptstudien freilich eine großflächigere Verbreitung verwehrt – eine Lücke, die, wie in [2.1] beschrieben, erst Apples GUI-basierter HyperCard-Ansatz zu füllen wusste.

Aufgrund des für die akademische Lehre zunächst überraschenden Erfolgs des Apple-Produktes wurde daher zunächst aus theoretischer Sicht versucht, sich der grundsätzlichen Herangehensweise des auch dem HyperCard-Paradigma zugrunde liegenden Hypermedia-Modells anzunähern. Das aus diesen Bemühungen resultierende *Dexter*-Modell [HaSc94] separiert daher den Gesamtkomplex der Hypertext-Systeme in drei verschiedene Schichten[4] und deckt somit die Anforderungen „der meisten Hypertextsysteme" ab [Bole98], lässt jedoch sowohl erweiterte Multimedia-Eigenschaften, Präsentations- und Interaktionsmöglichkeiten[5] zunächst unberücksichtigt. Der zunehmenden Integration zeitkontinuierlicher Medien in die Hypermedia-Anwendungen und dem damit einhergehenden Siegeszug Timeline-basierter Autorensysteme [s.2.1] war jedoch das auf dem eher statischen Hypertext-Konzept aus dem Jahre 1945 beruhende Dexter-Modell schon aus rein konzeptioneller Sicht nicht mehr gewachsen: Zwar war die Einbettung von Audio- und Videomaterial auf einem relativ statischen Hintergrund sozusagen als „Hypermedia-Erweiterung" möglich, darüber hinaus gehende Integration oder Zeitdynamik zwischen audiovisuellem Inhalt und der Hypertext-Umgebung mithilfe des Dexter-Modells jedoch nur erschwert realisierbar.[6] [vgl. Hard99, HORB99]

2.2.2 Erweiterte Modelle

2.2.2.1 Amsterdam Hypermedia-Modell

Auf diesem Hintergrund veröffentlichten Wissenschaftler des Niederländischen Zentrums für Mahematik und Informatik, CWI,[7] schließlich mit dem so genannten *Amsterdam Hypermedia-Modell* [HBR94], ein Framework, das auch die Synchronisation zeitabhängiger (kontinuierlicher) Medien in den Hypermedia-Kontext mit einbezieht und somit die Grundlage für den heutigen, insbesondere im *Streaming*-Bereich interessanten Multimedia-Standard SMIL bildet [s. 5.5]. Wie [Bole98] bemerkt, bleibt der von den AHM-Entwicklern verfolgte Ansatz jedoch „weit hinter den Möglichkeiten moderner Multimediasysteme zurück":

---

[1] vgl. [Rieh98] p.27

[2] vgl. ebenda, p.28

[3] Bspw. Das Hypermedia-System *Intermedia* der Brown University 1985, vgl. [Niel95] pp. 51ff.

[4] Storage layer (Datenbasis), Within-component-layer (Struktur, konkrete Medientypen), Run-time Layer (Zeit, Präsentation) [vgl. Bole95]

[5] „Spezielle Präsentations- und Interaktionsmöglichkeiten werden im Dexter Modell nicht weiter behandelt." [ibid]

[6] "Most hypermedia models and systems do not incorporate time explicitly. This prevents authors from having direct control over the temporal aspects of a presentation." [Hard99]

[7] Niederl.: Centrum voor Wiskunde en Informatica

> *So geben heutige Multimedia-Anwendungen dem Benutzer Interaktionsmöglichkeiten, die über eine Hypermediaanwendung hinausgehen und nicht als eine Teilmenge eines Hypermediasystems eingebunden werden können.*
>
> [Bole98]

Bereits im Rahmen der *Dexter*-Formulierung wurde daher die ausgesprochene Schwierigkeit deutlich, sowohl Interaktion als auch Zeitkontinuität innovativer Multimedia-Anwendungen zufrieden stellend abzudecken – nicht zuletzt, weil die Trennlinie zwischen diesen beiden Komponenten zusehends verwischt:

> *The old line between multi-dimensional hypertext and more linear [animated] multimedia has considerably blurred.*
>
> [Moul02]

Daher erscheint es auch wenig verwunderlich, dass trotz der Bemühungen der „scientific community", an dieser Stelle klare Definitionen und Modelle zu schaffen, die Begriffe Multimedia, Hypertext und Hypermedia laut [Blum98] „oft nicht deutlich voneinander unterschieden werden" und überdies „häufigem Wandel unterworfen" sind.[1]

2.2.2.2 MHEG

Besonders intensiv setzt sich daher die Spezifikation des MHEG-Standards [vgl Stei99:7] mit der Definition sowohl logischer Zusammenhänge als auch des begrifflichen Verständnisses multimedialer Systeme auseinander. Dies erscheint aufgrund der Abstraktheit des Standards selber auch nötig, da sich auf Basis der MHEG-Syntax zwar tatsächlich konkrete Anwendungen realisieren lassen [vgl. EHHP99], das Regelwerk sich jedoch erstlinig auf möglichst universelle Ablauffähigkeit konzentriert, um maximale Portabilität zu garantieren. Der Standard soll somit, wie etwa [Bole98] ausführt, „keine Grundlage für neue Autorensysteme bilden, sondern vielmehr in bestehende Systeme als Ausgabeformat integrierbar sein." Umso interessanter erscheint daher die Tatsache, dass die letztendlichen Implementierungen des MHEG-Standards trotz vorhandener Java-Schnittstellen [Morn98, EHHP99] nicht im Bereich des aufgrund seiner Heterogenität dafür prädestinierten *Internet*, sondern lediglich im Rahmen eines Präsentationssystems für Digitales Fernsehen zu finden sind [Birc99].

2.2.2.3 „Over-Complex" Standards: SRM-IMMPS und HyTime

Das Konzept der *Adaptierbarkeit* hypermedialer Anwendungen greifen wiederum die Wissenschaftler des CWI auf, um das von ihnen formulierte, bereits angesprochenene Amsterdam Hypermedia Modell unter zu Hilfenahme der weiteren, aufgrund ihrer „praktisch unanwendbaren" [ROHB98] Komplexität hier nicht näher erläuterten Hypermedia-Modelle SRM-IMMPS [2] und HyTime [s.3.5.2] im Hinblick auf verbesserte Anpassungsfähigkeit an verschiedene Endgeräte zu erweitern [vgl. RHOB99]. Dieser Ansatz ist zwar zugunsten „maximaler Adaptivität" noch weiter von der eigentlichen Darstellung entfernt als etwa der MHEG-Standard, die zugleich vorgestellte *Fiets*-Anwendung demonstriert jedoch zugleich die tatsächliche Umsetzbarkeit der CWI-Herangehensweise. Die hohe Komplexität des Erstellungsprozesses[3] bei Verwendung allgemein verfügbarer Standards und Modelle,[4] auch dies wird bei Untersuchung der als „prototypisches Vorbild" dargestellten Beispielanwendung in [RHOB99] mehr als deutlich, setzt freilich eine sehr hohe *User Sophistication* vorraus, die zugleich eine großflächige Anwendung der soeben vorgestellten Standards erheblich erschweren sollte.

---

[1] vgl. [Blum98]

[2] Standard Reference Model for Intelligent Multimedia Presentation Systems, vgl. hierzu [Bord97]

[3] vgl. hierzu das entsprechende Schaubild in [RHOB99] p.181

[4] in [RHOB99] kommen etwa SGML, XML, SMIL, HyTime, und DSSL zum Einsatz [vgl. ebenda]

Aufgrund dieser meiner Ansicht nach falsch eingeschätzten Ansprüche, die durch die soeben beschriebenen Hypermedia-Modelle an Software, Anwender und Präsentation gestellt werden, erscheint ein Grossteil dieser Ansätze, obgleich teilweise gar zum Standard erhoben [vgl. MeEf95], für eine Anwendung durch breit gestreute Anwenderkreise, wie noch das verhältnismäßig einfach gestrickte HyperCard erreicht, nur wenig geeignet. Gehen diese doch recht vielschichtig gehaltenen, akademischen Ansätze stets von einem professionellen Gestaltungsansatz aus, der hoch geschultes Personal im Bereich der Systemarchitektur und Konzeption, des Designs und darüber hinaus der Programmierung erfordert, so findet de facto ein „Unterwanderungsversuch" durch einen Präsentations-Ansatz statt, der sowohl von einem weitaus niedrigeren Niveau der „user sophistication", wie auch von einer gänzlich unterschiedlichen Herangehensweise an den Begriff „Multimedia-Präsentation" ausgeht: Die Rede ist, natürlich, von der „auch in weite Teile des Multimedia-Designs vordringenden" [Endi01] Software *PowerPoint* und dem damit verbundenen Ansatz der eigentlich nicht primär auf Bildschirmausgabe und Multimedia-Funktionalität beruhenden *Business-Präsentation*.

2.3 PowerPoint

Bildeten noch bei der soeben betrachteten, „klassischen" Multimedia-Anwendung Bildschirm- sowie Sound-Ausgabe die einzigen verwendeten *Means of Communication*, d.h. die Präsentation ist selbständiges Hauptelement eines zumeist interaktiven Kommunikationsprozesses, so stellt die zumeist linear voranschreitende, wenig interaktive Multimedia-Anwendung im Rahmen der Business-Präsentation lediglich die visuelle Komponente[1] einer in der Regel vom verbalen Vortrag dominierten Kommunikation dar und tritt hinter dem „reellen" Medium, also Sprache, Mimik und Gestik des Vortragenden, zurück.

Die Tatsache hingegen, dass ebendieser ursprünglich rein statischen, visuellen Komponente interessanterweise aufgrund des Anwendungsprinzips der Authoring-Software an sich [vgl. Park01] mittels multimedialer Funktionalität heute eine weitaus bedeutendere Funktion zukommt [vg. Jaco02b],[2] legt an dieser Stelle eine eingehendere Untersuchung sowohl des Programms als auch des damit assoziierten Formates nahe. Darüber hinaus jedoch, und dies macht eine detailliertere Analyse der Anwendung auch im Rahmen dieser Diplomarbeit besonders interessant, konnten die PowerPoint-Hersteller Microsoft überdies aus der bereits seit längerem dominierenden Stellung im Bereich der reinen Präsentationssoftware [Mane99] in weite Bereiche auch des „Multimedia-Authorings" vordringen [vgl. Endi01]. In der heutigen Geschäftswelt werden daher bedeutend mehr auch primär als selbständige Multimedia-Anwendungen konzipierte Präsentationen mithilfe der schlicht erscheinenden und überdies „kostengünstigen" [Brow02][3] PowerPoint-Applikation erstellt, als „mit jedem anderen" der bereits eingangs[4] betrachteten, professionellen Multimedia-Authoring-Tools[5] – ganz zu schweigen von der bis dato quantitativ völlig unbedeutenden Implementierung der in [2.2] angesprochenen, komplexen Standards und Systeme. Da die somit erzielten Ergebnisse aufgrund des „Dilettantismus" der PowerPoint-Anwender [vgl. Mane99], der „mangelhaften, ästhetischen Reglementierung" seitens des Programms [Park01] sowie dessen fehlender grundsätzlichen Eignung zur Erstellung primär multimedialer Anwendungen [vgl. Endi01, Wald02] jedoch „völlig indiskutabel"[6] ausfallen [Godi01], werde ich mich im folgenden etwas eingehender mit der Frage auseinandersetzen, inwieweit das Anwendungsprinzip des Programms und auch das damit assoziierte Dateiformat selber die bereits kurz erwähnten „*shortcomings*" [Brow02] hinsichtlich inhaltlicher Formulierung, technischer Struktur und Ästhetik der erzielten Ergebnisse direkt bedingt.

---

[1] Anm: Daher auch der Begriff „Visuals" im Sinne von Vortragsfolien.

[2] „Die Technik dominiert, der Mensch tritt zurück..." [Jaco02b] p.43

[3] "PowerPoint is a low-cost tool…" [Brow02]

[4] s. 2.1

[5] vgl. [Mane99]

[6] "Almost every PowerPoint presentation sucks rotten eggs." [Godi01] p.3

2.3.1 Microsofts PowerPoint-Dominanz, historisch betrachtet

Da, wie die späteren Betrachtungen zeigen werden, ein Grossteil der Kritik an der Ästhetik, der „intellektuellen Zweifelhaftigkeit“ [Stew01][1] und marktbeherrschenden Dominanz des Programms entwicklungsgeschichtlich begründet liegt, ist an dieser Stelle zunächst eine historische Analyse der Präsentationsbezogenen Entwicklung zum besseren Verständnis dieser Problembereiche von Interesse.

2.3.1.1 Notwendigkeit von Präsentationssystemen

Der Umstand, dass der menschliche Vortrag in Form der „Business-Präsentation“, einer Vorlesung oder Verkaufsgespräches ja bereits seit geraumer Zeit eine bedeutende Rolle in westlichen Industrienationen einnimmt, legt eigentlich die Vermutung nahe, dass bei bereits seit den 60er Jahren eingesetzten IT- und Mainfraimesystemen auch entsprechende Software zur Erstellung visueller Begleitmedien oder gar bereits projizierbare, rudimentär multimediale Präsentationssoftware verfügbar gewesen sein müsste und die Entwicklungsgeschichte der vortragsunterstützenden Präsentationssoftware daher bedeutend weiter zurückreicht als die bereits eingangs erwähnten, originären Multimedia-Authoringsysteme. Paradoxerweise ist dies, aufgrund seinerzeit zurückhaltender Software-Produzenten [vgl. Wine88],[2] daher „unzureichend Brauchbarem“ und somit einer ebenso zögerlichen Anwenderschaft, jedoch nicht der Fall:

> *Die Benutzer wurden eher abgeschreckt als zur Benutzung animiert.* [Scha90:XI]

Darüber hinaus ist außerdem die enge Verknüpfung von Präsentationssoftware speziell mit den Anforderungen der Geschäftswelt für eine gewisse entwicklungstechnische Verzögerung verantwortlich: Obgleich sich auch die akademische Welt die Erforschung struktureller und ästhetischer Präsentationsaspekte zueigen gemacht hatte [Mack86, FaSu97], konzentrierten sich diese Ansätze zumeist auf Benutzerschnittstellen-Design und Informationsdarstellung. Die Erstellung visueller Vortragskomponenten, für den akademischen Einsatz zumindest im Rahmen von Vorlesungen prinzipiell interessant, blieb aufgrund des ursprünglich von der wissenschaftlichen Welt nur zögerlich aufgenommenen Design-Konzeptes zunächst der Geschäftswelt vorbehalten, wo gestalterischen Aspekten im Rahmen geschäftsfördernder Präsentationen weit größere Bedeutung zukam und zudem größere finanzielle Mittel zur Verfügung standen.

Doch auch hier schuf laut [Park01] und [Sear98] erst ein betriebswirtschaftliches Phänomen, das Mitte der 70er Jahre zunächst insbesondere US-amerikanische Unternehmen durchzog, die wirkliche „Notwendigkeit“ für visuell unterstütze Präsentationen: Der Einsatz von technisch meist unbewandertem Marketing- und Vertriebspersonal, welches nun verkaufsfördernde Maßnahmen zentral voranzutreiben suchte, schuf zugleich ein gravierendes Kommunikationsproblem auf Seiten der Ingenieure, die sich zuvor lediglich mit Führungskräften der gleichen Abteilungen „in der gleichen Sprache“ [Park01] auseinander zu setzen hatten. Die „Lösung“ für die Entwicklungsingenieure stellten daher visuell unterstütze, zunächst firmeninterne *Präsentationen* dar [vgl. Godi03:3], die meist komplexe Inhalte in eine auch für Vertriebspersonal verständliche, bildliche Sprache übersetzten. Die zusätzliche, überzeugende Wirkung der hierbei zum Einsatz kommenden Business-Grafiken[3] machte sich daher schließlich auch die nach außen gerichtete Firmenkommunikation in Kundenpräsentationen und „sales pitches“ [Mane99, Park01] zunutze.

2.3.1.2 Rudimentäre Präsentationstechnik

Wiederum im Gegensatz zu den zuvor angesprochenen, nativen Multimedia-Anwendungen war jedoch auch hier eine direkte OnScreen-Anwendung bzw. Projektion von Media-Daten zunächst nur sehr erschwert möglich und zudem sündhaft teuer: Bei einer bereits digitalen, Bildschirmtextbasierten Präsentati-

---

[1] Frei übersersetzt: “intellectually suspect” [Stew01]

[2] “Apple was [not] interested in the database…” [Wine88]

[3] „[...] Nicht nur Grafiken zu Präsentationszwecken ... sondern auch solche zur Entscheidungsunterstützung:“ [Scha90] p.5

on war beispielsweise die Anwendung von optischen Projektoren zwar möglich und auch für das Vortragspublikum durchaus beeindruckend, allerdings zugleich mit horrenden Tagesmieten der überdies einen 5-Quadratmeter-Raum vollständig ausfüllenden Projektoren verbunden, was die zeitgleiche Anmietung zweier Techniker der Herstellerfirma zudem stets mit einbezog

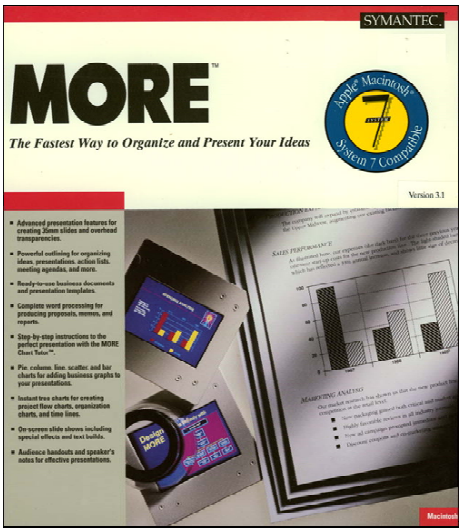
Die Lösung dieses Problems und zugleich Grund für die rasante Verbreitung der Präsentationstechnik stellten schließlich realisierbare Anwendungen auf Basis der bereits seit den 50er Jahren erhältlichen Tageslichtprojektoren da: Durch kopiererbeständige und zudem den Temperaturen des Projektors standhaltende Druckfarben war es nun möglich, „professionelle Inhalte“, in der Regel also „eintönige Textwüsten in Schreibmaschinenschrift“ [vgl. Bell00],[1] in akzeptabler Größe zu projizieren.

Da die „Gestaltung“ derartiger Projektionen, im günstigsten Falle, zumeist von der Sekretärin des jeweiligen Verfassers durchgeführt wurde [Park01], ging bereits mit diesem Schritt eine gewisse „Entprofessionalisierung“ des Erstellungsprozesses einher – die Kreation sämtlicher ästhetisch anspruchsvolleren Präsentationen, mit denen eine gewisse Wirkung auf Kunden oder Führungspersonal erzielt werden sollte, wurden jedoch weiterhin von den professionellen Grafikabteilungen der jeweiligen Unternehmen durchgeführt. Diese verfügten überdies über fortgeschrittenere technische Mittel wie der Erstellung von Diapositiven und somit eine absolute „Monopolstellung“ bezüglich optisch ansprechender Präsentationen.[2]

Mit der weiten Verbreitung von Computer-Mainframes und mittlerer Datentechnik in praktisch allen größeren Unternehmen sowie der zu Beginn der 80er Jahre aufkommenden PC-Revolution bestand nach dem überwältigenden Erfolg der ersten grafischen Business-Applikation *VisiCalc* auf dem Apple II nun ein großes Vakuum hinsichtlich einer vom Markt herbeigesehnten Software, die auch dem „schlichten Geschäftsmann“ [Park01] eine einfache Erstellung schmucker Geschäftspräsentationen ermöglichen sollte.

2.3.1.3 Frühe strukturierte Präsentationsansätze: VisiText, ThinkTank und MORE

Der frühe Erfolg einer solch gearteten Lösung wurde jedoch paradoxerweise durch eben dasselbe Unternehmen, das zuvor für die Entwicklung von VisiCalc verantwortlich zeichnete, zunächst verhindert: Ein entsprechendes Produkt – *VisiText* – war bereits 1980 entwickelt worden und wartete auf seine Auslieferung, als sich die Unternehmensführung im Sommer 1981 nach einem turbulenten Management-Wechsel entschloss, das Programm nicht zu vermarkten.[3] Der Entwickler der Software, Dave Winer, gründete daraufhin sein eigenes Unternehmen und feierte schließlich mit der Einführung der Anwendung unter dem Namen ThinkTank bzw. MORE, freilich erst einige Jahre später, Mitte der 80er Jahre grandiose Erfolge. [vgl. Wine88].



Das Programm selber verfügte über ein „geniales“ [Sear98] Anwendungsprinzip, dass von den später in den Präsentationsmarkt vordringenden Produkten wie PowerPoint hinsichtlich der grundsätzlichen Architektur abwich: Im Gegensatz zur primär optisch und gestalterisch orientierten WYSIWYG-Konkurrenz[4] waren die Produkte von Winers Firma „Living Video Systems“ sogenannte *Outlining*-Programme, die die Bearbeitung verschiedener Satzelemente, Stichworte und Argumente („Gliederungen“) in hierar-

[1] “Any font you want as long as it was Courier 12” [Bell00] p.3

[2] Daher auch die noch heute bestehende Analogie des Begriffes „Slides“ in der Englischen PowerPoint-Version (Die deutsche Fassung bedient sich hier an die Metapher der Overhead-Folie)

[3] vgl. [Wine88]

[4] What You See Is What You Get: Die Optische Darstellung und Benutzerschnittstelle entspricht dem später ausgedruckten

chisch geordneten Datenstrukturen erlaubte.

Die Bildschirm- und Druck-Ausgabe einzelner Folien samt Überschriften, in Wines MORE schlicht „Bullet Charts“ getauft [Sear98], stellte zunächst eher ein unbeabsichtigtes „Abfallprodukt“ [Wine88] der Software dar, welches sich später freilich zum Verkaufsargument der Anwendung schlechthin entwickeln sollte. Die Besonderheit des Programms bestand daher nicht in der Bearbeitung diskreter, grafischer Elemente durch den Benutzer – derartige Design-Spielereien, so mutmaßten die Entwickler, seien für Texter und mit Konzeptionierungsfragen betraute Mitarbeiter doch ohnehin „zu niveaulos“[1] – sondern in der automatisierten Generierung der grafischen Darstellung auf Basis dramaturgisch strukturierter Inhalte, in deren Erzeugung ja die eigentliche Aufgabe des Programms bestand.

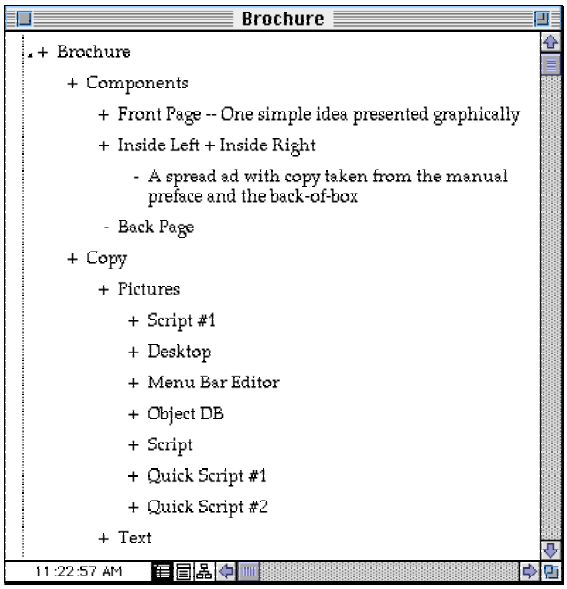


Abb. 2.3.1.3.1: Outlining-Screenshot von MORE (rechts)

Ebendiesem Ansatz – dem grafisch orientierte Editieren einzelner Folien – folgten hingegen die ebenfalls im Zeitraum der MORE-Veröffentlichung erscheinenden Konkurrenzvarianten des Produktes. Umso überraschender erscheint es daher, dass auch das Resultat dieser Herangehensweise kein Ergebnis bemühter Entwicklungsarbeit darstellt, sondern lediglich ein eher nebenher entwickeltes „Abfallprodukt“ einer gänzlich anderen Forschungsarbeit war: So hatte der Mathematiker Whitfield Diffie, in Vorbereitung auf einen Vortrag am Bell Northern Research Laboratory, lediglich ein simples Programm geschrieben, um einfache Kommentare und Textzeilen seiner Vorlesungen, inmitten eines Schwarzen Textrahmens platzieren und ausdrucken zu können. Die so erzielten Ergebnisse ließen sich dann wiederum auf Overhead-Folie kopieren oder auch an die Grafikabteilung weiterreichen, damit entsprechende Dias für seinen Vortrag belichtet werden konnten. Mit diesem schlichten, in Anbetracht der heutigen multimedialen Möglichkeiten geradezu „prähistorisch“ [Bell00:3] wirkenden Arbeitsvorgang hatte Diffie, wie *New Yorker*-Autor Ian Parker bemerkt, bereits „den Weg in Richtung PowerPoint gewiesen“ [Park01:2].

2.3.1.4 Erste zögerliche Schritte in Richtung PowerPoint

Schließlich sollte es jedoch nicht der friedensbewegte[2] Wissenschaftler Diffie, sondern sein Kollege am Bell-Forschungslabor, Robert Gaskins sein, der das wahre Geschäftspotential in Diffies Vorarbeit erkannte:

> *[He] now had his idea: A graphics program that would […] put together, and edit, a string of single pages, or “slides.”*
>
> [Park01:3]

Gaskin verlässt daraufhin das Forschungslabor, um sich mit seiner Idee (vorläufiger Arbeitstitel: *Presenter*) und dem eigenen Unternehmen „Forethought“ selbständig zu machen. Nach 16 Monaten Entwicklungsarbeit[3] unter Mithilfe von Dennis Austin,[4] einem Rechtsstreit um den Produktnamen und dem rettenden

[1] “I’ve always felt that graphics products like page layout programs, draw programs, paint programs, were too low-level to be useful to word and concept people.” [Wine88]

[2] vgl. [Park01] p.2

[3] vgl. [Bell00] p.4

[4] vgl. [Park01] p.3

Wort-Einfall „unter der Dusche“[1] bringt Gaskin seine „Vision“ (Diffie) schließlich unter dem Titel „PowerPoint“ auf den Markt.

2.3.1.5 PowerPoint 1.0: Radikales *User Empowerment*

Das unter der Version 1.0 im April 1997 ausgelieferte Endprodukt basierte, obgleich nur in Schwarz-Weiß und noch ohne direkte WYSIWYG-Schnittstelle, auf einem weitaus radikaleren Design-Gedanken als der automatisierte, strukturorientierte Ansatz des zuvor veröffentlichten MORE: Während MORE-Entwickler Winer die Tatsache, dass sich „nicht jeder einzelne Pixel kontrollieren“ ließe, unbekümmert als „usual tradeoff“ hinnimmt [Wine88], lässt Gaskins im Handbuch der Erstlingsversion seiner Präsentationssoftware explizit den folgenden, (als einzigen in der gesamten Begleitschrift) in kursiver Schrift ausgefertigten Satz abdrucken:

> *[PowerPoint] allows the content-originator to control the presentation.*[2]

Für Gaskins stellte dies den eigentlichen Kern und das wirklich revolutionäre an seinem Produkt dar: Sämtliche „intermediären Kräfte“ zwischen dem Autor der Präsentation selbst und seinem Werk zu beseitigen – „ungeachtet sämtlicher Konsequenzen“ [Park01] und dem Nutzer somit die „vollständige Kontrolle“ über die Gestaltung des Ergebnisses zu verleihen. Dass mit Beseitigung dieser „Zwischeninstanzen“ – zumeist die auch von vielen Führungskräften ob ihrer bisherigen „Vorherrschaft“ [s.o.] ungeliebten Grafik-Designer der *Corporate Communications*-Abteilungen – allerdings auch die einzigen in Kommunikationstheorie und Grafikdesign geschulten Kräfte aus dem Produktionsprozess mit PowerPoint erstellter Präsentationen entfernt wurden, bedeutete freilich einen weiteren Verlust „ästhetischer Kompetenz“ und zugleich eine fortschreitende Entprofessionalisierung visueller Business-Präsentationsmedien.

Gaskins hingegen ließ der Vorwurf, er lasse den einen „derart wichtigen“ [Jaco02a] Prozess „in die Hände von Amateuren“ [Mane99] fallen,[3] weithin unbekümmert – im Gegenteil: Obgleich laut [Park01] „verschiedene Kollegen“ Gaskins', in der Hoffnung, zumindest die schlimmsten „Design-Katastrophen“ vermeiden zu können, immer wieder die grafische Gestaltungs-Funktionalität des Programms zu beschneiden versuchten, setzte der eigenwillige Berkeley-Absolvent seine radikalen Vorstellungen von „absolutem *Empowerment*“ des Benutzers konsequent durch – unter steter Berufung auf einen Leitsatz von Thoreau:

> *Ich kam in diese Welt nicht unbedingt, um aus ihr einen guten Platz zum Leben zu machen, sondern schlicht, um in ihr zu leben, sei es nun gut oder schlecht.*[4]

2.3.1.6 Markterfolg und Dominanz

Trotz berechtigter Kritik an diesem Konzept [s. hierzu auch 2.3.2.2] sollte der überwältigende Erfolg den Entwicklern zunächst recht geben: War der Markt für die von Forethought herausgebrachte Software zunächst noch „sehr klein und spezialisiert“ [vgl. Wine88], so landete Gaskins bereits mit der Erstauslieferung seiner Business-Anwendung einen „Riesenhit“ [Park01]. Der Erfolg des „Vorreiters“ war in der Tat derart überwältigend, dass sich fortan nicht nur eine Schlacht mit der rasch in den vielversprechenden Markt drängenden Konkurrenz entfaltete, sondern ebenso ein in unverminderter Härte tobender Kampf um die Übernahme des zunächst sehr kleinen und somit hinsichtlich einet Akquisition interessanten Pionier-Unternehmens Forethought. Nachdem unmittelbar nach der Veröffentlichung von PowerPoint 1.0 Macintosh-Produzent Apple in die bis dato nur auf Apples Heim-Plattform entwickelnden Firma investiert hatte,[5]

---

[1] ibid.
[2] zitiert ebenda.
[3] “Too much PowerPoint is falling into the hands of amateurs” [Mane99]
[4] zitiert aus: [Park01]
[5] Quelle: [Mane99]

entschied Betriebssystem-Gigant Microsoft die Schlacht schließlich im Veröffentlichungsjahr 1997 mit einem 14 Millionen Dollar-Angebot für sich. Noch im selben Jahr wurde auch Konkurrent MORE dem Software-Spezialisten Symantec in einer ähnlich bezifferten Übernahme einverleibt [Wine88].

Nach massivem Aufgebot der Konkurrenz wurde der Markt jedoch nach kurzer Zeit in der Macintosh-Welt von Aldus' *Persuasion*, im Geschäftsbereich der IBM-kompatiblen PCs hingegen von *Harvard Graphics* beherrscht [vgl. Bell00:7]. Microsoft, dessen PowerPoint derweil trotz der seit Version 3 im Mai 1992 eingeführten Multimedia-Funktionen wie Folien-Übergängen, animierten Textzeilen, Sound und Video keine dominierende Stellung erlangt hatte,[1] konzentrierte daraufhin seine Marketing-Bemühungen auf ein sogenanntes *Office*-Paket, das neben PowerPoint auch die bereits die Marktführerschaft innehaltenden Produkte Word und Excel umfasste - eine Strategie, die PowerPoint schließlich „unverwundbar" machen sollte, wie [Park01] anmerkt.

Die im gleichen Zeitraum auf den Markt drängenden Konkurrenten *Lotus Freelance* und *WordPerfect* verfolgten daraufhin zwar ähnliche Strategien einer Office-Integration mit „SmartSuite" und „PerfectOffice", wurden jedoch bald, ebenso wie Mitbewerber Aldus, von jeweils größeren Unternehmen [2] aufgekauft, die wiederum jedoch die Bemühungen der PowerPoint-Konkurrenten nicht unterstützten und schließlich gegenüber der Microsoft-Dominanz „der Verrottung überließen" [Bell00:9] Die Herstellerfirma von Harvard Graphics, welche im Verlaufe der Schlacht „keinen Tanzpartner" gefunden hatte, wie Microsoft-Produktmanagerin Cathleen Belleville höhnisch anmerkt,[3] starb indes, ohnmächtig der gnadenlosen Marketing-Maschine des übermächtigen Microsoft ausgesetzt, einen „langsamen und qualvollen Tod".[4]

2.3.2 Systemkritik

Seit dieser bereits 1994 entschiedenen Schlacht „beherrscht PowerPoint die Welt",[5] auf der es wiederum zum heutigen Tage an Unmöglichkeit grenzt, Anwender von Präsentationssoftware aufzuspüren, „die nicht das ungeliebte Microsoft-Produkt verwenden" [Mane99]. Obgleich die soeben beschriebene Entwicklungsgeschichte deutlich macht, dass sich diese so „unangefochtene"[6] Marktführerschaft vielmehr auf unternehmerische Taktik denn durch wirklich überzeugende Produkteigenschaften begründet, stellt sich in den Augen vieler Kritiker [Sear98, Stew01, Godi01] dennoch die berechtigte Frage, ob diese „weitverbreitete Plage" [Mane99] denn derart leichtfertig hingenommen und noch unterstützt werden muss, wie etwa [Jaco02] demonstriert:

> *Businessleute erwarten heute einen perfekten Auftritt – präsentiert mit PowerPoint…*
> *„Bei uns kommen fast ausschließlich PC-Präsentationen mit PowerPoint zum Einsatz", unterstreicht Sascha Maurer, Kommunikationstrainer und Berater bei PriceWaterhouseCoopers in Frankfurt.*
>
> [Jaco02a,b] pp.40,42

In der Tat hat sich durch die „Unausweichlichkeit" [Stew01] des Produktes mitsamt zahlreichen Fehlern, „Frustrationsfaktoren" [7] und Versäumnissen eine regelrecht verschworene „Hassgemeinschaft" [Brow02] ausgebildet, die ihre Kritik an der umstrittenen Software durch zahlreiche Publikationen öffentlich kultiviert:

---

[1] vgl. [Bell00] p.7
[2] Dies sind bei Lotus: IBM, bei WordPerfect: Novell sowie bei Aldus: Adobe.
[3] vgl. [ibid] p.8
[4] zitiert ebenda, p.9
[5] vgl. [Mane99]
[6] vgl. [Bell00] p.10
[7] vgl. [Roto02b]

> *As part of this counterculture, people have written articles and books on the art of the presentation, the history of PowerPoint, and how the application will be the undoing of the business community. Of course, these are topics of interest relevant to all.*
>
> [Brow02]

Abgesehen von der existentiellen Frage, ob die Verwendung von PowerPoint schließlich mit dem „Untergang der industriellen Geschäftswelt" gleichzusetzen ist [Brow02], stellt sich darüber hinaus jedoch für jeden Angestellten das bedeutend konkretere Problem, ob er denn, wenn auch „offensichtlich scheint, dass PowerPoint ein katastrophales Produkt ist",[1] dasselbe dennoch zur Erstellung der geforderten Geschäftspräsentationen verwenden sollte:

> *After all, you're used to PowerPoint, it does the job, it's the corporate standard, and you're not a techie trying to impress an audience with your know-how.*
>
> [Wald02]

Diese Frage bedarf freilich einer etwas umfassenderen Erörterung – schon allein deswegen, weil die Problematik mehrschichtig ausgelegt ist: Zu betrachten sind hierbei nicht ausschließlich Anwendungsprinzip und Design-Konzept des erstellenden Programms selber, sondern ebenso Verbreitungs- und strukturelle Kriterien des zugehörigen Dateiformates. In der öffentlichen Diskussion treten jedoch technische Formataspekte, da sich dies in der Regel[2] auf die einzige hierfür verfügbare Anwendung beschränkt, hinsichtlich der deutlicher erkennbar zutage tretenden Mängel des Programms selber deutlich zurück. Letztere Problematik erfordert jedoch wiederum eine weitere inhaltliche Aufteilung: Die die öffentliche Diskussion beherrschenden Vorwürfe richten sich demnach primär gegen die vorgeblich sowohl in *intellektueller* als auch *ästhetischer* Hinsicht „verheerenden Konsequenzen" [Park01] der PowerPoint-Anwendung, weswegen im Folgenden beide Kategorien der geäußerten Kritik separat thematisiert werden sollen.

2.3.2.1 Inhaltlich-strukturelle Kritik: *„It's intellectually suspect."* [Stew01]

> *How many of us have been frustrated at seeing too many presentations where PowerPoint's […] visual aids obscure rather than enhance the point?* [Norv99]

Die Kritik, die der Direktor für lernende Maschinen des Suchmaschinen-Spezialisten *Google*, Peter Norvig, in [Norv99] zum Ausdruck bringt, illustriert bereits anschaulich den Grundsätzlichsten aller Vorwürfe, die zahlreiche Anwender [vgl. Sear98, Mane99, Godi01] gegen das Präsentationsprogramm richten: Die Frage, ob und in wieweit die Anwendung, neben allen „bunten Glöckchen und Wortgeklingel" [Roto02a][3] überhaupt noch den eigentlich vorgesehenen Zweck des Programms erfüllt, nämlich der anschaulichen Verdeutlichung der vorgebrachten Argumente [vgl. Norv99, Endi01].[4]

Dies jedoch erscheint zumindest bei Betrachtung der einschlägigen Veröffentlichungen [vgl. Park01] mehr als fraglich: Demnach ist nicht nur das ursprünglich als simples Tool zur strukturierten Erstellung visueller Begleitmaterialien konzipierte PowerPoint zur „unheimlichen"[5] Multimedia-Maschine mit animierten Übergängen und automatisch generiertem Textinhalt [Park01:2] förmlich „mutiert" [Brow02] – das Anwendungsprinzip hat offenbar auch die Benutzung und Verwendungsweise des Programms selber verändert: Statt daher den Inhalt des im Vortrag geäußerten sinnvoll zu illustrieren, dient das Programm, so Norvig,

---

[1] "Even though PowerPoint sucks, should I use it for my deliverables?" [ibid] Vgl. Hierzu auch [Godi01]

[2] s. hierzu auch 2.3.4

[3] Anm.: aus dem Amerikanischen: „Bells and whistles" [Roto02a]

[4] Die zitierten Autoren sprechen im Original von „to illustrate a point" sowie "get[ting the] message across" [Endi01]

[5] "Gaskins is skeptical about the product that PowerPoint has become—AutoContent and animated fades between slides…" [Park01] p.2

vor allem der „Verschleierung“ der Argumentation,[1] als auch, wie [Godi01] ausführt, weiteren kommunikationsfremden Zwecken „von denen keiner einer guten Präsentation dienlich ist“.[2]

2.3.2.1.1 Stichwort-Zwang

Über diese Zweckentfremdung hinaus meinen einige Kritiker zudem eine aus der Benutzung von PowerPoint erwachsende, bedrohliche „gesellschaftlich Veränderung“ [Park01] zu erkennen: Durch den „Zwang“, sämtliche Sachverhalte in Microsofts Monopolprodukt präzise in Gestalt einzelner, stichwortartiger Satzfragmente darlegen zu müssen, wie etwa Steve Jurvetson in [Mane99] ausführt, gerieten einige Anwender in einen regelrechten Zustand „mentaler Blockade“, durch welche sie sich „ausschließlich in der Form stichwortartiger Aufzählungs-Listen ihrer Umwelt mitteilen können“.[3]

Ebendiesem Zwang zur Kürze, den insbesondere PowerPoints „AutoContent Wizard“ dem Benutzer „aufdiktiert“ [Sear98], können andere Beobachter hingegen auch durchaus positives abgewinnen: „Schließlich regt PowerPoint letztendlich dazu an, treffender und präziser zu formulieren“.[4] [Brow02] Die überwiegende Mehrheit der in der Öffentlichkeit zu Wort kommenden Autoren wendet sich jedoch entschieden gegen diese „gefährliche Tendenz“ der Sprach-Fragmentierung:

> *No paragraphs, no pronouns – the world condensed into a few upbeat slides, with seven or so words on a line, seven or so lines on a slide.*
>
> [Park01:1]

Fortune-Kolumnist Thomas Stewart geht gar noch einen Schritt weiter, und bezeichnet diesen Hang zur drastischen Komplexitäts-Reduzierung, den Microsofts Präsentationsprogramm dem Nutzer auferlegt, als „intellektuell suspekt“:[5]

> *Complexity exists, really! […] In real life, bullet points kill.* [Stew01]

Hierbei kommt erneut speziell dem PowerPoint „Wizard”, wie der von Microsoft entwickelte Software-Agent zur nahezu vollautomatischen Erzeugung „schlüsselfertiger“ Präsentationen praktisch ohne Eingriff des Benutzers offiziell genannt wird, die aus intellektueller Sicht reichlich „problematische“ [vgl. Sear98, Stew01] Aufgabe zu, den Nutzer zur Verwendung vorgefertigter, schier „beliebig austauschbarer“ Business-Floskeln zu „nötigen“:[6] „Solutionese“ nennt etwa *Linux Journal*-Herausgeber David Searls diesen „aalglatten [und] im Prinzip nichtssagenden“ Business-Sprech, deren reichliche Anwendung Microsofts Content-„Zauberer“ über die Template-Maske nahelegt:

> *List the products and features, and how each addresses a specific need or solves a specific problem.*[7]

Wie bereits Stewart und Norvig, die ebenfalls die „Austauschbarkeit“ von PowerPoint-Präsentationen wehleidig beklagen [vgl. Norv99, Stew01], findet auch Searls derartige, vorgeblich durch PowerPoint auferlegte Phrasendrescherei „einfach nur zum Gähnen“.[8]

2.3.2.1.2 Fehlender Memorierungs-Effekt

---

[1] “[...] Obscure rather than enhance the Point...“ [Norv99]

[2] “Unfortunately, rather than communicating, PowerPoint is used to accomplish three things, none of which leads to a good presentation.” [Godi01] p.5

[3] “I’ve seen people who get in a mental rut and are unable to write anything other than bullet-point lists.” [Mane99]

[4] “Ultimately, PowerPoint forces you to be a better writer…” [Brow02]

[5] “Why ban PowerPoint? It’s intellectually suspect.” [Stew01]

[6] “PowerPoint prompts you to talk in solutionese” [Sear98]

[7] Ausgabe des AutoContent-Assistenten PowerPoints (Typ „Marketing-Meeting“).

[8] [ibid]

Als weitaus tragischer, über Aspekt der reinen Langeweile hinaus, macht Searls jedoch den „ausbleibenden" Memorierungs-Effekt auf diese Weise erstellter Präsentationen aus:

> *The speaker feels like he said something and the audience feels like something got said; but in reality nothing got communicated at all… All anybody remembers – including the speaker – is that a bunch of slides got shown.*
>
> [Sear98]

In [Park01] findet sich schließlich eine aufschlussreiche Theorie, die zur Erklärung des von Searls festgestellten „PowerPoint-Phänomens" herangezogen werden kann: Parker bemüht hierfür den Stanford'schen Soziologen Clifford Nass, der wiederum den „Verlust des gedanklichen Prozesses" für die fehlende Memorierung der Präsentationen verantwortlich macht [vgl. Park01:6ff]. Da die in PowerPoint zur Folienerstellung implizit geforderten, primär Lösungs-bezogenen,[1] Stichwortfragmente [vgl. Sear98] lediglich das *Ergebnis* eines gedanklichen Prozesses, nicht aber dessen Entstehung abzubilden vermögen,[2] fehlen auch die zur längerfristigen Erinnerung oft beitragenden kognitiven und emotionalen Details, wie etwa die „elegante Herangehensweise" an ein bestimmtes Problem.

2.3.2.1.3 Emotionale Überzeugungskraft

Dieser Grundgedanke findet sich auch in [Godi01] wieder – da im Gegensatz zu Nass' primär akademischem Ansatz hier jedoch eine eher geschäftsbezogene Motivation zum tragen kommt, betont Autor Seth Godin in seinen Ausführungen insbesondere die Wichtigkeit des Emotionalen zugunsten längerfristiger Erinnerungswerte. Erstliniges Ziel überzeugenden Präsentationsmaterials sei es daher, so Godin, den Wahrheitsgehalt der vorgetragenen Argumente durch gefühlsbetonte Optik ("emotional proof") zu unterstreichen, als lediglich deren Akkuratheit, etwa durch nackte Zahlenreihen, zu belegen.[3] [Godi01:7] Dieser Ansatz, so merkt allerdings [Brow02] an, sei freilich allenfalls für „Anfänger" geeignet – anspruchsvolleren Gestaltern legt Informationsarchitekt Brown hingegen eine offensichtlich Scott McClouds Ausführungen zur Symbolik des Comics [McCl94] entlehnte Metapher des „Tanzes zwischen Wort und Bild" nahe:

> *Words and pictures go hand in hand to convey an idea that neither could convey alone… When pictures carry the weight of clarity in a scene, they free words to explore a wider area*
>
> [McCl94:155,157]

Brown merkt jedoch zugleich an, dass die "Opportunities of Exploration", also das im übertragenen Sinne kreative Potential der textlichen Komponente, im Kontext einer Präsentation, die stets einem konkreten Zweck dienen muss, doch relativ „beschränkt" bleibe.[4] Auf die „Gefahr" der hier lediglich angedeuteten, in [Godi01] schließlich unverblümt geforderten *Dominanz der Bilder* gegenüber einem in Richtung strikt endergebnis-orientierter, stichwortartiger Satzfragmente „gedrängten" [Sear98] Textteil, die bereits die PowerPoint-eigenen Wizard-Komponenten dem Nutzer, so Searls, regelrecht „aufdiktieren",[5] weist überdies eine Studie der Arizona State University hin: So konnte ein dreiköpfiges Forscherteam um Psychologie-Dekan Robert Cialdini durch empirische Experimente nachweisen,[6] dass insbesondere multimedial aufbereitete PowerPoint-Materialien Urteilskraft und Entscheidungsfindung der Probanden erheblich beeinflussen. Diese ließen sich etwa von den Segnungen animierter Balkengrafiken des Programms derart beeindrucken,

---

[1] Anm: Dieser Allegorie entstammt auch Searls' Begriff des „Solutionese"

[2] s. d. vorhergehende Anmerkung

[3] "Create slides that demonstrate, with emotional proof, that what you're saying is true not just accurate." [Godi01] p.7

[4] "But a presentation is a means to an end (usually either selling or educating) and the opportunities to explore are limited." [Brow02]

[5] vgl. [Sear98]

[6] vgl. [Park01] p.3

dass die Beurteilung der PowerPoint-Daten deutlich von der Analyse derselben Information in tabellarischer oder rudimentär grafisch aufbereiteter Form abwich [vgl. Park01]:

> *PowerPoint seems to be a way for organizations to turn expensive, expert decision-makers into novice decision-makers.* [1]

Über die Schlussfolgerungen Cialdinis hinaus, der die vorgefundenen Ergebnisse als „schlicht beängstigend" kommentierte,[2] interpretiert Ian Parker die Anwendung gar als „gesellschaftspolitisches Instrument" mit gewaltigem Potential, welches, so Parker, bereits die Gedankenwelt amerikanischer Kindergartenkinder, die vermehrt schon in frühen Jahren zu PowerPoint-unterstützten „show-and-tells" angehalten werden,[3] deutlich zu prägen und strukturieren vermag. Seine Analyse des „subtilen Einflusses" PowerPoint-basierter Multimedia-Präsentationen kondensiert der Autor schließlich in der folgenden Aussage:

> *It edits our thoughts.* [Park01]

Stewart[4] zieht an dieser Stelle eine Parallele zur amerikanischen Literatur, in dem er den „Glauben an das Geschriebene", den der amerikanische Autor Ralph Waldo Emerson in „The American Scholar" anprangert, auf den modernen Präsentationsbegriff PowerPoints erweitert:

> *Emerson warned against the tendency to believe something just because it is written down. How much greater the danger when it is also boiled down.*[5]
>
> [Stew01]

Interessant ist im Rahmen der hier geäußerten Kritik auch die frappierende Analogie eines PowerPoint vorgeblich zugrundeliegenden, "faschistoiden" Anwendungs-Konzeptes: So lässt sich [Sear98] etwa zu einem Vergleich des PowerPoint-AutoContent-Assistenten mit „Gestapo-Verhör-Methoden" hinreißen: "Vee haff vays uff making you talk".[6] Auch USAToday-Autor Kevin Maney wird in seinem vielbachteten Essay [vgl. Norv99] nicht müde, den PowerPoint-„Fanatismus" des „Germanischen Giganten" SAP zu unterstreichen [Mane99]. Ob derartige Entgleisungen allerdings allein durch die problematische „soziale Funktion" [Park01] des Programms zu rechtfertigen oder gar angebracht sind, erscheint in meinen Augen allerdings mehr als fraglich.

2.3.2.1.4 Problematisches Prinzip

Selbst bei Verzicht auf die derart scharf attackierte *AutoContent*-Funktionalität des Programms selbst grenzt es hingegen an Unmöglichkeit, auch den dahinterstehenden, hartnäckigen „Spirit" [Park01] abzuschütteln: Der Verlockung, dem mittlerweile zum „Standard" [Wald01] avancierten Schema mit Stichworten aufgefüllter, endlos aneinandergereihter PowerPoint-Folien zu erliegen, steht in der Regel „kein besonders großer Impuls, sich diesem Zwang zu widersetzen",[7] gegenüber. Allzu bequem scheint es, diesem „Slide-Staccato"[8] unreflektiert nachzufolgen, nicht zuletzt, um der diesbezüglichen Erwartungshaltung der „Business Community" [vgl. Jaco02a] zu entsprechen. Überdies, so argumentiert Parker, böten audiovisuell bereicherte PowerPoint-Präsentationen eine willkommene Tarnung, sich hinter dem Aufmerksamkeit absorbierenden

---

[1] zitiert ebenda.

[2] "'It's frightening,' Cialdini says." [ibid]

[3] "Somehow, a piece of software designed, fifteen years ago, to meet a simple business need has become a way of organizing thought at kindergarten show-and-tells." [ebenda]

[4] vgl. [Stew01]

[5] Anm: Die amerikanische Redewendung „to boil [sth.] down" bezieht sich hier auf die komprimierte textuelle Form der PowerPoint-Stichworte

[6] Anm: Der geäußerte Satz (zu deutsch „Wir haben unsere Wege, Sie zum Sprechen zu bringen"), ist zugleich Referenz auf entsprechende Nazi-Verhörmethoden, Verballhornung deutschen Akzents im Englischen sowie den „Zwang" seitens des Assistenten, textliche Inhalte aus dem Nutzer förmlich „herauszupressen" [vg. Sear98]

[7] vgl. [Park01]

[8] ibid.

„Multimedia-Geklingel“ [Roto02a] zu verstecken: „Die Technik dominiert, der Mensch tritt zurück“ [Jaco02b:43].

Durch dieses Phänomen kann zwar die Angst des Vortragenden, der sich in der Vorbereitung statt der schwierigen Problemstellung, mit dem Zielpublikum effektiv zu kommunizieren, nun vorrangig mit vergnüglichem „Moviemaking“ [Park01] beschäftigen darf [vgl. Ziel02], gezielt bekämpft werden – neben der „menschlichen“ Komponente des Vortrags tritt allerdings hierbei auch das eigentlich Gesagte deutlich zurück; die Wirkung „verpufft“, wie Larry Gottlieb vom kalifornischen Livermore-Forschungslabor in [Searl98] anmerkt: Da das PowerPoint-erzeugte Präsentationsmaterial aufgrund dessen vorgeblich „erzwungener Oberflächlichkeit“ [Park01] zur bloßen „nichtssagenden Prothese“ (Gottlieb) oder gar zum „Teleprompter“ [Godi01:5] verkomme, so der übereinkommende Tenor der Kritiker [vgl. Mane99, Stew01, Park01, Godi01] könne auch eine erhoffte, überzeugende Wirkung oder selbst nur ein Memorierungseffekt [s.o.] der präsentierten Inhalte nicht erzielt werden.

Ungeachtet der hohen Emotionalität, der vielen Beiträgen im Rahmen der inhaltlichen Diskussion um die „intellektuelle Konsequenz“ des PowerPoint’schen Anwendungsprinzips anhaftet, bleibt an dieser Stelle nüchtern festzuhalten, dass das Programm über seine ursprünglich angedachte Funktionalität hinaus offensichtlich nicht nur eine bestimmte Form der Formulierung bzw. Fragmentierung zu präsentierender Inhalte nahe legt, sondern insbesondere durch konkrete Anwendung oder zumindest „subtilen Einfluss“ [vgl. Park01] des AutoContent-Assistenten auch die Inhalte der Präsentation selber in eine gewisse Richtung zu drängen vermag.

2.3.2.1.5 Verrückte Idee: Der AutoContent-Wizard

Der vielumstrittene „Wizard“ des automatisch Inhalte generierenden AutoContent-Systems erscheint hierbei auch entwicklungsgeschichtlich interessant: So waren denn auch die Entwickler des Programms selber gegenüber der ihnen von der Microsoft-Geschäftsführung nahegetragenen Aufgabe offenbar „reichlich skeptisch“ [Park01:2]: Nachdem die Marktforschung des Software-Riesen eine „ungutes Gefühl“ der Benutzerschaft hinsichtlich völlig leerer PowerPoint-Seiten zu Beginn des Erstellungsprozesses ausgemacht hatte, machten sich die Programmierer schließlich zu Beginn er 90er Jahre an die Realisierung dieser „völlig verrückten Idee“ (Gaskins), welche diese ungeliebten Lücken automatisch schließen sollte. Unter dem zunächst als „interne Verballhornung“ [Park01] kursierenden Namen „AutoContent“ fand das Modul bereits bei Programmversion 4.0 im Februar 1994 den Weg in die PowerPoint-Produktlinie – ein laut Parker „seltenes Beispiel direkter Verhöhnung“ der eigenen, potentiellen Kundschaft,[1] welche diese Funktionalität jedoch nur zu dankend annahm: „Wer nimmt sich schließlich schon die Zeit, denn automatisch erzeugten Text dann auch noch zu bearbeiten?“ [FlRi01:9]

So erstaunt es nur wenig, dass [Godi01] und [Mane99] in PowerPoint selber die Ursache einer „Debilisierung“ heutiger Business-Präsentationen zu erkennen meinen. [Sear98] mahnt derweil eine „Rückbesinnung“ auf das gedankliche Strukturen und Dramaturgie unterstützende Anwendungsprinzip von MORE aus den 80er Jahren [vgl. Wine88] an. Da durch „dramatische Überbenutzung“ des PowerPoint-Programms sowohl die Urteilskraft sogar standfester Führungskräfte untergraben wird (Cialdini), als auch Überzeugungs- und Memorierungseffekte PowerPoint-basierter Präsentationen sehr kritisch beurteilt werden müssen [vgl. Stew01, Godi01, Park01], habe sich die Effektivität PowerPoints, so [Kirb94], mittlerweile „deutlich relativiert“.[2] Aufgrund dessen rückt speziell in höheren Manager-Kreisen die Anwendung von PowerPoint zugunsten einer verstärkten Konzentration auf Inhalte und Ideen, „immer mehr in den Hintergrund“:

---

[1] vgl, ebenda, p.3

[2] “[…] widespread overuse has marginalized their effectiveness with live audiences.” [Kirb94]

> *PowerPoint is very rare at CEO conferences… Captains of industry like to see a speaker think, not watch him read.*
>
> [Stew01]

2.3.2.1.6 Back to the Roots: Gliederungsansicht als Struktur-Eingeständnis

Im Zuge dieser gedanklichen Rückbesinnung lenkt [Sear98] unsere Aufmerksamkeit auf das seit dem Scheitern von MORE mittlerweile in PowerPoint integrierte [Bell00:7], jedoch „stiefmütterlich behandelte" (Searls) *Outlining*-Tool des Präsentationsprogramms: So ist es nun auch in PowerPoint möglich, ungeachtet der visuellen Gestaltung Präsentationsinhalte rein auf Basis der inhaltlichen Gliederung zu konzipieren, statt direkt Design-orientiert vorzugehen. Ein Hinweis, dass aufgrund der nachfolgend besprochenen „Design-Sünden" des Programms eine ebensolche, verstärkt inhaltlich-strukturelle Herangehensweise mittlerweile auch der Vorstellung der Microsoft-Entwickler entspricht, stellt unter anderem die Tatsache dar, dass die Gliederungsansicht des Programms nicht mehr, wie noch von [Sear98] beklagt, „nur sehr versteckt zu erreichen ist", sondern auch in der werkseigenen Default-Einstellung immer mehr in den Vordergrund rückt.

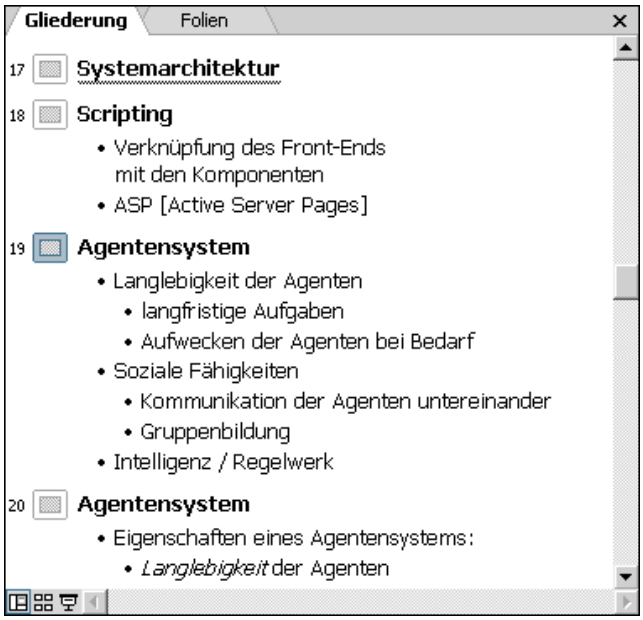


Abb. 2.3.1.1: Gliederungsansicht in PowerPoint [XP]

Dies allerdings als „Schuldbekenntnis" des Softwarekonzerns zu werten, durch Anwendungs-Prinzip und „Multimedia-Funktions-Overkill" [Godi01] sowohl inhaltlich als auch ästhetisch unvorteilhafte Präsentationen regelrecht zu „erzwingen" [Sear98], wäre an dieser Stelle jedoch voreilig – die erheblich erweiterten Multimedia-Eigenschaften (Animationen, Übergänge), über die insbesondere die aktuelle Version[1] verfügt, weisen in eine andere Richtung: Aufgrund nahezu vollständiger Marktabdeckung auf dem klassischen Präsentationssektor drängt die Software schließlich verstärkt in andere Geschäftsbereiche [vgl. Park01:1].

2.3.2.1.7 PowerPoint als Multimedia-Autorenwerkzeug?

Im Falle von PowerPoint, welches, obgleich nicht primär hierfür geeignet [vgl. Wald02] paradoxerweise auch bei CD-ROM-basierten Multimedia-Anwendungen einen relativ hohen Marktanteil genießt, spricht etwa Präsentations-Consultant Jim Endicott von einem „großflächigen, gefährlichen Missbrauch" [Endi01] der PowerPoint-Software als Multimedia-Authoring-Tool.

Ebendiese, mittlerweile durchaus beachtliche Multimedia-Funktionalität PowerPoints führte jedoch, wie Microsofts ehemalige PowerPoint-Produktmanagerin Cathleen Belleville [Bell00] anspricht, zu einer „recht bizarren Situation" in der heutigen Geschäftswelt:

> *Now we've got highly paid people sitting there formatting slides – spending hours formatting slides – because it's more fun to do that than concentrate on what you're going to say.*
>
> [Park01:3]

Das von PowerPoint-Entwickler Gaskins erträumte „Empowerment des Benutzers" hat sich, so lässt sich nach Auswertung zahlreicher Benutzer-Statements deutlich feststellen, offensichtlich zu einem sowohl aus ästhetischer [Godi01, Sipp02] als auch betriebswirtschaftlicher Hinsicht [Mane99, Park01, Ziel02] voll-

[1] Microsoft PowerPoint XP (Versions-Nummer: 10)

ständigen „Albtraum“ [Endi02b] entwickelt: Anstatt sich gedanklich Dramaturgie und Inhalt geplanter Präsentationen zu widmen, „vergnügen“ [Park01] sich hochbezahlte Manager in stundenlanger Arbeit mit im Vergleich dazu leicht und angenehm erscheinenden Design-Fragen. Bereits MORE-Entwickler Dave Winer wies 1988 auf die Gefahr hin, derartig verlockend „triviale“ Aufgaben zum Bestandteil einer für Autoren und Führungskräfte konzipierten Software zu machen:

> *I've always felt that graphics products like page layout programs, draw programs, paint programs, were too low-level to be useful to word and concept people.*
>
> [Wine88]

Stattdessen entschied sich Gaskins (wie bereits in [2.3.1] beschrieben) gegen den Widerstand seiner Entwicklerkollegen indes dafür, ursprünglich an Grafikabteilungen delegierte Design-Aufgaben „in die Hände von Amateuren“ [Mane99] fallen zu lassen. Neben der aus betriebswirtschaftlicher Hinsicht [vgl. Ziel02] bedauernswerten Tatsache, dass grafisch ungeschulte, jedoch in der Regel hochbezahlte Manager wertvolle Zeit mit gestalterischen Fragen „verschwenden“, [Park01] ist der weit dramatischere Effekt dieses Benutzerkonzeptes jedoch die „völlig ungenügende“ Ästhetik „nahezu aller mit PowerPoint gestalteter Präsentationen“ [Godi01:3][1] Auch die Frage, ob nun Programm oder Benutzer Schuld an diesem augenscheinlichen „Design-Debakel“ tragen, scheint nach [Godi01] bereits längst geklärt: “The fault lies with Microsoft”.[2]

2.3.2.2 Design-Kritik

Im Rahmen dieser Diplomarbeit empfiehlt es sich freilich, diese Problematik ein wenig differenzierter zu betrachten: So bezieht etwa die Frage nach der „Schuld“ Microsofts auch stets die Betrachtung der Zielgruppe mit ein – und hier kann in der Tat eine *Fehleinschätzung* der Entwickler hinsichtlich der Design-Kompetenz der Zielgruppe zu beobachtet werden: Ist etwa in sehr mächtigen, grafisch orientierten Anwendungen wie etwa 3D Studio Max oder Photoshop der Anwenderkreis durch den hohen Preis und die Komplexität der Bedienung bereits recht präzise auf eine professionelle Klientel ausgerichtet, so richtet sich Microsofts PowerPoint in erster Linie an „in grafischen Fragen eher unbewanderte Business-Kunden“ [Pirn01], hinsichtlich deren ästhetische Ansprüche Alan Cooper den folgenden, legendären Satz geprägt hat:

> *The word "design" is toxic in the world of business.* [Ande01]

2.3.2.2.1 PowerPoint als Design-Desaster

Unter diesem Aspekt macht das Microsoft-Tool dem ungeübten Benutzer grafische Manipulationen in der Tat „zu einfach“ [Mane99], da es aufgrund fehlender Restriktionen „Design-Desaster“ [Park01] nicht zu verhindern vermag:

> *[PowerPoint] mutated when Microsoft added functionality to give users more power in manipulating the text and images.*
>
> [Brow99]

Zum großen Leidwesen der Grafikabteilungen, deren Untergang der Erfolg PowerPoints zugleich einleitete, erfreuen sich die grafischen Funktionen des Programms daher seit ihrer Einführung in den 90er Jahren schier „allzu“ [Kirb94] großer Beliebtheit – zusätzlich noch „angeheizt“ durch wohlmeinende Empfehlungen selbsternannter „PowerPoint-Experten“, die in unzähligen Bulletins [Treu95, Roto02a, Roto02b] zum reichlichen Gebrauch optischen Geklingels anregen[3] und gleichzeitig stets die eigene, „wahre“ Design-Lehre verkünden.

---

[1] “Almost every PowerPoint presentation sucks rotten eggs.” [Godi01] p.3

[2] ebenda.

[3] “Use design to communicate that you know your listeners.” [Roto02a]

2.3.2.2.2 Hilfreiche Tipps und Wettstreit der Design-Experten

Die Lektüre eines Großteils dieser zumeist wenig hilfreich erscheinenden Ratgeber, die aufgrund ihrer schieren Masse bereits als eigenständige, nicht unbedeutende Kategorie im Computerbuchhandel firmieren, macht aufgrund ihrer großen Widersprüchlichkeit vor allem eines deutlich: Die offensichtlich ebenso mangelnde, grafische Ausbildung ihrer Autoren. So empfiehlt [Murd02] etwa „Schriftarten mit Serifen" für die textliche Ausgestaltung der Multimedia-Präsentation,[1] während wiederum [Simi02] die Verwendung serifenloser Fonts nahe legt und von serifierter Typografie „unbedingt" abrät; Auch die Verwendung des Firmenlogos auf jeder Folie, so kommentiert [Roto02a], sei „doch ein netter Zug" – was hingegen [NGWA02] unbedingt vermieden sehen will: „Logos sind Platzverschwendung und lenken von der eigentlichen Information ab".[2] [Treu95] rät derweil, „stets viel Platz" am Folienrand zu lassen, während [NGWA02] ebendies strikt ablehnt.

Die große Verwirrung, die auch im Kreise sogenannter „Experten" über die vorgeblich optimale Gestaltung PowerPoint-basierter Präsentationen ganz offensichtlich herrscht, weist hingegen nicht nur auf die Designbezogene Hilflosigkeit selbst präsentationserfahrener Benutzer hin,[3] sie macht auch fehlende Hilfestellungen und Restriktionen seitens des Programms selber deutlich. Im Gegenteil – anstatt konkrete, gestalterische Grenzen aufzuzeigen, „verleitet" die Anwendung regelrecht dazu, „sofort mit der Gestaltung der Präsentation loszulegen", [Jacob02b] und lässt die Anwender durch die „schier erschlagende Fülle an Optionen" [Treu95] eher „verwirrt und überwältigt" [FlRi01:7] zurück.

2.3.2.2.3 PowerPoint-ClipArts: Subkultur des schlechten Geschmacks?

Indem die Anwendung dem Benutzer überdies eine Bibliothek aus ästhetischer Perspektive „völlig indiskutabler" [Godi01:3] *ClipArts* bereitstellt und zudem zur großzügigen Verwendung auch aus neutralem Blickwinkel „unprofessionell erscheinender" Schriftarten direkt anregt [vgl. Sipp02], habe PowerPoint, so Ian Parker, darüber hinaus eine regelrechte „Subkultur des schlechten Geschmacks" geprägt, deren Kritiker die bereits beschriebene, vermeintlich intellektuelle „Austauschbarkeit" PowerPoint'scher Präsentationsmaterialien [vgl. Norv99, Stew01] um eine vorgebliche „Ästhetik der völligen Uniformität des enthaltenen Bildmaterials" [Pirn01] ergänzt sehen.

2.3.2.2.4 Automatische Erzeugung: Die ungenutzte Chance

Nun lehrt uns Forschung [Mack86, WeWi94] und Entwicklung [Wine88], dass intelligente, automatische Generierung grafischer Elemente anhand vorgegebener Daten (beispielsweise zu präsentierende Inhalte eines Vortrags) zwar ausgesprochen „schwierig und komplex",[4] die bei Mitarbeit fähiger Grafiker und Designer erzielten Ergebnisse jedoch stets ästhetisch und funktional zufriedenstellend ausfallen. Darüber hinaus kann dieser Automatisierungsprozess die Anwendung, die ja bei trivialen Textinhalten aus schlichter Dateneingabe besteht, für ungeübte Benutzer erheblich vereinfachen.

Die Einführung automatischer Grafikgeneration, wie ja zuvor bereits in konkurrierende Präsentationsanwendungen [vgl. Wine88] integriert, hätte daher eine zumindest theoretische Chance der Microsoft-Entwickler bedeutet, das grafische Design zumindest der automatisch generierten Präsentationen durch sowohl aus struktureller als auch ästhetischer Sicht überzeugende Regelwerke deutlich zu verbessern. Um das Ergebnis jedoch vorwegzunehmen: Mit der Integration grafischer Templates und später der sogenannten (bereits unter inhaltlichen Gesichtspunkten angesprochenen) AutoLayout- und AutoContent-Assistenten

---

[1] "7: [...] Use fonts that have serifs for text blocks" [Murd02]

[2] "Logos waste space and detract from the important information." [NGWA02]

[3] "Presenters, especially those with little or no design background, oftentimes lack a sense of what's professional in appearance", Jim Endicott in [Pirn01]

[4] "Building […] interfaces that intelligently present information is a difficult task." [Mack86]

(ab 1994) *verschlimmerte* das Software-Unternehmen in den Augen der Grafik-Designer [vgl. Norv99, Godi01, Sipp02, Brow02] sogar noch das optische Erscheinungsbild auf diese Weise erstellter Präsentationen:

> *Those "helpful" tools are the main reason that we've got to live with page after page of bullets, with big headlines and awful backgrounds.*
>
> [Godi01:3]

Neben den „wahrhaft uninspirierten" [Will01] Hintergründen, die das Microsoft-Designer-Team einem „überwältigenden Angebot" [FlRi01:7] verschiedener *Design-Templates* zugrunde legt, prägten, wie etwa *Google*-Forscher Peter Norvig bemerkt, vor allem PowerPoints *Layout-Wizards* automatisch erstellten Präsentationen „sämtliche Regeln schlechten Grafik-Designs" [Norv99] geradezu idealtypisch auf. Die einhellige Ablehnung der durch Microsofts Design-Assistenten „aufdiktierten" Ästhetik von Seiten professioneller Grafiker [vgl. Brow99] konnte indes die überwältigende Akzeptanz des Moduls durch in grafischen Fragen unbewanderte Benutzer nicht verhindern, die das somit bereitgestellte Assistenten-System als willkommene, „nützliche" [vgl. Mora00, FlRi01:9] Hilfestellung dankend annahmen und, wie Microsofts Design-Kritiker zähneknirschend notieren, zu einer „Schwemme unansehnlicher" [Godi01] Multimedia-Dokumente beitrugen, welche die Ästhetik geschäftsbezogener Präsentationen seither in sehr erheblichem Maße zu prägen vermögen.

2.3.2.2.5 Die Schuld des Anwendungsprinzips

Obgleich sich der Katalog durch PowerPoint motivierter „Design-Sünden" [vgl. Park01, Godi01] an den bereits angesprochenen, ästhetischen Kriterien beiweiten nicht erschöpft, sei an dieser Stelle lediglich abschließend bemerkt, dass bereits das Anwendungsprinzip des Programms offensichtlich nicht nur problematischen Einfluss auf Inhalt [s. 2.3.2.1], sondern, wie eben ausgeführt, augenscheinlich auch auf das visuelle Erscheinungsbild der erstellten Dokumente ausübt. Eine weitere, betriebswirtschaftliche Komponente der allgemeinen, primär an die PowerPoint-*Entwickler* gerichteten Kritik stellt zudem der unverhältnismäßig hohe Zeitaufwand [vgl. Ziel02] dar, den die oft „hochbezahlten Anwender"[1] der PowerPoint-basierten Erstellung multimedialer Präsentationen entgegenbringen müssen. Aus dieser Perspektive erscheint auch die durch PowerPoint eingeleitete „Beseitigung" [vgl. Park01] professioneller Grafik-Designer aus dem Erstellungsprozess geschäftlicher Präsentationen in finanzieller sowie ästhetischer Hinsicht durchaus problematisch

> *It would be much more efficient to offload that work onto someone who could do it significantly better in a tenth of the time, and be paid less. Millions of executives around the world are sitting there going, 'Arial? Times Roman? Twenty-four point? Eighteen point?'*
>
> (Cathleen Belleville)[2]

[1] vgl. Catherine Bellevilles Aussagen in [Park01]
[2] zitiert ebenda.

2.3.3 Formatkritik

Neben der rein Programmbezogenen Betrachtung des Anwendungsprinzips und der Eigenschaften der Applikation selbst [1] ist aufgrund der bereits in [2.3.1] dargelegten, „epidemischen" [Mane99] Verbreitung PowerPoints aus Medieninformatiker-Perspektive freilich auch eine Analyse des darauf aufbauenden Dateiformates von Belang: Insbesondere relevant erscheint unter diesem Blickwinkel die Frage, wie die multimedialen Daten logisch und strukturell in der Datei repräsentiert werden, und wie daraufhin mögliche im- und exportierende Dateifilter realisiert werden könnten. Da die fast vollständige Marktabdeckung des Programms [vgl. Bell00:10] zugleich eine sehr hohe Verfügbarkeit entsprechender PowerPoint-Dateien bedingt, können an dieser Stelle auch die in [2.3.2-3] angesprochenen, gravierenden Design-Mängel des Programms die Bedeutung umfassender Informationen über den Aufbau des Formates nicht mindern – im Gegenteil: Gerade *weil* die PowerPoint-Anwendung selber erwiesenermaßen[2] zu unzureichend formulierten Präsentationsdaten verleitet [vgl. Sear98, Godi01], sind für eine darüber hinausgehende Anwendung in einem erweiterten Kontext (*Import*-Filter), und insbesondere für alternative Authoring-Ansätze, die mit aus Design-Sicht unter Umständen ausgereifteren Inhalten auch auf den *Export* in das mittlerweile als *de-facto*-Standard [Wald02] etablierte PowerPoint-Format angewiesen sind, detailliertere Spezifikationen zum Aufbau des Dateiformates bzw. sogar ganze Schnittstellen-Komplexe wie SDKs oder Filter-APIs von großem Interesse. [vgl. O'Don97]

2.3.3.1 PowerPoint-Dateiformat: PPT, das unbekannte Wesen

Leider, und dies als ernüchternde Feststellung vorweg, muss dass PowerPoint zugrunde liegende Dateiformat (mit der Erweiterung *.PPT, bzw. *.POT für entsprechende Template-Dateien) als größtenteils *intransparent* bezeichnet werden – was es trotz seiner großen Verbreitung auch zugleich von einem „echten", offenliegenden Format bzw. *de-jure*-Standard unterscheidet: Die von Microsoft veröffentlichten Informationen über Struktur und Aufbau des Formates [Micr97] sind sowohl unvollständig, relativ veraltet und daraufhin für eine theoretisch interessante Realisierung einer Filter-Schnittstelle relativ uninteressant, als auch die in der Tat erhältlichen SDKs nicht im weiteren Sinne „öffentlich" verfügbar, sondern augenscheinlich stets mit beträchtlichen Lizenzgebühren verbunden sind.[3] Darüber hinaus macht auch die problematische Versionsabhängigkeit der Dateien[4] deutlich, dass die PPT-Dateiarchitektur ursprünglich nie als Austauschformat konzipiert, sondern stets nur im Hinblick auf interne Speicherung entwickelt wurde.

2.3.3.1.1 Verwirrung um Versionen

Als Beispiel hierfür kann etwa der frappierende Kompatibilitätssprung der 97er-Versionen angeführt werden: Trotz „stark verwirrender" [Bell00:9] Versionsnummern (so trug die im Februar 1994 erscheinende Version 4.0 zugleich die Versionsnummer 7, firmierte aber anschließend unter dem Namen PowerPoint 95)[5] hatte das Dateiformat einen aufgrund nahezu unkomprimierter Grafik-Einbettung immensen „Platzbedarf" bis 1997 beibehalten – mit der „siebten"[6] Version (unter dem Namen „Power Point 97") führten die Entwickler im Mai desselben Jahres aufgrund der Implementierung des OLE2-Modells[7] schließlich eine völlig überarbeitete Speicherarchitektur ein, die sich wiederum für sämtliche Vorläuferversionen als gänzlich

[1] s. 2.3.2

[2] s. ebenda

[3] Anm: Dies wird aus den enorm hohen Preisvorstellungen der bislang erhältlichen PowerPoint-Export-Tools (Impatica, Presedia, Wanadu iCreate, PowerConverter) ersichtlich.

[4] vgl. hierzu ebenso die Dokumentationen der Export-Tool-Hersteller, etwa [Impa02]

[5] Quelle: [Bell00] p.9

[6] Anm: Tatsächlich „fuhren die Versionsnummern" mit ebendieser auslieferung „zur Hölle": [Bell00:9] – der Index „sprang" von Version 4 auf 7.

[7] Erklärung hierzu s. weiter unten

unlesbar erwies, auf der anderen Seite jedoch für die kommenden Generationen Rückwärtskompatibilität und erhöhte Speichereffizienz versprach.

Während verschiedene Kritiker derweil zumindest den von Microsoft beschworenen, „platzsparenden" Effekt des neuen Speicheralgorithmus anzweifeln [vgl. Park01, Brow02], so ist zumindest die Aufwärtskompatibilität der seither folgenden Versionen (bis XP) eine sehr positiv anzumerkende Eigenschaft – wenn auch die XML-bezogenen Ankündigungen hinsichtlich der für 2003 angekündigten Folgeversion [vgl. Gall02] diesbezüglich etwas aufhorchen lassen. Für die eigentliche OnScreen-Darstellung speziell PPT97-spezifische Funktionalität[1] nutzender Präsentationen ist allerdings die bis heute relevante, erst seit der 8.0-Version eingeführte, überarbeitete *Rendering-Engine* zu erwähnen, die (auch wenn dieselben Dateien dank Rückwärtskompatibilität ebenfalls in den weit verbreiteten, PPT97-basierten *Viewern*, freilich deutlich unästhetischer, wiedergegeben werden können) optisch deutlich ansprechendere Darstellung desselben Präsentationsmaterials ermöglichen.

2.3.3.1.2 Wohlbehütetes Kind: PowerPoints File Format SDK

Aufgrund der Tatsache, dass auch das einzige, öffentlich verfügbare Dokument, welches aufschlussreiche Details über die interne Struktur des PowerPoint-Speicherformats preisgibt,[2] auf der 97er-Generation des Programms beruht, ist es – trotz unvollständiger Informationen über die Realisierbarkeit in Folgeversionen implementierter Erweiterungen (wie etwa fortgeschrittener Animationsmöglichkeiten) – zwar sehr mühsam, aber dennoch möglich, rudimentäre Im-/Exportfunktionalität zu entwickeln. Das von der Herstellerfirma „für Entwickler verschiedener Anwendungen zur Interaktion mit PowerPoint-Dateien" [Micr97][3] veröffentlichte „Microsoft PowerPoint 97 File Format SDK" enthält darüber hinaus noch eine Auswahl interessanter Hilfsmittel, wie etwa nützliche Code-Fragmente[4] oder einen so genannten *FileViewer*, der den internen Aufbau der PowerPoint-Dateien in Form einer Baumstruktur visualisiert.

2.3.3.1.3 Aufbau des SDK-Formats

Grundkonzept der PowerPoint97-Dateien, und damit überdies auch nahezu aller neueren Microsoft-Office-Dateiformate, ist der strukturierte Dateiaufbau nach dem ole 2 Compound-Format. Dieses Speicherverfahren, eine Implementierung des so genannten „Structured Storage"-Modells,5 stellt eigentlich ein kleines Datei-System innerhalb einer Datei selbst dar, d.h. es beherbergt eine hierarchisches Struktur von fester Speicherorte (storages) und den flexibleren Streams.6 Da eine nähere Erläuterung dieses doch recht komplexen Prinzips an dieser Stelle sicherlich den Rahmen sprengen würde, sei hier der Einfachheit halber auf [Broc95] verwiesen, wo Grundidee und Architektur des Verfahrens eingehend und verständlich dargelegt werden. Zum Verständnis der PowerPoint zugrunde liegenden Speicher-Architektur sei an dieser Stelle lediglich erwähnt, dass alle PowerPoint-Dateien ab 97er Versionsnummer ole DocObject-

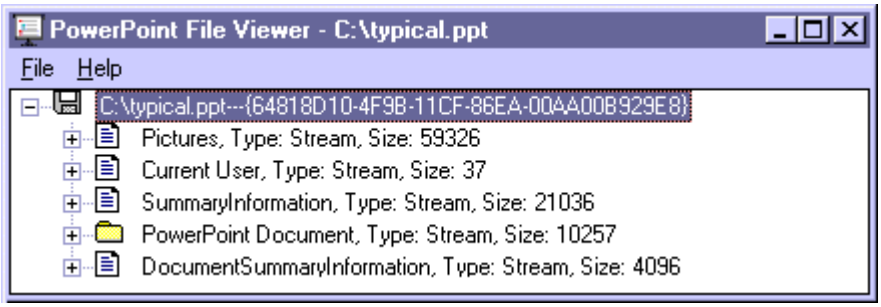


Abb. 2.3.3.1:Die verschiedenen Document-Streams, einschließlich des Power Point Document-Containers ( )

[1] Gemeint ist hiermit insbesondere die Unterstützung transparenter GIF-Dateien

[2] vgl. [Micr97]

[3] vgl. ebenda, Abschnitt "Purpose and Scope" [S62FDC]

[4] Hierbei ist insbesondere die in C++-Syntax verfertigte Datei *serial.h* interessant, in der die einzelnen Dateielemente zum einfacheren Zugriff bereits anrealisiert sind.

[5] Zu deutsch etwa: Strukturierte Speicherung

[6] Datenkette ohne nötigerweise definierten Anfang bzw. Ende, s. hierzu auch [3.5.1]: „Streaming"

Dateien darstellen, welche sich aus lediglich 4 verschiedenen Stream-Arten zusammensetzen: Benutzerinformationen, Bildinformationen, Dokumentinformationen und schließlich dem PowerPoint Document Stream, der die Hauptdaten der Präsentation (Texte, Vektorgrafiken etc.) beinhaltet.

Interessanterweise setzt sich die PowerPoint-Dateistruktur streng Baum-hierarchisch zusammen: So enthält jeder Präsentations-Baum *Container*, die wiederum weitere Elemente enthalten können, und so genannte *Atome*, die die Endknoten des Baumes bilden und die eigentlichen Informationen der Datei beinhalten. Die wichtigsten Daten der Präsentation werden im Falle von PowerPoint freilich in *Slide*-Containern abgelegt, deren einzelne Atome wiederum die jeweiligen Elemente der in der Präsentation enthaltenen Folien repräsentieren.

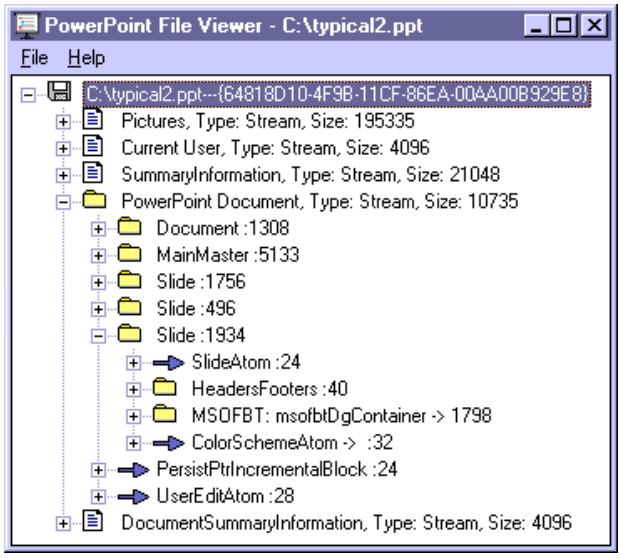


Abb. 2.3.3.2: Slide-Container im PowerPoint-FileViewer (rechts)

Auf weitere Details der im Übrigen recht systemnahen SDK-Dokumentation soll hier verzichtet werden (vgl. hierzu [Micr97]) – festzuhalten ist an dieser Stelle jedoch der verblüffend einfache Dateizugriff, den das Microsoft-SDK vor allem durch die darüber hinaus bereitgestellten, direkt verwendbaren Code-Snippets ermöglicht. Leider hat das Unternehmen nach Veröffentlichung der Spezifikation jedoch offensichtlich das Marktpotential dieser existentiellen Schnittstelle erkannt – die Bearbeitung PowerPoint-basierter Dateien ist seither bedauerlicherweise nur noch durch vom Format selber recht weit abstrahierte Software-Module möglich und die Verwendung Microsoft-eigener, stets wartungsbedürftiger Schnittstellen zum Dateizugriff an offensichtlich hohe Lizenzforderungen geknüpft.

2.3.3.2 Programmatischer Zugriff: Die COM-Schnittstelle

Die einzig „legale“ Möglichkeit,[1] auf PowerPoint-Dateien mittels automatisierter Programmiersprachen-Logik Zugriff zu erlangen, stellt daher Microsofts COM-Schnittstelle dar, über die sich relativ unkompliziert verschiedene Microsoft-Module skriptgesteuert kontrollieren lassen. Allerdings ist hierfür sowohl die Verwendung *Windows*-basierter System-Umgebungen als auch eine bereits vorliegende PowerPoint-Installation erforderlich. Für universale, serverseitige Lösungen ist diese Lösung somit weithin uninteressant – trotz bereits existierender „Brücken“ zur weit vielseitigeren Programmiersprache Java: Durch einen (relativ dünnen, aber immerhin logisch aufgebauten) COM-„Wrapper“ lässt sich beispielsweise die PowerPoint-Generierung über ein Java-Programm automatisieren. [Micr99] zeigt anhand eines (in diesem Falle „trivialen“) Beispiels auf, wie die PowerPoint-Applikation durch in Visual J++ realisiertem Code quasi „ferngesteuert“ werden kann:

```
import msppt8.*; // PowerPoint-Support. Für 2000: msppt9.* bzw. msppt.* für 2002
...
Application app = new Application(); // PowerPoint starten
Presentation pres = app.getPresentations().add(1); // Präsentation hinzufügen
// Slide mit Text-Layout hinzufügen:
Slide slide = pres.getSlides().add(1, PpSlideLayout.ppLayoutText);
Item title = slide.getShapes().getTitle(); // Titel-Element
title.getTextFrame().getTextRange().setText("Mein Titel!"); // Text zuweisen!
...
```

Listing 2.3.3.1: Erstellung einer PowerPoint-Präsentation über den COM-Wrapper *von* Visual J++.

[1] Anm: Hierbei würde ein durch die SDK-Spezifikation bereits angedachter Low-Level-„Hack“, also die direkte Manipulation von PowerPoint-Dateien ohne Berücksichtigung der Microsoft-Auflagen, ausgeschlossen

Über einen ggf. komplexeren, „intelligenten“ Algorithmus ließen sich auf diesem Wege zwar keine pixelgenaue, dem Format vollständig entsprechende Transformationen erreichen, aber immerhin etwa relevante Informationsdaten automatisiert *in* eine PowerPoint-Datei *schreiben* oder auch relevante Inhalte aus bestehenden Präsentationen *extrahieren*. Eine geringfügig komfortablere Möglichkeit zur „Microsoft-Fernbedienung“ stellt darüber hinaus die kommerzielle COM-Brücke „J-Integra“ der Firma Intrinsyc dar, die die PowerPoint-Automation ebenfalls mit konkreten Realisierungsansätzen [vgl. Intr98] eindrucksvoll zur Schau stellt. Auch diese Lösung, die im Übrigen lediglich die Bereitstellung der COM-Schnittstelle vereinfacht, macht hingegen, ebenso wie der bereits von Microsoft veröffentlichte Ansatz [Micr99], meiner Einschätzung nach eher den Eindruck eines recht umständlichen „Workarounds“, als eine „echte“, Java-basierte Alternative einer möglichen Server-Anwendung zur PowerPoint-Verarbeitung aufzuzeigen.

2.3.3.3 Export-Workaround: WMF

Ein weiteres „Workaround“ stellt indes die Verwendung der von Microsoft selbst angebotenen WMF-Schnittstelle dar [s. hierzu auch 4.3.2.1]: In Verbindung mit den soeben angesprochenen, Java-basierten COM-Wrappern ließe sich etwa der in alle PowerPoint-Versionen integrierte WMF-Export der Präsentations-Slides automatisieren, um diese, möglicherweise wiederum als Bestandteil einer Java-basierten Serveranwendung, weiter zu verarbeiten. Hierfür geeignete Konvertierungs-Module, die eine Verarbeitung der WMF-Grafiken in Java ermöglichen, sind bereits in Großer Anzahl vorhanden: Neben den primär zur Applet-Anwendung konzipierten WMF-Decodern des lizenzfrei zugänglichen Viewers in [Klei00] sowie dem durch Piet Jonas von der Universität Greifswald öffentlich gemachten Java-WMF-Framework [Jona02] sind dies zumeist Konverter-Module, die sich speziell auf den Internet-Vektorstandard SVG konzentrieren. Insbesondere seien an dieser Stelle die diesbezüglichen Vorarbeiten von Carmen Delessio [Dele97,99] sowie der WMFtoSVG-Transcoder des ebenfalls Java-basierten Batik-Projekts der Apache Group [vgl. Croo01] erwähnt.

Trotz der Eleganz dieser durchaus gangbaren WMF-Option erscheint im Hinblick auf die durchaus beeindruckenden, multimedialen Funktionalitäten PowerPoints die Tatsache, dass durch die eben angesprochene Konvertierung der einzelnen *Slides* in deren WMF-Entsprechung sämtliche dynamische, interaktive, auditive und audiovisuelle Information und somit ein Grossteil der prägenden Dynamik einer Präsentation verloren geht, als doch sehr beträchtliches Manko gegenüber einer „echten“ Filter-Lösung, in der sämtliche präsentationsspezifischen Eigenschaften des PowerPoint-Dateiformates berücksichtigt würden.

2.3.3.4 Fazit und Ausblick: SVG als Hoffnungsschimmer?

Abschließend muss daher leider festgestellt werden, dass aufgrund der *Intransparenz* des Dateiformates selber sowie der nur mäßigen Bemühungen von Seiten des Hersteller-Unternehmens Microsoft, bereits existierende Ansätze [vgl. Micr97] zu soliden, großflächig genutzten Schnittstellen auszubauen [vgl. O'Don97], Filter-Ansätze zur Bearbeitung des PowerPoint zugrunde liegenden Speicherformates zwar prinzipiell *möglich*, jedoch, wie soeben aufgezeigt, recht mühsam zu realisieren sind. Den einzigen Lichtblick stellt an dieser Stelle lediglich die Ankündigung [Paol02] Microsofts dar, für die künftige Version 11 ihres Office-Pakets einschließlich der PowerPoint-Komponente vollständigen *SVG*-Im- und Export bereitzustellen [vgl. LiJa03]: Dies würde insbesondere im Hinblick auf die im Rahmen dieser Diplomarbeit und auf Basis von SVG [s.5.4] realisierte Präsentationslösung[1] eine fast *ideale* Schnittstelle darstellen – da sich die Office11-Suite jedoch derzeit erst dem „Beta 2“-Stadium nähert und eine Veröffentlichung der Software daher noch einige Zeit in Anspruch nehmen wird (von einer entsprechende Verbreitung ganz zu schweigen), ist dies jedoch momentan noch Zukunftsmusik.

[1] s. hierzu Kap. 6

2.3.4 Web-Based PowerPoint

Derweil rückt allerdings auch das Software-Unternehmen selber vom mit der Einführung der 97er-Version zunächst angedachten Konzept eines eigenständigen, *nativ webfähigen* PowerPoint ab. So merkt etwa Microsoft-Produktmanagerin [Bell00] an, dass sich der Fokus des Produktes ab Mitte 1997 in Richtung „elektronischer Präsentationen und Online-Dokumente" verschoben habe.[1] Zu diesem Zweck wird das Programm neben dem zur optimalen Portabilität vorgesehenen „Pack and Go"-Konzept [2] um eine *Save-to-HTML*-Funktion ergänzt, mit der sich die Präsentationssoftware auch für das Internetzeitalter gerüstet sieht. Trotz dankbarer Akzeptanz und intensiver Benutzung durch die Anwender stößt die Funktionalität des Web-Exporttools jedoch bei Internet-Experten auf herbe Kritik:

> *The web-enabled form of PowerPoint […] is not good to begin with, since PowerPoint is not good on the Web, no matter what Billy Gates says.*
>
> [Will02]

2.3.4.1 Nativer Web-Export

2.3.4.1.1 Save to Web / Save to HTML

Und in der Tat macht die Betrachtung der recht unästhetisch [3] in Richtung Web-Deployment exportierten, rein statischen JPEG-Bilder und insbesondere eine Analyse des sehr unstrukturiert erzeugten HTML-Codes deutlich, mit welch heißer Nadel die Internet-Funktionalität ganz offensichtlich gestrickt wurde. Auch das in der Folgeversion 1999 nachgelegte „Save to Web"-Modul „verschlimmbesserte" die Funktionalität des Export-Filters noch zusehends: So gelang es Microsoft zwar, sowohl den eigenen VML-Standard zur Beschreibung vektorbasierter Grafiken [s. 5.3.3.4], als auch sehr weitgehende JavaScript-, CSS- und DHTML-Unterstützung in die Webdokumente zu integrieren – allerdings deutlich auf Kosten struktureller „Sauberkeit": Neben einer überwältigenden Fülle generierter „Export"-Dateien gesellt sich auch innerhalb der erstellten Webdokumente ein heilloses Chaos ungeordneter, HTML-konformer wie auch Microsoft-eigener Tags, sodass eine Weiterverwendung derart „aufbereiteter" Dateien völlig unmöglich erscheint[4] – von einer „professionellen, web-geeigneten Lösung" [Microsoft-Pressemitteilung] kann also an dieser Stelle meiner Einschätzung nach *nicht* die Rede sein.

2.3.4.1.2 PowerPoint Producer

Diese doch sehr offensichtlichen, strukturellen Mängel des PowerPoint'schen Internet-Exports versuchte das Unternehmen schließlich noch Ende 2001 [Lock01] mit einer leider ausschließlich für die XP-Version konzipierten[5] Zusatzanwendung zu beheben: Der für PowerPoint-Benutzer von Microsoft kostenfrei [vgl. Kara02] bereitgestellte „*Producer* für PowerPoint 2002" [6] erweitert die bereits in der Vorgängerversion zum Web-Export vorgesehenen Möglichkeiten noch um zusätzliche *Voice-Over*- und *Streaming*-Funktionalität.[7] Leider macht eine genauere Betrachtung des durch das Zusatzprogramm bereitgestellten Export-Algorithmus abermals deutliche Schwächen des generierten Materials deutlich: Eine Verbesserung der recht verflochtenen, verworren erscheinenden Code-Struktur ist im Vergleich zum PPT2000-Export nicht ersicht-

---

[1] vgl. [Bell00] p.10

[2] Das „Pack and Go"-Konzept sieht lediglich die Komprimierung und Einbettung verwendeter Schriftarten und Meta-Dokumente vor, um PowerPoint-Dokumente per Diskette auf andere Endgeräte übertragen zu können.

[3] Neben deutlich sichtbaren Bild-Artefakten versagt der PowerPoint'sche Export-Filter den resultierenden Grafiken ein optisch zumindest angenehmeres *Anti-Aliasing*.

[4] s. hierzu auch 5.3.3.2

[5] „Überdies ist die Exklusivität hinsichtlich der ausschließlichen 2002-Unterstützung zu bedauern…", aus: *ASCII PC Explorer*-Magazin (o.V) „Microsoft Producer for PowerPoint 2002." Tokio, 27. Februar 2002 [mittlerw. offline]

[6] Offizielle Referenz-Addresse: http://www.microsoft.com/office/powerpoint/producer

[7] s. hierzu auch 3.5.1

lich, zudem verhindern lange Ladezeiten und fehlende Synchronisierung zwischen Audio, Video und eigentlichen *Slide*

Abb. 2.3.4.1: Producer-Show (rechts)

Da die mit Microsofts *Producer* erstellten, vorgeblich „webfähigen" Präsentationen überdies ausschließlich auf der Windows-basierten Browser-Software *Internet Explorer* der Versionen 5.5 und später [vgl. Lock01] ablauffähig sind, stellt sich an dieser Stelle meiner Ansicht nach die Frage, inwiefern an dieser Stelle noch von einem „an alle" Endgeräte gerichteten Web-Export gesprochen werden kann, da ein bei Verwendung der durch den Software-Riesen angebotenen Funktionalität ein Großteil[1] der Internet-Endbenutzer ausgeschlossen wird. Aufgrund dessen empfiehlt sich Ansicht nach an dieser Stelle schon eher die direkte Verwendung *nativer* PPT-Dateien auch auf HTTP-Basis, wie etwa von vormaligen Microsoft-Entwicklern selbst vorgeschlagen [vgl. Bell00]: Durch die bereits in [2.3.3] angesprochene COM-Funktionalität innerhalb der als gegebene Voraussetzung (s.o.) angenommenen Windows-Umgebung lassen sich PowerPoint-Dateien (ähnlich der Browser-Integration des PDF-Viewers in [4.2]) direkt im Internet Explorer betrachten. [Brow01] spricht überdies einen weiteren bedeutsamen Vorteil dieser Vorgehensweise an: Im Gegensatz zu in der Regel ausschließlich zur *Betrachtung* exportierter Web-Dokumente[2] lassen sich PowerPoint-Dateien ohne Qualitätsverlust sogar komfortabel weiterverarbeiten und ermöglichen somit eine Internet-basierte, dynamische Zusammenarbeit („*Online Collaboration*"). Unter diesem Aspekt lässt sich überdies, neben im Gegensatz zum reinen Web-Export ebenfalls weiterhin verfügbaren Folienübergangs- und Dynamikfunktionen, eine weiterer Vorzug des PowerPoint-Dateiformates [3] nutzen: So ist im Rahmen der neueren PowerPoint-Versionen etwa die *Einbettung* verwendeter Schriftarten durchaus möglich – wenn auch der Weg dorthin recht umständlich erscheint und daher, wie [Endi02a] anmerkt, „keine exakte Wissenschaft" darstellt.[4]

Das Einfügen gesprochener Kommentare oder gar ganzer Video-Sequenzen, die „Web-basierten" Präsentationen den „realen Vortrag" ersetzen bzw. den präsentierten PowerPoint-Inhalt angemessen begleiten könnten, ist aufgrund des enormen Platzbedarfs [vgl. Endi01] und insbesondere der mangelnden *Streaming*-Funktionalität der nativen PPT-Dateien bei direkter Web-Verwendung PowerPoints jedoch nur erschwert möglich und erscheint überdies wenig Internet-tauglich. Aufgrund dessen erstaunt es nur wenig, dass neben dem mit 2002 reichlich spät erschienenen, bislang recht spärlich genutzten *Producer*-Zusatzprogramm des PowerPoint-Herstellers selber eine Fülle weiterer Drittanbieter mit zahlreichen Programmen und Komplettlösungen die auch aufgrund der [bereits beschrieben] unbefriedigenden Internet-Exportfunktionalität PowerPoints entstandene Marktlücke zu füllen versucht.

[1] Etwa Browser-Endgeräte, die mit der Internet-Explorer-Engines unterhalb der 5.5er-Versionsmarke betrieben werden, sowie freilich auch sämtliche, alternative Browser wie Netscape, Mozilla, Opera etc.

[2] Zu dieser Kategorie können freilich fast sämtliche Internet-Formate gezählt werden: HTML, JPEG, Flash … Die Ausnahme stellt hierbei das Vektor-Format SVG und in begrenztem Umfang [vgl. Brow01] auch das PDF-Format.

[3] Ab Version 7.0, s. [2.3.3]

[4] "[…] Font-embedding is not an exact science and adds to the overall file size." [Endi02a]

2.3.4.2 Third-Party-Lösungen

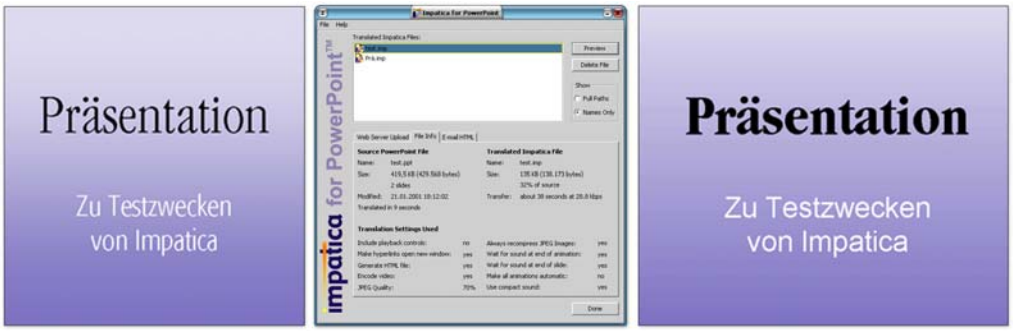
2.3.4.2.1 RealPresenter / PresenterOne

Ein der Microsoft *Producer*-Entwicklung vorausgegangener Ansatz, PowerPoint-basiertes Präsentationsmaterial durch aufgesprochene Kommentare oder gar visuelle Vortragssequenzen für das Internet aufzubereiten, stellt beispielsweise das für die RealONE-Plattform entwickelte Produkt *RealPresenter* (mittlerweile *PresenterONE*) der Firma Accordent dar.[1] Im Gegensatz zum reichlich später veröffentlichten Microsoft-Pendant, das neben augenscheinlich dem *RealPresenter* entlehnten Elementen für die Präsentationsdarstellung auf mit proprietären Microsoft-Tags „ergänzten" HTML-Code setzt,[2] bettet diese *Streaming*-orientierte Lösung die einzelnen, statischen Slides in einen derweil in Microsofts *Producer* schmerzlich vermissten[3] SMIL-Kontext [vgl. Moss01] ein,[4] der das Audio/Video-Zusatzmaterial mit den sichtbaren Präsentations-„Visuals" synchronisiert.

Obgleich die PresenterONE-Plattform über die reinen PowerPoint-Daten hinaus die Integration weiterer Elemente wie etwa *Flash*-Content[5] ermöglicht, bleiben die eigentlichen Möglichkeiten der Präsentation indes auf rein statische Slide-Bilder beschränkt – interaktive oder dynamische Elemente (wie etwa Folienübergänge) der PowerPoint-Daten selber können in der noch auf SMIL 1.0 aufbauenden[6] RealPresenter-Lösung wie auch im Nachfolger PresenterONE nicht abgebildet werden. Zusätzliche Multimedia-Funktionalität lässt sich zwar im Nachhinein per SMIL-Syntax relativ einfach wieder hinzufügen, die „Authoring Simplicity" PowerPoints und somit das letztendlich ja einzig für das Microsoft-Format sprechende Argument ginge jedoch aufgrund des hierfür vorausgesetzten SMIL-Know-Hows wieder verloren [vgl. Moss01]. Überdies schafft der auf diese Weise realisierte Web-Export, da die generierten Daten zwar PowerPoint-unabhängig, aber nun wiederum auf einen SMIL-fähigen Player angewiesen sind,[7] zusätzliche Abhängigkeit von einem spezifischen Wiedergabe-Tool auf dem jeweiligen Endgerät.[8]

2.3.4.2.2 Applet-basierte Lösungen

Die universelle, betriebssystemunabhängige Programmiersprache *Java* und hier insbesondere das im Kontext einer Browser-Umgebung interessante Konzept der Java-*Applets* [s.3.5.3], welche aufgrund ihrer großen Verbreitung nahezu überall[9] ablauffähig sind, stellt an dieser Stelle wiederum eine möglichen Ausweg aus der „Kompatibilitätsfalle" dar. Das kanadische Software-Unternehmen *Impatica* bietet daher mit dem gleichnamigen Produkt „Impatica for PowerPoint" eine elegante Lösung an, die PowerPoint-Dateien automatisch in mittels Java-Applet automatisch ablauffähige Präsentationen umzuwandeln. Die im Zwischenschritt erzeugten .IMP-Dateien sind dabei nicht nur in der Lage, einen Großteil [vgl. Impa02] der durch PowerPoint realisierten Dynamik-Funktionalität Internet-fähig und systemunabhängig abzubilden, sondern erreichen darüber hinaus dank JPEG-Optimierung eine zusätzliche Komp-



Impatica-Konversion

[3] vgl. die entsprechende Kritik des japanischen *ASCII PC Explorer*-Magazins: „Microsoft Producer for PowerPoint 2002." Tokio, 27. Februar 2002 [mittlerw. offline]

[4] zu SMIL siehe 5.5

[5] s. 4.5

[6] s. 5.5

[7] s. hierzu die Player-Diskussion in [5.5]

[8] vgl. hierzu auch den entsprechenden Kommentar des deutschen Oplayo-Marketingleiters auf [Kara02].

[9] Mit Einschränkungen, nicht nur bei alten, sondern insbesondere auch neuen Microsoft-Browsern [s. hierzu 3.5.3]

rimierung der Präsentationsdaten.

Trotz dieser beeindruckenden Funktionalität macht [Abb. 2.3.4.2.2.1] die signifikante Schwäche *Impatica*s hinsichtlich typografischer Konvertierung deutlich: So werden vom Windows-Standard abweichende Schriftarten weder in die *Impatica*-Präsentationsdatei eingebettet, noch durch das Viewer-Applet korrekt dargestellt, sodass die Online-Präsentation von der ursprünglichen PowerPoint-Datei in erheblichem Maße negativ abweicht. Die Rendering-Engine des Applets verfügt überdies über keinerlei Anti-Aliasing-Funktionalität, sodass das sichtbare Ergebnis in ästhetischer Hinsicht relativ unbefriedigend ausfällt.

Einen weitaus spannenderer, ebenso vollständig auf der Java Applet-Technologie aufbauender Ansatz ist hingegen das vor allem auf Echtzeit-Video-Streaming hin orientierte Präsentationssystem *Oplayo*, das darüber hinaus über eine angesichts der problematischen Java-Performance beeindruckend schnelle Rendering-Engine verfügt. So ist das stark optimierte[1] Viewer-Applet etwa in der Lage, selbst ganze Video-Streams flüssig und in Echtzeit darzustellen. Daher verwundert es kaum, dass sich die Funktionalität des Systems mittlerweile insbesondere auf Java-fähige, mobile Telekommunikations-Endgeräte konzentriert [vgl. Voge02]. Die Fähigkeiten *Oplayo*s hinsichtlich eines auch diesbezüglich ausgesprochen interessanten PowerPoint-Imports verbleiben jedoch – offensichtlich aufgrund fehlender Lizenzabkommen mit Microsoft – auf einem derzeit relativ enttäuschenden Niveau: Im Gegensatz zur konkurrierenden *Impatica*-Lösung, welche fortgeschrittenere PowerPoint-Funktionalität mit unterstützt, ist das Authoring-Tool von Oplayo lediglich in der Lage, einzelne, statische PowerPoint-Slides zu importieren. Dennoch besteht darüber hinaus die Möglichkeit, im Rahmen der *Oplayo*-eigenen Authoring-Tools „Media Designer" [Hans02] und „Composer" [vgl. Muel02b], PowerPoint sogar weit überlegene Präsentationen einschließlich optisch ansprechenden Text-Renderings zu erstellen.

Abb. 2.3.4.2.2.2: Oplayo in Aktion

Sowohl die erstellenden Autoren-Programme als auch die überzeugenden Performance-Daten des proprietären Video-Codecs MVQ [vgl. Lero02] wie auch des flotten Viewer-Applets machen daher diese vollständig Java-basierte Alternative des finnischen Software-Herstellers *Oplayo* zu einem würdigen Kandidaten für eine denkbare, streaming-orientierte Präsentationslösung. Als mögliches „Master-Format" für web-basierte Multimedia-Präsentationen kann diese strukturell durchaus interessante Lösung dennoch aufgrund der kommerziellen, proprietären Vermarktungspolitik des Formates sowie der bislang unzureichenden[2] PowerPoint-Schnittstellen auf der Authoring-Seite leider nicht in Betracht gezogen werden.

Die überdies bislang noch unbefriedigende Integration leistungsstarker *Java*-Engines in führende Internet-Browser[3] rückt darüber hinaus noch eine weitere Alternative zum Java-basierten Präsentations-Deployment ins Blickfeld: Die Verwendung des weitaus verbreiteteren Vektor-Animations-Formates *Flash*, das laut [Macr02a] bereits „bei 97,8 % aller Internet-Nutzer ohne zusätzliches Wiedergabe-Programm ablauffähig ist".[4] Da in Kapitel [4.5] die weiteren Details des Formates ohnehin noch ausführlich dargelegt werden, sei an dieser Stelle lediglich die im Hinblick auf den PowerPoint-bezogenen Internet-Export relevante Eigenschaft erwähnt, dass das *Flash*-eigene Internet-Deployment-Format SWF über die ansonsten übliche Autho-

[1] vgl. Lou Hirsh: "New Tech Aims To Speed Video to Mobile Devices." NewsFactor Network, 12. März 2002 http://sci.newsfactor.com/perl/story/16745.html [20.2.03]

[2] Die Integration ist bislang lediglich via einzel-Slide-Import [wie oben erläutert] oder der direkten Integration PowerPoint-nativer Daten möglich.

[3] Insbesondere ist hierbei die Weigerung Microsofts, die Applet-untersützende JVM standard-mässig in derzeitige und künftige Versionen des Internet Explorer zu integrieren [vgl. Viol01, Roth98], relevant (s. hierzu auch 3.5.3)

[4] vgl. [Macr02] p.1

ring-Option mittels des Standard-Animationstools „Flash MX“ hinaus auch als File-Spezifikation offiziell *publiziert* wurde [Macr98,02b], und überdies durch offizielle SDKs sowie weitere Schnittstellen verschiedener Drittanbieter [Main00, Dabb01] relativ komfortabel im- und exportiert werden kann.

2.3.4.2.3 Flash-basierte Lösungen

Diesen Umstand macht sich nun eine ganze Reihe findiger Software-Unternehmen [Ice01, Fult01, Pres02, Wana02] zu Nutze: Aufgrund der Tatsache, dass sich ein Grossteil der PowerPoint-spezifischen Präsentationsdaten ohnehin aus text- und vektorbasierten Grafikelementen zusammensetzt und sich überdies auch Animationen sowie Übergänge optimal durch das SWF-Format abbilden lassen, besteht die Funktionalität gleich vier unterschiedlicher Software-Lösungen in der Hauptsache darin, die PowerPoint-basierten Dateiinformationen möglichst exakt in deren Web-Pendants zu konvertieren.

Da SWF-exportierende Schnittstellen, wie bereits angesprochen, bereits zuhauf öffentlich und kostenfrei verfügbar sind [vgl. Star01], stellt das Hauptproblem dieses Prozesses dabei freilich die Extraktion relevanter Präsentationsdaten aus dem Microsoft-eigenen PowerPoint-Format dar.[1] Zu diesbezüglichen Äußerungen ließ sich bedauerlicherweise bislang keines der herstellenden Unternehmen hinreißen – allein das enorme Preisniveau der angebotenen Lösungen weist an dieser Stelle jedoch abermals auf augenscheinlich beträchtliche Lizenzzahlungen hin, die mit dem Zugriff auf geeignete Schnittstellen offensichtlich verbunden sind.

Eine relativ schlichte, aber solide erscheinende Konvertierung ohne zusätzliche Streaming-Funktionalität bieten hierbei die daher auch ohne Internetverbindung funktionsfähigen Programme IceSLIDE der Firma IceWEB Communications [vgl. Ice01] sowie der „PowerCONVERTER“ (vormals „PowerPresenter“) von PresentationPro [Pres02]. Ein Großteil der optischen Eigenschaften der Präsentation (wie Schriftarten und Grafikobjekte) konnte, wie verschiedene Testläufe zeigten, auf diese Weise recht zufrieden stellend in das *Flash*-Format überführt werden – die unzureichende Konvertierung etwa Verlaufs-befüllter Hintergründe sowie mangelhafte Sound-Unterstützung[2] hinterlassen an dieser Stelle allerdings einen etwas durchwachsenen Eindruck.

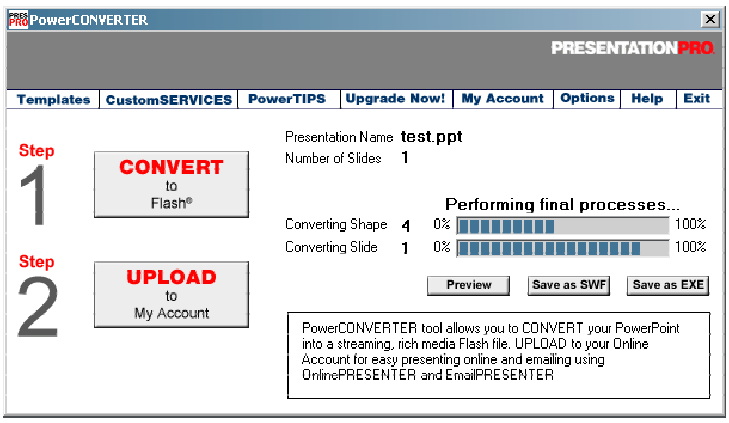


Abb. 2.3.4.2.3.1: PowerCONVERTER in Aktion

Weitaus interessanter erscheint an dieser Stelle hingegen die modularisierte Converter-Engine des kalifornischen Software-Herstellers Wanadu namens *iCreate*. Aufgrund der bereits in [2.3.3] angesprochenen Baumstruktur der PowerPoint-Architektur bildet der iCreate-Konverter die PowerPoint-Datei mittels „vector-to-vector“-Algorithmus [Wana02] zunächst als strukturierte, textbasierte XML-Datei ab,[3] zusammen mit den dazugehörigen XSL-Style Sheets und indiduellen Medien-Elementen.[4]

Leider war von Seiten des Unternehmens keinerlei Information über Struktur und eigentlichen Aufbau dieser intermediären XML-Dokumente erhältlich – die prinzipielle Transparenz dieser modularisierten, flexib-

---

[1] s. 2.3.3

[2] vgl. Geetesh Bajaj: “PowerCONVERTER Reviewed.” *Indezine* Magazine, Hyderabad 2000
URL: http://www.indezine.com/products/powerpoint/addin/powerconverter.html [12.2.03]

[3] s. hierzu auch d. XML-Kapitel 5

[4] “The Content Extraction and Publishing engine extracts all media from PowerPoint documents in a vector-to-vector conversion into a unified XML format, along with corresponding XSL style sheets and individual media elements…” [Wana02]

len Architektur der intermediären Präsentationsinformationen, in dem sich laut Firmeninformationen überdies auch „alle Animations-Effekte, Hintergründe, Templates, Schriftarten, Übergänge und selbst Audio-Informationen“ abbilden lassen,[1] erlaubt jedoch prinzipiell sämtliche erdenklichen Internet-Exportformate.

So wäre beispielsweise aus theoretischer Sicht auch eine (mittlerweile durch Microsoft ohnehin selbst angekündigte)[2], verlustlose Überführung der Präsentations-Daten in den Internet-Vektorstandard SVG [Kap. 5.4] denk- und machbar – indes konzentrieren sich die Wanadu-Entwickler bislang ausschließlich auf Macromedias *Flash* als bisher einziges Export-Format, vorgeblich „um ein Streaming der Inhalte zu ermöglichen und lange Download-Zeiten zu vermeiden.“ [Wana02]. Da der gesamte Konversionsprozess darüber hinaus ein so genanntes *Black Box*-System bildet, sind bedauerlicherweise bisher keine weiterführenden Verwendungsformen der wohl aufgrund des immensen Verkaufspreises[3] dieser Software-Lösung intransparent gehaltenen Konversions-Engine möglich. Prinzipiell jedoch stellt diese Herangehensweise gerade im Hinblick auf deren XML-basierte Vielseitigkeit einen äußerst interessanten Ansatz der PowerPoint-Konvertierung dar. Unter (ausschließlicher) Verwendung der durch Flash's MX-Technologie realisierten Mikrofon- und WebCam-Unterstützung bietet das Unternehmen neben dem bereits angesprochenen *iCreate* (übrigens sowohl als Desktop- als auch Serverlösung verfügbar) mit *iConference* zudem eine Konferenz-basierte Präsentation der konvertierten Präsentationsdaten an – aufgrund des nicht unbedingt Internet-optimierten Flash-Videos fällt die Streaming-Funktionalität dieser Konferenzlösung aufgrund ihrer starken Abhängigkeit von der MX-Umgebung jedoch etwas ernüchternd aus.

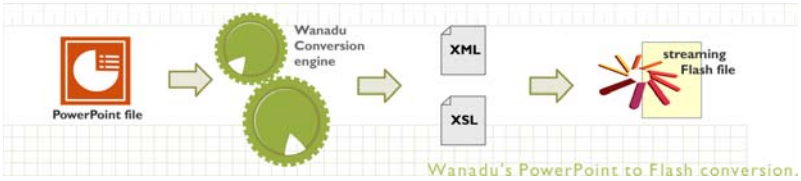


Abb. 2.3.4.6: PowerPoint-to-Flash-Konvertierung mit Wanadu's iCreate

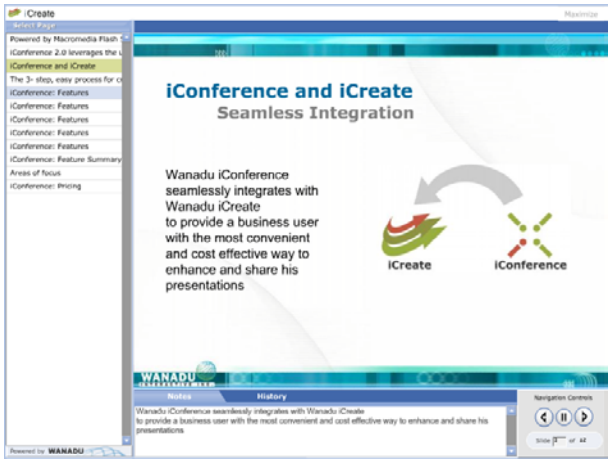


Abb. 2.3.4.7: iCreate in Aktion

Die ebenfalls auf der Flash-Technologie aufbauende Konkurrenzlösung *Presedia* [vgl. Fult01] macht hingegen deutlich, dass die *Streaming*-Problematik hinsichtlich einer zusätzlichen Video-Integration erkannt und daher im Gegensatz zur rein *Flash MX*-basierten Wanadu-Lösung mittels alternativer Übetragungsprotokolle[4] erheblich flexibler angegangen wurde. Interessanterweise hat sich bei den Entwicklern des primär auf e-Learning-Anwendungen ausgerichteten Presedia-Frameworks offensichtlich eine Erkenntnis durchgesetzt, die bereits in einem Rahmen eines e-Learning Semesterprojekts der FH Furtwangen veröffentlicht[5] wurde: Dass hinsichtlich einer speziell Informations- und Lehrzwecken dienende Online-Präsentation bzw. Vorlesung neben den eigentlichen, visuellen Präsentationsdaten lediglich auditive Zusatzinformation von Belang[vgl. FaSu97], die optische Übertragung des Vortragenden[6] (allein aus Bandbreiten-bezogenen Überlegungen) jedoch redundant ist. Daher kommt bei der Integration PowerPoint-basierter Präsentationen im Rahmen des Flash-Exports lediglich ein auditives, überraschend hochqualitativ gestreamtes *Voice-Over* zum Einsatz, was die kognitive Konzentration auf die

[1] vgl. ebenda.
[2] vgl. [Paol02]
[3] Die Firma selbst nennt in [Wana02] Einstiegspreise „ab bereits“ 10.000 US-$ für das erweiterte iConference-System
[4] Im Rahmen der Presedia-Lösung kommen zusätzlich die Real- und Windows Media-Streamingtechnologien zum Einsatz
[5] Projekt „E-Learning“ SS/WS 2000/2001, Leitung: Prof. Schäfer-Schönthal
[6] s. hierzu die Abbildungen 2.3.4.1 und 2.3.4.2

eigentlichen Präsentationsinhalte deutlich Erleichtert. Bei zusätzlich angeforderter Video-Übertragung bietet Presedia darüber hinaus hingegen die Verwendung der auf diesem Bereich deutlich weiter entwickelten *Real-* und *Windows Media*-Technologien an. Es verwundert daher kaum, dass neben der bereits im Hochpreissegment angesiedelten[1] Konvertierungssoftware selbst auch umfassende Hosting- und Streaming-Services integrale Bestandteile des Presedia-Businessplans bilden.[2]

2.3.4.3 Fazit

Abschließend bleibt an dieser Stelle festzustellen, dass die durch das PowerPoint-Paket selber erzeugten Internet-Präsentationen recht unbefriedigend ausfallen [vgl. Will02],[3] die verfügbaren Schnittstellen hinsichtlich einer Weiterverarbeitung der Dateiformates durch eigene Applikationslogik jedoch, wie bereits in [2.3.3] ausgeführt, unter diesem Aspekt ebenso uninteressant erscheinen. Die daher in großer Zahl angebotenen [vgl. Ice01, Fult01, Pres02, Wana02] Konvertierungs-Lösungen, die das native PowerPoint-Format in die großflächig unterstützen [vgl. Macr02a, Bels97] Internet-Technologien *Flash* und die auf der Programmiersprache Java aufbauende *Applet*-Plattform überführen, bieten zwar mitunter hochinteressante [vgl. Wana02] und ebenso funktionstüchtige Ansätze, liegen jedoch, wohl aufgrund der von Microsoft auferlegten Lizenz-Bestimmungen, allesamt preislich sogar noch erheblich *über* dem Niveau des PowerPoint-Programms selber. Dies hat sowohl eine nennenswerte Verbreitung dieser Lösungen bislang verhindert, als es auch deren Relevanz im Rahmen dieser Diplomarbeit zugleich deutlich herabsetzt, da im Hinblick auf flächendeckende, sowohl Internet- als auch multimediataugliche Präsentationsmöglichkeiten primär frei zugängliche und somit „massentaugliche" Konzepte für diese Arbeit von Interesse sind.

Daher stellt sich an dieser Stelle die berechtigte Frage, inwieweit die soeben besprochenen Anwendungsprizipien [s. 2.3.2] und die damit verbundene Format-Architektur PowerPoints [2.3.3] denn überhaupt für ein Internet-orientiertes Authoring geeignet sind. Von Interesse ist unter diesem Gesichtspunkt insbesondere die Diskussion, ob das sowohl der Anwendung als auch dem Format zugrunde liegenden Konzept dramaturgischer *Linearität* nicht bereits von Grundsatz her der primär auf Interaktivität beruhenden Leitidee des *World Wide Web* widerspricht. Des weiteren legt die in [2.3.2] besprochene Anwendungsumgebung bereits vor Beginn des Gestaltungsprozesses eine den Bilschirm-Dimensionen entlehntes Bildverhältnis von 4:3 fest, während WWW-basierte Endanwendungen hinsichtlich ihrer pixel-orientierten Ausmaße i.d.R. stark variieren und daher eine weit höhere „räumliche"[4] Flexibilität erfordern. Darüber hinaus wirft die soeben behandelte Problematik des Web-Exports nicht zuletzt die kritische Frage auf, inwieweit Datenstrukturen und Inhalte multimedialer (PowerPoint-) Präsentationen für das Internet verfügbar gemacht[5] und zugleich etwa ästhetische Darstellungskriterien des *World Wide Web* erfüllt werden können (was ich wiederum eher durch die zuletzt angesprochenen Java/Flash-Ansätze zufrieden stellend umgesetzt sehe). Natürlich dient auch diese Diskussion zu einem Gewissen Grad auch der Hinführung auf einem möglichen *3. Weg*, der beispielsweise mithilfe des Web-Standards SVG eine überzeugendere Alternative aus dieser verzwickten Problematik darstellen könnte.

---

[1] Den Preis des Basic-Pakets beziffert Geetesh Bajaj (*Indezine*-Magazin) auf allein 395 US-$; Das Unternehmen gibt Preisinformationen derweil nur auf Anfrage bekannt.

[2] vgl. hierzu Getesh Bajajs entsprechende Produkkritik („Presedia Reviewed") im Internet-Präsentationsmagazin *Indezine*: http://www.indezine.com/products/powerpoint/addin/presedia.pdf [24.2.03]

[3] "The web-enabled form of PowerPoint […] is not good to begin with, since PowerPoint is not good on the Web, no matter what Billy Gates says…" [Will02]

[4] Allegorie zum an dieser Stelle weitaus treffenderen *spatial* des Englischen.

[5] Anm: „Verfügbar" bezieht sich in diesem Falle auch auf *Durchsuchbarkeit* wie etwa, im Gegensatz zu *Flash-* und *Java*-basierten Deployment-Lösungen, beim direkten Web-Export PowerPoints der Fall.

3 Internet-Präsentation

Zum Themenkomplex der Internet-Präsentationen, zumeist auf Basis des *World Wide Web* muss freilich zunächst einführend erwähnt werden, dass die zwei Begriffe „Internet" und „WWW", entgegen der landläufigen Meinung, nicht ein und denselben Sachverhalt, sondern in der Tat durchaus verschiedene technische Systeme repräsentieren: Während „das Internet" lediglich die Gesamtheit aller weltweit über das IP-Protokoll verbundenen Rechner darstellt,[1] bildet das WWW schließlich eine konkrete, hypertext-orientierte Anwendung, das auf dieser Grundlage aufsetzt.[2] Die Tatsache, dass das World Wide Web aufgrund seines selbst für dessen Entwickler überraschenden Erfolges den heute bekanntesten und am weitesten verbreiteten Internet-Dienst bildet, sollte jedoch nicht darüber hinwegtäuschen, dass „dies nur ein Teil des Anwendungsgebietes" des Internet darstellt [Rieh01:23]. In der Tat ist es aus struktureller Sicht [s.3.2.1] eher „unglücklicher Zufall", dass die WWW-Suite heute weit über den Kontext der reinen Hypertext-basierten Informationsdarstellung hinaus quasi als Standard-Benutzerschnittstelle für jegliche Art von Anwendung „missbraucht" [Muen98] wird. Dabei hätte sich neben den relativ spezialisierten, weiteren Diensten des Internet (FTP, eMail, Telnet…) durchaus eine weitere, unter Umständen eher auf multimediale Präsentation oder grafische Darstellung gemünzte Applikation wie etwa *HyperCard* [vgl. Niel99][3] als allgemein genutzter Standard etablieren können – wenn denn nur, wie auch Jakob Nielsen bedauernd anmerkt,[4] die geschichtliche Entwicklung und „die betriebswirtschaftlichen Vorraussetzungen andere Gewesen wären".[5]

Da allerdings aufgrund der heute absolut unantastbaren Stellung des WWW eine (technisch auf Basis des Internet freilich durchaus mögliche), „von Grund auf neu" konzipierte Anwendung nur wenig Chancen besäße, sich gegen die universell etablierte, vielschichtige WWW-Suite durchzusetzen, muss sich eine Analyse hinsichtlich multimedialer Präsentationsmöglichkeiten „im Internet" leider hauptsächlich auf die Gestaltungs- und Implementierungsmöglichkeiten im Rahmen des World Wide Web konzentrieren. Darüber hinaus bleiben Ansätze, die hingegen von der durch die WWW-Protokollsuite bereitgestellten Standards abweichen (wie etwa *Streaming*-Anwendungen [s.3.5.1], die ebenfalls auf Basis eigener Protokolle und Darstellungsmechanismen arbeiten), natürlich weiterhin von Interesse, sind jedoch bislang erheblich weniger verbreitet [vgl. Bult01] und im Vergleich zu WWW-basierten Präsentationsformen daher im Rahmen dieser Diplomarbeit von weitaus geringerer Relevanz.

3.1 WWW – Prinzipien und Historie

Das *World Wide Web* selber stellt derweil, wie bereits erwähnt, gleich eine ganze Suite von Protokollen, Formaten und Konventionen zur Verfügung, die vorrangig der Erstellung, Übertragung und des „Retrievals"[6] logisch strukturierter, zunächst rein textbasierter Informationen dienen sollten [vgl. Rieh01:30]. Die hieraus ersichtliche Struktur, Aufgabe und Motivation der bereitgestellten Komponenten, begründet sich freilich aus dem primär akademischen Entstehungshintergrund der WWW-Suite: Ursprüngliches Ziel der WWW-Entwickler am CERN-Forschungszentrum[7] in Genf war es, wissenschaftliche Dokumente, Papers und Dissertationen zu strukturierten Textdokumenten zusammenzufassen und über ein Computernetzwerk ver-

[1] "The internet is the entirety of all computers which are networked (using various networking technologies) and employ the Internet protocol suite (IP) on to of their networking systems" [Wild99] p.18

[2] vgl. [Wild99] p.18

[3] s. hierzu auch die entsprechende Diskussion in [2.1]

[4] vgl. [Niel99]

[5] "If Apple had played its Cards better…", frei übersetzt aus [ibid]

[6] Hiermit sind sowohl Lokalisierung (Suchen und Auffinden) als auch das Betrachten der Dateien gemeint, nach Peter Daub: *Theorie und Praxis des Information Retrievals*. Bibl. FH Furtwangen 2000

[7] CERN : Centre Européen pour la Recherche Nucléaire (Europäisches Kernforschungszentrum)

fügbar zu machen. Hierfür entwickelte CERN-Entwicklungsleiter Tim Berners-Lee das recht einfach gestrickte, „nicht sehr intelligente" [Blei96:11] Protokoll HTTP[1] zur Übertragung, die URL-Konvention[2] zur Lokalisierung sowie das HTML-Format zur Strukturierung und Beschreibung der Dokumente, wobei letzteres freilich die für unsere Zwecke[3] interessanteste Komponente der WWW-Suite darstellt.

HTML, die HyperText *Markup*-Sprache, lehnte derweil sich ihrer ursprünglichen Fassung [1.0: 1989] noch sehr eng an Vannevar Bushs Hypertext-Begriff der assoziativen Informationsverknüpfung an [vgl. Bush45] und bedient sich als Implementierung dieser „semantischen Netzwerke" (Parsaye et al 1989)[4] der *Hyperlink*-Funktionalität zur zunächst unidirektionalen, jedoch mehrfach möglichen Verknüpfung von Dokumenten:

> *Ein Link ist ein Verweis auf eine andere Ressource, dem der Benutzer mittels „Aktivierung" folgen kann. Die so entstehende Struktur erinnert an ein komplexes Informationsnetz, daher die Bezeichnung Web.*
> [Schm99:16]

Grundlage der Beschreibungs-Syntax des Formates bildet der sehr mächtige Dokumenten-Standard SGML[5] [SGML86], der als so genannte „Metasprache" Vorschriften zur Definition beliebiger Auszeichnungssprachen bereitstellt [vgl. Duck01:15ff]. Konkret bedeutet dies, dass SGML *nicht*, wie von verschiedenen Autoren irrtümlich angenommen,[6] eine direkte Format-Implementierung, sondern lediglich Anweisungen und Konventionen zur Formulierung der eigentlichen Dokumentstrukturen bereitstellt. So ist auch das in [Born90] vorgestellte Format zur Erstellung strukturierter, wissenschaftlicher Dokumente, entgegen der irrtümlichen Ansicht des Autors,[7] keine Beschreibung „*des* SGML-Standards", sondern lediglich eine im Annex E.1 des ISO-Dokuments [SGML86] erläuterte, beispielhafte Anwendung auf Basis der SGML-Konvention dar, in der Komponenten wissenschaftlicher Schriften wie etwa `<abstract>`, `<bibliog>` oder `<glossary>` in Form so genannter *Tags* logisch ausgezeichnet werden können. Diese *Elemente* können nun für jede beliebige Anwendung, so sieht es der SGML-Standard vor, in einer so genannten DTD, also einer Dokumenten-Typ-Definition, in Struktur und Syntax definiert und somit leicht auf ihre Gültigkeit (oder „Wohlgeformtheit") hin überprüft werden. Da eine detailliertere Beschreibung der SGML-Konvention sicherlich den Rahmen sprengen würde, sei an dieser Stelle zur genaueren Spezifikation dieses Standards auf [Duck01:41ff, Wild99:10ff] sowie (mit den bereits genannten Einschränkungen) [Born90:359ff.] verwiesen.

Als problematisch hinsichtlich eines möglichen Einsatzes des SGML-Frameworks im Internet bzw. als direkte Komponente des Word Wide Web ist jedoch die Komplexität von SGML zu beurteilen, die die Entwicklung von SGML-Anwendungen „teuer und kompliziert" machte[8] und bisher einer weiten Verbreitung von SGML entgegenstand [vgl. Bosa97]:

> *SGML erweist sich [daher] wegen seiner hohen Komplexität als im Internet nur begrenzt einsetzbar*
> [Star01:25]

Daher orientierte sich Berners-Lee bei der Entwicklung des HTML-Formates zwar lose an der soeben beschriebenen Annex E.1-Dokumentstruktur, entschloss sich jedoch zu einer drastischen Reduzierung der durch SGML und auch der letztgenannten Konvention bereitgestellten Komplexität – das Ergebnis dieser

---

[1] HTTP: HyperText Transfer Protocol

[2] URL: Uniform Resource Locator. Syntax: protokoll://subdomain(s).domainname.topleveldomain/verzeichnis(se)/datei.erweiterung?parameter

[3] s. Kap. 1

[4] Aus: Kamran Parsaye, Mark Chignell, Setrag Khoshafian und Harry Wong: *Intelligent Databases*. John Wiley & Sons, Toronto 1989

[5] Abk.: Standard Generalized Markup Language

[6] s. hierzu etwa [Born90] pp.360ff

[7] vgl. ebenda

[8] vgl. [Star01] p.25

Bemühungen bildete schließlich die 1990 veröffentlichte[1] „HTML 1.0"-DTD. Im Gegensatz zur „Mutter-Metasprache" stellte diese äußerst vereinfachte Dokumentstruktur ein radikal netzorientiertes Hypertext-System dar, das mit einer sehr übersichtlichen Tag-Auswahl selbst die Einbindung vektorbasierter „Artworks", wie noch in SGML möglich,[2] nicht zuließ und zunächst auch keinerlei darüber hinausgehende, gestalterische oder gar Präsentationsorientierte Funktionalität vorsah.

Gerade aufgrund dieser verblüffenden Schlichtheit setzte sich die von den CERN-Forschern lediglich für interne Zwecke entwickelte WWW-Suite zunächst im Rahmen des wissenschaftlichen Anwendungsfelds [vgl. Kali00] sehr schnell durch, um dann schließlich nach Veröffentlichung des ersten grafischen Browsers *Mosaic* [vgl. Andr93] im Januar 1993 regelrecht zu „explodieren" [Rieh01:30]. Dennoch merkt Kurt Cagle[3] „ohne den Verdienst Berners-Lee's schmälern zu wollen", aufgrund der parallel zu den CERN-Bemühungen ebenfalls voranschreitenden Arbeiten weiterer Internet-Initiativen an, dass „früher oder später" auch ohne die Veröffentlichung des HTML-Frameworks von Berners-Lee ein „SGML-mäßiger"[4] Internet-Standard zur Übertragung und Beschreibung von Hypertext-Dokumenten im gleichen Zeitraum entstanden wäre [vgl. Cagl01:9].

3.2 HTML: Das missbrauchte Format

Nach genauerer Betrachtung des ursprünglichen HTML-Formats bleibt an dieser Stelle jedoch lediglich festzuhalten, dass es sich sowohl hinsichtlich des Grundprinzips, der Syntax und Semantik der Sprache sowie aufgrund fehlender Gestaltungsmöglichkeiten bei der 1.0-Version HTMLs um ein rein *textbasiertes* Hypertextformat zur Veröffentlichung logisch strukturierter, wissenschaftlicher Dokumente handelt. Der bereits erwähnte, selbst für die Web-Entwickler überraschende Erfolg der gesamten WWW-Suite und insbesondere des HTML-Formates machte die Markup-Sprache jedoch über die Grenzen der rein akademischen Anwendung hinaus auch für kommerzielle Ersteller- und Nutzergruppen interessant [vgl. Kali00].[5] Speziell die auf Basis von HTTP sehr einfach zu realisierende, weltumspannende, und aufgrund der alsbald rasch steigenden Internet-Anbindungszahlen auch quantitativ beeindruckende Verbreitungsmöglichkeit Web-basierter Dokumente förderten die enormen Wachstumszahlen des WWW auch im unternehmerischen Bereich. Lediglich die aufgrund der restriktiven, schlichten (da primär auf wissenschaftliche Dokumente optimierten) HTML-Syntax fehlenden Gestaltungsmöglichkeiten des Hypertext-Formates waren den nun dominierenden, kommerziellen Parteien ein Dorn im Auge: So stellte das WWW bislang in der Tat kein echtes Hyper*media*-System im Sinne von [Stei99:11] dar, da grafische Elemente nicht oder nur rudimentär eingebunden werden konnten. Auch handelte es sich bei dem von Berners-Lee konzipierten World Wide Web um kein *Präsentations-* sondern ausschließlich ein *Informations-Medium* zum dezentralisierten, nicht-linearen und assoziativ verknüpften Angebot und „Retrieval" textbasierter Informationen:

> *Browsers are specifically designed to retrieve and display text […] The whole purpose of the Web is to create pages that any machine can view, with any software, thanks to the universality of HTML…*
>
> [Balo00:2]

Um "genau zu sein", so Robert Caillau, neben Berners-Lee Urvater des Web am CERN [Mint03], ist freilich auch die „Präsentations"-Komponente elementarer Bestandteil auch des ursprünglichen WWW und seiner Browser-Umgebungen – jedoch in anderer Hinsicht: gemäß der im 1. Kapitel definierten Kategorie primär

---

[1] vgl. [Stei99] p.1

[2] In SGML in Form von *CGM*: siehe [4.3.1] sowie [Grae98, HeGe94, LiPl97]

[3] vgl. [Cagl01]

[4] "It is entirely possible that had Berners-Lee not written HTML, an SGMLish language would have emerged soon thereafter because the research community had been working with SGML as data structures even around that time." [ibid] p.9

[5] "[…] The Web began its exponential growth – and, eventually, commercial growth…" [Kali00]

auf die *Darstellung von Informationen* gemünzter „Präsentation", so Caillau, stehe an dieser Stelle die Struktur der Information sowie nutzerbezogene Flexibilität im Vordergrund:

> *You have here a book on presentation. But it is presentation of information that should also remain structured, so that your content can be effectively used by others, while retaining the specific visual aspects you want to give it.*
>
> [Cail97]

3.2.1 Präsentation versus Information

Dennoch, oder vielmehr eben deswegen, orientierten sich weite Teile der kommerziellen Welt in Richtung der zunehmend populären WWW-Plattform, versuchten jedoch zugleich intensiv, die dem Web zugrunde liegende HTML-Konvention trotz deren (eben angesprochenen) mangelnden Präsentationseigenschaften in eben diese Richtung zu drängen. Obgleich sich zu Anfang das Gros der als Pioniere in das Web drängenden Unternehmen den strikten Hyptertext-Konventionen des WWW zunächst unterwarf und bemüht war, Firmeninformationen gemäß des HTML-Prinzips *strukturiert* zu hinterlegen, forderten insbesondere Design-orientierte Firmenvertreter vehement grundlegende Erweiterungen der HTML-Syntax. Die Wünsche der Unternehmen konzentrierten sich hierbei in erster Linie auf erweiterte gestalterische Möglichkeiten und die Web-Integration des sich zugleich im Offline-Bereich stark entwickelnden Multimedia-Zweigs[1] [s. 2.1 und 2.3]. Diesen Forderungen standen jedoch Bemühungen der so genannten „Strukturalisten" [Rieh01] gegenüber, welche eine „Aufweichung" des HTML-Standards strikt ablehnten und im Gegenzug unter Hinweis auf die Nutzerbezogene Informations-Orientierung des Formates für verbesserte Web-*Usability* sowie eine noch konsequentere Trennung von Inhalt und Struktur, als bereits im ursprünglichen HTML-Standard vorgesehen, eintraten.

In der Tat führten die „Strukturalisten" unter Federführung des Usability-Gurus und „klügsten Kopfes des Internet",[2] Jakob Nielsen [vgl. Niel95,99,00] durchaus überzeugende Argumente zu Felde, weswegen das als wissenschaftliche Beschreibungssprache konzipierte HTML-Format für Design- und Präsentationsbedürfnisse bereits im Grundsatz ungeeignet sei: So stehe bereits das dem World Wide Web zugrunde liegende Prinzip des auf multiplen Verknüpfungen beruhenden, vernetzten Hypertext-Prinzip dem Wunsch der „Design"-Vertreter nach primär linearer (ergo Verknüpfungs*loser*) Internet-Präsentation entgegen. In deren naturgemäßer Orientierung an dramaturgisch aufgebauter, sequentieller Argumentation verkennen die so genannten „*Presentationists*"[3] jedoch, so das Argument der „Struktur-Verfechter", die Haupt-Eigenschaft der assoziativen Interaktivität und der damit verbundenen, Nutzerorientierten Verhaltensänderung (der Nutzer sucht, findet und liest gezielt einzelne Informationen, statt einer linearen Informationskette, wie bei Buch oder Fernsehen, passiv ausgesetzt zu sein), die durch das Hypertext-Prinzip des WWW bedingt wird, in „eklatanter Weise". Eine aufschlussreicher Kommentar zu diesem Konflikt zwischen direkter Übertragung der gewohnten, linearen Dramaturgie auf verlinkte Hypertext-Medien und dem Vorwurf der „Strukturalisten", die besonderen Eigenschaften und Grundprinzipien des WWW damit zu ignorieren, findet sich – diesmal interessanterweise aus der „Designer-Perspektive" – in den medien-theoretischen Ausführungen Neville Brodys:[4]

> *The main drawback […] is the lack of risk being taken in addressing a new form and in developing that form as a new language that reflects a change in human behaviour. When we "use" an electronic publication, our very activity is altered. When the television was invented it was designed to appear as a radio with a picture, because that is what we were used to. The formatting of early television programmes was based on the "staged" appearance of theatre, because we felt safe within the limits of something we knew.*

---

[1] Insbesondere Audio/Video und „interaktive" Multimedia-Applikationen wie Director, PowerPoint etc.

[2] vgl. [Depe02] p.22

[3] Terminologie aus dem „Structuralist vs. Presentationist Discourse" [vgl. Zieg99]

[4] vgl. [Brod96]

> *Most CD-ROM titles todayare little more than books read using VCR controls… This demonstrates that when faced with any new technology, we revert to an aping of a previous technology because we feel threatened by it. Here we see the equivalent of a reversal trend in architecture that of "Veneer Vernacular", or the need to dress up in a modernistic or futuristic approach behind a facade of familiarity.*
>
> [Brod96]

Ohne jedoch zu weit in den Diskurs der „Strukturalisten" und „*Presentationists*" um die vorgeblich „Web-gerechte" Form der Informationsdarstellung bzw. –Präsentation eintauchen zu wollen, bleibt an dieser Stelle lediglich ein offensichtlich kontrovers diskutierter Konflikt zwischen herkömmlichem Präsentations-Begriff und dem interaktiven, Informations-orientierten Hypertext-Prinzip des WWW zu vermerken [vgl. hierzu Rieh01].

3.2.2 Lost in Hyperspace

Weiteres Kennzeichen dieser Problematik ist neben der Frage nach der *Berechtigung* Präsentationsorientierter Inhalte im WWW freilich auch die theoretische *Unmöglichkeit*, Web-basierte Anwendungen starr linear zu dramaturgisieren: Aufgrund der sehr mächtigen Kontroll- und Navigationsmöglichkeiten, der auf den individuellen *Nutzen* fixierten Informations-Orientierung sowie des im Internet sehr eng gesteckten Lese-Zeitrahmens des Internet-Nutzers von nur einigen Sekunden [vgl. Niel01] kann im World Wide Web *nicht* mehr von einer streng sequentiellen Lesart der Inhalte ausgegangen werden:

> *Es gibt nicht nur eine Reihenfolge des Lesens, der Leser entscheidet über den Lesepfad.* [Stei99:700]

Die dadurch äußerst reduzierten dramaturgischen Möglichkeiten, die sich den potentiellen Erstellern und Distributoren Web-basierter Präsentationen im Rahmen des HTML-Konzepts bieten, werfen daher die berechtigte Frage auf, inwiefern das WWW „in seiner ursprünglichen, reinen Form" [Voge96] für konventionelle Präsentations-Anforderungen überhaupt geeignet ist, da sich weder eine durchgängige Argumentationsfolge noch emotionale (i.e. grafische) Elemente im WWW wirkungsvoll realisieren lassen. Darüber hinaus, so gibt Ralf Steinmetz zu bedenken, hat „diese Technik [noch weitere] kritisch zu bewertende Eigenschaften",[1] wonach beim Lesen eines Web-Dokuments etwa „Überblick und Kontext" verloren gingen: „Der bekannteste Effekt ist '*Getting Lost in Hyperspace*'" [Stei99:719]: „Während ich den Hypertext las, wusste ich oft nicht, wie ich dahin zurückkehren sollte, wo ich gerade herkam" [NiLy90]. Da das Web als so genanntes „Spaghetti-Buch" [Bole98] somit nur wenig Orientierungsmöglichkeiten oder vertraute Strukturen anbietet, erscheint auch die Umsetzung eines ganzheitlichen, inhaltlichen Aufbaus möglicher Internet-Präsentationen äußerst schwierig realisierbar. Aus diesem Grund sind im Übrigen auch unter anderem bi- oder gar multidirektionale Link-Strukturen wie etwa im *Intermedia*-Systems implementiert,[2] [vgl. Niel95:51ff] aufgrund ihrer hohen, Nutzerbezogenen Komplexität zum Scheitern verurteilt:

> *[Solche Anwendungen] fristen im Web ein trauriges Dasein, da die Nutzer nicht bereit sind, „hochentwickelte" Bedienungsphilosophien zu erlernen.*
>
> [Hent98:19]

Auch verschiedene,[3] als „Lösungsvorschläge" für diese Problematik angebotenen Methoden [vgl. Bole98] vermögen diese dem Präsentationskonzept entgegenstehenden Schwierigkeiten nicht zu kompensieren. Als theoretisch interessantestes Modell erscheint unter diesem Aspekt einzig das Prinzip der „Directed Paths" bzw. „Guided Tours" [vgl. Trig88, Zell89], welches einen linearen, optionalen Pfad (von dem der Nutzer bei Interesse wiederum interaktiv abzweigen kann) durch das nur schwierig dramaturgisch zu kontrollieren-

---

[1] vgl. [Stei99] p.719

[2] Das Hypermedia-System Intermedia wurde ab 1985 an der Brown University, am Institute for Research in Information Scholarship (IRIS) entwickelt. Konzepte: Bidirektionale Links, Speicherung von Bemerkungen.

[3] „Back-Funktion, History-Mechanismen, Annotationen…" [Bole98]

de Hypertext-Gewirr anbietet. Trotz Einführungsversuche verschiedener, derartiger „Web-Guides" Mitte der 1990er Jahre konnte sich auch dieser Ansatz jedoch in der Praxis nicht durchsetzen, weil selbst diese Art der angeblichen „Bevormundung" durch die Internet-Nutzer offensichtlich nicht akzeptiert wurde. Dies liegt meiner Einschätzung nach in dem hohen Grad der Interaktivität und der Informations-/Nutzen-Orientierung „echter" Hypertext-basierter Internetanwendungen begründet, die einen Mehrwert derartiger „pseudo-interaktiver", primär linearer *Guide*-Systeme nur schwer erkennen lassen.

3.2.3 „Then we descended into the Dark Ages…" [Cail97]

Dennoch hielten die soeben angesprochenen Schwierigkeiten bzw. die teils mangelnde Eignung des HTML-Formates selbst hinsichtlich grafischer Gestaltung und dramaturgischer Präsentation insbesondere kommerzielle Anwenderkreise nicht davon ab, die anfangs recht bescheidenen[1] kreativen Möglichkeiten des zum Quasi-Standard avancierten Formats sowohl intensiv zu nutzen als auch deren Funktionalität zunehmend zu „missbrauchen" [Muen98]. Ursprünglich zur rein logischen Auszeichnung bzw. rudimentären Textformatierung vorgesehene HTML-Elemente wurden so zusehends gezielt gestalterisch eingesetzt – „ein Fehler, den heute immer noch sehr viele Anwender beispielsweise im Gebrauch von Textverarbeitungen begehen." [NeWi00]

Statt etwa die speziell hierfür vorgesehenen Einrückungs-, Aufzählungs-, oder Absatzelemente HTMLS zu verwenden, wurde es insbesondere bei wenig professionellen HTML-„Designern" alsbald zur verbreiteten Unsitte, sich zur Vergrößerung des Textabstands des nicht-umbrechenden Leerzeichens (` `) zu bedienen – einer Erweiterung, dessen Einführung die Autoren der HTML 1.1-Spezifikation alsbald bereuten. War nämlich eine mehrfache Wiederholung des einfachen Leerzeichens (‚·') noch in der ursprünglichen Sprachfassung nicht möglich, so verleitete das als separate, so genannte „Entity" auszuzeichnende NBSP-Element[2] zu häufigem Missbrauch, da es eine (scheinbare) Positionierungs-Funktionalität alternativ zur automatisch dargestellten, logischen Auszeichnung anbot.

Als Paradebeispiel des „vergewaltigten HTML-Elements" kann jedoch eindeutig das ebenfalls erst in späteren HTML-Versionen eingeführte *Tabellen*-Objekt bezeichnet werden, welches zunächst – freilich noch zugunsten der Erstellung wissenschaftlicher Dokumente – einzig und allein die Darstellung tabellarischer Daten ermöglichen sollte. Zum großen Entsetzen der „Strukturalisten" [Rieh01:55] dient bis zum heutigen Tage jedoch nur ein Bruchteil der im Rahmen von HTML eingesetzten `table`-Tags tatsächlich diesem Bestimmungszweck: Die überwältigende Mehrheit aller im Web vorkommenden, zumeist unsichtbaren[3] Tabellen müssen hingegen zum „Absoluten Positionieren von Elementen" [vgl. Baum98:25,65] herhalten – ein „wahrer Albtraum", nicht nur aus struktureller Sicht:

> *Although apparently suited to layout on the surface, under the hood it becomes clear that tables do a pretty lousy job of page construction. Among their shortcomings is the implied bias of the code towards presentation rather than structure, the necessity to nest tables in order to achieve the most basic of layouts, and enough redundant bandwidth-hogging tags to feed a large family of tag eating monsters for literally a month.*
>
> [McLe02]

3.2.3.1 Erste Erweiterungen

> *Although the web began as a medium to exchange physics research papers, it seems naive to expect it to remain predominately text-based.*
>
> [Clon00]

---

[1] s.3.2.1

[2] Non Breaking Space (Nicht-umbrechendes Leerzeichen), etwa „Wort1····Wort2"

[3] Hiermit ist eigentlich der Rand (border=0) der Tabelle gemeint.

Selbst der soeben angesprochene "Missbrauch"[1] der ursprünglich nicht hierfür vorgesehenen HTML-Funktionalität erschien den zunehmend Design-orientierten Nutzern der WWW-Suite jedoch aus gestalterischer Perspektive schließlich „völlig unzureichend", weswegen alsbald auf weitreichendere Änderungen und Erweiterungen der HTML-Sprache gedrängt wurde. Die ersten, die diesen Wünschen „nur zu bereitwillig nachkamen" [Star01:1], waren jedoch nicht die Autoren der HTML-Spezifikation selbst, da diese freilich an der semantischen „Sauberkeit" und der ursprünglichen Absicht des Formates zur logischen Strukturierung Web-basierter Dokumente (zunächst) festhielten, sondern der Entwickler des grafischen Browserprogramms *Mosaic*, Marc Andreessen, welcher das zunächst rein universitäre NCSA-Projekt[2] [Andr93] mit seiner daraufhin verselbstständigten Firma *Netscape* zu kommerziellem Erfolg führte. Mit der gezielten Einführung nicht-konformer Erweiterungen wie etwa „visueller Tags", der so genannten *Frames* [vgl. Niel96] und der (vorerst proprietären) Skriptsprache *JavaScript* wich Andreessen zunehmend vom ursprünglichen HTML-Standard ab und preschte den CERN-Entwicklern durch die erweiterte Funktionalität der eigenen Browsersoftware „Netscape Communicator" weit voraus.

Da letztere jedoch bereits aufgrund ihrer großflächigen Verbreitung den „Quasi-Standard" der WWW-Clienttechnologie darstellte, richteten sich die Autoren Web-basierter Anwendungen im Verlaufe dieser Entwicklung zunehmend nach den erweiterten Fähigkeiten und Darstellungseigenschaften des *Browsers* statt der eigentlichen, „strikten" HTML-Spezifikation. Dies führte insbesondere zur verstärkten Anwendung der speziell durch Netscape forcierten, *visuellen* HTML-Tags, die, etwa mittels `<font>` oder `<b>` (Fettschrift) eine verbesserte Kontrolle über das eigentliche Aussehen der Web-Dokumente ermöglichte. Schwerwiegender Nachteil dieser Auszeichnungs-Methode und daher ein Dorn im Auge der „Strukturalisten" war und ist jedoch der hiermit einhergehende Verlust der Semantik, „eine weitere Aufweichung der Struktur sowie eine Aufblähung der Dokumente." [Rieh01:55]

> *Dazu kam, dass Web-Autoren [nur] noch diese proprietären Standards […] und Formatierungen im strukturierten Text verwendeten, anstatt die [durch die HTML-Spezifikation] vorgeschlagene Trennung vorzunehmen (z.B . fett oder kursiv für Zitate, anstelle Erstellung des HTML-eigenen Formates für „Zitat")*[3]
>
> [NeWi00]

Als schließlich Microsoft ebenfalls in den zunehmend lukrativen Browser-Markt in Gestalt des eigenen *Internet Explorer* einstieg, schien die ursprünglich von Tim Berners-Lee erhoffte strukturelle Trennung zwischen Inhalt und Darstellung endgültig „zum Scheitern verurteilt" [NeWi00]. Der daraufhin entfesselte, so genannte „Browser-Krieg" [Wild99:14] führte nicht nur zu „Wildwuchs" und der Einführung jeweils eigener, proprietärer Standards,[4] sondern in der Reaktion hierauf ebenso zur Gründung des *World Wide Web-Konsortiums* (W3C) durch WWW-Erfinder Berners-Lee, welches diese unkontrollierbare Entwicklung mithilfe universeller Standards einzudämmen suchte. Bedauerlicherweise hinkten auch die durch das W3C verabschiedeten HTML-*Recommendations* der rasanten Entwicklung und sogar dem Akzeptanzverhalten der Autoren und Web-Anwender weit hinterher und vermochte auch die Bewegung der HTML-Sprache vom reinen Hypertext-Format hin zu einem von den kommerziellen Web-Autoren erwünschten *Präsentations-Medium* in letzter Konsequenz nicht aufzuhalten – im Gegenteil: Die „eingeschmuggelten" Frame-, JavaScript- und dHTML-Funktionen sorgten nicht nur für Inkompatibilität [vgl. ChNg02] und Verwirrung seitens der Web-Entwickler und –Nutzer, sondern verwischten auch zusehends die Grenze zwischen semantischem In-

[1] vgl. [Muen98]
[2] NCSA-Mosaic: National Center for Supercomputing Applications at the University of Illinois
[3] Anm: Für Zitate stellt der HTML-Standard etwa die Tags cite sowie q (mit Quellenangabe) vor.
[4] vgl. [NeWi00]

halt der Webseiten und ihrer grafischen Darstellung, welche insbesondere durch den Missbrauch der HTML-Tabellenelemente[1] und die Einführung von dHTML zunehmend verschmolzen.

Aufgrund dessen erscheint es in diesen Zusammenhang nur wenig verwunderlich, dass insbesondere selbsternannte „Usability-Experten“[2] um Jakob Nielsen diese Entwicklung aufs Schärfste kritisierten[3] und sich mit den vorgeblich fehlgeleiteten „*Presentationists*“ rüde Wortgefechte lieferten. Da diese kontroverse Diskussion jedoch bereits etwa in [Rieh01:51ff] oder [Clon00] sehr ausführlich dargelegt wird, sei an dieser Stelle lediglich festgehalten, dass im Verlauf dieses Diskurses nicht nur der Wunsch der so genannten „Designer“[4] nach weitgehenderen, visuellen Gestaltungsmöglichkeiten auf Basis von HTML, sondern durch die vehemente Kritik der „Strukturalisten“ ebenso die durchaus problematische *Usability* und Inkompatibilität der damit einhergehenden „fragwürdigen“[5] Erweiterungen deutlich wird.

Die Tatsache, dass jedoch selbst aktuelle Versionen der im Übrigen auch heute noch teils inkompatiblen[6] dHTML-Komponenten, *Frame*-Sets und JavaScripts nur rudimentäre oder aber relativ umständlich zu realisierende Layout- und multimediale Funktionalität bietet, legt an dieser Stelle zunächst den Schluss nahe, dass das HTML-basierte Web auch aufgrund der in [3.2.1] gewonnenen Erkenntnisse nur *in sehr begrenztem Umfang* als ein für Präsentationszwecke „geeignetes“ Medium bezeichnet werden kann.

3.2.3.2 Der Lösungsversuch: Cascading Style Sheets

Das Hauptbestreben des World Wide Web Konsortiums, die durch verschiedene, in [3.2.3.1] bereits angesprochenen „Erweiterungen“ der HTML-Syntax bedrohte Trennung der Web-Inhalte von ihrer Darstellung wiederherzustellen, manifestierte sich gegen Ende der 90er Jahre nach längerem, scheinbar unkontrolliertem „Wildwuchs“ durch den „Browser-Krieg“ [vgl. NeWi00, Rieh01:51ff] endlich neben zuvor vergeblichen Standardisierungsversuchen der HTML-Spezifikation[7] in der Einführung der so genannten Cascading Style Sheet-Spezifikation [CSS96]. Dieses Regelwerk erlaubt anhand der eigenen, von der SGML-Konvention abweichenden Style Sheet-Syntax „endlich“ [Cail97]# die Definition rein visueller Elementeigenschaften, wie etwa Typografie- oder Farbattribute:

```
h1 { color:dark-red; font-size:36pt; font-family:RotisSansSerif; }
```

Listing 3.2.3.2.1.1: Definition von Farbe und Typografie des Haupt-Überschriftentyps H1 *in CSS*

Um die somit angestrebte Trennung dieser Eigenschaften von den semantischen HTML-Inhalten zu gewährleisten, sieht die CSS-Spezifikation statt der direkten Adressierung visueller Tag-Attribute nun die einfache Referenzierung ex- oder interner Style-Sheet-Definitionen vor, während die in der HTML-Datei verwendeten *Tags* lediglich inhaltliche Zusammenhänge direkt repräsentieren sollten:

[1] s. 3.2.1 (Anfang)
[2] vgl. [Clon00]
[3] s. hierzu etwa [Niel96b,99,00]
[4] s. hierzu [Rieh01] p.51
[5] vgl. hierzu: Marius Lübbe, Jonas Stiller, Mathias Schäfer, Lars Janssen und Maximilian Fuchs: “‚Popups’ mit JavaScript.” URL: http://home.t-online.de/home/dj5nu/js-popup.html [12.2.03]
[6] vgl. [Zeld99]
[7] vgl. [Rieh01] pp.30ff: „Der folgende offizielle Standard HTML 2.0 wurde allgemein als Enttäuschung empfunden da gerade Netscape in seiner Entwicklung schon viel weiter war…“ [Rieh01:30]

```
<html>
   <head> ...
      <style type="text/css"
       h1 { // siehe oben ... }
      </style>
   </head>
   <body>
      <h1>Grosse &Uuml;berschrift</h1>
   </body>
</html>
```

Listing 3.2.3.2.1.2: Anwendung von CSS in einre HTML-Seite

Abb. 3.2.3.2.1: Darstellung im Browser.

Obgleich diese separate Architektur zumindest in der Theorie eine Auftrennung von Darstellung und Inhalt erlaubte, kann jedoch die zu wenig konsequente und recht laxe Umsetzung des Style-Sheet-Ansatzes [vgl. Zeln98] als Hauptursache für das letztendliche Scheitern dieser Bemühungen bewertet werden: Während den „Strukturalisten" Umfang und Funktionalität des Frameworks noch lange nicht weit genug gingen [vgl. MaMu99], brachte schließlich die inkonsequente CSS-Implementierung der Browser-Hersteller die hehren Absichten der CSS-Entwickler zu Fall. Da sowohl Microsoft als auch Netscape die bereits vorab erschienene Style-Sheet-Spezifikation sowohl sehr unterschiedlich[1] als auch teils nur lückenhaft[2] umsetzten, erlaubten die Autoren des Web-Konsortiums im Rahmen des noch im Folgejahr veröffentlichten HTML 4.0-Standards [HTML97] „für die Übergangszeit" eine so genannte „Transitional"-Form der Hypertext-Sprache, der zugunsten Berücksichtigung älterer Browser und mangelhafter CSS-Implementierungen eine parallele Verwendung Style-Sheet-konformer wie auch direkter, visueller HTML-Elemente erlaubte. Ebendieses Vorgehen erwies sich jedoch als folgereicher, strategischer Fehler, da diese Möglichkeit durch die Web-Designer bei weitem nicht nur „übergangsweise" (wie von den W3C-Vertretern vorgesehen), sondern durchaus langfristig genutzt wurde – und bis heute wird:

> *Für Rückwärtskompatibilität sollten Sie den Font-Marker mit CSS zusammen verwenden.* [Baum98:114]

Statt der erhofften, vollständigen Trennung von Inhalt und Darstellung trägt jedoch die häufig parallele Anwendung von CSS-Syntax und „rückwärtskompatiblem", visuellem HTML zu einem recht unüberschaubaren Code-Gewirr bei, das einen Großteil der heute im Web auffindbaren HTML-Seiten mit einer aus inhaltlich-struktureller Hinsicht relativ unbrauchbaren Architektur hinterlässt [vgl. SuCi02]. So macht unter anderem der recht aufwendige *Parsing*-Algorithmus in [Duff03] deutlich, dass sich eine semantische Analyse HTML-codierter Webseiten ganz und gar nicht mehr, wie noch von Berners-Lee und Nielsen vorgesehen,[3] aus der Struktur einer HTML-Seite selbst ergibt, sondern inhaltliche und logische Zusammenhänge bestenfalls nur sehr umständlich und unter großem Aufwand vollständig zu rekonstruieren sind.

Eine mehrheitliche, konsequente Abtrennung inhaltlicher Struktur von deren grafischen Darstellung, bleibt also, da in der praktischen Umsetzung durch entgegensetzte Bemühungen Design-Orientierter Web-Entwickler[4] unterlaufen, bedauerlicherweise vorerst „Wunschtraum" der W3C-Visionäre.[5] Meiner Einschätzung nach hätte an dieser Stelle ein entschlosseneres Vorgehen des Web-Konsortiums mittels weitaus rigiderer Reglementierungen den vorsichtigen „Schlingerkurs" der Browser-Hersteller und Web-Entwickler verhindern und somit eine etwas übersichtlichere Situation herbeiführen können – die vorgeblich „pragma-

[1] „Wie bei der HTML-Spezifikation wurden jedoch auch bei den CSS von den Browserherstellern eigene Vorstellung verwirklicht. Die Darstellung bleibt daher von Browser zu Browser unterschiedlich, wie auch die Unterstützung einzelner Styles der Spezifikation. Zusätzlich wurden von den Browserherstellern eigene, proprietäre Attribute umgesetzt, die in der CSS-Spezifikation nicht enthalten sind." [Rieh01] p.56

[2] Insbesondere die Netscape-Implementierung wies zunächst deutliche Mängel auf, vgl. [SuCi02]

[3] vgl. hierzu [Bern99] sowie [Niel01]

[4] Diese Bezeichnung bezieht sich vorrangig auf Web-Designer, die eine maximale Kontrolle über das visuelle Erscheinungsbild einer Web-Anwendung erreichen wollen.

[5] vgl. [Rieh01:57]

tische“ Strategie aller Beteiligten jedoch hinterlässt bei Betrachtung der sehr inhomogenen, chaotischen Struktur der heutigen WWW-Landschaft einen eher zweifelhaften Eindruck. Interessanterweis fasst Flash-Hersteller Macromedia dieses „HTML-Dilemma“ [Star01:24] im Hinblick auf Web-Präsentationen am treffendsten zusammen:

> *HTML […] relies on obscure formatting techniques such as pixel positioning, frames, and table layout to emulate screen regions, or on style sheets that are plagued with cross-browser challenges.*
>
> [Alla02:5]

3.3 Web-Grafik

Trotz der vorangehend beschriebenen, radikalen Erweiterung der HTML-Syntax durch visuelle und gestalterische Komponenten bleibt daher festzustellen, dass sich in HTML selbst unter großzügiger Verwendung dieser nahezu grafischen Funktionalität keine ästhetisch opulenten Ergebnisse erzielen lassen. Dies liegt daran, dass in den bisherigen Betrachtungen lediglich HTML-spezifische Funktionen isoliert berücksichtigt wurden – das durch die *Mosaic*-Entwickler 1993 eingeführte[1] `<img>`-Tag erweitert den ursprünglichen Hypertext-Begriff jedoch um die Referenzierungsmöglichkeit externer Bilddateien:

```
<img src="../bilder/tanzbaer.xbm" width="368" height="383" alt="Tanzb&auml;r">
```

Listing 3.3.1: Einbindung einer Bild-Referenz mittels img.

Diese „kritische Schritt“ [Magl98][2] der Einbettung so genannter „Inlined Images“ (*Mosaic*-Terminologie)[3] veränderte die Grundlegende Architektur der HTML-Sprache in zweierlei Hinsicht: Obgleich bereits die grundlegende Hypertext-Eigenschaft durch Link-Referenzierung die Integrität HTML-basierter Seiten von der Existenz *externer* Quellen abhängig machte, gewährleistete an dieser Stelle noch die Tatsache, dass erst die Aktivierung des Links die eigentliche Anzeige der referenzierten Datei herbeiführte [vgl. Schm99:16] zunächst die Darstellungs-bezogene *Abgeschlossenheit* ursprünglicher HTML-Seiten. Da die „Inline“-Einbindung, d.h. die direkte Einbettung der referenzierten Grafik auf derselben Seite im Gegensatz zur zuvor praktizierten, so genannten „externen Bildreferenzierung“ [4] die Darstellung ebenjener HTML-Komponente jedoch unmittelbar beeinflusst, kann an dieser Stelle bereits von einer *Fragmentierung* der HTML-Architektur gesprochen werden, da die entsprechenden Dateien (genauer: deren Darstellungseigenschaften) nicht mehr in sich *abgeschlossen*, sondern stets von der Existenz weiterer (u.U. externer) Grafikdateien abhängig sind [vgl. Morr01:41].

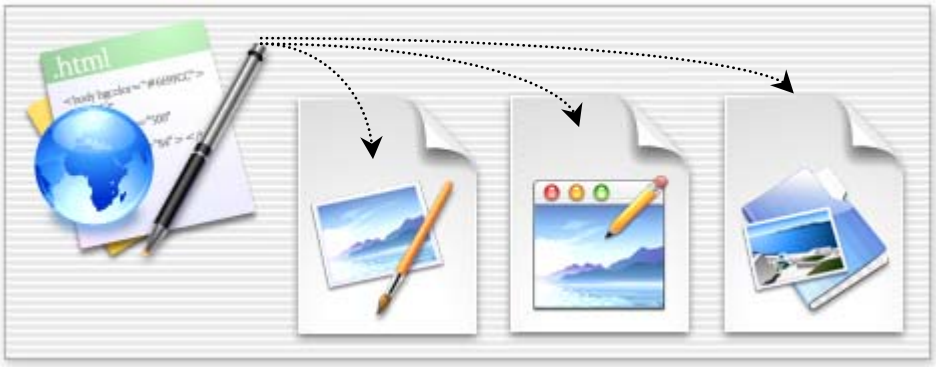


Abb. 3.3.1: Referenzierung mehrerer(Bild)- Dateien aus einer HTML-Datei heraus.

[1] s. hierzu Marc Andreessens “IMG Tag Proposal” im WWW-Talk-Forum vom 25. Februar 1993 [mittlerweile offline]

[2] “Mosaic’s introduction of inline images (the “img” tag) is widely considered to be a critical step in the evolution of the World Wide Web…” [Magl98]

[3] Aus: „Inlined Images for NCSA Mosaic”, *Mosaic*-Handbuch des National Center for Supercomputing Applications, Urbana-Champaign IL 1994

[4] s. hierzu auch [Morr01] p. 41

Der zweite, bemerkenswerte Effekt dieser Bildreferenzierung ist der Übergang des bislang als rein textbasiertes Hypertextmedium betrachteten Formates zu einem echten Hyper*media*-Format, in dem sowohl textliche als auch bildhafte Komponenten auftreten können: „Sind auch multimediale Daten (z.B. Bild[er]…) eingebunden, spricht man von Hypermedia." [Rieh01:23]. Es sei jedoch darauf hingewiesen, dass selbst diese Definition umstritten ist: So merkt etwa Ralf Steinmetz an, dass „nicht jede beliebige Kombination von Medien"[1] die Verwendung des Begriffs „Multimedia" rechtfertige:

> *Von Multimedia in unserem Sinn sollte man […] erst reden, wenn sowohl diskrete als auch kontinuierliche Medien betrachtet werden.*
>
> [Stei99:12]

Da es sich bei der Einbindung zunächst freilich noch statischer Bilddaten somit nicht um ein zeitkontinuierliches Medium handelt,[2] kann an dieser Stelle lediglich von einem Hyper-Medium „im weiteren Sinne" die Rede sein. Allein dies rückt das HTML-Format jedoch bereits, zumindest tendenziell, in die Richtung eines potentiellen Präsentations-Formates, da nun über die relativ begrenzten, textlichen Gestaltungsmittel der Hyptertext-Sprache hinaus nun auch weitgehende grafische Design-Möglichkeiten offen stehen: Im Gegensatz zur zunächst rein logisch strukturierten HTML-Syntax sind den Inhalten des eingebundenen Bildmaterials aus gestalterischer Hinsicht freilich keinerlei Grenzen gesetzt. Einschränkend sei hier allerdings bemerkt, dass sich der ursprüngliche Zweck der Grafik-Einbettung zunächst ausschließlich auf Illustrationszwecke wissenschaftlicher Dokumente beschränkte. Daher stellte auch der Schritt von der indirekten Referenzierung der Abbildungen zur sofortigen Darstellung [vgl. Morr01:41] zumindest in den Augen der WWW-Entwickler zunächst eine eher theoretische Unterscheidung dar. Web-„Erfinder" Tim Berners-Lee spielt in einem direkten Kommentar auf Marc Andreessens Image-Tag-Vorschlag jedoch bereits auf den bislang im Rahmen der Hypertext-Diskussion vernachlässigten Präsentations-Aspekt an:

> *Often, the reader (rather than the author) might want to choose which figures are expanded inline rather than put in a separate window, and also that the reader (rather than the author) might want to choose whether related things are PRESENTed automatically or not…*
>
> (Tim Berners-Lee)[3]

Die Tatsache, dass sich Andreessen, entgegen Berners-Lees Vorschlag, jedoch schließlich für die stets unangefragte, automatische Anzeige direkt referenzierter Bilder im Rahmen der *Mosaic*-Darstellung entschied,[4] verlagerte den Aspekt der Kontrolle über Darstellung und Präsentation der WWW-Dokumente erneut weg vom Nutzer der Web-Anwendung und hin zum Browserprogramm bzw. dem Web-Dokument selber. Dieser Umstand verhalf dem HTML-Format schließlich zu seiner letztendlich ja unfreiwillig angestrebten Rolle als Präsentations-Format, da, ähnlich wie bereits beim gezielten „Missbrauch" visueller HTML-Tags als grafische Gestaltungsmittel [s.3.2.3], das ursprünglich lediglich „wissenschaftlich-illustrative"[5] *Image*-Tag zur ausschließlich visuellen Gestaltung HTML-basierter Websites verwendet wurde.

Da aufgrund dieser Funktionalität auch grafisch durchaus opulente „Web-Designs"[6] möglich und „in erschreckendem Ausmaß" [vgl. Niel01] auch umgesetzt werden, rücken an dieser Stelle freilich Betrachtungen hinsichtlich Größe und Usability des eingebundenen Bildmaterials in den Vordergrund: Bildgewaltige Web-Dokumente sind nicht nur aus Erstellersicht erschwert zu verwalten und insbesondere durch behin-

---

[1] vgl. [Stei99] p.12

[2] Dies wäre etwa die Einbindung von Animationen [3.4] und Audio/Video-Material [3.5.1]

[3] Kommentar auf Marc Andreessens Vorschlag, WWW-Talk-Forum vom 26. Februar 1993 [mittlerweile offline]

[4] s. hierzu sein Kommentar im WWW-Talk-Forum vom selben Tag.

[5] Anm.: Im Sinne von Abbildungen (wie in dieser Diplomarbeit, siehe Abbildungsverzeichnis): Auch Andreessen und Berners-Lee sprechen an dieser Stelle zunächst ausschließlich von „Figures", was auf eben diesen Verwendungszweck hinweist.

[6] Anm: Aufgrund der Bild-Eigenschaft von HTML wäre, dank *Image*-Tag, auch diese Bezeichnung nun zutreffend

derte WWW-Benutzer lediglich eingeschränkt bedienbar,[1] „auch die Größe – und damit die Ladezeit – des betroffenen Dokuments insgesamt steigt damit“ in immensem Umfang an [Rieh01:55]. Aus diesem Grund erscheint speziell die Auswahl der im Rahmen des WWW zur Anwendung kommenden Bildformate interessant. Diese obliegt jedoch, im Gegensatz zur eigentlichen HTML-Spezifikation bei visuellen Tags und Einbettungs-Syntax, nicht, wie man meinen möchte, den Entwicklern der HTML-Sprache, sondern stets den Programmierern der jeweiligen Browser-Implementierungen:

> *Browsers should be afforded flexibility as to which image formats they support… If a browser cannot interpret a given format, it can do whatever it wants instead.*
>
> (Marc Andreessen)[2]

Umso überraschender erscheint unter dem Blickwinkel der soeben ausgeführten Überlegungen hinsichtlich Dateigröße und Ladezeit daher die Tatsache, dass *Mosaic*-Entwickler Andreessen an dieser Stelle zunächst völlig unkomprimierten Bildformaten den Vorzug gibt:

> *XBM and XPM are good [image formats] to support, for example…*[3]

Dieser Umstand lässt sich jedoch bei näherer Betrachtung der beiden Dateiformate etwas erhellen: So stellen die von Andreessen angesprochenen (und in der Tat bereits mit der *Mosaic* 0.10-Auslieferung unterstützten) Dateiformate *X-BitMap* (XBM) sowie *X-PixMap* (XPM) schlicht die seinerzeit in der Unix-Welt übliche Konvention zur Speicherung von Icons dar. Da aufgrund der starken Unix-Orientierung der akademischen Welt insbesondere das X-Motif-System in universitären Einrichtungen wie dem CERN und NCSA weite Verbreitung genoss, lag es daher (auch angesichts der relativ späten Windows-Portierung des Mosaic-Programms) durchaus nahe, sich dieses Quasi-Standards zur Bildkodierung zu bedienen. Aufgrund der Tatsache, dass, wie eingangs erwähnt, Bildreferenzen zunächst ohnehin rein wissenschaftlichen Abbildungen dienen sollten, fiel daher auch anfangs der ausschließlich schwarz-weiße, unkomprimierte Speicheralgorithmus des Formates nicht weiter ins Gewicht: Der Vorteil, dass sich XBM-Grafiken in reiner Textform und somit zumindest in der Theorie (wie auch durch die Mosaic-Entwickler angedacht) „inline“, also statt externer Referenz prinzipiell *direkt* im HTML-Code abbilden ließen, wog in den Augen Andreessens die strukturelle Schlichtheit des Formats bei weitem auf. Bei genauerer Betrachtung des XBM-Formats wird überdies deutlich, dass der Text-Code nicht lediglich einzelne Bilddaten enthält, sondern verblüffenderweise reinen *C*-Quelltext darstellt, was die Verwendung des Formates im Kontext weiterer (da primär C-basierter) Unix-Programme freilich begünstigt:

```
#define directory_width 20
#define directory_height 23
static char directory_bits[] = {
   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x01, 0x00, 0x04, 0x02, 0x00,
   0x02, 0x04, 0x00, 0x01, 0xf8, 0x01, 0x01, 0x00, 0x02, 0x01, 0x00, 0x04,
   0x01, 0x00, 0x0e, 0x01, 0x00, 0x0d, 0x01, 0x00, 0x0e, 0x01, 0x00, 0x0d,
   0x01, 0x00, 0x0e, 0x01, 0x00, 0x0d, 0x01, 0x00, 0x0e, 0x01, 0x00, 0x0d,
   0x01, 0x00, 0x0e, 0x55, 0x55, 0x0d, 0xaa, 0xaa, 0x0e, 0xfc, 0xff, 0x07,
   0xf8, 0xff, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
```

Listing und Abb. 3.3.2: Kodierung eines Directory-Icons in XBM, sowie dessen Darstellung

Außerhalb der Unix-basierten Welt, so merkt unter anderem [Mian99] an, ist dieses vielmehr Icon- als Bildorientierte Format[4] allerdings „äußerst selten anzutreffen“[5] – obschon es zumindest laut [Blei99] nicht nur das *erste* von Web-Browsern unterstützte, sondern überdies angeblic immer noch „das bevorzugte Bit-

[1] vgl. [Niel96a]

[2] s. hierzu Marc Andreessens “IMG Tag Proposal” im WWW-Talk-Forum vom 25. Februar 1993 (*mittlerweile offline*)

[3] zitiert ebenda.

[4] “An XBM file is actually more analogous to an icon file on Windows than to a BMP” [Mian99] p.3

[5] “XBM [is] rarely used outside that environment…” [ibid]

map-Bildformat im Internet“ darstellt.[1] Bei zunehmender *Online*-Anwendung der WWW-Suite und der damit einhergehenden Vernetzung heterogener Systeme wurde jedoch verstärkt deutlich, dass Unix-fremde Internet-Clients mit dem eigenwilligen Format sowohl recht wenig anfangen konnten, als auch dessen verschwenderisches Speicher-Verfahren für anfangs schier unerträgliche Übertragungszeiten sorgte und im Verlaufe der Entwicklung daher die Frage nach den zunächst kargen Bandbreiten des Internet entsprechenden Algorithmen aufwarf, die eine entsprechende Komprimierung der Bilddaten und eine damit einhergehende Reduktion der Übertragungszeit ermöglichen sollten.

Zum besseren Verständnis der daraufhin im Rahmen des *World Wide Web* zum Einsatz kommenden, komprimierten Bildformate soll daher zunächst kurz auf verschiedene, diesen Formaten zugrunde liegende Kompressionsverfahren eingegangen werden. Da diese Konzepte überdies die Basis für heutige, großteils freilich darüber hinausgehende Techniken hinsichtlich Internet-basierter *Streaming*-Formate[2] sowie Web-fähiger Multimedia-Anwendungen[3] bildet, sind die im Folgenden knapp dargestellten Ideen dieser grundlegenden Kompressions-Algorithmen somit auch unter dem Gesichtspunkt der Übertragung (multimedialer) Präsentationsdaten im Internet relevant.

3.3.1 Grundlagen der Bildkomprimierung

Im besonderen Detail auf die einzelnen Kompressionsalgorithmen einzugehen, würde an dieser Stelle sicherlich den Rahmen dieser Diplomarbeit sprengen, daher sei zum Verständnis der den heutigen Internet-Bildformate zugrunde liegenden Prinzipien lediglich oberflächlich erwähnt, dass nach Identifikation des *Ziels* der Datenkompression (Reduzierung des Speicherbedarfs von Bildinformationen) das Hauptprinzip zur Erreichung dieses Ziels nach [Arno92] das „Eleminieren von Redundanzen und Irrelevanzen“ darstellt. Man unterscheidet in diesem Zusammenhang auch Codierungsverfahren der „Entropiecodierung“ bzw. „Quellencodierung“.[4]

3.3.1.1 Verlustlose Verfahren

„Entropiecodierung“ bezeichnet in diesem Falle die Kompression ohne Verlust von Bildinformationen (verlustfreie Komprimierung), wobei primär der Aspekt der unterschiedlichen Häufigkeitsverteilung einzelner Bildinformationen ausgenutzt wird:

> *Klassisches Beispiel für reine Entropiecodierung ist die Statistische Codierung. Hier geht es ausschliesslich um das Verringern von Redundanzen. Redundanzen finden wir in vielen statistisch bekannten Eigenschaften des Bildsignals und in den Abhängigkeiten der Bildelemente untereinander.*
>
> [Kres95:13]

Das wohl einfachste Verfahren ist unter diesem Aspekt sicherlich die so genannte *Lauflängenkodierung*, auch als RLE bezeichnet („Run Length Encoding“). Hier werden gleichartige, aufeinander folgende Informationen (etwa 5,5,5) schlichterdings zusammengefasst (im Beispiel: 3x5) und auf diese Weise komprimiert. Einer etwas anderen Herangehensweise bedient sich indes die *LZ*-Kodierung: Dieses nach ihren Erfindern Lempel und Ziv benannte Verfahren [LZ77] speichert mehrfach auftretende Bitfolgen in einem so genannten „Pixel-Value Dictionary“. Statt der eigentlichen Bildinformation enthält der codierte Bit-Strom somit oftmals lediglich *Verweise* auf einzelne Tabellen-Einträge und kann damit erheblichen Speicherplatz einsparen. Die wohl bekannteste und meistverwendete Form der Entropiekodierung ist allerdings die Codierung nach *Huffman*, die aufgrund eines Wahrscheinlichkeits-Binärbaumes die effiziente Speicherung un-

[1] vgl. [Blei99] p.9
[2] s. 3.5.1
[3] so liegen dem Shockware- bzw. Flashformat etwa eine ganze Reihe der in [3.3.1] genannten Techniken zugrunde
[4] vgl. [Kres95] pp. 13ff.

terschiedlicher Informationen in variabler Bitlänge ermöglicht, wobei den am häufigsten vorkommenden Zeichen, ähnlich des Morse-Codes, jeweils die kürzesten Codeworte zugewiesen werden.

3.3.1.2 Quellencodierung: Verlustbehaftete Kompression

Im Gegensatz zu den soeben beschriebenen Entropie-Codierungen bedient sich die so genannte *Quellen-Codierung* nicht der eigentlichen, diskreten Bildwerte, sondern vielmehr der *Semantik* der zu codierenden Informationen. Da die verschiedenen Verfahren jeweils die „Irrelevanzen", also die vom Benutzer subjektiv weniger gut wahrnehmbaren Bildinhalte eliminiert, ist die Quellencodierung zwar grundsätzlich Verlust-*behaftet*, erzielt dadurch jedoch erheblich bessere Kompressionsraten. Grundlegendes Kennzeichen dieser Kodierung ist die Hin- und Rücktransformation aus dem Bild- in den so genannten *Frequenzbereich*, in dem jeweils die eigentliche Datenreduktion erfolgt: hohe Bildfrequenzen werden aufgrund zunehmender Irrelevanz einfach „abgeschnitten". Bekannteste Erscheinungsform ist hierbei das so genannte DCT-Verfahren, bei der die diskrete Kosinus-Tranformation[1] den Umwandlungsprozess realisiert.

3.3.2 Konventionelle Internet-Bildformate

Zur Anwendung kommt diese „sehr komplexe" [Schm99:24] Technik etwa im JPEG-Format, dem derzeit wohl meistverbreiteten Internet-Bildformat. Eigentlich korrekterweise mit dem Kürzel JFIF[2] bezeichnet,[3] werden hierbei die nach der Kosinus-Rücktransformation gewonnenen, bereits komprimierten Bilddaten zur mit RLE- bzw. Huffmankodierung weiter in ihrer Speichergröße reduziert. Aufgrund der Verlust-Behaftetheit dieses Verfahrens[4] bietet sich das JFIF-Format daher primär für photografische Abbildungen an, erscheint jedoch hinsichtlich der Kodierung computergenerierter Grafiken nur wenig geeignet.

Diese „Lücke" weiß wiederum das GIF-Format,[5] welches bereits vor der langwierigen (da Gremien-basierten)[6] Entwicklung JPEGs eingeführt und daher frühzeitig im Rahmen der Browser-Bildunterstützung implementiert werden konnte,[7] elegant zu füllen: Da sich die GIF-Kodierung einer durch Terry Welch verbesserten Variante des bereits erwähnten LZ-Verfahrens (LZW)[8] bedient, können insbesondere Grafiken mit großen, homogenen Flächen-Anteilen äußerst effizient komprimiert werden. Der Nachteil dieser Lookup-Technik begründet sich hingegen ebenfalls in der Paletten-Basiertheit dieses Algorithmus': So sieht die konkrete Implementierung der GIF-Komprimierung lediglich Farbtiefen von maximal 8 Bit[9] vor, was das Format wiederum für photorealistische Bilder uninteressant und bei Einsatz von *Dithering*[10] zudem ineffizient macht. Nach anfangs (insbesondere durch das Engagement des Online-Dienstes *CompuServe*) nahezu epidemischer Verbreitung ist das Format durch rechtliche Forderungen des LZW-Patenteigners Unisys überdies heute zusehends „zur Bedeutungslosigkeit verdammt" [Mian99:171][11]

Hauptgrund für den berechtigten Niedergang des GIF-Formates bildet wiederum die eigens im Rahmen der *Unisys*-Patentstreitereien initiierte Entwicklung des daraufhin bereits 1996 veröffentlichten PNG[12]-Standards [Bout96]: Dieses ebenfalls auf dem LZ-Verfahren (wennauch freilich auf dessen patentfreier LZ77-

---

[1] alternativ: Fourier-Transformation

[2] JFIF: JPEG File Interchange Format

[3] Anm: JPEG bezeichnet lediglich den Namen seiner „Erfinder", nicht aber des Formates selber.

[4] Anm: Es existiert zwar ein entsprechendes „verlustloses" JPEG-Format (JPEG-LS), dieses greift aber wiederum auf einen anderen Algorithmus zurück und ist aufgrund geringer Kompressionsraten nur wenig verbreitet.

[5] GIF: Graphic Interchange Format

[6] JPEG stellt eine Zusammenarbeit gleich zweier Gremien dar: Der ISO sowie der CCITT (daher auch „joint")

[7] s. hierzu Marc Andreessens "IMG Tag Proposal" im WWW-Talk-Forum vom 25. Februar 1993 [mittlerweile offline]

[8] LZW: Lempel-Ziv-Welch.

[9] 8 Bit = 2^8 = 256 Farben

[10] Statistische Verteilung von Pixeln (im Gegensatz zur *Nearest-Color*-Methode, die stets auf den nahe-liegendsten Farbwert zurückgreift)

[11] "Legal entanglements have certainly condemned it to obsolescence." [Mian99] p.171

[12] PNG: Portable Network Graphics

Komponente) beruhende Format [vgl. Bert96] verbindet durch Aufteilung des Bildes in so genannte „Chunks“ die Vorteile des GIF-Formates (Verlustfreiheit) mit den höheren Kompressionsraten und insbesondere der höheren Farbtiefe von JFIF. Da PNG überdies die bislang problematischen Eigenschaften des JPEG-Formates hinsichtlich mangelhafter Farbtreue [vgl. Mian99:35ff] mittels erweiterter Palletten-Funktionalität zu beheben weiß und darüber hinaus auch endlich einen separaten Alpha-Kanal zur Speicherung von Bildtransparenzen anbietet, stellt es nicht nur ein ideales intermediäres Format zur Speicherung verschiedenster Bilddaten, sondern ebenso eine beeindruckende Alternative hinsichtlich des „Deployments“, also der eigentlichen Präsentation von Grafiken im Internet-Umfeld dar und kommt daher auch im Rahmen der des Diplomarbeit zugrunde liegenden Präsentationskonzepts *XML » SVG Presenter* [Kap.6] extensiv zum Einsatz.[1]

Einschränkend muss an dieser Stelle jedoch erwähnt werden, dass PNG gerade hinsichtlich *multimedialer* Präsentationen derzeit noch einen erheblichen Nachteil gegenüber dem GIF-Konkurrenten aufweist: Während eine rückwärtskompatible GIF-Erweiterung des auch rudimentäre *Animationen* ermöglicht, wird das entsprechend erweiterte Animations-Pendant des PNG-Formates, MNG [Rand97],[2] derzeit noch von keinem der gängigen Internet-Browserprogramme unterstützt, da es die PNG-Autoren wohl aufgrund des hektischen Entwicklungsprozesses[3] bedauerlicherweise versäumt haben, bereits zum Veröffentlichungszeitpunkt des Vorgängers die entsprechende Funktionalität bereitzustellen.[4]

3.3.3 Innovative Komprimierungsverfahren

Vor genauerer Betrachtung dieser Animationseigenschaften [s. hierzu 3.4] erscheint jedoch an dieser Stelle noch die Frage relevant, inwiefern trotz der bereits „guten Komprimierung“ [Woda02] im Rahmen der bisher vorgestellten, mittlerweile zum Browser-Standard zählenden (und immerhin bereits 10 Jahre „alten“) Dateiformate darüber hinaus alternative bzw. fortschrittlichere Datenkompressionsverfahren vorhanden und verfügbar sind. Insbesondere sind hier natürlich der Aspekt der *Verbreitung* entsprechender Formate im Umfeld des *World Wide Web*[5] sowie die Existenz von En- und Decodierungsschnittstellen bei diesen Überlegungen von Interesse. Und in der Tat stellen in der Hauptsache gleich zwei unterschiedliche Komprimierungsverfahren durchaus interessante Funktionalität und Möglichkeiten bereit, an dieser Stelle weitaus beeindruckendere Datenreduktionsraten zu erzielen. Aus diesem Grunde sollen diese zwei grundlegenden Ansätze im Folgenden kurz Erwähnung finden:

3.3.3.1 Fraktale Bildkomprimierung

Der wohl bekannteste und in meinen Augen faszinierende Ansatz stellt hierbei die *fraktale Kompressionstechnik* dar, die unter Ausnutzung des „Prinzips der Selbstähnlichkeiten im Bild“ [Kres95:44] überwältigende Kompressionsraten von „bis zu 1:10.000“ [BaSl87] ermöglicht. Da das dieser Technik zugrunde liegende Verfahren nach deren Entwicklung am Georgia Institute of Technology (GATech) in Atlanta längere Zeit „unter Verschluss“ gehalten wurde,[6] ist die fraktale Kompression in der Bildverarbeitung heute zwar allgemein bekannt, wird allerdings bedauerlicherweise immer noch „nicht sehr häufig angewandt“ [Ruhl97:4]. Erst die Dissertation des GATech-Laboranten Arnaud Jaquin [Jaqu89] ermöglichte der Fachwelt schließlich Einblicke in die Funktionsweise des entsprechenden Algorithmus: Demnach wird im Rahmen der so genannten „iterierenden Funktionssysteme“ (IFS) zunächst versucht, nicht überlappende Teilbe-

[1] s. 6.3

[2] MNG: Multiple-image Network Graphics (sprich: „Ming“)

[3] “The final version was released on October 1, 1996, with just over a year and a half after the project began.” Jahre [Mian99:190]

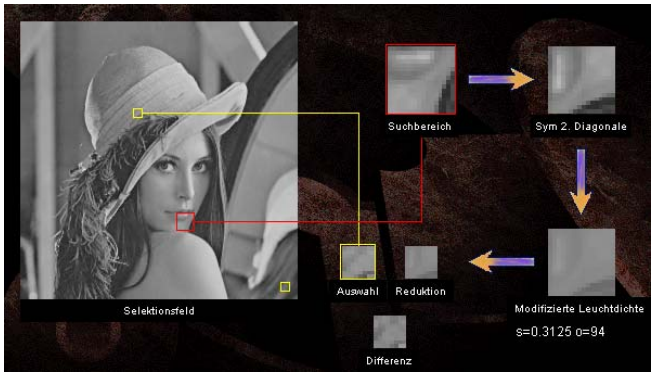
[4] Anm: Dies ist insofern Relevant, als dass zu diesem Zeitpunkt die entsprechende Animations-Erweiterung des GIF-Formates bereits seit längerem veröffentlicht war [vgl. SeLo97]

[5] Hier ist natürlich speziell von entsprechender Browser-Unterstützung die Rede.

[6] vgl. [Kres95] p.42 sowie [Ruhl97] p.4

reiche eines Bildes, die „Ranges“, durch jeweils eine kontrahierende Abbildung die auf einen anderen Teilbereich, der sogenannten „Domain“, des Bildes ausgeführt wird, möglichst gut anzunähern. [vgl. Fick99] Eine interaktive Java-Anwendung des französischen Eurécom-Instituts [Duge99] visualisiert diesen Sachverhalt auf anschauliche Weise:

Abb. 3.3.3.1: Eine Range wird an eine Domain angenähert.[1]



Wie in [Duge99] deutlich wird, ist die „Domain“ in der Regel größer als die „Range“, auf die sie abgebildet wird,[2] da die Überführung des ersten in den zweiten Bildbereich stets durch eine *geometrische* (Rotation, Spiegelung etc.) sowie eine *Helligkeits*transformation durchgeführt wird.[3] Hauptproblem der fraktalen Bildkompression ist nun die Bestimmung eines IFS für eine Vielzahl möglicher „Ranges“, das die jeweilige Bildvorlage möglichst gut beschreibt.

> *Es gibt [jedoch] unendlich viele solcher affinen Transformationen… [Daher] ist die Frage, wie viele man davon braucht, bzw. wie „optimal“ eine gewählte Transformation ist.*
>
> [Kres95:44]

Die Mathematik kommt uns hier allerdings einen Schritt entgegen: So müssen die jeweiligen Bildbereiche nicht „optimal identisch“ sein, sondern sich lediglich „hinreichend“ ähneln. Der hierdurch entstehende Informationsverlust („Artefakte“) bildet somit das Abbruchskriterium des ansonsten „NP-harten“ [4] Algorithmus [vgl. Ruhl:5,43,63], sodass die Qualität bei fraktaler Bildkompression nicht allein durch den Speicherbedarf, sondern einen ebenso wichtigen Zeitfaktor determiniert wird. Bei Bilddaten mit großer Selbstähnlichkeit[5] lassen sich jedoch bereits mithilfe dieses Ähnlichkeits-Annäherungsverfahrens durchaus ansehnliche Resultate erzielen, die überdies (schließlich werden mittels IFS-Verfahren Pixel-Daten in mathematische Gleichungen „übersetzt“) im Gegensatz zu den Raster-Originalen frei skalierbar sind.

Entsprechend dieses in [Fish98] auch praktisch dargelegten Verfahrens existieren daher bereits mehrere „schlüsselfertige“ Anwendungen, die eine direkte Verwendung fraktaler Bildkompression erlauben. Neben des im Rahmen einer Diplomarbeit an der Universität Ulm entstandenen, C-basierten[6] *FraComp* von Andreas Kassler [vgl. Kass95] sind dies vor allem Java-Klassenbibliotheken [Fick99,[7] DeLo02], obgleich die Java-Programmiersprache aufgrund ihrer „unerträglichen Langsamkeit“ [8] für den Rechenzeit-intensiven, asymmetrischen IFS-Algorithmus nur wenig geeignet erscheint.

Neben verschiedenen, wenn auch „nicht sehr zahlreichen“ [9] Fraktal-Encodern konnte sich darüber hinaus das sogenannte *Fractal Image Format* (kurz: FIF), dem der IFS-Algorithmus der Firma Iterated Systems [vgl.

---

[1] Nach [Duge99]

[2] s. Abb. 3.3.3.1. In der Regel ist die „Range“ in der praktischen Anwendung somit 8x8 Pixel und die entsprechende „Domain“ jeweils 16x16 Pixel breit.

[3] Quelle: [Fish98] Kap. 6 (pp.119ff)

[4] Das heißt, dass es keinen *polynomiellen* Algorithmus für dieses Problem gibt, falls P != NP gilt, also auf deutsch, dass es keine ‘schnellen’ Verfahren für optimale Kompression gibt.

[5] “Fractal encoders are effective for images composed of isolated straight lines and constant regions since these features are self-similar” [EfSt98] p.35

[6] Anm: Leider ausschließlich für Windows-Systeme verfügbar [vgl. Kass95]

[7] Der Autor stellt neben seinen theoretischen Ausführungen überdies eine konkrete Fraktal-Implementierung zum Download bereit: http://www.stud.uni-siegen.de/markus.fick/ZIPS/ZFC10.ZIP [31.1.03]

[8] vgl. die Code-Kommentare in [Fick99]

[9] vgl. [Ruhl97] pp.4,17

Lu97] zugrunde liegt, durch die anfangs große Verbreitung des zugehörigen Shareware-Programms *Fractal Imager* als Quasi-Standardformat für fraktale Bildkomprimierung durchsetzen. Dies lag nicht einzig an dem relativ flotten Encoding-Prozess, sondern ebenso an den beeindruckenden Ergebnissen, die sich mithilfe des *Fractal Imagers* erzielen ließen: So konnten digitale Bilddaten dank der Iterated-Software nicht nur drastisch in ihrer Grösse reduziert werden – das Kompressionsresultat sieht frappierenderweise sogar *besser* aus und „fördert Details zutage, die im ursprünglichen Bild nicht zu sehen waren". [SeLo97]

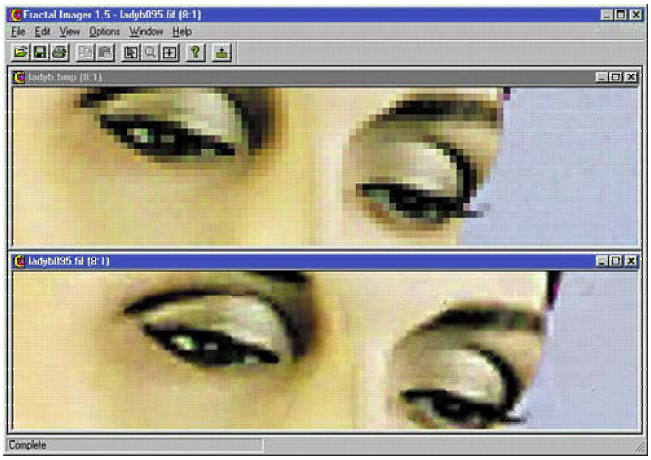


Abb. 3.3.3.2: Fraktale Bildkomprimierung mit dem Fractal Imager

Obwohl insbesondere das FIF-Format zunächst als „Beginn einer großartigen Entwicklung" angesehen wurde [vgl. Jasn99, Stei99:113] und unter anderem in der *Encarta*-CD von Microsoft zur Anwendung kam [BaLy96:1, EfSt98:35], konnte es sich bedauerlicherweise weder im Offline- noch im Internetbereich gegen den qualitativ zweifellos unterlegenen[1] JPEG-Standard durchsetzen. Nachdem auch die Entwicklung weiterer, fraktaler Kompressionsalgorithmen seit Ende der 80er Jahre nicht mehr sichtbar fortgeschritten ist, findet der FIF-Ansatz aufgrund mangelnder Browser-Unterstützung (trotz der Entwicklung eines entsprechenden Plug-Ins) im Internet praktisch keinerlei Verwendung. Da *Iterated Systems* trotz des Misserfolgs der ursprünglichen FIF-Version[2] statt einer Freigabe auf erhebliche Lizenzgebühren für das Fraktal-Format bestand, versinken auch existierende FIF-Implementierungen in zunehmender Bedeutungslosigkeit – ein gutes Beispiel, so [Voge02], wie man „gute Ideen kaputtmachen kann".

3.3.3.2 Wavelets

Erheblich langlebiger erwies sich an dieser Stelle hingegen der ebenso innovative, wenn auch ein gänzlich anderes mathematisches Prinzip verfolgende *Wavelet*-Ansatz: Dieser geht, wie bereits die „konventionelle" Fourier- bzw. DCT-Quellencodierung, von einer Überführung vom Orts- in den Frequenzbereich aus. Die Wavelet-Theorie stellt uns allerdings nun eine *Vielzahl* unterschiedlicher Transformationen zur Verfügung:

> *Man könnte die so genannte „Schnelle Wavelettransformation" auch als eine Folge von Hoch- und Tiefpassfiltern beschreiben, wobei die durch Filterung entstandenen Verluste mithilfe einer Differenzinformation abgespeichert werden und damit nicht verloren gehen*
>
> [Kres95:45]

Durch dieses Verfahren kann nicht nur eine erhöhte Qualität, sondern ebenso eine verbesserte Komprimierung der Bilddaten erreicht werden. Aufgrund dessen kommt die Wavelet-Technik derzeit etwa beim FBI zur Speicherung von Fingerabrücken[3] oder auch in der Medizintechnik [vgl. Lang99] zur Anwendung. Neben wie bereits in der eben angesprochenen Fraktalkompression durchaus erwähnenswerten Implementierungen der akademischen Welt, die überdies interessante Schnittstellen zwischen den beiden innovativen Verfahren offenbaren [KSH01,[4] FRS94] fand das Wavelet-Verfahren im Gegensatz dazu auch in Bereichen des Video-Encodings und ebenso des WWW Verwendung: Zwar waren die frühen Software-Produkte und

[1] „Der Qualitätsvergleich zwischen gleich großen JPEG- und FIF-Dateien fällt bei fotografischem Bildmaterial zugunsten der fraktalen Methode aus." [SeLo97]

[2] So findet sich auf der Webseite der Firma (trotz mehrfacher Verweise hierauf) keinerlei Hinweis mehr auf das Format, noch auch eventuelle, entsprechende Software: http://www.iterated.com [31.1.03]

[3] vgl. [Kres95] p.45

[4] "Wavelet coders are well suited for this purpose because the wavelet coefficients can be naturally ordered according to decreasing importance. Progressive fractal coding is feasible, but it was proposed only for hybrid fractal-wavelet schemes" [KSH01]

entsprechenden Wavelet-Plug-Ins [vgl. SeLo97] von *Summus*, *Infop* und *Image etc*. zueinander inkompatibel und daher schlussendlich ebenso wie bereits das FIF-Format nur wenig verbreitet – dafür hat das Wavelet-Verfahren allerdings seinen Weg in den JPEG2000-Standard [Boli00] gefunden und ermöglicht in diesem Rahmen eine insbesondere aus qualitativer Hinsicht beachtliche Bildkomprimierung:

Obwohl bis zur vollständigen Integration der letztgenannten Implementierung noch einige Zeit vergehen wird (auch das JPEG2K-Format wird, im Gegensatz zu seinem Vorgänger JFIF, derzeit noch von keinem Internet-Browser unterstützt), so kann doch davon ausgegangen werden, dass sich dieser Standard zumindest aus langfristiger Sicht im Rahmen des WWW-Deployments sicherlich durchsetzen wird [vgl. Bert01, Gold03]

3.3.4 Fazit

Abschließend betrachtet ist daher festzustellen, dass die überwiegende Mehrzahl des im Web verwendeten Bildmaterials derzeit bedauerlicherweise noch in Form veralteter Grafikformate vorliegt: So wurde das veraltete JPEG-Format JFIF zwar schon längst auf dem Papier durch den Wavelet-basierten JPEG2000-Standard ersetzt, – dieser ist, trotz Standardisierung, jedoch (zumindest derzeit) in keiner einzigen Browser-Umgebung implementiert. Derweil gerät das immer noch in durchaus beachtlicher Quantität zur Anwendung kommende[1] GIF-Format aufgrund technischer Mängel und patentrechtlicher Konflikte zwar zunehmend ins Abseits, kann jedoch durch den eigens hierfür entwickelten, technisch überzeugenden PNG-Standard – auch dank teils noch mangelnder Browser-Unterstützung[2] – noch „nicht vollständig ersetzt werden“ [SeLo74]. GIF-ähnliche Lizenzansprüche des Eigners des bislang wohl populärsten Formates auf Basis *fraktaler Bildkomprimierung*, dem FIF-Format,[3] standen überdies einer weiteren Verbreitung dieses aus technischer Sicht äußerst interessanten Verfahrens [vgl. Grae96] bisher entgegen, weswegen aufgrund derzeit ausbleibenden Entwicklungsfortschritts eine Anwendung des IFS-Algorithmus im World Wide Web auch langfristig eher unwahrscheinlich erscheint.

Aus ästhetischer Perspektive kann an dieser Stelle freilich hinzugefügt werden, dass die frühe Einführung pixelorientierter Grafikunterstützung in nahezu[4] sämtliche Browserumgebungen das „grafische“ Web-Design in dessen Verlauf „ganz erheblich“ beeinflusst hat: Insbesondere in Verbindung mit der „missbräuchlichen Verwendung“ von HTML-Tabellen [s.3.2] kamen GIF- und JPEG-Grafiken im Rahmen opulent ausgestatteter Web-Seiten in „schier unglaublichem Umfang“ [vgl. Wein97:10] zum Einsatz: Aufgrund der bereits ausgeführten, mangelhaften Textformatierungseigenschaften des HTML-Standards selbst satteln viele „Design“-orientierte Webentwickler auf die ausschließliche Anwendung grafikbasierter Schriftelemente [vgl. Rieh01:55] um – was bei „grafisch opulenten“ Webseiten gar dazu führen konnte, dass trotz der Verwendung des HTML-Standards mitunter nicht auch nur ein einziges „richtiges“ Textelement zur Anwendung kam und manche Seiten somit aus der Perspektive der Internet-Suchmaschinen unauffindbar und daher im Sinne der Web-Semantik gänzlich unbrauchbar wurden – und, dank der Einführung „hilfreicher Tools“ wie etwa Adobes *ImageReady*,[5] oftmals bis heute noch sind. Mit dieser Entwicklung einher ging überdies eine „explosionsartige“ Entwicklung der im Internet übertragenen Datenmenge, die sich innerhalb weniger Monate um „bis zu tausend Prozent“ vervielfachte.[6]

---

[1] „Heute steht die überwältigende Mehrzahl der Bilder im Web im GIF-Format.“ [Schm99] p.20

[2] Anm: Die 4er bzw. 5er-Generation des Microsoft Internet Explorer hat etwa „durchaus ihre Probleme“ bei der Darstellung des PNG-Formates (Buffer-Overflow, keine/fehlerhafte Transparenz-Darstellung)

[3] FIF: Fractal Image Format

[4] Ausnahmen: „Thin“ Browsers sowie so genannte Text-Browser (Lynx)

[5] Anm: Das Programm erlaubt es, große Bilder (die jeweils eine gesamte Webseite darstellen können) in Verschiedene Zellen („slices“) aufzuteilen und sodann automatisch in HTML-tabellierte Grafiken zu verwandeln

[6] vgl. [Wein97] p.10

Da sich neben diesen problematischen Aspekten intensiver Grafikbenutzung etwa nach Meinung Jakob Nielsens an dieser Stelle überdies erhebliche „*usability issues*“ auftun [vgl. Niel97], stellt die „übermäßige Anwendung“ [Engs96] von GIFs und JPEGs, in Verbindung mit der zuvor diskutierten, derzeitigen *Ignoranz* fortschrittlicher Kompessionstechniken [s.3.3.4], in den Augen zahlreicher Kritiker einen weiteren „Nagel im Sarg des Web“ dar:

> *The fundamentalists [are] really torn up inside about all "those newfangled freaks and perverts who've wrecked the Web with their GIFs and JPEGs".*
>
> [Balo00:2]

Unter dem Aspekt der „Webtauglichkeit“ erscheint der „exzessive Gebrauch“ [Rott01:2] grafischer Elemente allerdings nicht nur reichlich problematisch, sondern im Rahmen ansprechender Internet-Präsentationen überdies unzureichend: So macht bereits der zunächst rudimentäre „Web-Export“ PowerPoints [s.2.3.4] in Form statischer JPEG-Grafiken deutlich, dass diese Variante der „Konvertierung“ einer Präsentation neben dem Verlust der Text-Eigenschaft überdies weitere wichtige Möglichkeiten einer Multimedia-Präsentation (wie etwa Skalierbarkeit, FullScreen-Modus, Multimedia-Effekte etc.) außen vor lässt. Auch die in diesem Abschnitt betrachteten, fortgeschritteneren Komprimierungsverfahren vermögen (mit Ausnahme des Fraktalverfahrens, welches eine grafische *Skalierbarkeit* zumindest theoretisch erlauben würde) diese Mängel nicht zu beheben.

3.4 Bewegung kommt ins Spiel: Animationen im Web

Insbesondere im Hinblick auf multimediale Präsentationssysteme im Internet sind daher weniger die eben ausgeführten Kompressionseigenschaften der zur Anwendung kommenden Bildformate relevant, als vielmehr deren wirklich „multimediale“ Eigenschaften: So ermöglichte eine Spezifikations-Erweiterung der bereits 1989 veröffentlichten Variante des GIF-Formates (GIF89a) etwa schon frühzeitig die Erstellung einfacher, sequentieller Animationen. Obgleich bereits die ursprüngliche Fassung des Dateiformates[1] ein prinzipiell wiederholbares Bildsegment vorsah, bot hingegen erst die GIF89a-Fassung Raum für wichtige Informationen zum Ablauf-Timing.[2] Auch die Web-Browsersoftware *Netscape* (bzw. deren Vorgängerversion *Mosaic*) sah bereits im Rahmen seiner ersten *Releases* die grundlegende Unterstützung auch animationsbasierter GIF-Dateien vor – machte dies jedoch „in weiser Voraussicht“ erst einiges später im Zuge der Netscape 2.0-Einführung öffentlich [Baum98:93]. Nachdem das begehrte Feature jedoch schließlich von Seiten Netscapes selbst eindeutig dokumentiert wurde,[3] gab es für die „Web-Designer-Community“ schließlich kein Halten mehr – schließlich bedeutete diese Möglichkeit aus theoretischer Perspektive weitaus mehr als die Implementierung einer „interessanten Funktion“: Indirekt stellte die Browser-Unterstützung animierter GIF-Dateien, und dies im Kontext einer HTML-Umgebung, zugleich den Übergang des Hypertext-Mediums WWW bzw. HTML, entsprechend der bereits in [3.3] geführten Diskussion, in Richtung *Hypermedia* dar. Da in Form der entlang einer veränderbaren Zeitachse ablaufenden GIF-Animation überdies ein *kontinuierliches* Medium mit der *diskreten* Hypertext-Komponente verbunden werden konnte, kennzeichnet diese Funktionalität das WWW überdies als „echtes *Multimedia*-System“ auch im Sinne der strengen, diesbezüglichen Kriterien nach [Stei99:10ff].

Zugleich markierte die Einführung animierter GIF-Grafiken überdies einen „ästhetischen Wandel“ der Webseiten-Gestaltung: Selbsternannte „Web-Designer“ fassten den neu erschlossenen „Multimedia-Kontext“ (s.o.) der HTML-Umgebung auch als ebensolchen auf und bedienten sich der Animations-Funktionalität folglich ausgesprochen großzügig. Entgegen den Gestaltungsgrundsätzen *Stankowskis*

[1] GIF87a – Anm: zur Nachfolgeversion inkompatibel [vgl. Stei99:52]
[2] vgl. [SeLo97]
[3] „Netscape hatte die Animations-Möglichkeit (einige Zeit) nicht dokumentiert…“ [Baum98] p.93

[Stan94], nach denen Animationen nur dort Sinn machen, „wo das Wort oder der Gedanke Unterstützung benötigen“,[1] kam die in den Augen der oft Design-unerfahrenen HTML-Gestalter „begeisternde“ [Reib99] Animationsfunktionalität in schier epidemischem Ausmaß zum Einsatz, so dass fortan „rotierende Erdkugeln, flatternde Werbefahnen“ [SeLo97] und „blinkende Pfeile im Jahrmarkt-Stil“ [Jaco02a:41] das Erscheinungsbild einer „überwältigenden Anzahl“ [Schm99:20] von Webseiten und somit auch den (ästhetisch somit durchaus fragwürdigen)[2] Gesamteindruck des *World Wide Web* großteils prägten:

> *Animierte GIF-Grafiken [sind derzeit] als nervige Werbe-Wackelbilder und herzlich überflüssige Blink-Buttons verpönt.*
>
> [Chro02]

Über die zweifellos zunehmende Unbeliebtheit animierter Banner-GIFs hinaus stellt UI-Spezialist Jared Spool darüber hinaus fest, dass nahezu „sämtliche Benutzer“ Animationen im Web als grundsätzlich „irritierend“ empfinden [SSSA99:89f] – eine „ständige Animation“, so merkt auch [Fors98] an, kann dem Benutzer „sogar regelrecht auf die Nerven gehen“.[3]

3.4.1 Animated GIF

In der Tat überrascht es daher, dass die überwiegende Mehrheit der im Web zu Anwendung kommenden, animierten GIF-Grafiken „indeterminiert ge*loop*t“ sind, d.h. sich deren Bewegungsablauf unendlich oft wiederholt – und dies, obwohl sich im Rahmen der erweiterten GIF98a-Spezifikation überaus bequem die Anzahl der Iterationen, also der Animations-Wiederholungen, diskret angeben lässt.

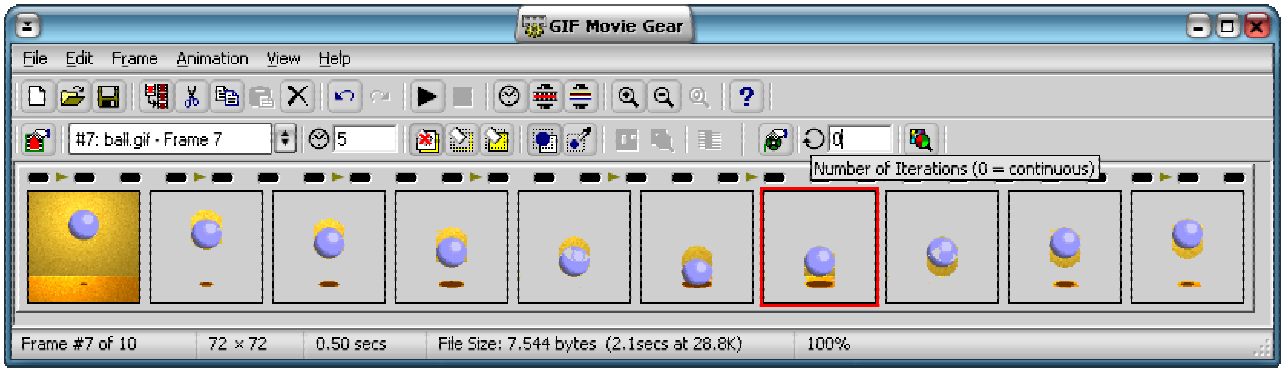


Abb. 3.4.1.1: GIF-Animation im GIF Movie Gear.

Überdies lässt sich auch die Dauer der Animation sehr präzise steuern, ebenso wie sich – ein entsprechend komfortables Bearbeitungs-Werkzeug wie den überaus „empfehlenswerten“ *GIF Movie* Gear [vgl. Chro02] natürlich vorausgesetzt – weitere, Speicherplatz sparende Eigenschaften des GIF-Formates ausnutzen lassen: So erlaubt es die Transparenz-Eigenschaft der GIF98a-Spezifikation etwa, anstatt der bei Animationen üblichen, vollständigen *Frames* lediglich einzelne, kleinere Bildfragmente[4] und/oder teiltransparente Folgebilder zu übertragen, um bei sich wenig ändernder Bildumgebung auf entsprechend „redundante“ Information verzichten zu können. Da überdies die Angabe unterschiedlicher Frame-Dauern („Durations“) für jedes einzelne Teilbild separat möglich ist, lassen sich auch unnötig erscheinende Frames komfortabel aus der Animation entfernen, um so zusätzlichen Speicherplatz einzusparen.[5]

Dennoch erweist sich gerade die eben angesprochene Transparenz-Eigenschaft im Hinblick auf deren „1-bittige“ Implementierung als mitunter problematisch: Da sich Bildflächen lediglich entweder 100%-ig

---

[1] vgl. [Stan94] p.20

[2] “[There] are some really ugly and annoying Animated GIFs that rate high on the Annoyance scale.” [Pomp99]

[3] vgl. [Fors98] p.127

[4] Anm: Einschließlich entsprechender ggf. Offset-Informaition

[5] vgl. hierzu auch [Chro02].

transparent oder eben vollständig opak darstellen lassen, können *Fades* oder mittels Anti-Aliasing geglättete Kanten nur „sehr enttäuschend" direkt umgesetzt,[1] [vgl. Schm99:23] oder aber mittels *Dithering* aufwendig „simuliert" werden. Da unter Anwendung dieser Technik jedoch normalverteilte „Störpixel" in die Grafik eingefügt werden, sinkt zugleich die Effizienz der LZW-basierten GIF-Komprimierung – für umfangreichere, komplexe Aufgaben ist die Animations-Komponente des GIF-Formates daher nur sehr begrenzt geeignet: „Ihre Dateigröße würde alle vernünftigen Grenzen sprengen" [SeLo97]. Über diesen rein technischen Aspekt hinaus gilt weiterhin natürlich auch für die animierte Version das, was bereits die Anwendung statischer GIF-Grafiken durchaus problematisch erscheinen lässt: Da LZW-Patenteigner Unisys natürlich auch für die Erzeugung animierter GIFs erhebliche Lizenzforderung stellt, ist von einer großzügigen Verwendung dieses Dateityps im Rahmen der Erstellung webbasierter Präsentationen mit Animations-Charakter daher nicht nur aus technischer und „ästhetischer", sondern ebenso aus wirtschaftlicher Sicht abzuraten.

3.4.2 MNG: Die ignorierte Alternative

Das entsprechende Animations-Pendant des GIF-Nachfolgers PNG, MNG,[2] vermag das Gros der soeben genannten Probleme jedoch durchaus zu beheben: Ähnlich wie bereits die entsprechende GIF-Erweiterung macht die Funktionsbeschreibung des Formates [Rand97] schnell deutlich, dass eine MNG-Datei in der Hauptsache lediglich einen zeitlich und organisatiorisch präzise steuerbaren „Array" aneinander gereihter PNG-Einzelbilder darstellt. Im Gegensatz zum GIF-Format erlauben an dieser Stelle jedoch die hervorragenden Transparenz-Eigenschaften der PNG-Spezifikation überaus elegante, ästhetische Effekte wie etwa film-ähnliche „Fade-Übergänge". Darüber hinaus unterstützt MNG zudem eine objektorientierte „Sprite-Funktionalität", die etwa die Komprimierung Objekt-basierter Animation deutlich begünstigt.

Dennoch besteht derzeit (zumindest von Seiten der Browser-Hersteller) bedauerlicherweise kein ausgesprochenes Interesse hinsichtlich einer nennenswerten Anwendung des Formates im WWW. Im Gegensatz zum Vorgänger GIF, dessen Animations-Komponente [s.3.4.1] rückwärtskompatibel und bereits vor „Erfindung" des WWW ausführlich spezifiziert wurde, stellt MNG eine *separate* Erweiterung der PNG-Spezifikation dar, die wohl aufgrund der zeitversetzten Veröffentlichung bisher nicht den Weg in Microsofts Internet Explorer gefunden hat – und dies wohl auch längerfristig nicht gelingen wird: Zwar existieren neben der erfreulichen, nativen *Mozilla*-Unterstützung[3] bereits funktionstüchtige Plug-Ins [Summ03] und sogar eine auch durch Quellcode-Freigabe weiterverwendbare Java-Implementierung [Iked00], doch trotz der Standardisierungs-Bemühungen seitens des W3C [vgl. Rand97] droht dem MNG-Format, da es derzeit aufgrund seines „OpenSource-Flairs" von Internet-Explorer-Hersteller Microsoft weitgehend ignoriert wird, der Absturz in die Bedeutungslosigkeit, sollte es der durchaus aktiven *LibMNG*-Initiative [Juyn00] auch in Zukunft nicht gelingen, das konzeptionell viel versprechende Format auch zahlenmäßig bedeutenderen Entwickler- und Nutzerschichten und insbesondere größeren Software-Unternehmen näher zu bringen.

3.4.3 Fazit

Zusammenfassend bleibt daher festzustellen, dass auf Basis der betagten GIF89a-Spezifikation derzeit eine Fülle „irritierender" [SSSA99:89f] und durch deren technische Mängel auch ästhetisch fragwürdiger,[4] animierter GIF-Grafiken [vgl. Pomp99] im Web kursiert. Neben deutlichen Vorteilen hinsichtlich Kompressions-Effizienz und Bildqualität ermöglicht das entsprechende PNG-Pendant MNG durch beeindruckende Transparenz-Funktionalität zwar auch aus ästhetischer Hinsicht (zumindest theoretisch) interessantere A-

[1] „Eine transparente Komponente in eine GIF-Grafik zu bringen, kann auch zu enttäuschenden Ergebnissen führen. [Schm99] p.23

[2] MNG: Multiple-image Network Graphics (sprich: „Ming")

[3] Dies schließt sogar die Netscape-Versionen 6 und höher mit ein [vgl. Juyn00]

[4] Aufgrund mangelhafter Transparenz-Unterstützung und dadurch bedingten „Unwegbarkeiten" bei der Anwendung von *Anti-Aliasing* kommt es bei der Verwendung von GIF-Grafiken oft zu „unansehnlichen, enttäuschenden Ergebnissen" [vgl. Schm99:23]

nimations-Ergebnisse,[1] ist jedoch aufgrund teils ausbleibender Browser-Unterstützung [vgl. Juyn00] im Umfeld des WWW derzeit nur wenig verbreitet.

3.5 Echtes Multimedia im Web?

Abgesehen von den rein technischen Realisierungs-Aspekten wirft die durch Animierte GIFs sowie MNG bereitgestellte Funktionalität jedoch die Frage auf, inwieweit diese Animations-Eigenschaft Selbstverständnis und Möglichkeiten HTML-basierter Präsentationen wirklich beeinflusst haben: So stellt die Einführung dieser Funktionalität zwar zumindest theoretisch, wie bereits in [3.4] diskutiert, einen „Übergang zu echtem Multimedia" dar [vgl. Stei99:11] – die genauere Betrachtung der tatsächlichen Animations-Eigenschaften auf dieser Basis realisierter Ergebnisse macht jedoch deutlich, dass sich die Umstände Web-basierter Präsentation im wesentlichen *nicht* geändert haben: Zwar vermag diese Technik durchaus „Bewegung" in den ansonsten eher statischen Gesamteindruck Internet-basierter „Präsentationen" auf HTML-Basis[2] zu bringen – der eigentliche Rezeptions-Kontext ist jedoch in der Regel nach wie vor derselbe. Lediglich, wenn die gesamte Präsentation im Rahmen einer einzelnen Animationsdatei[3] realisiert und dargestellt wird, kann der Eindruck einer konsequent *linearen* Präsentationsstruktur wiederhergestellt werden. Dies jedoch widerspricht nun einerseits (wie bereits in [3.2.1] diskutiert) wiederum dem Grundprinzip des Hypertext-Modells, in dessen Rahmen die Animationsdatei durch den HTML-Kontext ja eingebettet ist, als auch die derzeitige Spezifikation beispielsweise animierter GIFs derartige Lösungen allein aus technischer Hinsicht[4] erschwert [vgl. SeLo97].

Darüber hinausgehende Funktionen wie etwa die Synchronisation derart realisierter Animationen mit Audio-Material oder gar weiteren Multimedia-Daten (welche, etwa zeitversetzt, nach Ablauf des Bewegungsvorganges etc. entsprechend gesteuert werden könnten) sind im Rahmen der derzeitigen Implementierungen jedoch *nicht* möglich oder aber stets mit unverhältnismäßig großem Aufwand (etwa mittels dHTML oder aufwendigen JavaScript-Routinen) verbunden.[5] Auch entsprechend zeitabhängige Interaktion (die etwa Hyperlink-Verweise mit dem Fortschritt der jeweiligen Animation verknüpfen könnte) ist weder im Rahmen der eben angesprochenen, „schlichten" Animationsfunktionalität noch des freilich hierfür nicht ausgelegten Umgebungs-Frameworks HTML/JavaScript vorgesehen. Dies begründet sich naturgemäß in dem HTML zugrunde liegenden, theoretischen *Modell*, welches sich – trotz der Animationseigenschaft – nicht gewandelt hat: Obgleich sich einzelne HTML-Komponenten durch die Einführung animierter Bildelemente um einen temporalen Faktor erweitern lassen, bleibt das Prinzip durch Hyperlinks verknüpfter, prinzipiell jedoch statischer Seiten dasselbe. Da die gestalterischen Möglichkeiten sich somit immer noch im Rahmen des *Dexter*-Modells[6] [vgl. HaSc94] bewegen (müssen), ist somit der, „zeitliche Faktor", den HTML über die Animations-Funktionalität scheinbar anbietet, nur ein *gedachter*, da „spezielle Präsentations- und Interaktionsmöglichkeiten" auch in diesem Rahmen noch „weitgehend unberücksichtig" [Bole98] bleiben:

> *[Those] hypermedia hypermedia models and systems do not incorporate time explicitly. This prevents authors from having direct control over the temporal aspects of a presentation.*
>
> [Hard99]

[1] Anm: So ermöglicht der zusätzliche Alphakanal MNGs etwa elegante *Fades* und begünstigt überdies das dem ästhetischen Erscheinungsbild meist zuträgliche Anti-Aliasing der Bildelemente.

[2] Amn: HTML stellt zumindest bis dahin ja zumindest prinzipiell eher durch Hyperlinks nur *optional* verknüpfte, Informationsorientierte Datenstrukturen denn dramaturgisch planbare Präsentations-Einheiten dar [s.3.2.1-2].

[3] Anm: Ob nun auf GIF- oder MNG-Basis ist hierbei unter diesem Aspekt gleichgültig.

[4] „Ihre Dateigröße würde alle vernünftigen Grenzen sprengen." [SeLo97]

[5] Anm: Die eben angesprochenen Problematiken werden nun beispielsweise durch den SMIL-Standard [SMIL98] gelöst, s.5.5

[6] vgl. 2.2

Dies wiederum hat zur Konsequenz, dass Animationen auf Basis von Animated GIF und MNG derzeit lediglich zu *dekorativen* Zwecken weitgehend statischer Web-Sites, etwa in Form „nerviger Werbe-Wackelbilder und herzlich überflüssiger Blink-Buttons" [Chro02] zum Einsatz kommen, statt etwa dem weitaus interessanten Zweck der *Veranschaulichung* [vgl. This00:96] zu dienen,[1] oder aber – was schließlich die Erweiterung und logische Fortsetzung dieser Funktion entspricht – zur Erstellung (und Darstellung) ganzheitlicher Präsentationen angewandt zu werden.

Aufgrund der hierbei deutlich werdenden, den Möglichkeiten[2] bislang unangemessenen Nutzung [vgl. Chro02] sowie der soeben angesprochenen, mitunter unzureichenden Funktionalität und Synchronisation der Animationen bleibt daher festzustellen, dass die Möglichkeiten statischen HTMLs auch trotz der zuvor dargelegten Erweiterungen [s.3.2-4] im Hinblick auf die Anforderungen, die insbesondere unter dem Aspekt der *Multimedialität* [nach Stei99] an (auch Web-basierte) Präsentationen gestellt werden, bei weitem nicht ausreichen. Zu berücksichtigen ist an dieser Stelle überdies die bereits in [3.1-2] diskutierte Erkenntnis, dass das WWW an sich aufgrund seiner strukturellen Auslegung prinzipiell zu Präsentationszwecken nur sehr bedingt geeignet ist: So bleiben trotz der erheblichen, „heiß umkämpften" [Clon00] Erweiterungen die Gestaltungsmöglichkeiten, die HTML insbesondere hinsichtlich Typografie [vgl. MaKo02:363, Rieh01:55f, Stei99:443f], *Spatial Flexibility*[3] und Animation [s.3.4] bietet, immer noch weit hinter den Forderungen der Web-Designer nach einer Annäherung an die Funktionalität *klassischer* Multimedia-Anwendungen [s.2.1] zurück.

Aufgrund dessen rücken an dieser Stelle weitergehende Lösungen in den Mittelpunkt unseres Interesses, welche über die (wie soeben diskutiert) zweifellos enttäuschenden Präsentationseigenschaften des HTML-Umfelds hinausgehende Möglichkeiten eröffnen sollen. Insbesondere sind daher für unsere Interessen natürlich im Folgenden alternative Ansätze und Erweiterungen relevant, die sowohl dem Medium Internet[4] entsprechende Lösungen anbieten, als auch bezüglich ihrer grundsätzlichen Architektur und Formatstruktur prinzipiell weitergehenden, multimedialen Möglichkeiten sowie dem „Präsentationsgedanken" [s.Kap.1] an sich eher Rechnung tragen, als dies im Rahmen des Dokumenten-orientierten WWW-Modells derzeit der Fall ist.

3.5.1 Video-Streaming

Ein Ansatz, der etwa an der Übertragungstechnik des World Wide Web selber ansetzt, ist das Konzept des *Streaming*: Aufgrund der paketorientierten Netzerk-Übertragungtechnik des Internet mit teils problematischer Konstanz und Durchsatzeigenschaften erscheint es natürlich insbesondere im Hinblick auf ein möglichst „gleichmäßiges",[5] angenehmes Rezeptions-Erlebnis im Rahmen Internet-basierter Präsentation von Interesse, übertragene (Präsentations-)Inhalte bereits (zumindest teilweise) darzustellen, selbst wenn der Übertragungsprozess der entsprechenden Daten noch nicht vollständig abgeschlossen ist. Die allgemein gefasste Definition des Streaming-Prinzips bezieht sich daher zunächst vorrangig auf den Nutzer, also den

[1] „Grundsätzlich haben Bilder in Multimedia-Produkten drei Funktionen: Veranschaulichung, Strukturierung, Dekoration. Diese Funktionen lassen sich nur schwer vermischen…" [This00] p.96

[2] Hierbei sind insbesondere Bildschirm-Demonstrationen, wie sie etwa in [Chro02] erläutert werden, unter dem Aspekt der Veranschaulichung interessant.

[3] Dieser Ausdruck bezieht sich auf die Flexibilität, die HTML im Hinblick auf sehr unterschiedliche, mögliche Browser-Fenstergrößen und –Typen erfordert, verbunden mit zugleich eingeschränkten Kontrollmöglichkeiten bezüglich eines gewünschten (aber unerreichten) „pixel-genauen Webdesigns".

[4] Anm: Hier wird zwar auf das Internet allgemein Bezug genommen (insbesondere in Übertragungstechnischer Hinsicht) – es spielen jedoch durchaus auch darauf aufbauende WWW-spezifische Aspekte (wie etwa die Durchsuchbarkeit/Indizierung durch Suchmaschinen etc.) eine Rolle.

[5] Das englische Wort „smooth" wäre wiederum an dieser Stelle passender – es findet sich jedoch bedauerlicherweise keine äquivalente, deutsche Übersetzung, die hier ebenso geeignet wäre.

Client des Übertragungsprozesses: Obgleich (noch) von nicht allen Web-Browsern unterstützt,[1] sieht diese Herangehensweise eine kontinuierlich fortschreitende Darstellung der Inhalte vor, noch bevor sämtliche Daten übertragen sind. Anschaulich erfüllt bespielsweise die *progressive* Darstellung von Bilddaten (wie derzeit etwa von GIF, JPEG, PNG und JPEG2000 unterstützt) dieses Kriterium, da diese aufgrund ihrer erheblichen Größe übertragungstechnisch naturgemäß erheblich problematischer erscheinen als vergleichbare Textinhalte.

Weitaus bedeutender wird dieses Prinzip freilich bei der Übertragung von Audio- und speziell Video-Daten, da die erheblichen Datenmengen an dieser Stelle insbesondere die Fähigkeiten des diesbezüglich „nicht sonderlich intelligenten" [Blei96:11] HTTP-Protokolls übersteigen:

> *'Streaming' video is unique because playback begins as soon as sufficient content is downloaded into the player's buffer (memory)…*
>
> [Guhl01]

Neben verstärkter Kompression der Daten ähnlich der bereits in [3.3.1] beschriebenen Verfahren[2] konzentrieren sich die Streaming-Bemühungen somit auf die *Übertragung* der Daten selber. Somit definiert sich „Streaming" (im engeren Sinne) durch eine (im Gegensatz zum quittungsbasierten HTTP -Verfahren auf Basis von TCP) quittungslose, ergo (nahezu) unidirektionale Übertragung,[3] in deren Rahmen auch einzelne Paketverluste kompensiert werden können, um eine möglichst „flüssige" Darstellung auf der Client-Seite zu erhalten.

3.5.1.1 Streaming-Formate

Die mit der Streaming-Thematik verbundenen Probleme und Technologien an dieser Stelle näher zu beschreiben, würde im Hinblick auf die Zielsetzung dieser Diplomarbeit jedoch sicherlich den zu weit führen – aufgrund dessen soll an dieser Stelle lediglich kurz zusammengefasst werden, dass sich im Rahmen der Streaming-Technologie in der Hauptsache drei konkurrierende Systeme am Markt durchsetzen konnten: Der *RealMedia*-Ansatz der Firma RealNetworks, die *Windows Media*-Technologie von Microsoft sowie Apples *QuickTime*. Insbesondere letztgenanntes Framework macht jedoch deutlich, dass es sich bei sämtlichen, kommerziell angebotenen Streaming-Lösungen selten „nur" um die eigentliche Übertragungstechnik, als vielmehr um eine Verknüpfung eigens entwickelter Übertragungsprotokolle mit durchweg proprietären Bildkompressions-Verfahren in Form so genannter „ganzheitlicher" *Streaming-Suites* handelt. So fußt etwa Apples „Quicktime Streaming-Server" lediglich auf dem standardisierten, jedoch aufgrund allgegenwärtiger Firewall-Beschränkungen selten anwendbaren RealTime-Streamingprotokoll (RTSP). Dies macht freilich die Konzentration dieses Ansatzes auf das qualitäts-orientierte *Encoding* des QuickTime-Formates,[4] welches (naturgemäß wiederum aufgrund der „Unerfüllbarkeit" der ausschliesslichen UDP-Verfügbarkeit[5] von RTSP) im Web zumeist auf Basis des betagten HTTP-Protokolls vorliegt, mehr als offensichtlich. Auch im Rahmen dieser Diplomarbeit erscheint daher die Betrachtung der jeweiligen *Formate* natürlich weitaus interessanter als diesbezügliche Übertragungs-Protokolle [vgl. hierzu Kres95, Schm99 sowie Muel00a] – schließlich entscheiden erstlinig die jeweiligen Funktionalitäten der Formate über die entsprechenden Gestaltungsmöglichkeiten im Rahmen Streaming-basierter Internet-Präsentationen.

---

[1] So sind an dieser Stelle etwa Microsofts Internet-Explorer oder (insbes. die 4er-Generation von) Netscape zu nennen, welche es (im Gegensatz bspw. zu Opera) mitunter Vorziehen, eine Webseite erst am Schluss des Download-Prozesses „auf einen Schlag" darzustellen und auch LowSource-Referenzen nicht unterstützen.

[2] Wobei an dieser Stelle freilich noch Frame-*Übergreifende* Verfahren (*Inter-Frame-Coding*, vgl. [Wsg00]) zum Tragen kommen

[3] Anm: Optimalerweise fußt Streaming somit auf dem quittungslosen UDP-Protokoll

[4] Anm: Dieses basiert wiederum i.d.R. auf dem so genannten *Sorensen* Codec.

[5] Anm: UDP in diesem Falle als quittungsloses Übertragungsverfahren, im Gegensatz zum stets „Feedback" erfordernden TCP.

3.5.1.2 Streamed Presentations?

Und gerade unter diesem Aspekt muten zumindest die grundlegenden Spezifikationen der einzelnen Streaming-Formate zunächst ein wenig enttäuschend an: So konzentrieren sich sämtliche Formate in ihrer ursprünglichen Fassung ausschließlich auf die Kompression Frame-basierter Filmsequenzen und schränken deren Anwendbarkeit im Hinblick auf (durchaus auch nicht-filmische), ggf. interaktive Präsentationen somit deutlich ein. Lediglich das der konventionellen Video/Multimedia-Technik entstammende [vgl. Kres95:41, Pieg95:101, Schm99:42ff] und somit *nicht* primär Internetorientierte QuickTime-Format erlaubt grundsätzlich die Definition bewegter „Sprites", sowie die Integration separater Audio- Text- und Bildinformationen; Die Format-Angaben der Microsoft- und Real-Formate verraten hingegen in ihren ursprünglichen Spezifikationen nichts über die Unterstützung Audio/Video-fremder Informationen.

Zwar lassen sich unter Anwendung dieser Technik sämtliche Art von Text- und Bildinformationen ähnlich der Montage eines konventionellen Films integrieren – da beispielsweise Text auf diese Weise jedoch in *diskretisierter* Form, d.h. in Form seiner Pixel-Informationen anstatt als Text selber übertragen wird, würde – ähnlich wie bereits bei Verwendung von im GIF-Format aufgerasterten Textelementen im HTML-Format [vgl. Rieh01:55] – an dieser Stelle jedoch sowohl Übertragungskapazität „missbraucht", als auch die semantische Information des Textes selber durch Internet-Suchmaschinen unauffindbar werden [vgl. Muel00a]. Eine derartige Realisierung heterogen zusammengesetzter[1] Präsentationen erscheint somit wenig sinnvoll, da das Ergebnis weder in Inhalt oder Form der ursprünglich beabsichtigten Präsentation, noch dem Medium Internet an sich gerecht würde [s.3.5]. Zudem, so merkt SMIL-„Erfinder" Dick Bulterman in einem interessanten Statement an,[2] steht dem verhältnismäßig „leichten" Verfassen einer textbasierten Präsentationen (etwa im Rahmen des ungeliebten PowerPoint)[3] ein ungeheurer Aufwand bei der Produktion Video-basierter Präsentationen gegenüber:

> *The point is, of course, that the main reason that multimedia content is not streaming across the Web is because this content is very, very difficult to make. Part of the problem may be in indexing or transmission, but these technical issues pale in comparison to the creative effort of putting together even the simplest audio recording. We are all too spoiled by the syntactic/production quality of TV to easily accept the poor production quality of home-made presentations.*
>
> [Bult01]

3.5.1.3 Mehr als nur Video

3.5.1.3.1 Rudimentäre Erweiterungen

Diese Ansicht legt schließlich auch eine engere, native Integration separater Text- und Bildelemente in die multimedialen Streaming-Formate nahe: Neben dem bereits angesprochenen *QuickTime*, das über einfache Bild- und Textelemente hinaus auch die Integration komplexer Flash-Animationen [s.4.5] ermöglicht, stellen auch die Frameworks der Konkurrenten Real und Microsoft die unter anderem von Bulterman geforderten „*rich hyperlinks*" im Rahmen ihrer Streaming-Formate bereit: So erlaubt etwa die *RMEvents*-Funktionalität des späteren RealProducers[4] das Auslösen so genannter „Ereignisse": Zwar ließ sich im Rahmen des Erstellungsprozesses auch weiterhin keinerlei semantische Information in die Streaming-Datei einpflegen, doch mithilfe separater Metadateien, die später durch ein Kommandozeilen-Tool mit der eigentlichen Streaming-Datei „verschmolzen" werden konnten, stellte Real schließlich sowohl automatische „Triggers", als auch so genannte „Hotspots" zur Verfügung. In Form eines „Triggers" ließen sich etwa TimeCo-

[1] Anm: Hiermit ist die Zusammensetzung einer Präsentation aus Textelementen, Pixel- und Vektorbildern, Bild, Videos etc. gemeint.
[2] vgl. [Bult01]
[3] s.3.2
[4] Anm: Lediglich Versionen 7-9.

de-gesteuerte Verweise hinterlegen, die dann bei Erreichen des entsprechenden Zeitpunktes während der Streaming-Wiedergabe automatisch die spezifizierte Internet-URL in ein Browserfenster lud:

```
u 00:00:10.0 00:00:59.9 http://meine.seite.ms
```

Listing 3.5.1: Hinterlegen eines Triggers *zum automatischen Öffnen einer URL in externer* events.txt*-Datei*

Im Gegensatz hierzu stellten die so genannten „Hotspots" hingegen lediglich optional klickbare Verweise im Sinne eines herkömmlichen Hyperlinks dar. Auch die *Real*-Terminologie („Image Map") macht diese Analogie zum HTML-Standard deutlich: So schreibt etwa die Producer-Spezifikation das Erstellen der (wiederum extern abgelegten) *ImageMap* in einer dem entsprechenden HTML-Pendant enlehnten Syntax vor:

```
duration=0:0:0:40:0
<map start=0:0:0:0:0 end=0:0:0:5:20 coords=0,0,100,100>
   <area shape=circle coords=50,50,10 url="http://meine.seite.ms" alt="home page">
</map>
```

Listing 3.5.2: Hinterlegen einer Image Map zum optionalen Anklicken einer Url in externer image.txt*-datei.*

Auch im Rahmen von Microsofts Windows Media Format SDK ist neben der reinen Übertragung von Audio/Video-Daten das Einpflegen so genannter „Arbitary Streams" möglich, mithilfe derer sich parallel zum herkömmlichen „Bildstrom" zusätzliche Kommandos hinterlegen lassen, und die zugleich den RMEvents-ähnlichen, veralteten „ASF-Index" [vgl. Cove99] der früheren WindowsMedia-Generationen ersetzten.

Neben den so genannten „Web-" und „Script Streams", die – ähnlich wie bereits die *RMEvents* des Real-Konkurrenten – Verweise auf externe URLS ermöglichen, stehen darüber hinaus jedoch noch weitere Möglichkeiten zur Verfügung, wie etwa das Eröffnen eines Dateitransfers („File Transfer Stream") oder beliebige Skript-Kommandos mittels „Custom Arbitary Streams" [vgl. Micr02]. Die recht simpel zu realisierenden „Text-", und „Image Streams" erlauben überdies die Erstellung einfacher „Slide-Shows" und Präsentationen, die sich wiederum mit zusätzlichem Videomaterial anreichern lassen.

3.5.1.3.2 Multimedia-Integration durch SMIL?

Obwohl Microsoft diese Funktionalität mit der Veröffentlichung der neunten Generation der Windows-Media-Serie im vergangenen Jahr sogar noch deutlich ausgebaut hat, deutet sich jedoch bereits heute an, dass das „Arbitary Stream"-Modell bereits im Rahmen der nächsten Version durch die zweite Generation des SMIL-Standards (Codename: „SMIL Boston") ersetzt werden könnte: Die erste, bereits seit längerem standardisierte Fassung dieses Synchronisationsmodells [SMIL98] hat an dieser Stelle schon seit geraumer Zeit das bereits diskutierte RMEvents-Modell des Konkurrenten *Real* ersetzt, da im Gegensatz zu der relativ rudimentären Synchronisation und umständlichen Erstellung der RMEvent-„Triggers" und „ImageMaps" ein recht einfaches Synchronisations- und Präsentations-Framework die mittels HTML-ähnlicher und überdies XML-konformer Syntax überaus komfortable Bearbeitung multimedialer Präsentationen erlaubt. Obgleich auch Apple bereits im Rahmen der QuickTime 4.1-Veröffentlichung die SMIL 1.0-Spezifikation vollständig implementiert, weigerte sich die Microsoft-Konzernführung über längere Zeit hinweg [vgl. Piem98], den in ihren Augen „zu simplen" [vgl. Arci02][1] Standard auch in den Funktionsumfang des MediaPlayers oder Internet Explorers aufzunehmen:

> *Microsoft […] viewed the specification as a limited, proprietary response to the challenge of synchronizing time-based Web content and interactions.*
>
> [Cove99]

[1] "Lack of transitions is an example of SMIL 1.0's over-simplification, a problem corrected technically in version 2.0, but with serious image consequences for SMIL in the mind of many creative professionals" [Arci02]

Erst im Rahmen der neueren WindowsMedia-Frameworks schlägt sich die SMIL-Mitarbeit[1] des Software-Riesen endlich darin nieder, einen Großteil der in SMIL 2.0 [SMIL01] vorgesehenen Funktionalität in das Microsofts Windows Media Format SDK [vgl. Micr02] und somit auch in den Funktionsumfang des Windows Media Player zu integrieren.

3.5.2 Zeit-orientierte Hypertext-Erweiterungen: HTML+TIME und HyTime

Bei der SMIL-ähnlichen Synchronisation *Browser*-basierter Multimedia-Präsentationen jedoch beschritt der Redmonder Softwarekonzern abermals einen anderen, recht eigenwilligen Weg: Statt, wie allgemein erhofft und erwartet, das (wie bereits in [3.2] diskutiert) bezüglich Web-basierter Internet-Präsentationen nur „begrenzt geeignete" HTML-Format durch den hierfür eigens ausgelegten SMIL-Standard zu ersetzen, befanden die Microsoft-Entwickler zunächst, entsprechende dHTML-Erweiterungen seien hierfür bereits vollständig ausreichend:

> *Die Einstellung von Microsoft lässt sich etwa so zusammenfassen: CSS (Casacading Style Sheeds), HTML 4, Dynamic HTML und DOM (Document Object Model) reichen als Standards bereits aus, um Multimedia-Präsentationen in Web-Seiten zu integerieren. SMIL wird nicht gebraucht.*
>
> [Piem98:81]

Neben Anwendung des Microsoft-eigenen ASX-Formates zum Video-Streaming [vgl. Phil98:30] reagierten die Redmonder stattdessen zunächst mit einer deutlichen *Erweiterung* der mächtigen, jedoch „aufwendig zu realisierenden" [Graf02] dHTML-Funktionalität im Rahmen des Microsoft Internet Explorer. Als ebendies aufgrund der Inkompatibilität [vgl. ChNg02, Graf02] zum ebenfalls „Dynamic HTML" propagierenden *Netscape*-Browser und der hiermit oftmals verbundenen Mehrfach-Entwicklungen jedoch zu der bereits in [3.2.3.1] beleuchteten Entrüstung und Frustration großer Teile der Web-Entwicklerschaft führte, entschloss sich Microsoft schließlich, Teile des „SMIL Boston"-Modells [vgl. Cove99, SMIL01] in den Funktionsumfang des Internet Explorers mit aufzunehmen – dies jedoch wiederum freilich nicht dem Wortlaut des Standards entsprechend, sondern im Zusammenhang mit einem ganz eigenen Verständnis des Synchronisationsmodells. So wurde auf eine *direkte* Umsetzung des schlichten SMIL-Frameworks zugunsten der Implementierung eines eigens entwickelten „Kompromisses" verzichtet: Dem HTML+TIME-Modell [SYS98] – einer Umsetzung des ebenfalls von Microsoft-Entwickler Patrick Schmitz verfassten HTML+SMIL-„Sprachprofils" [Schm00]. Dieser den früheren *Drafts* der SMIL 2.0-Spezifikation beigefügte Ansatz sah statt einer vollständigen Ersetzung des veralteten HTML-Standards für Multimedia-Präsentationen durch die hierfür eher geeignete SMIL-Syntax vielmehr eine *Erweiterung* von HTML hinsichtlich SMIL-basierter Synchronisationsfunktionalität vor:

> *The HTML+SMIL Language profile […] specifies that HTML is used for layout in the HTML+SMIL profile rather than using the separate SMIL Language.*
>
> [Newm00]

Obgleich unsere Betrachtungen in [3.2] gezeigt haben, dass sich HTML mitnichten „für Layout-Funktionen hervorragend eignet"[2] - eine Ansicht, die Microsoft diesem Ansatz ja offenbar zugrunde legt – sondern dem hierfür zunächst „verschmähten" SMIL-Standard diesem Aspekt sogar erheblich nachsteht, ermöglicht das HTML+TIME-Modell die zeitlich gesteuerte Kontrolle über einzelne Elemente der HTML-Seite. Der Unterschied zum dHTML-Ansatz besteht hierbei darin, dass die Ansteuerung der Animation nicht mehr (wie in dHTML) *Skript*gesteuert,[3] sondern mittels SMIL-konformer Synchronisations-Syntax möglich wird: So können einzelne HTML -Elemente entsprechend dem SMIL-Modell nun zeitgesteuert parallel, sequentiell oder aufgrund entsprechender Benutzer-Interaktionen animiert werden:

---

[1] "Microsoft was involved during development, but in the end chose not to support SMIL 1.0…" [Cove99]

[2] vgl. [Newm00]

[3] In dHTML wird dies etwa mittels JavaScript/JScript oder VBScript realisiert.

```
<html xmlns:t ="urn:schemas-microsoft-com:time">
   <head>
      <style>.time { behavior: url(#default#time2) }</style>
      <?IMPORT namespace="t" implementation="#default#time2">
   </head>
   <body>
      <t:seq repeatCount="indefinite">
         <t:par>
            <img class="time" dur="3" timeAction="display" src="mountain.jpg"
                 style="height=150; width=300" />
            <div class="time" dur="3" timeAction="display">Olympic Mountains</div>
         </t:par>
      </t:seq>
      ...
   </body>
</html>
```

Listing 3.5.2.1: Animieren einzelner HTML-Elemente im Rahmen des HTML+TIME*-Frameworks*

Da dieser Ansatz jedoch insbesondere aufgrund der fragwürdigen Eignung HTMLs als „Layout-Trägerformat" in meinen Augen einen relativ „unsauberen" Kompromiss darstellt und überdies im Vergleich zu den direkten Umsetzungen des SMIL-Standards weitaus komplexer zu realisieren ist, erstaunt es daher nur wenig, dass sich bisher relativ wenige Multimedia-Präsentationen finden, die auf die HTML+TIME-Funktionalität (welche seit Einführung des Internet Explorer 5.5 in einer durchaus beachtlichen Anzahl von WWW-Endgeräten ablauffähig ist) zurückgreifen. Auch die Tatsache, dass das World Wide Web-Konsortium die sogar als W3C-*Note* eingereichte Spezifikation [SYS98] im weiteren Verlauf der SMIL-Entwicklungsarbeit völlig ignoriert hat, vermag unter diesem Eindruck nur wenig zu überraschen:

> *Microsoft [...] threw all of its energies into HTML+TIME, but it never made it to recommendation status.*
> [Cove99]

Das voraussehbare Scheitern des HTML+TIME-Ansatzes weist an dieser Stelle übrigens eine interessante Parallele zum Misserfolg einer ähnlichen, jedoch weitaus frühzeitigeren Initiative zur Erweiterung des Hypertext-Prinzips um zeitlich gesteuerte Animations-Funktionalität auf: Dem noch auf SGML fußenden *HyTime*-Standard [Gold97]. Dieses modulare Framework konnte sich gerade aufgrund seiner „Mächtigkeit […] und insbesondere auch der damit verbundenen Komplexität" [Bole98] nicht durchsetzen, da es der breiten Masse der potentiellen Nutzer schlicht und ergreifend „unzugänglich" [1] erschien: Aufgrund dessen wird die hochkomplexe DTD nicht nur von „praktisch keinem Vertreter der privaten Industrie in irgend einer Form verwendet" [ROHB98] – auch ist „bis heute keine kommerzielle HyTime-Engine verfügbar, die die ganze Funktionalität dieses Standards beherrscht".[2]

Der HTML+TIME-Ansatz ist indes aufgrund der Internet Explorer-Implementierung zwar theoretisch direkt anwendbar – aufgrund der recht mühsamen und (wegen der nahtlosen Verquickung mit dem ohnehin diesbezüglich problematischen[3] „Träger-Medium" HTML) tendenziell „unsauber" wirkenden Realisierung sowie insbesondere derzeit fehlender (für eine potentielle Verbreitung jedoch unabdingbarer) visueller *Authoring-Tools* [vgl. Cove99] macht dieses „Kompromiss-Modell" derzeit jedoch im Hinblick auf die komfortable Erstellung Web-fähiger Präsentationen einen wenig überzeugenden Eindruck. Eine direkte Anwendung der primär auf Internet-basierter *Präsentation* gemünzten SMIL-Syntax [s.5.5] erschiene unter diesem Aspekt daher deutlich „sauberer" und sinnvoller, da erhebliche HTML-„Altlasten" [s.3.2] dem Erstellungskomfort der Präsentationsdateien doch deutlich entgegenstehen und den HTML+TIME-Ansatz somit als denkbare Alternative für Internet-basierte Präsentationen wenig attraktiv erscheinen lassen.

---

[1] vgl. [ROHB98] p.191
[2] vgl. [Bole98] Kap.20.2.6
[3] vgl. hierzi die entsprechende Diskussion in [3.2]

3.5.3 Multimedia mit Java-Applets

Eine gänzlich andere Herangehensweise an die Problematik, dem statischen, dokumentenbasierten „HTML-Web“ multimediales Leben einzuhauchen, stellt hingegen der *Applet*-Ansatz der plattformunabhängigen Programmiersprache Java dar: Die von Sun Microsystems 1994 aus der Taufe gehobene[1] *Java*-Technologie vereint die Mächtigkeit einer objektorientierten Sprachumgebung mit einer sehr einfach zu verstehenden Syntax. Da die Ausführungsschicht der Programmiersprache jedoch nicht auf einer realen, sondern stets einer so genannten „virtuellen Maschine“ fußt, stellt das Java-Framework nicht nur eine *plattformübergreifende* (da vom jeweiligen Betriebssystem unabhängige), sondern eine ebenso „sichere“ Programmiersprache da, da der jeweils zur Laufzeit kompilierte Bytecode[2] stets innerhalb eines so genannten „Sandkastens“ ausgeführt wird und im Gegensatz zu C++ nicht auf diskrete Speicherbereiche direkten Zugriff erhält. Dies führt zwar einerseits dazu, dass Java-Programme in der Regel erheblich langsamer als entsprechende Lösungen etwa in C++ oder Pascal sind, jedoch zugleich nur über wenige „Absturz-Möglichkeiten“ verfügen.

Java-Applets stellen nun wiederum speziell auf den Einsatz im WWW gemünzte Java-Programme dar, welche auf Basis einer im Web-Browser integrierten Ausführungsschicht, der *Java Virtual Machine* (JVM) direkt im Internet ablauffähig sind. Auf dieser Grundlage lässt sich nun beliebiger Code auf dem Client-Rechner ausführen – da all dies jedoch im Rahmen des bereits erwähnten „Sandkasten-Modells“ vonstatten geht, gilt die Anwendung von Java-Applets aufgrund restriktiver Sicherheits-Einschränkungen der JVM trotz mitunter auftretender „Virus-Schlupflöcher“[3] [vgl. Oaks01] auch aufgrund verschiedener Zertifizierungs-Möglichkeiten[4] als relativ sicher. Bei der Erstellung Multimedialer Präsentationen sind an dieser Stelle jedoch keinerlei Grenzen gesetzt: Neben bereits in Fülle vorhandenen Grafik- und GUI-Funktionen, die bereits nativ integriert oder aber komfortabel über das Internet erhältlich sind, lassen sich erweiterte Möglichkeiten (etwa eine Rendering-Engine für geglättete (*Anti-Alias-*)Schrift, einen Slide-Generator o.ä.) beliebig und ohne jegliche Einschränkungen selber realisieren, was aufgrund der relativen Einfachheit der Java-Syntax selber auch ohne größere Probleme möglich ist. Da eine detailliertere Beschreibung möglicher Realisierungen im Rahmen von Java-Applets an dieser Stelle jedoch zu weit führen würde, gilt für mich im Rahmen meiner Diplomarbeit ebenso das, was bereits [Wild99] treffend ausführt:

> *Although Java evidently is one of the main spin-offs of the web, it is not covered in detail, since I feel that a detailed description of a programming language would not fit into the rest of the book, which is more focused on the description of architectural concepts.*[5]

3.5.3.1 Java: Selber programmieren macht fett

Im Hinblick auf die (im wahrsten Sinne des Wortes unbegrenzten) Möglichkeiten, die Javas Applet-Technik hinsichtlich der Erstellung Web-basierter Präsentationen eröffnet, sei daher an dieser Stelle lediglich auf die bereits in großer Zahl bestehenden Frameworks hingewiesen, auf die bei der Gestaltung und Realisierung Java-basierter Applet-Anwendungen zurückgegriffen werden kann, insbesondere den von Sun Microsystems selbst bereitgestellten Java Media APIs [vgl. Terra02]. Hierbei erscheinen im Rahmen grafischer Gestaltungsmöglichkeiten insbesondere die Photoshop-ähnliche Bildbearbeitungs-Funktionen erlaubende *Java Advanced Imaging*-API [JAI99] sowie das auf Multimedia- und Streaming-Anwendungen zugeschnittene Java Media Framework (JMF) von Interesse [vgl. Kaub00], welche bereits an sich die Realisierung beeindruckender Präsentationen ermöglichen. In Verbindung mit darüber hinaus im Internet verfügbaren, weiteren Code-Komponenten und sogar ganzen Frameworks zur komfortablen Umsetzung kompletter Prä-

---

[1] vgl. [YLQ98]

[2] Anm: Eigentlich „interpretierte“ – da Java an sich keine kompilierende, sondern eine *interpretierende* Programmiersprache darstellt.

[3] Anm: Diese *Viren* sind jedoch zumeist eher auf Sicherheitslücken der entsprechenden JVM-Implementierungen zurückzuführen

[4] vgl. [Oaks01]

[5] vgl. [Wild99] p.VII

sentationen [vgl. Baum98:125] stehen somit nahezu unbegrenzte Möglichkeiten zur Präsentationserstellung zur Verfügung.

Die Problematik, die sich trotz der ernormen Fülle an Möglichkeiten bei einer Applet-basierten Präsentationserstellung ergibt, besteht freilich einerseits in dem (etwa im Vergleich zum trivialen PowerPoint-Erstellungsprozess) unverhältnismäßig hohen Aufwand, der mit der Programmierung der Präsentationslogik (oder aber der entsprechenden Anpassung bei der Verwendung externer Komponenten) verbunden ist, sowie die vergleichsweise enormen Anforderungen an Programmierkenntnisse und IT-Kompetenz, die bei Realisierung der entsprechenden Präsentation an den Ersteller gestellt werden. Da überdies auch die prinzipiell beeindruckenden Ergebnisse, welche sich bei manueller Realisierung einer Applet-Präsentation erzielen lassen, in keinem Verhältnis zu dem damit verbundenen Zeitaufwand stehen, stellt zumindest die *direkte* Verwendung der Java-Technologie unter dem Aspekt der Neu-Erstellung („*from scratch*“) freilich keine ernsthafte Alternative zu etablierter Präsentationssoftware dar.

3.5.3.2 Automatische Applet-Generation: Die Java-Fabriken

An dieser Stelle kommen jedoch „ganzheitliche“, Java-basierte Präsentationsansätze ins Spiel, d.h. Präsentationslösungen, die zwar die Erstellung multimedialer Präsentationen im Rahmen eines komfortablen, intuitiven GUIs erlauben, zum „Deployment“ der eigentlichen Präsentation jedoch auf die Java-Technik zurückgreifen. Konkret bedeutet dies zumeist die Konvertierung der über die grafische Oberfläche determinierten Präsentationsdaten in direkt ausführbaren Java-Bytecode. Neben verschiedenen hilfreichen Frameworks, die lediglich die Generierung des Codes vereinfachen, wie etwa das (mittlerweile eingegangene) „Java Presentation Framework“ [vgl. Scar98], sind dies insbesondere die bereits in [2.3.4] beleuchteten, durchaus beeindruckenden Lösungen *Impatica* [Impa02] sowie die sogar Echtzeit-Streaming unterstützende Präsentationssoftware *Oplayo*, deren jeweils verschiedenen Zielgruppen angepassten Authoring-Tools „Media Designer“ [Hans02] und „Composer“ [vgl. Muel02b] die intuitive Erstellung überaus beeindruckender Präsentationen ermöglichen. Stillschweigend „verflossen“ sind hingegen bereits die ebenfalls auf Java basierenden Multimedia-Lösungen der (daraufhin in Konkurs gegangenen) Firmen *RandomNoise* mit „Coda“ [Bers97, Wagn98] und *Web Knight*'s „Instant Coffee“ [Hess97, Perr97], sowie „Jamba“ [Shah96, Para97] von *AimTech* [vgl. Kurz97]. Bedauerlicherweise hat sich auch die Herstellerfirma des letztgenannten, vom konzeptionellen Standpunkt durchaus faszinierenden Java-Multimedia-Produktes [vgl. Abb. 3.5.3.1] in dieser Form mittlerweile aufgelöst.

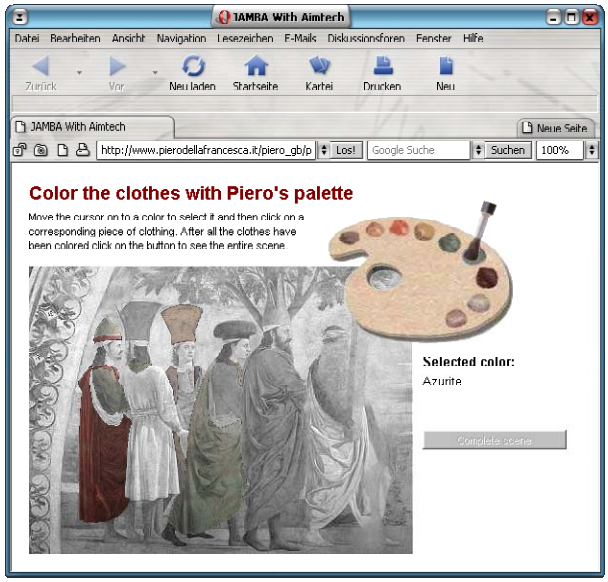


Abb. 3.5.3.1: Java-basiertes Multimedia leicht gemacht, dank AimTech's Jamba

Die einzige derartige Lösung, die (neben den derzeit aktuellen Ansätzen *Impatica* und *Oplayo*)[1] auch heute noch zumindest der Form halber weiterhin existiert, ist die *Barista*-Technik des kanadischen Grafik-Giganten Corel [Core98]. Aufgrund der Tatsache, dass diese Java-Lösung, die sich hingegen vornehmlich auf statische (wenn auch interaktive), Dokumenten-basierte Präsentationen konzentriert, bereits seit gut fünf Jahren nicht mehr aktualisiert wird und überdies trotz dieser langen „Wartezeit“ noch erhebliche „Macken“ aufweist [vgl. Gutz97], muss jedoch davon ausgegangen werden, dass sich die diesbezüglich pessimistische Prognose Ri-

[1] s. hierzu 2.3.4

chard Wagners,[1] der bereits die Einführung der „Coda"-Technologie 1998 als „todgeweihten" Ansatz bezeichnet hatte, leider durchaus zu bestätigen scheint:

> *While a Java-only solution may offer some capabilities that can't be fulfilled by HTML, it's never going to wildly succeed in the long run. No matter how popular Java gets, it will never surpass the document-publishing aspect of the Web.*
>
> [Wagn98]

3.5.3.3 (Noch) Ungelöste Probleme

„Wunder Punkt" der Anwendung Applet-basierter Präsentationstechniken bleibt darüber hinaus die ungewisse Frage, inwieweit die entsprechende *Runtime-Engine* bzw. die korrekte Version der „Java Virtual Machine" eine Ausführung des übermittelten Bytecodes auf den entsprechenden Client-Geräten erlaubt: Einerseits verhindert beispielsweise der große Anteil veralteter JMFs im Rahmen der jeweiligen Browser-Implementierungen etwa die Unterstützung fortschrittlicher GUI-Techniken auf Basis des „Swing"-Modells (welches jedoch wiederum eine recht aktuelle JMF erfordert), wodurch naturgemäß mit der Realisierung entsprechender Präsentationsansätze stets eine gewisse Unsicherheit bezüglich der tatsächlichen Unterstützung der entwickelten Funktionalität auf Seiten des Nutzers einhergeht. Zum anderen jedoch bleibt auch die letztendliche Verbreitung der Applet-Unterstützung an sich recht ungewiss [vgl. Roth98], da einerseits bekannte Sicherheitslücken [vgl. Oaks01] insbesondere Firmen-Administratoren zunehmend dazu verleiten, Java-Funktionalitäten von zahlreichen Web-Clients zu verbannen. Überdies kann auch die mangelnde Bereitschaft der Browserhersteller zur verstärkten Integration der Java-Plattform in ihre Internet-Clients als Hauptgrund für eine immer noch zu wünschen übrig lassende Java-Unterstützung [vgl. Bels97] benannt werden: So ließ sich etwa Microsoft nur unter dem Zwang eines kontroversen Rechtsstreites [vgl. Viol01] mit Erz-Rivalen und Java-„Erfinder" Sun Microsystems dazu bringen, eine entsprechende Java Runtime-Engine in den *Internet Explorer* einzubinden.[2]

3.5.4 Web-Portierungen nativer Multimedia-Anwendungen

Ein ähnliches „Schicksal" mangelnder Client-Unterstützung erlitt derweil auch ein Großteil der Vertreter eines bereits weit vor dem Aufkommen Javas verbreiteten Präsentationsansatzes: Die aufgrund des überraschenden „WWW-Booms" eilends „Internet-fähig gemachten" Web-Versionen der zuvor noch auf dem Offline-Sektor erfolgreichen, nativen Multimedia-Programme. So sahen sich die bereits in [2.1] beleuchteten Multimedia-Veteranen wie etwa das betagte HyperCard [Good90] sowohl durch die nachlassende Popularität CD-ROM-basierter Medien wie auch der explosionsartigen Entwicklung des HTML-Formates zunächst überrascht – reagierten jedoch nach längerer Zeit der Irritation zumeist in der Form proprietärer Plug-Ins: Während Apples HyperCard mittels QuickTime-Portierung und der CGI-basierten (Server-)Lösung *LiveCard* [Roya96] im Rahmen der WWDC-Konferenz noch 1996 eine durchaus überzeugende und vor allem aufgrund der Server-Seitigkeit allseits kompatible Internet-Portierung vorstellen konnte [vgl. Calh96, Neub96], ging bereits HyperCard-Imitator *SuperCard* im Rahmen der Vorstellung seines Plug-Ins *Roadster* [Heid97] andere Wege: Ebenso wie der noch später folgende Web-Adapter *Neuron* [vgl. Hols97, Musi02] des Mitkonkurrenten *Toolbook* [BrDw94] ermöglichte die Plug-In-Software zwar im Gegensatz etwa zum Web-Export PowerPoints [s.2.3.4] eine 1:1-Portierung der Multimedia-Inhalte innerhalb des Web-Clients – damit einher ging jedoch stets eine zunehmende *Abhängigkeit* von der Verfügbarkeit der jeweiligen, freilich untereinander völlig inkompatiblen Plug-Ins:

---

[1] vgl. [Wagn98]

[2] vgl. den entsprechenden Heise Newsticker-Eintrag von Erich Bonnert und Karsten Violka am 24.Dezember 2002: „Microsoft muss Java integrieren." http://www.heise.de/newsticker/data/kav-24.12.02-000 [12.2.03]

> *Bei einer Aktualisierungsgeschwindigkeit von ungefähr 1% pro Woche dauert es etwa ein Jahr, bis die Mehrheit der Nutzer auf die Möglichkeiten der von ihnen eingesetzten […] Technologie zugreifen kann – und gar zwei Jahre, bevor jeder diese Technologie hat.*
>
> [Niel00b:34]

3.5.4.1 Plug-In-Problematik

3.5.4.1.1 Unzählige Problembereiche

Neben der im Zuge der Plug-In-Diskussion noch hinzukommenden Problematik, dass aufgrund der speziellen Browser- Schnittstelle für jedes Betriebssystem ein separates Plug-In zu entwickeln ist, erschwert überdies die Tatsache, dass die von Netscape entwickelte „Plug-In-API" keinen offenen Standard, sondern ein auf den *Communicator* gemünztes Programmmodul darstellt [vgl. Blei96:68], die Entwicklung allgemeingültiger Plug-In-Komponenten, da somit auch die Kompatibilität mit anderen Browsern keinesfalls gewährleistet ist. Da überdies auch der Internet-Nutzer „nur selten willens ist, sich ein entsprechendes Plug-In herunterzuladen" [Star01:1] erscheint „der Einsatz von Techniken, die zusätzliche Plug-Ins verwenden, ohnehin fragwürdig." [Rieh01:68].

> *Vor der Verwendung sollte überprüft werden, wie verbreitet die betreffenden Plug-Ins bei den Nutzern sind, um nicht zu viele potentielle Nutzer auszuschließen. Daher sind derartige Technologien in der Regel erst dann zu verwenden, wenn sie sich als Standard durchgesetzt haben, d.h. eine längere Zeit auf dem Markt erprobt sind.*
>
> [Niel01]

3.5.4.1.2 Vollständige Abdeckung: Praktisch unmöglich

Aufgrund dieser Erkenntnis überrascht es daher nur wenig, dass es keinem der soeben genannten Multimedia-Tools schließlich gelang, eine ausreichende Verbreitung des jeweiligen Plug-Ins zu erreichen. Aufgrund der diese Problematik noch verschärfenden Stagnation des CD-ROM-basierten *Offline*-Marktes versank ein Grossteil der noch Anfang der 90er Jahre boomenden Authoring-Softwares rasch in Bedeutungslosigkeit oder konzentrierte sich, wie etwa Asymetrix' *Toolbook*, auf kleinere Marktnischen wie Internet-Lernsoftware [vgl. Musi02]. Das positive Beispiel HyperCards, das schließlich dennoch aufgrund schwindender Unterstützung von Seiten des Mutterkonzerns[1] die Waffen strecken musste,[2] macht dennoch zugleich die eigentliche Redundanz der entsprechenden Plug-In-Abhängigkeit deutlich: Da sämtliche, oben genannten Multimedia-Lösungen ebenso wie das HTML zugrunde liegende Strukturprinzip auf einer recht statischen Hypertext-Architektur nach dem *Dexter*-Modell[3] [vgl. HaSc94] basieren, ist, wie etwa die rein Server-seitige Lösung *LiveCard* [Roya96] deutlich macht, eine relativ exakte, aber dennoch dem Hypertext-Prinzip vollends entsprechende Überführung prinzipiell durchaus „sauber" möglich. Daher erscheint es auch im Hinblick auf die Kompatibilitätsbezogenen Vorzüge einer möglichen Server-Lösung mitunter nahezu unverständlich, warum dennoch größtenteils auf den (wie soeben diskutiert) „problematischen"[4] Plug-In-Ansatz zurückgegriffen und somit zumindest in Teilen zum Scheitern dieses (wie in [2.1] beleuchtet, strukturell durchaus überzeugenden) so genannten *Frame-based Paradigmas* [vgl. Bole98] beigetragen wurde.

3.5.4.1.3 Sinnvolle Plug-In-Verwendung

Weitaus plausibler erscheint dagegen die Plug-In-Notwendigkeit des dem HTML-Prinzip grundsätzlich eher entgegenstehenden Modells des „Timeline-basierten" Authoring-Paradigmas [vgl. Bole98], dessen *Animations*-Basiertheit eine direkte Konvertierung an dieser Stelle freilich unmöglich macht. Da ebendieses Grund-

---

[1] vgl. hierzu Jim Stephensons abschließenden Kommentar zu dessen „HyperCard Heaven"-Projekt vom 17. November 2001: http://members.aol.com/hcheaven/inactive.html [3.2.03]

[2] s. 2.1

[3] s. hierzu ebenda.

[4] vgl. hierzu auch [Depe02] pp.19ff.

prinzip das ansonsten eher statischen HTML-Web um ganz erhebliche, multimediale Möglichkeiten (insbesondere natürlich im Hinblick auf Animation) erweitert, erstaunt es auch angesichts der bereits in [2.1] diskutierten Marktführerschaft des Hauptvertreters dieses Timeline-Ansatzes, *Marcomedia Director*, daher nicht, dass sich – freilich auch dank geschickter Marketing-Bemühungen und strategischer Vereinbarungen des Macromedia-Konzerns, speziell mit Microsoft [vgl. Star01][1] – das entsprechende, so genannte *Shockwave*-Plug-In des zuvor im Offline-Bereich dominierenden *Director*-Tools aufgrund seiner letztlich „großen Verbreitung“[2] von (laut eigener Studie) bis zu 70 Prozent,[3] überraschend deutlich durchsetzen konnte.

3.5.4.2 Multimedia versus Web?

3.5.4.2.1 Anwendungsprinzip

Trotz des für semi-professionelle Anwender, wie bereits in [2.1] diskutiert, ungewohnten und (speziell im Vergleich mit der „Over-Simplicity“ [Maney99] PowerPoints) voraussichtlich „zu anspruchsvollen“ Anwendungsprinzips [vgl. Bole98], eröffnet *Shockwave* – eine auf dem Client-Rechner vorhandene Plug-In-Installation natürlich stets vorausgesetzt – durchaus beeindruckende, multimediale Möglichkeiten auch im WWW: Neben mittlerweile vollständig geglättetem[4] Schrift-Rendering, integrierter Chat- und XML-Funktionen [vgl. Dowd99] ermöglicht insbesondere eine mächtigen 3D-Engine [vgl. AuBa02] die zwar aufwendige, aber prinzipiell umso wirkungsvollere Erstellung multimedialer Präsentationen.

Die grafische Opulenz und Animationsmöglichkeiten, welche den Autoren von Multimedia-Präsentationen durch die Anwendung der Shockwave-Technik nun offenstehen (und die bei Betrachtung der tatsächlichen Umsetzungen in der Regel auch stets voll ausgespielt werden), geht jedoch mit einer im Hinblick auf *Webfähigkeit* empfindlichen Nachteil einher: So schlägt nicht nur der Download des Shockwave-Plug-Ins mit allein weit über 3 MB sehr „schwergewichtig“ zu Buche[5] – auch die Dateigröße der eigentlichen DCR-Dateien[6] selbst sprengt oftmals die Übertragungskapazität handelsüblicher Modems. Diese Problematik begründet sich freilich einerseits in der Animations-Basiertheit *Directors* per se, und zugleich natürlich ebenso in dessen ursprünglicher Offline-Orientierung: Da das Programm ursprünglich lediglich auf den „hybriden“ [7] Export auf CD-ROM-Basis ausgelegt war, sah die *Shockwave*-basierte Web-Export-Funktion zwar ausreichend kompressionstechnischen Spielraum zur rudimentären Datenreduzierung vor – allein das hinsichtlich möglicher Dateigrößen und Bandbreiten recht ignorante Anwendungsprinzip der *Director*-Anwendung selbst steht an dieser Stelle jedoch der Möglichkeit, auf dieser Grundlage tatsächlich Web-*optimierte* Anwendungen zu erstellen, eher entgegen.

3.5.4.2.2 CD-ROM-Multimedia und Internet: Unvereinbar?

Diese Problematik wirft an dieser Stelle naturgemäß die Frage nach der *Unvereinbarkeit* des ursprünglich CD-ROM-Basierten „Offline-Multimedia“ mit an sich bandbreitenschonenden Web-Anwendungen auf: Obgleich zweifellos feststeht, dass sich im Rahmen des „nativen“ Offline-Multimedia (wie bereits in [1.1] beleuchtet) zweifellos emotional „beeindruckendere“ [vgl. Godi01] Präsentationen erzielen lassen, kann jedoch ebenso eindeutig festgestellt werden, dass sich diese Ansätze (wie teils im Rahmen von *Directors Shockwave*-Plug-In versucht, s.o.) keinesfalls 1:1 übertragen lassen, da sie den in [3.1-2] teils angedeuteten, grundsätzlichen Eigenschaften des Web keinesfalls Rechnung tragen: So werfen nicht nur die enormen Da-

---

[1] „Die Verbreitung des Shockwave-Plugins basiert vielmehr auf einer geschickten Strategie von Macromedia: Durch Kooperation mit Firmen wie Microsoft...“ [Star01] p.11

[2] vgl. [Dahm01]

[3] vgl. die entsprechende Erhebung von NPD bzw. IDC User Forecast im Auftrag von Macromedia: „Shockwave Player Penetration“, San Francisco, Dezember 2002: http://www.macromedia.com/software/player_census/shockwaveplayer/penetration.html [4.2.03]

[4] Anm: Mittels *bikubischem Anti-Aliasing*.

[5] “Unfortunately, the Shockwave installation is very large, and it is difficult to download.” [Olip96] Kap.2

[6] DCR stellt das eigentliche *Shockwave*-Exportformat für das WWW dar [vgl. Dowd99, Baja02]

[7] Anm: „Hybrid“ bezeichnet in diesem Falle die Plattformunterstützung sowohl für Mac- als auch für Windows-Betriebssysteme.

teigrößen „multimedialer“ Präsentationen diesbezüglich Probleme auf – auch die textliche Durchsuchbarkeits-Eigenschaft des WWW, die *Spatial Flexibility*-Problematik,[1] sowie das Navigations- und Anwendungsprinzip des Internet selber [vgl. Niel95,99] stehen den „Multimedia-Präsentationen“ zugrunde liegenden Eigenschaften prinzipiell entgegen.

3.5.4.2.3 Die Lösung: Vektoren

Nicht nur *Director*-Herausgeber Macromedia hat mit der Integration des vektor-basierten *Flash* [s.4.5] in das Shockwave-Framework deutlich gemacht, dass das Unternehmen diese Problematik eindeutig erkannt hat – auch im Rahmen unserer Untersuchungen multimedialer Präsentationssysteme erscheint es interessant, sich im folgenden näher mit einem Ansatz auseinanderzusetzen, der zumindest Teile der soeben erörterten Problemstellungen zu lösen verspricht: dem Konzept *Vektor*-orientierter Internetanwendungen.

> *Yes, raster is faster – but Vector just seems more correcter!* (Tomlin, 1990)[2]

[1] Dieser Ausdruck bezieht sich auf die Flexibilität, die HTML im Hinblick auf sehr unterschiedliche, mögliche Browser-Fenstergrößen und –Typen erfordert, verbunden mit zugleich eingeschränkten Kontrollmöglichkeiten bezüglich eines gewünschten (aber unerreichten) „pixel-genauen Webdesigns“.

[2] zitiert aus [NeWi00]

4 Kurvenwunder: Vektorgrafik im Web

4.1 Grundlegende Konzepte

4.1.1 Manipulationstheorie, Bandbreite und Intelligenz

Da im Rahmen *Vektor-orientierter* Bilddateien grafische Grundelemente in Form mathematischer Gleichungen statt ihrer „aufgerasterten" Pixeldaten repräsentiert werden (eine Linie kann dieserart etwa lediglich anhand ihrer entsprechenden Koordinaten sowie grundlegender Attribute wie Linienstärke etc. definiert werden), ist maßgebliches Kennzeichen ihrer zugehörigen Formate zumeist die enorme *Einsparung von Speicherplatz*, die mit der Enkodierung vektorieller Bilddaten gegenüber ihren „Bitmap"-Pendants einhergeht:

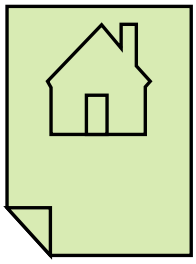


Abb. 4.1.1.1: Ursprüngliche Vektor-Grafik (im EPS-Format),[1] Dateigröße: 8 KB.

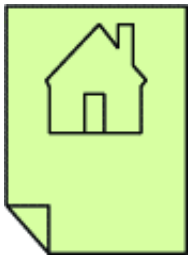


Abb. 4.1.1.2: Vektorisiertes Bitmap-Pendant (BMP-Format, 86x118 Pixel), Größe: 30 KB.

Auf diese Weise ermöglicht es die Speicherung vektorbasierter Grafiken im Internet prinzipiell, die Dateigrößen der im Rahmen entsprechender Web-Präsentationen zur Anwendung kommenden Bilddaten erheblich zu reduzieren – dementsprechend zuvor im Vektorformat vorliegende Grafiken natürlich vorausgesetzt. Hiermit einher geht freilich zudem die (bereits im Rahmen des PowerPoint-Web-Exports [s.2.3.4] besprochene) *Manipulations*-Eigenschaft der entsprechenden Bilddaten: Im Gegensatz zu den Pixel-orientierten *Rasterformaten*, deren Kennzeichen insbesondere in Zusammenhang mit Internet-gemäßer Datenkompression ein nennenswerter *Verlust* von Bildinformation darstellt, lassen sich Vektordaten auch nach einem u.U. erfolgten Export-Vorgang ins Internet hernach ohne jeglichen Verlust an Information oder Bildqualität wieder betrachten und bearbeiten. Sogar die Nachträgliche Bearbeitung oder gar Animation von Text oder einzelnen Elementen des Vektorbildes ist im Gegensatz zu „aufgerasterten" Pixelbildern ohne weitere Umstände möglich. Aufgrund dessen sind Vektorgrafiken im Allgemeinen, wie auch [Blum98] anmerkt, „den im Internet derzeit verwendeten Pixelgrafiken bezüglich der Wartbarkeit und Handhabbarkeit deutlich überlegen".[2]

Da grafische „Primitive"[3] und ihre Attribute, die ja die Hauptkomponenten vektorieller Bildformate darstellen, somit eine „höhere Ebene der Bildrepräsentation repräsentieren" [Stei99:59], kommt an dieser Stelle überdies ein weiteres insbesondere im Internet vorteilhaftes Merkmal vektorbasierter Grafiken zum Tragen, welches ich gerne unter dem Begriff der „Bild-Intelligenz" zusammenfassen würde: Da die einzelnen Komponenten der Grafik als direkte, mitunter sogar semantisch zusammenhängende Objekte und nicht lediglich in der Form „dummer",[4] unzusammenhängender Bild-Pixel abgelegt sind, wird nicht nur der darstellenden *Rendering-Engine* eine höhere funktionelle „Intelligenz" etwa in Form von Echtzeit-Anti-Aliasing zuteil [vgl. Moir97] – im Rahmen des so genannten „Semantic Web" [Bern01] ist es überdies entweder direkt oder aber unter Anwendung künstlicher Intelligenz durchaus möglich, den semantischen Inhalt (auf

[1] s.4.2
[2] vgl. [Blum98]
[3] Anm: Mit den "Primitives" werden i.d.R. grafische (vektorbasiert) Grundformen beschrieben: Kreise, Vierecke, Linien etc.
[4] "Bitmaps are dumb […] Bitmap file formats make little or not attempt to represent their images intelligently." [Moir97] – Einführung

dem niedrigsten Abstraktionslevel wäre dies schlicht die Durchsuchbarkeit des enthaltenen Textes) bzw. die Objekt*struktur* einzelner Grafiken einschließlich des inhaltlichen Zusammenhangs auch für die „Web-Öffentlichkeit" transparent zu machen und somit das Auffinden und die (bereits angesprochene) Verarbeitung vektorieller Bilder im Internet zu ermöglichen. Dieser Grundgedanke findet sich etwa der „abstrakten" Grafik-Repräsentation „höherer" Internet-Grafikstandards [s. hierzu 5.3.4] wieder, da hier Grafiken *nicht* als „diskrete", einzelne „Primitives" in Form etwa von konkreten Linien mit ihren jeweiligen Positionsdaten und Attributen, sondern auf einer „abstrakteren" semantischen Ebene (etwa „Objekt A befindet sich ‚unter' Objekt B") repräsentiert werden.

4.1.2 Strukturelle Unterschiede

Dieser Umstand an sich lenkt unsere Aufmerksamkeit bereits an dieser Stelle auf die Tatsache, dass die *Abbildung* einer Vektorgrafik an sich zwar eindeutig bestimmt sein kann – sich die „interne" Repräsentation der einzelnen Elemente jedoch auf verschiedene Weisen umsetzen lässt und somit erheblich komplexer zu realisieren ist als entsprechende Pixelbilder. Während Bitmap-Grafiken für jede Darstellung und für jedes Format vollständig und eindeutig bestimmt sind (d.h. sämtliche Pixel-Dateien einschließlich der in [3.3.2-3] besprochenen Kompressionsverfahren lassen sich in ein eindeutiges „Universal"-Format[1] überführen), gibt es bei Vektor-basierten Bildern *unendlich* viele Möglichkeiten, ein und dieselbe Grafik zu beschreiben; Ebenso wie in der Mathematik eine Parabel sowohl durch die Abbildungsvorschrift $f(x) = x^2$, wie auch etwa durch $f(x) = ½ x (x \cdot 2)$ darstellen lässt, gibt es im Hinblick auf Vektorgrafikformate unzählige Arten, eine Abbildung zu repräsentieren. So kann in dem einen Vektorformat ein Kreis etwa durch ein gedachtes *Circle*-Objekt beschrieben werden, während dieses in einem anderen Format möglicherweise gar nicht existiert und stattdessen durch einen gekurvten *Pfad* realisiert werden muss.[2] Dieser Sachverhalt verkompliziert nicht nur die (daher oft verlustbehaftete und mitunter gar unmögliche) Konvertierung eines Vektorformates in das andere [vgl. PMEB01], sondern erschwert überdies auch Ansätze der automatischen Erstellung und des Zugriffs auf verschiedene Formate, etwa über eine höhere Programmiersprache.

4.1.3 Formattypen

Über diese rein semantische Problematik hinaus stellen überdies technische Grundlagen vielfältige Möglichkeiten bereit, ein vektorbasiertes Grafikmodell syntaktisch zu enkodieren: Neben der rein textlichen Beschreibung des Modells bietet sich zur effizienteren Nutzung von Speicherplatz etwa die *binäre Speicherung* der Daten an, was freilich wiederum die „Lesbarkeit" der Dateien einschränkt und die Interpretationsalgorithmen beim Zugriff auf die Datei deutlich erschwert. Im Rahmen der insbesondere text-basierten Beschreibung vektorieller Bilddaten stellt überdies die Trennung von Elementen und Daten (etwa des „Circle"-Objekts mitsamt „konkreten" Positionsdaten und Stilattributen) ein Problem dar, welches jedoch durch etwa durch so genanntes *Markup* [vgl. Wild:142] kompensiert werden kann. Obgleich die Idee des Markup, da bereits im Rahmen von SGML frühzeitig eingeführt und etwa durch dessen konsequente Anwendung in HTML einer breiten Öffentlichkeit bekannt, gerade durch die Popularität des WWW nicht sonderlich neu und überdies allgemein akzeptiert ist, überrascht es um so mehr, dass unter anderem [NeWi00] entrüstet feststellen müssen, es gebe derzeit „keinen zufrieden stellenden Standard für Vektorgrafik im Internet" – und dies, „obwohl schon sehr früh (nämlich 1993) mit ersten Versuchen begonnen wurde."[3]

Bei der folgenden Untersuchung verschiedener Ansätze, vektorbasierte Grafik im Rahmen Web-fähiger Präsentationen zur Anwendung kommen zu lassen, spielen daher insbesondere Betrachtungen der semanti-

[1] Anm: In der Regel ist dies etwa ein unkomprimiertes Bitmap-Format wie z.B. BMP.

[2] Anm: Dies ist bereits eine Referenz auf das *Flash*-Format, in dem es mittels der hierfür ausschließlich angebotenen „quadratischen Béziers" recht umständlich ist, Kreise zu realisieren: „Its use of quadratic Beziers makes it a bit of a pig to draw circles" (Jon Katz, Slash-Dot)

[3] vgl. [NeWi00]

schen sowie syntaktischen Struktur der jeweiligen Formate eine Rolle, um sowohl herauszufinden, wie geeignet die jeweiligen Ansätze hinsichtlich Internet-basierter Präsentationen erscheinen, als auch die Schnittstellen zum möglichen Zugriff bzw. der potentiellen Verarbeitung der jeweiligen Formate zu identifizieren und im Hinblick auf eine mögliche Anwendung im Rahmen einer Präsentationslösung nutzbar zu machen.

4.2 Die Print-Experten: PostScript und PDF

4.2.1 PostScript

Im Zusammenhang der allgemeinen Betrachtung Vektor-basierter Bildtypen stellt die von Adobe bereits 1984 entwickelte *PostScript*-Sprache sicherlich *das* Vektor-Format schlechthin dar [vgl. Hent98:104]: Nicht zuletzt aufgrund der raschen Ausbreitung der Laserdrucker in den frühen 90er Jahren, deren internes Verarbeitungsformat in der Regel durch PostScript realisiert wird, hat sich das Format als allgemein akzeptierter und verwendeter „Quasi-Standard" etabliert, was sicherlich auch in dessen durchaus beachtlicher Funktionalität zu verdanken ist:

> *PostScript bietet die mögliche Integration von hochqualitativer, skalierbarer Schrift, Grafik und Bildern… Neben Möglichkeiten zur Beschreibung einer Seite stellt PostScript eine vollständige Programmiersprache dar, die unter anderem Variablen, Kontrollstrukturen, Prozeduren und Dateien umfasst.*
>
> [Stei99:51]

Obgleich PostScript daher ein rein *Print*-orientiertes Format ist, erscheint dennoch die *Text-Eigenschaft* der übrigens in „umgekehrter polnischer Notation"[1] gehaltenen Skript-Syntax der PostScript-Sprache [vgl. Born90:587ff] hinsichtlich einer möglichen Web-Anwendung durchaus nicht uninteressant: So stellt PostScript derzeit immerhin eines der wenigen Formate im Internet dar, dessen Inhalte vollständig etwa durch die Suchmaschine *Google* indizier- und auffindbar sind.[2] Mögliche Präsentations-Funktionalität sucht man in der PostScript-Spezifikation hingegen vergeblich: Da das betagte PostScript-Format noch weit vor der „Präsentations-Revolution" [s.3.2.1] aus der Taufe gehoben wurde, bietet der seit den frühen 90er Jahren nicht mehr aktualisierte[3] Sprachumfang (trotz Scripting-Funktionalität) neben der rein statischen Darstellung vektor-basierter Grafiken keinerlei darüber hinausgehenden „Features" – ein Manko, das schließlich durch dessen Nachfolger *PDF* mehr als behoben wird: Im Rahmen von Überlegungen, wie man insbesondere die PostScript-Ausgabe hinsichtlich möglicher Präsentations- und Dokumentenaustausch-Eigenschaften beschleunigen könne [vgl. MaKo02], entwickelte John Warnock, Mitbegründer und damaliger CEO von Adobe, Anfang der 90er Jahre das Konzept für PDF, dem *Portable Document Format*.

4.2.2 PDF : Portable Document Format

4.2.2.1 Web-Fähigkeit des PDF-Formates

Ursprünglich primär im Hinblick auf optimale Austauschbarkeit (daher auch der entsprechende Produktname) entwickelt und aufgrund dessen zunächst insbesondere im Grafik- und PrePress-Bereich erfolgreich,[4] richtete sich das PDF-Format jedoch bereits im Rahmen des so genannten „Amber"-Projekts auf die boomende WWW-Plattform aus [vgl. McHu96]: Aufgrund der rasch deutlich werdenden *Enttäuschung* Design-orientierter Web-Entwickler bezüglich der gestalterischen (Kontroll-)Möglichkeiten HTMLs [s.3.2] wurden schon bald Bemühungen seitens des Adobe-Konzerns deutlich, PDF als „grafische" Alternative zum Struk-

---

[1] Anm: Verfahren zur *Stack*-optimierten Speicherung von Anweisungen. Beispiel: 3,4,+ (= 3+4)

[2] vgl. hierzu etwa die entsprechende Meldung des *Heise-Newstickers* vom 6. November 2001: „Google findet Office-Dokumente" (von Jo Bager) http://www.heise.de/newsticker/data/jo-06.11.01-000 [4.2.03]

[3] vgl. hierzu [Stei99] pp.51f.

[4] vgl. [MaKo02] p.17

tur-orientierten HTML zu positionieren – zu offensichtlich schienen zunächst die Vorteile des PostScript-Nachfolgers: Neben den bereits in [4.1.1] diskutierten Vorzügen hinsichtlich der Platzeinsparung durch Atrribut-basierte (in PDF übrigens zusätzlich Huffman-codierte) Vektordaten garantierten überdies komfortable Schrifteinbettung und die zusätzliche Unterstützung JPEG-komprimierter Grafiken eine im Gegensatz zu HTML *vollständige* Kontrolle über das optische Erscheinungsbild der dargestellten PDF-Datei:

> *Inzwischen hat PDF auch im Web seinen Platz erobert und stellt eine Alternative zu HTML dar, insbesondere wenn die Typografie und das Layout im Vordergrund stehen.*
>
> [Hent98:110]

Der „Tagged-PDF"-Ansatz [vgl. MaKo02:367ff] bietet darüber hinaus noch die Möglichkeit, PDF-Dateien XML-basierte Strukturinformationen hinzuzufügen und die damit bereits komfortable Durchsuchbarkeit[1] im Web noch durch semantische Zusammenhänge zu bereichern. Zumindest prinzipiell stellt PDF somit eine wirkliche Alternative hinsichtlich Web-basierter Präsentationen dar, da eine beeindruckende Kontrolle über visuelles Erscheinungsbild mit den semantischen und interaktiven Möglichkeiten des WWW verbunden werden kann.[2] Da PDF überdies auch die positiven Eigenschaften der PostScript-Sprache auf sich vereinigt, kann somit auch eine korrekte, fehlerfreie Druckausgabe, bislang mit eines der Mankos des HTML-Formates, im Rahmen von PDF garantiert werden, weswegen das Format freilich auch in der DTP-Branche überaus geschätzt wird.[3]

4.2.2.2 Problemfall: Acrobat Reader

Der Grund, warum sich PDF bislang dennoch nicht gegen das ungeliebte[4] HTML-Format durchsetzen konnte, ist neben der im Web durchaus störenden Seiten-Orientierung des Formates insbesondere in der „opulenten" Umsetzung des PDF-Viewers zu suchen: Obgleich sich die Entwicklungsbemühungen im Rahmen eines universellen Viewing-Tools sich bereits frühzeitig auch auf die Erfordernisse der Internet-Browserumgebungen ausrichtete [vgl. McHu96], geriet der letztendlich verfügbare *Adobe Acrobat Reader*[5] erheblich zu „mächtig" und schwerfällig. So erscheint nicht nur der Download des Viewers, der Plug-In und *Standalone*-Applikation zugleich bereitstellt, mit derzeit rund 15 MB eine Spur zu umfangreich – auch die erheblichen Latenzzeiten, die sich beim Betrachten von PDF-Dokumenten im Web mithilfe des PDF-Reader-Plug-Ins ergeben, erscheinen angesichts der von [Niel97] angeführten „User-Toleranz", die vorgeblich im Bereich „weniger Sekunden" liegt,[6] kaum plausibel: Allein um die 10 Sekunden vergehen daher in der Regel bereits beim „Hochfahren" der Reader-Komponente nach Anforderung einer Online-PDF-Datei: Aufgrund des überdies intransparenten[7] und überdies unnötig langsam erscheinenden Ladevorgangs der in der Regel empfindlich großen im Web vorliegenden PDF-Dateien kann somit bis zu eine halbe Minute untätig verstreichen, noch ehe die ersten Lettern des PDF-Files am Bildschirm erscheinen. Verglichen mit der ursprünglichen Absichtserklärung Adobes zur Implemetierung eines „optimierten" Internet-Readers erscheint die derzeitige Performance-Situation daher geradezu ironisch:

> *Amber will […] allow for faster downloading and displaying of a page at a time across the Web… Users will be able to retrieve and view only the pages they want to see in any given document. Optimized PDF*

---

[1] Anm: PDF-Dateien stellen zumindest in großen Teilen immer noch Textdateien dar und sind aufgrund dessen in der Regel von Suchmaschinen gut indizierbar.

[2] vgl. [Hent98]: „Mit PDF haben Sie die gewohnte Kontrolle für die Gestaltung auf Papier, zusätzlich stehen Ihnen noch die interaktiven Möglichkeiten von HTML zur Verfügung." [Hent98] p.110

[3] vgl. [MaKo02] p.17

[4] s.3.2

[5] URL: http://www.adobe.com/products/acrobat/readstep.html [4.3.03]

[6] vgl. [Niel97]

[7] Anm: Innerhalb des PDF-Plug-Ins wird an keiner Stelle der Ladefortschritt der PDF-Dateien angezeigt.

> *files will render text first while larger objects, such as images, will be downloaded in the background and rendered last.*
>
> [McHu96:20]

Das zur Behebung dieser Problematik vor Urzeiten angekündigte Prinzip des „Embedded PDF" (EPDF), welches durch *block-basierte* Fragmentierung der PDF-Datei einen „Web-gemäßeren" Ladeprozess ermöglichen sollte [vgl. Smit95], wurde von Adobe jedoch bedauerlicherweise nie verfolgt, sodass „auch heute noch gilt: Erst laden, dann ansehen":[1]

> *Bisher war dem Portable-Document-Format daher der große Durchbruch im Netz verwehrt: Obwohl für alle wichtigen Plattformen Reader und sogar Netscape-Plug-ins existieren, liegen Dokumente dieses Formats häufig ungelesen auf den Servern.*
>
> [Meis97:49]

Dennoch sind darüber hinaus weitere Bemühungen des Adobe-Konzerns erkennbar, das PDF-Format für direkte Online-Präsentationen geeignet zu machen: So versuchte etwa der noch 1996 initiierte „Web Presenter" [Dyso96], das Format mithilfe des bereits Anfang der 90er Jahre von Adobe eigentlich „links liegen gelassenen" [Bell00:9] Präsentations-Programms *Persuasion* [s.2.3.1] sowohl „Web-fähig" zu machen, als es auch über den Print-Bereich hinaus für den lukrativen Bereich der Business-Präsentationssoftware attraktiv zu machen. Es gelangt dem Unternehmen zwar, noch 1996 im Rahmen des „Improving Net Expectations"-Symposiums einen voll funktionsfähigen Prototypen des als kostenfreie *Drag-and-Drop*-Software konzipierten „Web Presenters" vorzustellen[2] – da jedoch (wie oben diskutiert) das *Amber*-Projekt und somit die Bemühungen des Konzerns um ein „Internet-fähiges" PDF-Format, das dem Web Presenter ja zugrunde liegt [vgl. Essi96], nicht weiter verfolgt wurden, muss somit abschließend bedauerlicherweise festgestellt werden, dass PDF bis zum heutigen Tage als nur sehr eingeschränkt „Web-tauglich" bezeichnet werden kann. Dennoch lohnt es sich insbesondere angesichts der beeindruckenden Verbreitung und Akzeptanz des PDF-Formates und damit auch des *Readers*, die Präsentations-bezogenen Anzeigeeigenschaften von PDF – wenn auch, wie eben gesehen, primär im Offline-Bereich anwendbar – etwas genauer unter die Lupe zu nehmen, um somit eventuelle Rückschlüsse auf eine generelle Eignung des Formates bzw. seiner „Viewers" hinsichtlich einer (auch im emotionalen Sinne)[3] überzeugenden Darstellung ziehen zu können.

4.2.2.3 Darstellungseigenschaften

Obgleich die vollständig geglättete Textdarstellung im Rahmen des PDF-Viewers konsequent implementiert und überaus ästhetisch wirkt, können die Darstellungseigenschaften des Readers dennoch im Gesamtbild nicht überzeugen, da eingebettete Pixel- und sogar Vektorgrafiken stets ohne entsprechendes Anti-Aliasing dargestellt werden und daher einen relativ „klobigen" Eindruck hinterlassen. Auch die bereits angesprochene Seiten-Orientierung PDFs sowie die in bisherigen PDF-Versionen durchaus störende Eigenschaft, dass Texte im Rahmen von PDF-Dateien bislang ausschließlich einzeilig vorlagen und Textumbrüche nicht möglich waren, stehen dem durchaus vorhandenen Potential des Formates hinsichtlich Internet-basierter Präsentation derzeit noch entgegen: Entsprechende Lösungen sind in Form von „Tagged PDF" [vgl. MaKo02:367ff], welches dank XML-Orientierung unter anderem erfreuliche Textfluss-Funktionen und somit auch seitenübergreifende Anwendungen möglich macht, zwar bereits vorhanden, aber aufgrund der bislang enttäuschenden Unterstützung[4] der neuen PDF-Spezifikation 1.4 [BCM01] bislang leider wenig verbreitet.

---

[1] vgl. [Meis97] p.49

[2] vgl. hierzu die entsprechende Pressemitteilung Adobes vom 7. Mai 1996: "Free Web Presenter Drag and Drop Software." URL: http://www.pimall.com/nais/n.adobe.html [6.2.03]

[3] Anm: Schließlich spielt eine ästhetische Darstellung und somit der „emotionale Faktor" im Rahmen von digitalen Präsentationen eine wichtige Rolle [vgl. Godi01]

[4] Anm: Dies mag unter anderem auch durch den in der neuesten Version enormen Download-Umfang von 15 MB begründet liegen

Das einzige System, auf der PDF-Dateien dank „Quartz"-Engine [vgl. EnEh00] derzeit äußerst performant und überdies in vollständigem Anti-Aliasing dargestellt werden können, ist das nativ PDF-unterstützende MacOS X-Betriebssystem, auf dessen Basis ein von der Freiburger *Schubert IT* entwickeltes Plug-In [Schu02] „begeisternde" PDF-Erlebnisse ermöglicht [vgl. EnSc03]

4.2.2.4 Multimedia-Funktionalität

Nach Betrachtung der Dokument- und Anzeige-Eigenschaften des Formates erscheinen natürlich auch im Rahmen dieser Diplomarbeit – schließlich erscheint die Anwendung des PDF-Formates im Hinblick auf Web-basierte Präsentationen durchaus nicht uninteressant –eine eingehendere Untersuchung der „multimedialen" Eigenschaften, die PDF unter diesem Aspekt bereitstellt, angebracht: Nicht zuletzt gilt es, die diesbezüglich oft geäußerte Meinung zu überprüfen, PDF stelle im Allgemeinen ein „rein statisches" [Voel98] Format dar, dass „für komplexere Multimedia-Anwendungen und Interaktionen nicht geeignet" sei [vgl. KrTe02].

Zu diesem „Vorwurf" muss zunächst einmal freilich kleinlaut eingestanden werden, dass PDF von seinem Grundprinzip her in der Tat ein primär statisches, Seitenbasiertes Hypertext-Format auf Basis des Dexter-Modells[1] [vgl. HaSc94] darstellt – allerdings ist es im Gegensatz zur allgemein verbreiteten Meinung dennoch durchaus möglich, in Form so genannter „*Movie Player Annotations*" sowie Link-basierter „*Actions*" Sound- und sogar ganze (Quicktime-)Filmdateien in PDF-Dokumente einzubetten [vgl. BCM01]. Aufgrund der Hypertext-Eigenschaft des PDF-Formates können sowohl PDF-Hyperlinks als auch die eben erwähnten „Movie Actions" ausschließlich durch Aktivierung durch den Benutzer wiedergegeben (Adobe-Terminologie: ge-„triggered") werden. Animationen oder etwa die automatische Wiedergabe einzelner Sound- oder Filmelemente sind somit zwar nicht möglich, für vollständig interaktive Präsentations-Anwendung ist diese im Rahmen der PDF-Spezifikation bereitgestellte Funktionalität jedoch völlig ausreichend.

Neben einem insbesondere für Bildschirm-Präsentationen oder Multimedia-Anwendungen überaus interessanten FullScreen-Modus, welcher sich komfortabel über ein Optionsfeld aktivieren lässt[2] und bei Öffnen des entsprechenden PDF-Dokuemts dasselbe auf Bildschirm-Größe „maximiert", bietet darüber hinaus das ursprünglich lediglich für *Formulare* vorgesehene JavaScript-Framework des PDF-Formates zusätzliche Möglichkeiten an, interaktive Multimedia-Anwendungen zu erstellen. Zwar ist auch hier die dynamische Manipulation einzelner PDF-Elemente (was die JavaScript-Sprache ja etwa im Rahmen von HTML erst „interessant" macht) leider nicht möglich – dennoch lassen sich bei etwas programmiertechnischer „Kreativität" durchaus interessante Effekte und Funktionen erzielen.

4.2.2.5 Fazit

Abschließend kann daher festgestellt werden, dass das PDF-Format entgegen der allgemeinen Auffassung durchaus beachtliche Ansatzpunkte auch für multimediale, Web-basierte Präsentationen bietet – insbesondere die mangelhafte Umsetzung der entsprechenden *Reader*-Anwendung [s.4.2.2.2-3] sowie mitunter enttäuschende Einschränkungen (etwa bei der JavaScript-Umsetzung oder hinsichtlich fehlender Animationseigenschaften) hinterlassen jedoch unter dem Strich einen durchwachsenen Eindruck, der PDF zwar für einige Anwendungen recht interessant erscheinen lässt, jedoch die erwünschte Erstellung effektvoller, multimedialer Präsentationen insbesondere im Hinblick auf Performance-Schwächen und der damit verbundenen, eingeschränkten Web-fähigkeit der PDF-Implementierung in ein eher zweifelhaftes Licht stellt.

---

[1] vgl. 2.2

[2] Syntax: obj << ... /PageMode /FullScreen ... >> endobj

Etwas mehr Bewegung in den Bereich PDF-basierter, multimedialer Präsentationen hat Adobe hingegen erst kurze Zeit vor Fertigstellung dieser Diplomarbeit gebracht: Nachdem der Konzern noch im Januar 2003 der *Album*-Version der populären Bildbearbeitungs-Software *Photoshop* zu einer exklusiven Slide-Show-Funktion auf PDF-Basis verholfen hatte [Rask03], schob das Unternehmen gegen Ende des Monats mit dem *Adobe Image Viewer Plug-In* endlich die lang ersehnte, multimediale Funktionalität nach: Neben MP3-basierter Hintergrundmusik ermöglicht diese Zusatzschnittstelle nun Animationen, Übergangseffekte und insbesondere komplexe[1] Vektorgrafiken samt Anti-Aliasing – alles dank vollständiger Einbettung des Grafikstandards SVG [s.5.5] direkt in die PDF-Datei.

Da überdies zu erwarten ist, dass ebendiese, zentrale SVG-Funktionalität im Rahmen der nächsten *Reader*-Version vollständig in Adobes PDF-Viewer integriert sein wird, muss an dieser Stelle zwar ein erneutes Performance-Defizit dieses Betrachter-Tools befürchtet werden – zugleich jedoch eröffnen sich nun insbesondere im Offline-Bereich, wo das PDF-Format, wie zuvor ausgeführt, ja bereits seit längerem hohe Verbreitung genießt, nicht zu unterschätzende Möglichkeiten hinsichtlich einer portablen Präsentationslösung. Eine Beispiel-Anwendung im Rahmen der SVG Open-Konferenz [Jack03b] macht bereits die beeindruckenden Fähigkeiten der Erweiterung bezüglich dynamischer Vollbild-Präsentation deutlich:

> *Such a slideshow can support high-quality image presentation, and can include transitions, video, audio, and animations. The same file maintains the same high-quality results expected from PDF.*
>
> [Foss03]

Aufgrunddessen lässt das PDF-Format trotz der momentan eher zweifelhaften Präsentationsleistung des *Readers* insbesondere im Hinblick auf die verstärkte SVG-Integration - speziell auch im Rahmen multimedialer Präsentationen über das Internet – für die Zukunft noch einiges erhoffen:

> *There seems to lurk some very tantalising prospects for dynamic digital media – based on the use of embedded SVG files within PDF presentations.*[2]

4.3 Anwendungsbereiche vektorbasierter Formate im Web

Neben dem soeben betrachteten *Dokumenten*-Format PDF, welches in der Regel neben Text- und Vektorobjekten noch weitere Elemente wie etwa pixelorientierte Bitmaps beherbergt und somit streng betrachtet kein originäres Vektor- sondern vielmehr ein *Meta*-Dateiformat [vgl. Stei99:60] darstellt (bei Integration von Audio/Video-Material [s.4.2.2.4] kann sogar von einem „Multimedia-Format" gesprochen werden),[3] existiert freilich noch eine Fülle weiterer, vektorbasierter Bildformate, die jeweils anderen Zielsetzungen folgen und auf spezielle Bedürfnisse zugeschnitten sind. Obgleich sich unsere Betrachtungen im Rahmen dieser Diplomarbeit primär auf Web-fähige, multimediale Präsentationsformate beschränken, soll dennoch an dieser Stelle das „wohl wichtigste und bekannteste" Vektor-Format [Wild99:436] an dieser Stelle nicht verschwiegen werden: Das bereits 1987 standardisierte *Computer Graphics Metafile* (CGM), auch bekannt unter dem Kürzel ISO 8632 [CGM92].

4.3.1 CAD-orientierte Anwendungen

Da sich ein Großteil der Anwendungsbereiche Vektor-orientierter Zeichnungen Ende der 80er Jahre hingegen auf *technische* Aspekte und hier insbesondere auf den Bereich des *Computer-Aided Engineering* (CAD/CAM) beschränkte, verwundet es freilich nicht, dass sich die Spezifikation des als (im Gegensatz zu zuvor proprietären Formaten wie AutoDesk DXF) *offenen* Standard konzipierten CGM-Formates erstlinig an

[1] Anm: Dies schließt Photoshop-Ähnliche Filter wie Gaussche Unschärfeefekte und Schlagschatten etc. mit ein [s.5.5]
[2] vgl [Foss03]
[3] vgl. hierzu auch [Stei99]

den Bedürftnissen CAD-basierter, technischer Zeichnungen orientiert: Zwar sind neben reinen Linien-Primitives und Text[1] auch so genannte „B-splines“ sowie die Einbettung von Rastergrafiken möglich – aufgrund der „eher begrenzten“ [2] Auswahl grafischer Elemente findet sich (trotz des nicht unerheblichen Beitrags[3] des CGM-Filters der *Corel-Draw*-Suite)[4] das Gros der vorzufindenden CGM-Anwendungen jedoch nahezu ausschließlich im Bereich technischer CAD-Zeichnungen wieder.

4.3.1.1 Computer Graphics Metafile und WebCGM

Auch im Rahmen dieser Diplomarbeit relevant [s.Kap.1] erscheinen jedoch nicht die eigentlichen, eher bescheidenen Grafikfunktionen des CGM-Formates selber, sondern vielmehr dessen Eigenschaft der *Web-Fähigkeit*: Nicht zuletzt stellt CGM *das* Format dar, welches bereits im Anfangsstadium des WWW, da bereits zuvor „Leib-und-Magen“-Grafikstandard der HTML zugrunde liegenden SGML-Konvention [vgl. HeGe94, Grae98, Born90:471],[5] durchaus in die engere Auswahl zur direkten Anwendung im Rahmen Web-basierter Grafik kam. Obgleich sodann aufgrund der den damaligen *Mosaic*-Browser offensichtlich überfordernden Komplexität des CGM-Renderings[6] zunächst äußerst schlichten Bitmap-Formaten den Vorzug gegeben wurde [s.3.3], kam das Format im Rahmen einer Mitte der 90er Jahre Aufkommenden Diskussion um die lang ersehnte[7] Einführung vektorbasierter *Web*-Grafik wieder als mögliches, universelles Austauschformat auch im Rahmen des WWW wieder ins Gespräch:

> *We* [W3C-Funktionäre Chris Lilley und Roy Platon, Anm.d.Verf.] *believe that CGM fulfils most of the [World Wide Web Consortium's Scalable Graphics] requirements…*
>
> [LiPl97]

Ergebnis dieser Bemühungen war schließlich die Veröffentlichung eines so genannten *WebCGM-Profils* [WCGM99], welches die direkte Anwendung vektorieller Grafiken auf Basis des (entsprechend angepassten) WebCGM-Formates direkt in der Browser-Umgebung ermöglichen sollte. Da das Format jedoch, wie nicht nur [Wild99], sondern offensichtlich auch die Mehrheit der Browserhersteller befand, aufgrund spezieller Eigenschaften des CGM-Formates selber[8] im Grundsatz eigentlich nur begrenzt für den WWW-Einsatz geeignet erschien und die offensichtliche CAD-fixierung das CGM-Format eher als speziellen, nicht unbedingt „mehrheitsfähigen“ Standard klassifizierte, kam es schließlich nie zu einer Implementierung dieses eigenwilligen Profils in gängige Internet-Browserprogramme. Auch eine im Rahmen von WebCGM schmerzlich vermisste, mögliche Präsentations- oder gar multimediale Funktionalität schränkt an dieser Stelle nicht nur die Relevanz dieses Formates für die Zielsetzung dieser Diplomarbeit, sondern offenbar auch das Interesse der breiten (sowie kommerziellen) Öffentlichkeit an WebCGM deutlich ein. Die kaum verbreitete und zudem sehr übersichtliche Auswahl an überzeugenden oder auch nur funktionsfähigen Browser-Plug-Ins trug somit trotz der intensiven Bemühungen der unter Federführung Dieter Weidenbrücks [vgl LiWe01] durchaus aktiven *CGMOpen*-Initiative[9] mit dazu bei, dass dem CGM-Format derzeit im Web keine übermäßige Bedeutung zukommt.

---

[1] Anm: … wenn auch hier, im Gegensatz zu PostScript, durchaus auf eine fixe Auswahl an Schriftarten begrenzt. [vgl. LiPl97]

[2] vgl. [Wild99] p.436

[3] vgl. hierzu den Beitrag des späteren Leiters der SVG-Arbeitsgruppe Chris Lilley im WWW-Talk-Forum vom 22. November 1994: „Most [of today's CGM files] are 1987 CGMs generated by Corel Draw…” [mittlerweile offline]

[4] vgl. hierzu den Kommentar des CGM-Mitautors Lofton Henderson im WWW-Talk-Forum in Antwort auf die Anregung Chris Lilleys: „Vector Graphics for the WWW“ [WWW-Talk, 4. Dezember 1994, mittlerweile offline]:

[5] „Weiterhin können mit der […] Sprache SGML CGM-Dateien importiert werden.“ [Born90] p.417

[6] Anm: Neben der Reinen Interpretation des Formates herrscht offensichtlich insbesondere Unklarheit über die Form der Darstellung: Einerseits erschien Anti-Aliasing zu aufwendig, die rein lineare Darstellung erschien jedoch dem Format nicht angemessen

[7] “There has been a long-term requirement in the World Wide Web for an additional type of graphics object to display scalable or vector graphics.” [LiPl97] – vgl. hierzu auch [Stub96] sowie [4.1]

[8] “Due to some properties of CGM, WebCGM does not fulfill all requirements for scalable graphics on the web as specified by W3C” [Wild99] p.436

[9] URL: http://www.cgmopen.org [5.2.03]

4.3.1.2 SVF: Simple Vector Format

Ähnlich erging dies bereits einige Zeit zuvor dem bereits in den Frühen 90er Jahren von den „Vätern" des *Mosaic*-Browsers am NCSA entwickelten,[1] so genannten *Simple Vector Format* [SVF95], welches ebenfalls auf Basis eines äußerst reduzierten *Primitive*-Umfangs nahezu ausschließlich „für das Anzeigen von CAD-Zeichnungen im Internet" [vgl. NeWi00] konzipiert war. Obgleich sich der SVF-Funktionsumfang nur auf das aller-Notwendigste beschränkte, gilt das Format heute als der „erste Ansatz zur Darstellung von Vektoren im WWW" überhaupt.[2] Trotz Veröffentlichung eines funktionstüchtigen Browser-Plug-Ins [Sher98] gelang den Entwicklern der Spezifikation jedoch weder eine zunächst erhoffte Standardisierung,[3] noch eine nennenswerte Verbreitung des Formates.[4]

4.3.1.3 Autodesks Design Web Format

„Weitaus erfolgreicher", so zumindest [Oebb97], konnte sich hingegen das proprietäre *Design Web Format* (DWF) der Firma Autodesk am Markt behaupten: Da Autodesks CAD/CAM-Lösung *AutoCAD* noch Mitte der 90er Jahre eine nahezu Monopol-artige Stellung auf dem CAD-Sektor einnahm, gelang es dem Software-Unternehmen überaus geschickt, eine beachtliche Verbreitung des zugehörigen *Whip!*-Plug-Ins[5] und somit eine nicht zu unterschätzende Akzeptanz des DWF-Formates zu erreichen. Obgleich die sich ständig ändernden Funktionen und Versionsnummern des (mittlerweile in die 6. Generation [vgl. Fish02] übergegangenen) Formates wie auch des dazugehörigen Plug-Ins, das mittlerweile zudem unter dem Namen *Autodesk Express Viewer* firmiert,[6] doch deutlich zur Verwirrung der Anwenderschaft beigetragen haben, kann *DWF* dennoch, trotz entgegengesetzter Bemühungen der CGM-Community wie auch konkurrierender Unternehmen, als derzeit erfolgreichstes, Web-basiertes CAD-Format bezeichnet werden.

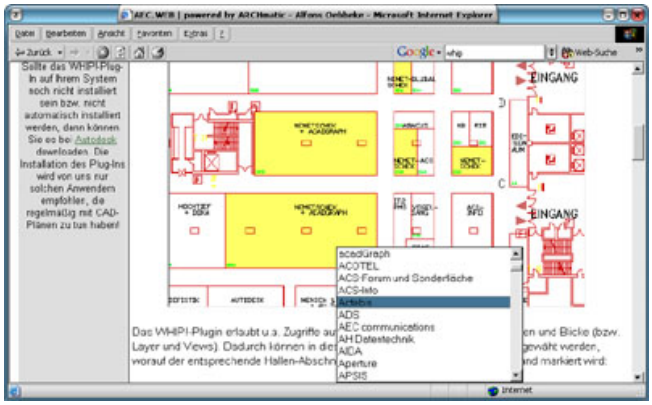


Abb. 4.3.1.1: Anwendung des Whip!-Plug-Ins (r.)

Dennoch muss natürlich zugleich einschränkend festgestellt werden, dass dies auch ausschließlich für die spezielle Nische der CAD-Anwendungen zutrifft – zur Anwendung oder auch Verallgemeinerung dieser Ansätze auf die Anforderungen multimedialer Präsentationen erscheinen die derzeit verfügbaren Lösungen hingegen völlig ungeeignet:

> *Es gibt keine Animationsmöglichkeiten, und dadurch ist die Ansprechbarkeit durch JavaScript nur bedingt nützlich.*
>
> [NeWi00]

Zwar lassen sich, wie auch [Oebb97] anmerkt, insbesondere dank der verschiedenen JavaScript-Funktionen, die eine Interaktion von DWF-Dateien insbesondere mit der jeweils umgebenden HTML-Seite erlauben, „interessante Effekte" erzielen – ein „Missbrauch" des CAD-Webformates im Rahmen einer vektor-basierten Web-Präsentation erscheint jedoch nicht nur unzweckmäßig, sondern angesichts durchaus vorhandener Alternativen nur wenig sinnvoll: So stellt an dieser Stelle gleich eine Fülle hierfür weitaus eher

---

[1] vgl. [Oebb97]
[2] vgl. [NeWi00]
[3] Anm: Ersteres ist zweifellos auf die parallelen Bemühungen, den CGM-Standard mittels WebCGM „Internet-fähig" zu bekommen, zurückzuführen.
[4] Nie aussage aus [NeWi00], „ab 1997" habe es keine Entwicklung mehr gegeben, ist jedoch unzutreffend [vgl. SWF95]
[5] vgl. hierzu [Oebb97]
[6] URL: http://www.autodesk.com/global/expressviewer/installer/ExpressViewerSetup.exe [5.2.03]

geeignet scheinender, primär grafisch orientierter Vektor-Formate dank entsprechender Web-Schnittstellen alternativ zu den bisher diskutierten Ansätzen mitunter überaus beeindruckende, visuelle Möglichkeiten hinsichtlich interaktiven Präsentationen bereit.

4.3.2 Grafische Internet-Vektor-Formate

4.3.3 CMX und QuickSilver

Zumeist stellen diese vorgeblich „web-fähigen" Vektorformate zwar lediglich entsprechend „aufgemöbelte" Internet-Pendants bereits etablierter Grafikprogramme bzw. –Formate dar, wie beispielsweise Corels CMX-Format [Core95, vgl. Arpa96] – andere Ansätze vermögen hingegen die Vorzüge der vektor-basierten Bilddaten eines eigenen Grafikformates mit darüber hinausgehenden Web- und Präsentationseigenschaften zu verbinden: So bietet beispielsweise das aus Mircografx' *Designer*-Suite erzeugbare *QuickVector*-Format [Kauf96, Stub96] welches zunächst unter dem Namen *QuickSilver* [vgl. SeLo97, Kobl98] firmierte, zusätzliche Interaktivitäts- und Scriptingmöglichkeiten: Ähnlich wie der bereits etablierte Multimedia-Konkurrent *Shockwave* [s.3.5.4] lassen sich so dynamisch Bildinhalte manipulieren und sogar animieren:

> *In addition to its use of vector graphics, QuickSilver allows for interactivity within a Web page… [It] can make graphics change dynamically as the user hovers or clicks the mouse on certain areas. In addition to making visual changes to graphics, "hot" areas on a graphic can function as hyperlinks that take the user to new Web sites.*
>
> [Kauf96]

Im Vergleich zu dem bereits in [3.5.3] diskutierten, Animations-orientierten[1] *Director*-Plugin erscheint der Funktionsumfang von QuickVector freilich „äußerst eingeschränkt" [Kauf96] – Hierbei sollte jedoch die Tatsache nicht außer Acht gelassen werden, dass die im Hinblick auf Web- und Multimedia-Tauglichkeit hinzugefügten Funktionen lediglich vorsichtige Erweiterungen eines ursprünglich rein statischen Vektorformates darstellen. Aufgrund der konsequenten Schlichtheit des Formates – schließlich wird zusätzlich zum Grafikprogramm Designer kein weiteres Multimedia-Authoring-Tool in Anspruch genommen – ist die erzielte Funktionalität trotz des schmerzlich vermissten Anti-Aliasings dennoch durchaus beachtlich und auch hinsichtlich Web-basierter Präsentationen durchaus interessant. Bereits weit vor dem Aufkauf der Herstellerfirma Micrografx versank dieser Ansatz aufgrund des geringen Marktanteils des Unternehmens jedoch zusehends in der Versenkung.[2] Trotz der heute (mit aufgrund der Ignoranz seitens der Corel-Konzernführung) vollständigen Bedeutungslosigkeit des QuickVector-Ansatzes erscheint es jedoch durchaus ironisch, dass auch die Web-fähigen Vektorgrafik-Formate des Mutterkonzerns Corel in Form von CMX [vgl. Arpa96] und dem Java-basierten *Barista* [Core98], die beide einen eher präsentations- und dokumentorientierten Ansatz verfolgen,[3] mangels allgemeiner Akzeptanz als gescheitert bezeichnet werden müssen.

4.3.4 Das Plug-In-Wunder: Xara Flare

Durchaus erwähnenswert erscheint unter diesem Zusammenhang auch die nähere Betrachtung eines ebenfalls vektor-basierten Web-Formates des („zwischendurch" im Übrigen ebenfalls von Corel aufgekauften),[4] britischen Softare-Unternehmens *Xara* mit dem klangvollen Titel „Flare" [Moir97]. Bei diesem Ansatz überzeugen auch nach näherer Untersuchung nicht nur die Eigenschaften des Formates selber, sondern eben-

---

[1] vgl. hierzu die entsprechende Diskussion in [2.1]

[2] vgl. hierzu den entsprechenden Artikel des Nachrichtendienstes Golem.de von Andreas Donath am 18.12.1999: http://www.golem.de/9912/5581.html [6.2.03]: „Mit Quicksilver 3 versucht Micrografx seit geraumer Zeit Macromedias Flash beizukommen, doch leider ist die Verbreitung des erforderlichen Plug-Ins nicht sehr groß."

[3] Anm: Dennoch stellen beide Ansätze weiterhin *Vektor*-orientierte Formate dar. [vgl. SeLo97]

[4] Anm: Dies gilt jedoch lediglich im Zusammenhang mit der Software CorelXara (1 und 2) – mit der Programmversion Xara X sind sowohl Produkt als auch Unternehmen wieder vom kanadischen Corel-Konzern unabhängig.

so die Display-Funktionalität des zugehörigen *Viewers*: So unterstützt das „äußerst kompakte“ [Bens98] Flare-Format mit der Erweiterung “.WEB” nicht nur komplexe Verläufe, Transparenzen und weichgezeichnete Schlagschatten nativ und ohne Qualitätsverlust – auch selbst kreierte Muster und Photoshop-ähnliche Filter lassen sich in Flare-Grafiken direkt abbilden, ähnlich wie es im Rahmen des derzeitigen SVG-Standards [SVG01, s.5.4] erst seit kürzerer Zeit möglich ist. Im Gegensatz zum XML- und damit auch textbasierten SVG[1] werden Flare-Grafiken jedoch äußerst effizient *binär*-komprimiert und belegen somit lediglich „bis zu ein Sechstel“ des ursprünglichen Speicherplatzes.[2] Auch das zugehörige, neben den für unsere Zwecke relevanten Vektor-funktionen zusätzlich noch eine so genannte Xara3D-Engine[3] unterstützende Plug-In [Xara02] ist mit rund 250 KB zwar „recht schlank“ geraten,[4] beherrscht aber dennoch automatisches, vollständiges Anti-Aliasing selbst bei 8-bittiger Farbauflösung in „schier unglaublicher Schnelligkeit“ [vgl. Arah01]. Ein weiteres, beeindruckendes Feature stellt überdies die „hervorragende“ [5] *Zoom*-Eigenschaft des Flare-Viewers dar, der Echtzeit-Vergrößerungen von bis zu 25.000 % spielend meistert. Obgleich die Performance des Viewers zumindest subjektiv „etwas schwächer“ ausfällt als die entsprechende, „wahnsinnige“ [6] Rendering-Geschwindigkeit des Authoring-Tools (wobei dies natürlich an der Code-Umgebung des Internet Explorer liegen mag)[7], stellt Xara’s Flare zumindest nach Ansicht zahlreicher Grafik-Designer [vgl. Maci00] „das universelle, vektor-basierte Grafikformat des Web schlechthin“ dar.[8]

Auch bei genauerer Betrachtung der im Web verfügbaren, tatsächlichen Implementierungen der Flare-Technik (d.h. tatsächlich vorliegender Xara-Grafiken)[9] fällt auf, dass die entsprechenden Ergebnisse nicht nur aus technischer Hinsicht, sondern – sicherlich unterstützt durch das gut umgesetzte Echtzeit-Anti-Aliasing – auch aus ästhetischer Perspektive nahezu durchweg positiv ausfallen, was sicherlich nicht nur an der „beeindruckenden“ Usability der Xara-Tools selbst,[10] sondern ebenso an den interessanten, technischen Möglichkeiten (und hierbei insbesondere den Schatten- und Transparenzfunktionen) des Flare-Formates liegen dürfte.

Trotz der Existenz einer daher kaum verwunderlichen, „verschworenen“ Xara-Fangemeinschaft ist es dem Web-Format dennoch nicht gelungen, eine größere Verbreitung sowohl des Plug-Ins als auch einer Flare-Verwendung selber zu erreichen:

> *The plug-in has been available since 1997 and it doesn’t seem to have caught on…* [Bens98]

Web-Experte Daniel Will-Haris bemerkt überdies, das, „obwohl .WEB ein wunderbares Format ist“,[11] ein langfristiger Erfolg des Formates dennoch eher unwahrscheinlich sei:

> *Corel* [zwischenzeitlicher Distributor von *CorelXara*, Anm.d.Verf.] *is not behind it, and on its own, Xara lacks the political clout to make it a standard on the web.*
>
> [Will01]

---

[1] vgl. hierzu auch die entsprechenden Vergleiche der *Flare-* mit korrenspondierender SVG-Technologie in [Arah99,01] sowie [Maci00]

[2] vgl. [Bens98]

[3] vgl. hierzu http://www.xara.com/products/xara3d [6.2.03]

[4] vgl. hierzu auch entsprechenden Kommentar der kartographischen Flare-Implementierung des britischen *Coventry and Warwickshire Network* (1997) http://www.coventry.org.uk/maps : “The XARA plug-in is only 220k in size and will therefore only take a minute or two to download. It is also straightforward to install.” [URL am 6.2.03 verifiziert]

[5] s. ebenda.

[6] vgl. [Arah01]

[7] vgl. hierzu den entsprechenden Beitrag von Marko Raos (2000) *A Word on File Formats* [*mittlerweile offline*]

[8] s. ebenda.

[9] s. hierzu beispielsweise http://xara.com/gallery [6.2.03] oder http://www.xaranordic.dk/samples.htm [6.2.03]

[10] vgl. einen entsprechenden Kommentar des US-amerikanischen *PC Magazine*: “Using CorelXARA is like driving a sleek red Ferrari convertible in the springtime through a beautiful park. With an interface so clean and simple, CorelXARA will make you wonder why people thought illustration software was so hard to master in the first place.” [zitiert aus: Marko Raos, “A Word on File Formats”, s.o.]

[11] vgl. [Will01]

Dieser „erhebliche Wettbewerbsnachteil“, so Will-Harris, begründe sich unter anderem auch darin, dass dem Flare-Format zugleich die „Fähigkeiten zur Interaktion und Animation“, über die etwa das marktführende Animations-Format *Flash* [s.4.5] verfügt, derzeit noch fehlten. Obgleich im Gegensatz zu Will-Harris' Ansicht durchaus interaktive Elemente möglich sind,[1] hat sich offensichtlich die pessimistische Prognose des Autors hinsichtlich der „politischen Entwicklung“ vektor-basierter Web-Formate bedauerlicherweise bestätigt: So konnte sich das Format trotz beeindruckender Funktionalität bedauerlicherweise nicht durchsetzen, und auch die Tatsache, dass die Flare-Spezifikation seit deren Erstveröffentlichung [Moir97] nicht mehr aktualisiert wurde, weist darauf hin, dass das Unternehmen seine Bestrebungen hinsichtlich des .WEB-Formates mittlerweile eingestellt hat.

Abschließend muss daher im Hinblick auf das Gros der Vektor-orientierten, grafischen „Web“-Formate[2] festgestellt werden, dass es hauptsächlich aufgrund der bereits in [s.3.5.4] beleuchteten *Plug-In*-Problematik sowie marktstrategischer „Schierigkeiten“ [vgl. Will01] im Umfeld des WWW[3] bislang keinem der soeben betrachteten Ansätzen gelungen ist, über das reine „proof of concept“-Stadium hinaus ausreichende Akzeptanz oder auch nur Bekanntheit im Rahmen der „Web Community“ zu erlangen. Trotz mitunter durchaus interessanter Ansätze wie etwa dem Xara Flare-Format [vgl. Arah01] konnte weder angeblich „fortschrittlichen“[4] Web-Entwicklern noch den maßgeblichen Browser-Herstellern der in [4.1] dargelegte Mehrwert auch statischer,[5] Vektor-orientierter Grafiken nahe gelegt werden. Dies schränkt nun wiederum die derzeitige Attraktivität der soeben betrachteten Formate hinsichtlich Web-basierter Präsentationen (die wir nach den in [4.1] angestellten Betrachtungen ja durchaus im Bereich einer vektororientieren Lösung ansiedeln wollen) im Rahmen dieser Diplomarbeit bedauerlicherweise doch deutlich ein.

4.4 Vektorbasierte Präsentationsdaten im Web

4.4.1 Das Scheitern der Präsentations-Plug-Ins

Angesichts des bereits in [2.2] betrachteten, außerordentlichen Marktpotentials Business-orientierter Präsentationslösungen und in Verbindung damit natürlich die besondere Attraktivität des Internet für diese Zwecke [s.3.1-2] erstaunt es daher freilich um so mehr, dass es nicht nur den soeben betrachteten, rein statische Vektordaten repräsentierenden Grafikformaten bislang nicht gelungen ist, im Internet erfolgreich Fuß zu fassen, sondern dem Großteil auch primär auf *Präsentationen* gemünzten Web-Formaten bisher ein ähnliches diesbezügliches Schicksal beschieden war: So führten die Bemühungen der Softwarefirma RadMedia um eine entsprechend „Web-fähige“ Präsentationsläsung namens „PowerMedia“ [Nevi96] lediglich zum anschließenden Kurswechsel mit PowerPoint-„Kuschelstrategie“ [vgl. Froo96] und der letztendlichen Auflösung des Unternehmens selbst.

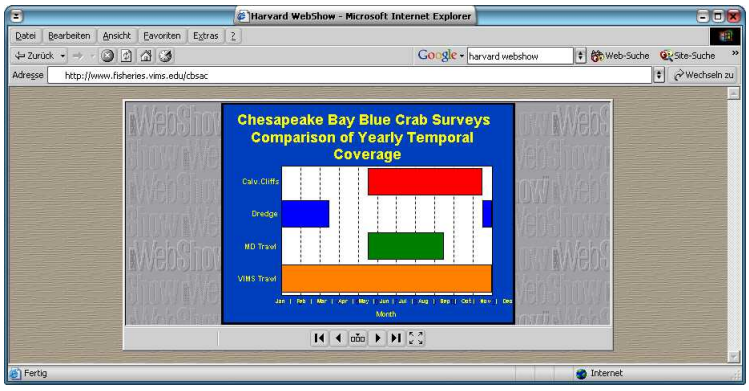


Abb. 4.4.1: Web-Präsentation mit Harvard WebShow (l.)

[1] s. hierzu etwa: *Coventry and Warwickshire Network* (1997) http://www.coventry.org.uk/maps [6.2.03]

[2] Ann: In den bisherigen Untersuchungen wurden bislang ausschliesslich binär-codierte (Ausnahme: CGM, das sowohl einen binär- als auch einen textbasierten Modus besitzt), statische Formate betrachtet.

[3] Anm: Dies bezieht sich insbesondere auf die „Verbreitung“ eines Formates bzw. Plug-Ins, das nicht nativ durch Internet-Browser unterstützt wird („Eine nahezu unmögliche Aufgabe“), vgl. [Will01]

[4] Anm: Den Begriff der „Progressivität“ bei Webdesignern mag ich hingegen durchaus anzweifeln, da in der Regel meist eine „Konservative Lösung“ (etwa in Form von Animierten GIFs oder Flash) angestrebt wird, um maximale Kompatibilität zu erreichen.

[5] Anm: Diese Unterscheidung bezieht sich hierbei auf das fast ausschließlich Animationsbasierte *Flash*-Format [4.5], welches hingegen im Gegensatz zu den soeben beleuchteten Ansätzen durchaus Erfolge feiern konnte.

Nicht anders erging es dem ebenfalls auf Business-Präsentationen gemünzten „Astound Web Player“ [vgl. Brow96] der Firma *Gold Disk*, die sich nach dem Aufkauf durch „Genesys“ nunmehr auf Konferenz-Software konzentriert. Auch die entsprechenden Präsentations-Plug-Ins von Harvard Graphics [Shne02],[1] welches zunächst als „ASAP“ [vgl. Olip96, Brow99], seit 1997 jedoch unter dem Namen „Harvard Web-Show“ [SPCO97] eher unbemerkt und ungenutzt „dahinsiecht“ [Bell00:9], und des ehemaligen *Freelance* [s.2.3.1] mit dem klangvollen Namen „Lotus Web Screen Show Player“ [FLSP99], wurden nach ihrer Veröffentlichung Ende der 90er Jahre und des daraufhin leider ausbleibenden Erfolges seither nicht mehr aktualisiert, verbleiben aber dennoch wohl, eher aus Verlegenheit, noch auf den Webservern der Unternehmen zum Download.[2]

Frappierenderweise erging es dem im Offline-Bereich dominierenden Präsentations-Monopolisten Microsoft in Form des „Bestsellers“ PowerPoint an dieser Stelle ebenso: Da das Unternehmen den Internet-Boom und die Attraktivität des WWW als interessantes Medium zum Präsentations-Deployment zunächst „vollständig verschlafen“ hatte [vgl. Bell00] und erst im Zuge der 97er-Version einen entsprechenden (wenn auch technisch zunächst völlig unbefriedigenden)[3] HTML-Exportfilter in sein wohlbehütetes Präsentationsprodukt integrieren konnte [s.2.3.4], existierte etwa im Vorfeld noch der 95er-Version keinerlei Ansatz, die liebgewonnene Cash-Cow fit für das Internetzeitalter zu machen – eine Marktlücke, die als erste die Entwickler des kleinen Software-Unternehmens *Net-Scene* zu füllen wussten: über das so genannte „PointPlus“-Plugin[4] eröffnete der israelische Pionier Web-orientierten PowerPoint-Benutzern die daraufhin auch rasch genutzte Möglichkeit, PowerPoint-basierte Präsentationen in Form des intermediären so genannten CSS-Formates[5] kaum komprimiert über das Internet nutzbar zu machen. Schon bald jedoch erkannte der Software-Riese Microsoft selbst das enorme Potential einer solchen Lösung und veröffentlichte noch 1995 ein eigenes Plug-In (freilich wiederum mit einem weiteren, zur PointPlus-Variante inkompatiblen Übertragungsformat) unter dem klangvollen Titel „Microsoft PowerPoint Animation Player and Animation Publisher“ [vgl. PaAd98].

4.4.2 Microsofts Plug-In-Flop

Obwohl sich diese Lösung zwar aufgrund der weiten Verbreitung PowerPoints aus Erstellersicht zunächst großer Beliebtheit erfreute, bliebt die Download-Rate des Plug-In-Programms selber und somit auch die Verbreitungsmöglichkeit dieser ausschließlich für Windows verfügbaren[6] und offensichtlich „mit der heißen Nadel gestrickten“ Lösung doch erschreckend gering – wohl auch mit deshalb, weil das Modul mit gut 800 KB für damalige Internet-Bandbreiten offensichtlich empfindlich zu umfangreich ausfiel. Da jedoch bereits 1997 eine (wenn auch, wie in [2.3.4] erkannt, recht zweifelhafte) „native“ HTML-Exportfunktion zur Verfügung stand, und PowerPoint-Dateien überdies via COM-Schnittstelle alsbald direkt im Internet Explorer dargestellt werden konnten,[7] versank diese recht unausgereift erscheinende Zwischenlösung zur großen Erleichterung des Software-Konzerns alsbald in Bedeutungslosigkeit. Die Tatsache jedoch, dass das Unternehmen sich heute jegliche Referenz auf die damalige „Jugendsünde“ verbietet, weist jedoch mitunter darauf hin, dass das erfolgsverwöhnte Unternehmen diese für Microsoft ungewohnte (da überwiegend negative) Erfahrung mit der diffizilen Plug-In-Problematik wohl am liebsten schnellstmöglich vergessen will.

Die verblüffende Erkenntnis, dass es selbst dem mächtigen Software-Riesen *nicht* gelang, mit einem Plug-In des wohl populärsten Präsentationsformates schlechthin eine auch nur nennenswerte Verbreitung im Web

---

[1] Hersteller: Software Publishing Corporation, Fairfield (New Jersey)

[2] vgl. [SPCO97, FLSP99]

[3] vgl. [Will02]

[4] vgl. hierzu einen entsprechenden Artikel des *Byte*-Magazins: “Now You Can Put PowerPoint Shows on the Internet”, Oktober 1996

[5] Anm: Ironisch ist auch hier natürlich die Namenserweiterungs-Überschneidung mit dem Späteren Cascading Stylesheets-Standard.

[6] Anm: Es handelt sich bei dem genannten Produkt bezeichnenderweise ausschließlich um eine ActiveX-Komponente

[7] s. hierzu die entsprechende Diskussion in [2.3.4]

zu erreichen, lässt an dieser Stelle freilich tief blicken: Aufgrund des zum Veröffentlichungszeitpunkt noch äußerst schwachen Browser-Standbeins des Unternehmens (Version 1.0 des Internet Explorers war unter den Code-Namen „Chicago“ bzw. „O'Hare“ erst Mitte desselben Jahres erschienen) und dem noch frischen Zerwürfnis mit Noch-Maktführer Netscape schien eine *Integration* des Plug-In-Moduls in führende Browser-Produkte seinerzeit unmöglich – ebendiese Integrationsstrategie, so kann nach den in diesem Kapitel sowie bereits in [3.5.3] gewonnenen Erkenntnissen zweifellos festgestellt werden, stellt jedoch eine zentrale Bedingung für eine grundsätzliche Verbreitung der entsprechenden Plug-In-Software dar.

Da jedoch (im Gegensatz übrigens zu bereits in [3.3.2] diskutierten, Pixelorientierten Formaten) eine Integration der in diesem Kapitel beleuchteten vektorbasierten Formate (und dies schließt die soeben angesprochenen, durchweg auf diesem Konzept[1] beruhenden Präsentations-Formate mit ein) in keiner Form im Rahmen eines „mehrheitsfähigen“ Browsers stattgefunden hat, überrascht es aufgrund dieser Erkenntnis nur wenig, dass dem primär auf dem Hypertext-Modell fußenden (ergo größtenteils statischen) Vektor-Formaten eine größere Verbreitung im WWW verwehrt blieb. Daher beinhalten diese auf dem Plug-In-Modell fußenden Format-Ansätze zwar aus technischer Hinsicht durchaus interessante Ideen, sind hinsichtlich einer praktischen Anwendung, in unserem Falle etwa im Rahmen einer Web-basierten Präsentationslösung, aus „pragmatischer“ Perspektive[2] jedoch nur eingeschränkt relevant.

4.4.3 Der integrale Ansatz: Adobes Bravo und Vertigo

Aufgrund dessen sind im Gefolge des „abschwellenden Plug-In-Booms“ [Schr00] schon bald Bemühungen insbesondere auf Grafik-Software spezialisierter Unternehmen deutlich geworden, umfassende (und ebenfalls vektorbasierte) Grafik-Frameworks zu entwickeln, die eine „Print-ähnliche, grafische“ und überdies „stets exakt gleiche“ Darstellung nicht nur auf verschiedene Systeme optimieren [vgl. Airb96], sondern deren zugrunde liegendes Format darüber hinaus auch über mehrere Web-Schnittstellen vertrieben werden kann. Als mustergültiges Beispiel für einen derartigen Ansatz kann etwa Adobes 1996 vorgestellte *Bravo*-API angeführt werden, welche „die Grafikqualität im Internet erheblich verbessern und Transfer sowie deren Aufbau beschleunigen“ sollte.[3] Zur Optimierung der systemübergreifenden Darstellung setzt die Bravo-Grafikschnittstelle beispielsweise sowohl auf den Windows-nativen Grafiktreiber GDI, als auch auf die QuickDraw GX-Engine der Macintosh-Betriebssysteme auf [vgl. Meis97]. Setzte das interne Grafikformat des Bravo-Frameworks noch auf dem PostScript-Modell [s.4.2.1] zur Repräsentation grafischer Primitives auf [vgl. Essi96], so wuchert die plattform-unabhängige Darstellungsschicht derweil mit 24-Bit-Echtfarben, einstellbarer Transparenz und vollständigem Anti-Aliasing [vgl. Flüg96].

Da dieses zentrale Framework, Anfang 1996 von Adobe vorgestellt, jedoch noch völlig Realisierungsunabhängig war, bedurfte es neben konkreten Player-Implementierungen freilich noch funktionsfähiger Authoring-Software. In Zusammenarbeit mit den Firmen Sun, Microsoft und AT&T,[4] welche unter anderem die konkreten Implementierungen des Grafik-Renderings übernehmen sollten, kündigte Adobe daher schließlich die auf dem Bravo-Ansatz fußende Player- und Authoring-Software *Vertigo* an, die durch zusätzliche Animations- Interaktions- und Multimediaeigenschaften eine „direkte Alternative“ zum Marktführer Director [s.2.1, 3.5.4] eröffnen sollte [vgl. Burk96]. Darüber hinaus stellten die Bravo-Entwickler überdies eine baldige Integration des Frameworks in die Programmiersprache Java [s.3.5.3] in Aussicht, mit der sich die soeben diskutierte Plug-In-Problematik „elegant umschiffen“ ließe:

---

[1] Anm: Hiermit ist selbstredend das Konzept der Vektor-Orientierung wie in [1.1] besprochen gemeint.

[2] Anm: Dies bezieht sich freilich erstlinig auf die clientseitige Unterstützung des jeweiligen Formates und mit ihr natürlich auch die entsprechende Multimedia-Lösung. Im Konkreten ist dies etwa die Frage „Wie wahrscheinlich ist es, dass der letztendliche Internet-Nutzer die Präsentation tatsächlich (ohne zusätzlichen Installationsaufwand) problemlos betrachten kann?“

[3] vgl. [Meis97]

[4] vgl. [Essi96]

> *JavaSoft, the spin-off from Sun that will carry forward all Java updates, has said that it will make Bravo the standard 2D imaging model of a future version of Java.*
>
> [Dyso96]

Am Beispiel der kurz zuvor von derselben Firmenallianz angekündigten Internet-Fontlösung "OpenType", welche zugleich Grundlage der Schrift-Einbettung Bravos bilden sollte, zeigte sich jedoch bereits, wie [Meis97] nahezu hellseherisch kommentierte, „der Wert solcher Vereinbarungen": Während OpenType aufgrund der Versäumnisse der einzelnen Partner nie fertiggestellt wurde und das WWW daher noch heute vergeblich auf ein standardisiertes Font-Embedding wartet,[1] blieb auch die Bravo-Allianz die Umsetzung der Vertigo-Software der Öffentlichkeit bislang schuldig. Ebensowenig in die Tat umgesetzt wurde überdies eine Java-Integration der Bravo Grafik-API: So sucht man im derzeitigen Java Imaging-Model die von Seiten Adobes lauthals angekündigten „grafischen Funktionen"[2] derzeit vergebens – von den in [Dyso96] versprochenen Bemühungen JavaSofts um einen diesbezüglichen Ausbau der Programmiersprache kann also, wie an dieser Stelle bedauerlicherweise festgestellt werden muss, keinesfalls die Rede sein.

4.4.4 Applet meets Vector: Die Java-Offensive

Wenn auch das aus technischer Hinsicht überaus interessante Konzept dieses Frameworks wohl aufgrund verschiedener Unstimmigkeiten zwischen den einzelnen (im Prinzip ja „verfeindeten")[3] Unternehmen der Bravo-Allianz bereits vor einer möglichen Realisierung gescheitert ist, so existiert durchaus eine ganze Reihe von Ansätzen, die insbesondere den Grundgedanken, Vektordaten neben der problematischen Plug-In-Schnittstelle über die systemübergreifende und via Applet-Technik überdies Web-fähige Java-Plattform [s.3.5.3] im Internet verfügbar zu machen, aufgegriffen haben.

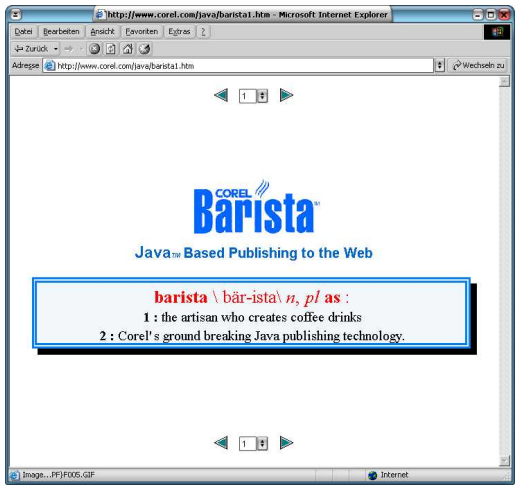


Abb. 4.4.4.1: Corels Barista-Applet in Aktion (rechts)

Neben Corels „schwerfälliger" [vgl. Gutz97], „fehlerbehafteter" [4] und mittlerweile daher nicht mehr weiterentwickelter[5] *Barista*-Technologie [Core98, s. hierzu auch 3.5.3] erscheint im Rahmen vektorbasierter Java-Lösungen besonders das *Colada*-Format des Grafik-Programms *Canvas* von Deneba Systems von besonderem Interesse: Neben der bereits obligatorischen Einbettung diverser Rasterformate vermag an dieser Stelle insbesondere das vollständige Anti-Aliasing[6] der Schrift- und Vektordaten zu überzeugen. Trotz noch erheblicher Effizienz-Steigerungen bei der Einführung der zweiten Colada-Generation [vgl. Dudr98] ist es diesem interessanten Ansatz (dem im Übrigen auch ein durchaus überzeugendes Grafik-Programm zugrunde liegt) dennoch nicht gelungen, nennenswerte Verbreitung im Web zu erlangen – „nicht einmal auf der Webseite des Hersteller-Unernehmens Deneba selbst", so höhnt etwa [Duda99], komme die Colada-Technik noch in „irgend einer Form zum Einsatz".[7]

---

[1] Anm: Es gibt zwar bereits eine Lösung, die jedoch keinerlei Implementierung in irgend einem Browser aufweisen kann: „Vom breiten Publikum unbeachtet ist seit längerer Zeit Bitstreams TrueDoc lieferbar…" [Meis97]. Siehe zu dieser Thematik auch [Wild99] pp.443f.

[2] Anm: Insbesondere die 24-bit-Farbtiefe, Transparenz und vollständiges Anti-Aliasing wären an dieser Stelle freilich überaus interessant gewesen.

[3] s. hierzu auch die unterschiedliche Haltung einerseits von Sun und Adobe sowie auf der anderen Seite Microsofts bezüglich XML-basierter Vektorformate im Vorfeld der Entwicklungsarbeiten zu SVG [5.3.2-3]

[4] vgl. Tom Arah: "Deneba Canvas 6", *Designer Info*, April 1999.

[5] vgl. hierzu die entsprechende Diskussion in [3.5.3]

[6] vgl. den Bericht „Colada" der japanischen MacWeek-Ausgabe (Tokio, 7. Oktober 1997)

[7] vgl. [Duda99] p.54

Unter Berücksichtigung der bereits in [3.5.3] beleuchteten, problematischen Aspekte in Verbindung mit der Anwendung Java-basierter Applets lässt die Tatsache, dass den soeben betrachteten, vektorbasierten Ansätzen ein quantitativer[1] Erfolg bislang versagt blieb, in Verbindung mit dem bereits zuvor[2] betrachteten *Scheitern* Pixel-orientierter Multimedia-Lösungen auf Applet-Basis („Coda" von RandomNoise [Bers97, Wagn98], *Web Knight*'s „Instant Coffee" [Hess97, Perr97], sowie „Jamba" [Shah96, Para97, Kurz97] von *AimTech*) daher den Schluss zu, dass auch die Java-Technologie an dieser Stelle im Vergleich zum Plug-In-Prinzip trotz erheblicher Vorzüge [s.3.5.3] offensichtlich bislang nicht den erhofften „Schlüssel zum Erfolg" darstellt.

Im Zusammenhang mit Web-basierter Echtzeit-Präsentation kommt an dieser Stelle insbesondere dem bislang unter diesem Aspekt kaum behandelten *Performance*-Problem der Java-Umgebung erhebliche Bedeutung zu: Nicht nur der Lade-Vorgang des Applet-Loaders sowie der JVM, sondern ebenso die bereits ausgesprochene „Trägheit" [Glue96] des Runtime-Systems (welches ja schließlich ein *interpretierendes* ist),[3] beeinträchtigen naturgemäß die Ablaufgeschwindigkeit des auszuführenden Programms und somit auch den Präsentationseindruck des Nutzers. Speziell rechenintensive Effekte wie etwa aufwendige Übergänge oder auch (aus ästhetischer Sicht durchaus wichtiges) Anti-Aliasing von Schrift- und Vektorelementen können somit bei Realisierung der Präsentationslösung im Rahmen eines Java-Applets durchaus kritisch werden: So merkt etwa [Duda99] an, dass sich etwa mithilfe des Java-basierten Colada lediglich „langsame nicht-editierbare Webseiten" generieren ließen,[4] während [Wagn98] den „Dokumenten"-Aspekt Java-basierter Applet-Grafiken zu recht schmerzlich vermisst.[5] Wesentlicher „Knackpunkt" Applet-basierter Lösungen ist und bleibt hingegen die *Performance*-Schwäche der Programmiersprache selbst, aus der sich letztendlich auch eine kontroverse Diskussion um die mitunter angezweifelte Eignung des Java-Ansatzes hinsichtlich durchaus rechenintensiver Multimedia-Effekte entwickelt hat:

> *Embedded Java […] is not fully suited for multimedia rich content.* [Shoy01]

Neben diesem, in meinen Augen durchaus lösbaren Aspekt[6] gilt es bis zur vollständigen, sinnvollen Anwendung der Java-Umgebung[7] im Rahmen multimedialer Web-Präsentaionen freilich überdies die derzeit noch unzureichende, clientseitige Ausbreitung und Akzeptanz der Applet-Technologie [s. hierzu auch die entsprechende Diskussion in 3.5.3] erheblich zu steigern. Derzeit vermag zwar die primär aus unternehmensstrategischen Überlegungen motivierte Blockadehaltung insbesondere des Microsoft-Konzerns [vgl. Viol01] noch eine vollständige Verbreitung entsprechender „Virtual Machines" auf sämtlichen Browser-Clients zu verhindern – meiner Ansicht nach steht aber der Java-Sprache selber, dem „Hottest Thing on the Web" [Schm95], und im Hinblick auf multimediale, webfähige Multimediapräsentationen insbesondere natürlich dem Applet-Framework der eigentliche Durchbruch im Bezug auf großflächige Anwendung im Web erst noch bevor.

Während die soeben angesprochenen Probleme statischer Vektorformate im Hinblick auf „Web-Akzeptanz" auch heute noch teils ungelöst im Raum stehen, konnte sich derweil der Vertreter einer gänzlich anderen (da animationsbasierten) Herangehensweise bereits die diesbezügliche „Marktführerschaft" er-

---

[1] Anm: Die Quantität bezieht sich in diesem Zusammenhang auf die Anzahl der mit den entsprechenden Techniken realisierten Lösungen.

[2] s. hierzu [3.5.3]

[3] Anm: Im Gegensatz zu den kompilierenden Systemen, in denen die Programmlogik bereits in direkt ausführbarer Maschinensprache vorliegt, muss in diesem Falle der „vorbehandelte" Byte-Code noch zur Laufzeit ausgewertet („interpretiert") und ausgeführt werden.

[4] "Problem is, Colada produces slow, uneditable, Java-based Web pages." [Duda99] p.54

[5] "While a Java-only solution may offer some capabilities that can't be fulfilled by HTML, it's never going to wildly succeed in the long run. No matter how popular Java gets, it will never surpass the document-publishing aspect of the Web." [Wagn98]

[6] Anm: Neben intensiven Optimierungs-Bemühungen zahlreicher Java-Entwickler wird sich das Problem aufgrund erheblich steigender Rechenleistungen natürlich in der Theorie ohnehin von selbst lösen.

[7] s. hierzu [2.3.4] sowie [3.5.3]

gattern: Das „rich Internet content“-Format[1] *Flash* des kalifornischen Grafiksoftware-Konzerns Macromedia stellt durch mittlerweile nahezu vollständige[2] Player-Abdeckung und einer – wohl hierdurch bedingten – enormen Anwendung im Rahmen unzähliger Flash-Applikationen den mittlerweile unangefochtenen „de-Facto-Standard“[3] [vgl. NeWi00] multimedialer Internet-Anwendungen dar:

> *It's even possible that [the] Flash Player is now the most widely distributed piece of software on the Internet – ahead of Internet Explorer, Netscape Navigator, and Real Player.*
>
> [Gay01]

Aufgrund dessen erscheint freilich nicht nur die Frage, inwieweit sich diese dominierende Stellung der Flash-Technik etwa durch technische Aspekte des Flash-Formates begründen und verifizieren lässt, eine wichtige Rolle in der nun folgenden Betrachtung dieser Technologie, sondern ebenso marktbezogene Betrachtungen hinsichtlich der von Seiten Macromedias sowie Konkurrenz- und Partnerfirmen verfolgten Strategien und der daraus resultierenden, letztendlichen Verbreitung des es Flash-Ansatzes, von Interesse. Auch im Zusammenhang mit der Zielsetzung dieser Diplomarbeit [s.1.1] gilt es freilich unter anderem auf Basis dieser Erkenntnisse, die Eignung des Flash-Formates im Hinblick auf einen möglichen Einsatz im Rahmen multimedialer Web-Präsentation zu untersuchen und natürlich nicht zuletzt die Frage zu beantworten, inwiefern diese Technologie dem oftmals im Bezug auf Flash geäußerten Anspruch, das „multimediale Präsentationsmittel des Webs schlechthin“ [Pusc00] darzustellen, denn überhaupt gerecht wird.

4.5 Flash

4.5.1 Einführung und Entwicklungsgeschichte

4.5.1.1 Anders und doch gleich

Zunächst jedoch sticht bei entwicklungshistorischer Betrachtung der angeblich „so anderen“ [4] Flash-Technologie sogleich die Tatsache ins Auge, dass Ursprung und Herangehensweise des Formates den soeben behandelten Ansätzen zunächst in geradezu frappierender Weise ähneln: So stellte auch das erst erheblich später in das *FlashStudio* Authoring-Tool übergegangene Programm „SmartSketch“ [vgl. Mart95] der Firma FutureWave zunächst ein rein statisches, vektorbasiertes Zeichenprogramm dar – und der hierfür „eilends“ [Gay01] entwickelte Web-Player lediglich das korrespondierende, zunächst recht schlichte Internet-Pendant des zugrunde liegenden Formates:

> *1995 veröffentlichte die Firma FutureWave aus San Diego das vektororientierte Illustrationsprogramm SmartSketch und ein dazugehöriges Plug-In namens FutureSplash-Player, um die mit SmartSketch erzeugten Illustrationen mit einem Webbrowser betrachten zu können.*
>
> [Star01:6]

Interessant erscheint überdies, dass die erste Version dieser Web-Schnittstelle wie bereits die soeben betrachteten Lösungen in Form eines Java-basierten Applets realisiert wurde und daher durchaus überraschende Ähnlichkeit etwa mit Denebas *Colada* oder Corels *Barista* aufwies.

---

[1] vgl. [Macr02c]

[2] Im September 2002 lag die „Penetration“ des Players laut einer IDC NPD-Research-Studie bei 97.8% aller Browser-Endsysteme, vgl. [Macr02a]

[3] “Over the years, the Macromedia Flash File Format, as it is officially called by Macromedia, has become the de facto standard for vector graphics on the Web.” [Arty02]

[4] vgl. Michael Gregory (2000) “Flash is Different.” New York, 21. Februar 2000: http://1callhosting.com/help/flash_is_different.html [9.2.03]

4.5.1.2 Pen-Computing: Flashs dunkle Vergangenheit

Der Ursprung der SmartSketch-Software selber folgte jedoch einer gänzlich anderen Herangehensweise als die im vorausgegangenen Kapitel behandelten, professionellen Grafikprogramme und ihrer Formate: Obwohl auch Flash-Entwickler Jonathan Gay mit den zuvor Entwickelten Grafikprogrammen *SuperPaint II* [s.Abb.4.5.1.1] und *IntelliDraw* ebenso wie bereits Corel und Deneba ein PostScript-Ähnliches Modell verfolgt hatte, war Anwendungsprinzip und „*Painter's Model*" von SmartSketch hingegen auf die speziellen Anforderungen des so genannten *Pen Computing* hin ausgerichtet: Da das gesamte SmartSketch-Projekt auf dem Prinzip der (heute insbesondere in Form von *Palm*-Organizern bekannten) stift-basierten Handhelds und hier speziell der *GO*-Plattform [vgl. Meyr91] fußte, wurden insbesondere Füllungen nicht nach der altbekannten PostScript-Methode [s.4.2.1], sondern einem „unkonventionellen", linienbasierten Verfahren [vgl. Dowd00, Prob00a] realisiert, was sich natürlich ebenso in dem mit SmartSketch assoziierten Format niederschlug.

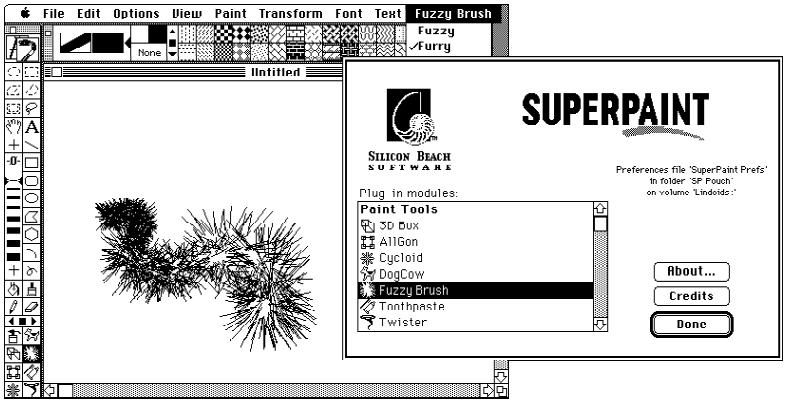


Abb. 4.5.1.1: SuperPaint (links)

Da nach dem Scheitern der GO-Handhelds 1993 dem SmartSketch-Ansatz sozusagen „die Basis entzogen" worden war [vgl. Wald00], portierte Gay die Software schließlich auf Mac- und Windows-Systeme – allerdings, ohne das Anwendungsprinzip des Palmtop-optimierten (und natürlich weiterhin statischen) Programms jedoch grundsätzlich zu ändern. Im Rahmen der SIGGRAPH'95 wurde den FutureWave-Entwicklern schließlich diskret nahe gelegt, das Web-Pendant des SmartSketch-Formates mit einer zusätzlichen Animations-Komponente zu versehen:

> *In the summer of 1995, we were at SIGGRAPH and got lots of feedback from people that we should turn SmartSketch into an animation product. We were starting to hear about the Internet and the Web, and it seemed possible that the Internet would become popular enough that people would want to send graphics and animation over it. So we began to add animation to SmartSketch.*
>
> [Gay01]

Das Ergebnis bildete schließlich der so genannte „SmartSketch Animator", dessen korrespondierendes Java-Applet die Wiedergabe vektorbasierter Animationen im Internet erlaubte. Dieses fiel jedoch zunächst derart „erbärmlich langsam" [Gay01] aus, dass der ursprünglich an der SmartSketch-Software interessierte Adobe-Konzern, schließlich auf eine Akquisition des Unternehmens verzichtete. Nach einem radikalen „Umbau" der Web-Komponente, aus Performance-Gründen nun wiederum auf Basis der *Netscape Plug-In API* [vgl. Olip96, s.3.5.4], sowie einer erfolgreichen Kooperation mit dem Microsoft-Konzern [vgl. Wald00] trat mit Macromedia Anfang 1996 endlich ein aussichtsreicher Kaufinteressent an die FutureWave-Entwickler heran. Nach einer verwirrenden Namenspolitik des Unternehmens (das Programm wurde zur Hervorhebung der Animations-Fähigkeiten zunächst von SmartSketch in „CelAnimator" umbenannt, um dann schließlich nur wenige Wochen später wiederum als „FutureSplash" vermarktet zu werden) veröffentlichte Macromedia die nur wenig modifizierte Version der Animations-Software schließlich im November 1996 unter dem Titel *Macromedia Flash 1.0*.

In diesem Zusammenhang verbleibt jedoch abschließend anzumerken, dass obgleich mit Macromedia MX derzeit bereits die 6. Edition der Animations-Software verfügbar ist, sowohl das Authoring-Tool als auch das Flash-Format selber noch immer einen durchaus beachtlichen Anteil der ursprünglichen SmartSketch- und FutureSplash-„Altlasten" mit sich herumtragen:

Flash has retained a good amount of code that was written for the GO pen computers. [Wald00]

4.5.1.3 Mit Animation zum Marktführer

Dieser Umstand konnte freilich nicht verhindern, dass es Macromedia "insbesondere dank äußerst aggressivem Marketing" [vgl. Raux01] sehr erfolgreich gelungen ist, Flash zur derzeit unbestrittenen Marktführerschaft zu verhelfen, sodass das zugehörige SWF-Format derzeit als „de-Facto-Standard" [vgl. NeWi00] bezüglich Webbasierter Animation und Multimedia zu betrachten ist. Dieser unzweifelhafte Erfolg ist indes gleich auf mehrere Faktoren zurückzuführen: So kommt neben den aus Designer-Perspektive verlockenden Funktionen des Flash-Players (speziell der vollständigen Kontrolle über Aussehen und Darstellung der Anwendung, sowie vollständiges Anti-Aliasing) insbesondere der enorme Verbreitung des Flash-Plug-Ins eine zentrale Rolle zu: Mit laut eigenen Angaben [vgl. Macro02a] derzeit „knapp 98 Prozent" kann in der Tat bereits von „nahezu vollständiger Abdeckung" der Flash-Funktionalität auf Browser-Ebene gesprochen werden.

> *Aufgrund dessen kann an dieser Stelle festgestellt werden, dass die Entscheidung für Flash längst nicht mehr allein einer rein ästhetischen Motivation folgt, sondern ebenso stark von ökonomischen und strategischen Beweggründen beeinflusst wird.*
>
> [Zmoe00][1]

Indes sollte die alleinige Feststellung dieser vollständigen, so genannten „Player Penetration" [Macr02a] freilich durchaus kritisch hinterfragt werden: So sind an dieser Stelle natürlich nicht nur die nackten Zahlen sowie Methodik der Erhebung der deutlich anzuzweifeln[2] – auch ist trotz der freilich nicht zu unterschätzenden Funktionalität des Flash-Formates wie auch des zugehörigen Players nicht davon auszugehen, dass sich diese „dominierende Marktstellung" [vgl. Zmoe00] allein aus deren technischen Vorzügen begründet: So haben unter anderem eine „aggressive Marketing-Kampagne" [Raux01], zahlreiche Kooperationen des Unternehmens insbesondere mit dem Microsoft-Konzern [vgl. Star01:11] zur engen Integration des Flash-Players in führende Browser-Umgebungen,[3] sowie eine sehr geschickte, subtile Blockade-Strategie im Bezug auf alternative Formate (wie etwa der Verhinderung des PGML-Standards durch Offenlegung der SWF-Spezifikation und Mitarbeit[4] am VML-Gegenentwurf)[5] letztendlich „ausschlaggebenden" Anteil am Erfolg des Flash-Ansatzes.

> *Diese angeblichen „98 Prozent Abdeckung" spiegeln jedoch nicht etwa die Beliebtheit Flashs beim Konsumenten wieder, sondern fußen weitgehend auf der entscheidenden Marketingidee, der Webgemeinde den Player als Bestandteil der gängigen Betriebssystem- und Browserdistributionen unterzujubeln.*
>
> [Curt01]

Bei der Feststellung des „Erfolges von Flash" muss natürlich wiederum differenziert werden zwischen der Akzeptanz und Verwendung des Flash-*Formates* (SWF), dem Verbreitungserfolg des Flash-*Players*, der die Unterstützung und Wiedergabe des Formates im Rahmen der verschiedenen Webbrowserumgebungen erst ermöglicht, und natürlich den (für den Macromedia-Konzern) letztendlich ausschlaggebenden *Verkaufs*erfolg des Flash-*Authoring-Tools*, welches schließlich für die *Erzeugung* der SWF-Formatdateien als quasi „monopolistisches Werkzeug" [Zmoe00] verantwortlich zeichnet.

---

[1] Frei übersetzt: "Therefore it can be argued that the decision to develop […] in Flash nowadays is not alone an aesthetic one, but also an economic and strategic one." [Zmoe00]

[2] vgl. hierzu Appendix I der Macromedia-Studie: [Macr02a] p.11, sowie [Curt01]

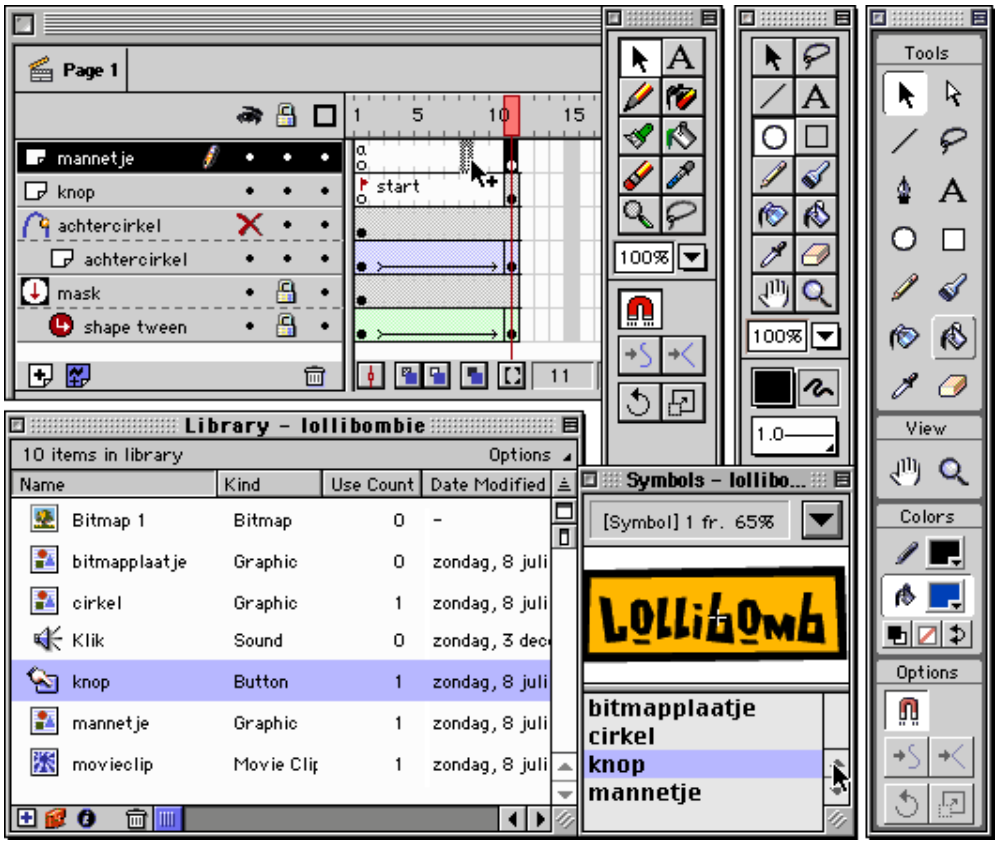
[3] „Die Verbreitung des Shockwave Flash-Plugins basiert vielmehr auf einer geschickten Strategie von Macromedia: Durch Kooperation mit Firmen wie Microsoft, Netscape und Apple wird es inzwischen mit jedem neuen Windows- und Macintosh-Betriebssystem ebenso wie mit jedem neuen Webbrowser für diese Systeme (Internet Explorer, Netscape, AOL) standardmäßig mitgeliefert…" [Star01] p.11

[4] „Um positiv auf die Verbreitung und die Akzeptanz des Flash-Dateiformates einzuwirken und sich damit gegen ehemalige Konkurrenten wie Adobes PGML zu schützen…" [ibid] p.12

[5] s. hierzu auch [5.3.2-3]

4.5.2 Das Authoring-Tool: Macromedia Flash

Um es vorwegzunehmen: Bei Vergleichen derzeit verfügbarer Authoring-Tools zur Erstellung multimedialer Web-Anwendungen sucht die *Macromedia Flaxh MX*-Suite natürlich derzeit ihresgleichen: Das ohne Zweifel äußerst mächtige Profi-Programm vereint Grafik- und Vektorillustrationsprogramm, Animationstool sowie eine umfangreiche Programmierumgebung für gleich zwei verschiedene Skriptsprachen (FlashScript und ActionScript) unter einem Dach. Neben recht komfortabler, teils der



ColdFusion Benutzerschnittstelle[1] entliehener *Coding*-Funktionalität und der noch aus der SmartSketch-Engine stammenden *Drawing*-Kompo-nente dominiert jedoch vor allem das framebasierte *Timeline*-Modul, welches nahezu unverändert aus dem Director-GUI übernommen wurde [s.2.1], das Anwendungsprinzip des Programms.

Abb. 4.5.2.1: GUI-Elemente des Flash-Authoring-Tools in verschiedenen Entwicklungsstadien[2] (V.l.o.n.r.u: Timeline (f.5), Toolbar (f.3-5), Bibliothek (f.4,1))

Während das (wie in [4.5.1] erläutert) noch dem Pen-Computing-Ansatz entstammende Zeichen-Werkzeug im Vergleich mit anderen, professionellen Illustrationsprogrammen doch recht ungewohnt aber, wie etwa [Dowd00] behauptet, mitunter „deutlich intuitiver" zu bedienen ist,[3] erscheint insbesondere das Zeitleisten-basierte Editing, wie bereits im Rahmen der ähnlichen Diskussion in [2.1] dargelegt, zwar für Animationen hervorragend geeignet, erweist sich jedoch bei steigendem Interaktivitäts- und Scriptinganteil zunehmend problematisch:[4] Da insbesondere stark verzweigte Interaktions-Strukturen durch ihre Nichtlinearität gekennzeichnet sind [vgl. „Lost-in-Hyperspace"-Syndrom, s.3.2.2] und sich somit nur umständlich im Rahmen einer Zeitleisten-gesteuerten GUI-Umgebung realisieren lassen, kann auch die Bearbeitung interaktiver Anwendungen mitunter äußerst komplex geraten. Überdies macht insbesondere die „steile Lernkurve"[5] das Programm gerade für Anfänger eher uninteressant:

> *The learning curve for Macromedia's Flash is perhaps its most salient weakness [and] the interface is also not very user-friendly or intuitive for non-designers*
>
> [Abb00]

[1] Anm: Auch ColdFusion (ehemals Allaire) wurde 2001 von Macromedia aufgekauft.

[2] Quelle: flashfreaks.nl [9.2.03] – daher teils niederländ. GUI-Komponenten

[3] "Those who have spent time with Flash's drawing tools can do complex things quite quickly, in many cases more quickly than you can in a PostScript-style drawing tool." [Dowd00]

[4] vgl. hierzu auch [Abbe00]: "For example, users must learn new concepts and jargon, such as Flash's timeline and stage…"

[5] vgl. auch

4.5.2.1 Komplexität

Insbesondere für die Erstellung geschäftlicher Präsentationen erscheint Flash daher erheblich zu kompliziert und zu mächtig: Wie auch [Wald02] zu bedenken gibt, würde bei einer Anwendung der Flash-Suite nicht nur „mit Kanonen auf Spatzen geschossen" – angesichts der im Regelfall technisch und ästhetisch eher unbewanderten Ersteller Business-orientierter Präsentationen [vgl. Scha90, Pirn01] bestünde natürlich insbesondere die Gefahr einer Überforderung der Nutzer. Aber auch bei professioneller Erstellung Flash-basierter Inhalte stellt die Mächtigkeit des Autorenwerkzeugs zugleich ein erhebliches Problem dar: Ergab sich beispielsweise noch im Rahmen HTML-basierten Web-Designs die Arbeitseinteilung aufgrund der einzelnen WWW-Komponenten (Grafiker für Bildelemente, HTML-Producer für deren Integration, Clientside JavaScripter für Seitendynamik sowie Serverside-Programmierer für Backend-Lösungen etc.) quasi *automatisch*, so erfordert das Anwendungsprinzip der Macromedia-Lösung zur „ganzheitlichen" Erstellung Flash-basierter Anwendungen nun sowohl grafische Fähigkeiten (zum Erstellen der Illustrationen und Bildinhalte direkt in Flash)[1], Programmierkenntnisse zur Realisierung der Flash- und Action-Scripts, sowie überdies Animations-Kompetenz zu Umsetzung der Flash-„Filme" selbst. Da dies eine Arbeitseinteilung nicht nur erschwert sondern, wie etwa [Fren02] anführt, gar „unmöglich macht", könne, so French, eine konsequente und vollständige Anwendung der Flash-Suite mitunter wichtige Kommunikationsketten sowie die „soziale Struktur" etwa einer Multimedia-Agentur „völlig zerstören":[2]

> *Several employees became so disillusioned with the changes brought about by the company's new commitment to Flash that they handed in their resignation notices… [The company's] rushed, even forced adoption of Flash precipitated major changes within the organization, and ultimately contributed to its undoing.*
>
> [Fren02]

4.5.2.2 Freie Wahl der Werkzeuge?

4.5.2.2.1 Das offene Format als Argument

An dieser Stelle tritt freilich das Argument der "Offenheit des Formates", das insbesondere das Macromedia-Marketing „gerne anführt" [Zome00] in Aktion: Da Macromedia die Spezifikation des Flash-Export-Formates SWF bereits 1998 veröffentlicht und zur Weiterverwendung freigegeben hat [s.4.5.5.3.1], sei die Erstellung Flash-basierter Online-Medien, so der Tenor des Softwarekonzerns, ja nicht allein auf die Verwendung des Macromedia-eigenen Flash-Tools festgelegt. Da eine Vielzahl an Unternehmen auf Basis der Spezifikation und des zugehörigen SDKs [Macr98,02b] bereits das SWF-Format zumindest in Form eines Export-Filters unterstützen [s.4.5.3.3.2], stünde es dem Nutzer somit frei, ein beliebiges, so genanntes *3rd-Party*-Produkt zur Erstellung der Flash-Medien zu Verwenden.

Erscheint diese Sichtweise bei oberflächlicher Betrachtung noch durchaus plausibel, so liegt in der Praxis der „logische Fehler" dieser Argumentation letztendlich im Detail der Umsetzung: So ist das „Deployment-Format" SWF [s.4.5.3] zwar in der Tat öffentlich zugänglich und (wenn auch zeitverzögert)[3] lückenlos dokumentiert – das nicht minder interessante, interne .FLA-Format des Autorenwerkzeugs, welches neben den rein visuellen und zur Präsentation unbedingt benötigten Daten überdies erweiterte Zusammenhänge etwa über Animationsverläufe oder hochqualitative Bilddaten beinhaltet, wird von Seiten des Macromedia Kon-

---

[1] Dies beschied mir auch Sascha Eschmann, Projektmanager bei Bartenbach Neue Medien in Mainz, in einem Kurzinterview: Stolz gab er an, „alle seine Web-Designer" erstellten vektorbasierte Zeichnungen nun „direkt in Flash", statt dies zuvor in einem anderen Werkzeug (etwa Illustrator oder Freehand) zu erledigen und anschließend zu importieren. [Pers. Interview am 3.6.2002 in Mainz]

[2] vgl. [Fren02]

[3] Anm: Der Zeitunterschied zwischen Veröffentlichung des Flash 6-Formates (Januar 2002) und der entsprechenden Spezifikation (Oktober 2002) betrug immerhin knapp 10 Monate

zerns jedoch zugleich unter Verschluss gehalten, „und es erscheint derzeit wenig realistisch, dass Macromedia die entsprechende Spezifikation irgendwann in näherer Zukunft veröffentlichen wird“:[1]

> *Sure, it would be a good move for Macromedia to make the FLA format open so the designers could move their project files from one authoring environment to another, [but] it just doesn't seem to be in their best interest as a business, because it's one of the ways that Macromedia can use to protect their share of the Flash authoring tools market.*
>
> [Arty02]

4.5.2.2.2 Verdrängungswettbewerb

Die „aggressive“ Blockadehaltung des Software-Konzerns gegenüber Entwicklungsbemühungen in Richtung alternativer Flash-Autorentools macht auch über diese rein passive Verhinderungs-Strategie hinaus die zwiespältige Haltung Macromedias hinsichtlich der tatsächlichen „Offenheit“ des Flash-Formates deutlich. Ein Hinweis darauf, dass die SWF-Veröffentlichung überwiegend strategisch motiviert und „sicherlich nicht aus idealistischen oder gar selbstlosen *Open-Source*-Beweggründen zustande kam“ [vgl. Star01], stellt etwa die Rolle des Unternehmens im Bezug auf das Scheitern eines äußerst interessanten, alternativen Flash-Werkzeuges dar: So machte sich etwa Mitte 2001 ein hoffnungsvolles OpenSource-Programmierteam um „Flash-Pionier“ [2] Arthur Stevens an die Entwicklung der portablen,[3] leistungsfähigen und vor allem kostenlosen Flash-Alternative *OpenLGX*, welche überdies erweiterten XML-Support sowie „Echtzeit-3D“ auf Basis des Flash-Formats bieten sollte. Neben „einigen dramatischen Entwicklungen“ innerhalb des Programmierteams selbst[4] machten externe Beobachter jedoch insbesondere den „Druck des Unternehmens“ auf die Entwicklungsmannschaft dafür verantwortlich, dass das Projekt zur großen Enttäuschung der aufmerksamen Flash-Gemeinschaft[5] letztendlich „abgewürgt wurde – ebenso wie bereits zuvor andere Ansätze, die in der Vergangenheit den Versuch unternommen hatten, konkurrierende Produkte zu entwickeln.“ [Berg01][6]

4.5.2.2.3 Fragwürdige Third-Party-Tools

So bleibt unter dem strengen Auge des Flash-Herstellerkonzerns Macromedia den Entwicklern Flash-basierter Alternativ-Software indes lediglich das wenig attraktive Marktsegment „flashiger Effekt-Tools“ wie etwa des Text-Effektgenerators Swish zum Erstellen wenig komplexer und zumeist rein linearer SWF-„Filmchen“ – angesichts mangelnder Interaktivität und hoher Textlastigkeit erscheint die Sinnhaftigkeit derartiger Flash-Anwendungen jedoch „höchst fraglich“:

> *Once upon a time before a product called Swish a cool text effect was considered good, [but] now, the program has just become too popular…*[7]

Neben Repräsentanten dieses so genannten “nutzlosen Flash”[8] reicht neben dem glücklosen und konzeptionell wenig überzeugenden „LiveMotion“ [vgl. WoHi00] des Konkurrenten Adobe daher derzeit kein einziges, Flash-basiertes Authoring-Tool an die Komplexität und Mächtigeit der Flash-Suite selbst heran, da potentiell „gefährliche“ Entwicklungen wie etwa das viel versprechende *OpenLGX*-Ansatz unter der latenten

---

[1] “We probably shouldn't count on Macromedia publishing the .FLA specification anytime soon” [Arty02]

[2] vgl. [Berg01]

[3] Im Gegensatz zu Macromedia's Flash war Stevens' Produkt unter anderem für PocketPC und Linux geplant

[4] Tomas Apodaca: “OpenLGX.” *Pushby Queue*, San Francisco 2001: “Some drama:… Dave Smith has left the team in a rogue fashion… [He] was kind enough to disable the servers…”
http://www.pushby.com/queue/computing/001273.html [10.2.03]

[5] “A huge shame and disappointment to all of us who got so excited about the program…”, in Guy Watson: OpenLGX – Arthur Stevens.” *FlashGuru* UK, 19. August 2002: http://www.flashguru.co.uk/000124.php [10.2.03]

[6] “The project risks being shut down before they get to release their first public beta, just like other projects trying to make competing products have been in the past.” [Berg01]

[7] aus: Lesley Paone: Flash and Design – what separates the good from the bad.” Ursprgl. auf *Aritali*, [mittlerw. offline], derweil auf *OrganicPixel* archiviert: http://wow.organicpixel.com/mc/articles/flashdesign.html [10.2.03]

[8] „Useless Flash“, vgl. ebenda, sowie [Niel00]

Androhung „rechtlicher Schritte“ [vgl. Berg01] seitens des Macromedia-Konzerns stets erfolgreich abgewehrt werden konnten. Bei Erstellung professioneller Multimedia-Präsentationen führt daher derzeit an der Benutzung des mit immerhin gut 500 Euro doch empfindlich kostenintensiven Authoring-Tools bedauerlicherweise „kein Weg vorbei“.[1]

Da, wie bereits diskutiert, das interne interne .FLA-Format der Flash-Suite darüber hinaus streng unter Verschluss gehalten wird [vgl. Arty02], erstaunt es natürlich kaum, dass die Flash-Anwendung auch zugleich das unbestrittene, zentrale Authoring-Tool des „Web-Export-Formates“ SWF darstellt, da nur die Flash-Suite selbst stets Zugriff auf den gesamten Funktionsumfang bietet, der von dem jeweils aktuellen Flash-Player unterstützt wird: 3rd-Party-Programmierer, die sich auf die Exportfunktion des „öffentlichen“ Flash-Formates stützen, sind daher ausschließlich auf die Informationen angewiesen, die von Seiten Macromedias mit stets großzügigem Zeitversatz zur Verfügung gestellt werden. So mussten die Entwickler etwa bei Einführung des komprimierten Video-Embeddings im Rahmen der 6. Version des SWF-Formates Anfang 2002 [vgl. BGTM02] ganze 10 Monate auf die Veröffentlichung einer entsprechenden Spezifikation warten.

Nicht zuletzt aufgrund dessen existiert neben der Flash-Anwendung selber derzeit beispielsweise kein einziges Softwareprodukt, welches die Bearbeitung dieses und weiterer, im Rahmen des MX-Frameworks erheblich erweiterten Funktionen [vgl. Cham02] des SWF-Formates erlaubt. Allein die sich hieraus ergebende, äußerst enge Verbindung zwischen Anwendungsprogramm und Exportformat macht an dieser Stelle deutlich, dass es sich bei Flashs „universellem Web-Format“ [2] SWF in der Tat *nicht* um einen allgemeinen Standard, sondern vielmehr eine immer noch *proprietäre* Technologie des Macromedia-Konzerns handelt [vgl. Abbe00, Doug01]

4.5.3 Das Format SWF

Bereits die Ausführungen in [Star01:11f] machen deutlich, dass die Veröffentlichung der entsprechenden Datei-Spezifikation keinesfalls freiwillig oder gar aus selbstlosen, idealistischen Motiven zustande kam, sondern vielmehr durch eine „konkrete Gefährdung“ der Flash-Technologie als erforderlich eingeschätzt wurde: So publizierte Macromedia die SWF-Interna nicht von ungefähr nur kurze Zeit nach der Einreichung eines PGML-Vorschlags [s.5.3.2] des „Erzrivalen“ [3] Adobe an das W3C-Konsortium: Da die durchaus interessante Funktionalität[4] des PGML-Entwurfs eine unmittelbare Bedrohung für das im direkten Vergleich „proprietär und untauglich“ [vgl. Arty01][5] erscheinende Flash darstellte, sah sich die Macromedia-Konzernführung schließlich genötigt, die SWF-Spezifikation „zähneknirschend“ offen zu legen:

> *Um positiv auf die Verbreitung und die Akzeptanz des Flash-Dateiformates einzuwirken und sich damit gegen ehemalige Konkurrenten wie Adobes PGML zu schützen, veröffentlichte Macromedia bereits im Juni 1998 im Rahmen des „Flash Open File Format“-Programms das Dateiformat der Flash-Filme*
>
> [Star01:12]

Wenn auch diese erste Spezifikation [Marc98] noch „unvollständig und sogar fehlerbehaftet“[6] war, so erlaubte sie nun doch durchaus interessante Einblicke in den internen Aufbau des Web-optimierten Flash-Formates. Vor einer kurzen Betrachtung dieser aufschlussreichen Struktur erscheint an dieser Stelle jedoch noch eine weitere, begriffliche Differenzierung notwendig: So weist unter anderem [Dowd99] auf die Tatsache hin, dass neben der bereits diskutierten Unterscheidung zwischen Flash-internem .FLA-Format, wel-

---

[1] „Wenn es um Vektorgrafiken geht, führt derzeit kein Weg an Macromedia Flash vorbei...“, aus der Presseerklärung von Macromedia Deutschland: „Flash 4: Web-Grafiken im Vektormodus.“ Neustadt a.d. Donau, 3. Juni 1999

[2] vgl. David Tyrer: “What is Flash?”, *Click as a Flash*, Leichhardt/Austral. 2002: “Flash has become a universal format”… http://www.clickasaflash.com/flash_info.html [10.2.03]

[3] vgl. [Piff02]

[4] s.5.3.2.2

[5] “[Many critics] consider [SWF] to be closed, unreadable to humans, and unfit for the enterprise…” [Arty01]

[6] vgl. [Star01] p.12

ches „im Prinzip lediglich ein Speicherabbild der Flash-Anwendung darstellt“ [Bryn99],[1] und dem Weboptimierten SWF-Format eine „allgemeine Verwirrung“ bezüglich der Unterscheidung zwischen „Shockwave“ und „Flash“ bestünde: Die dem Flash-Format zugrunde liegende Dateierweiterung „SWF“ ist zwar in der Tat die Abkürzung für „ShockWave Flash“ – das Dateiformat unterscheidet sich jedoch von der Architektur des „eigentlichen“ Shockwave-Formates in ganz erheblicher Weise. So können die aus *Director* stammenden Shockwave-Dateien (mit der Erweiterung .DCR) im Allgemeinen *nicht* von dem mit gut 200 KB recht schlanken Flash-Player wiedergegeben werden, während der (als 2,8 MB-Plugin schon erheblich „behäbiger“ zu Buche schlagende) Shockwave-Player die Flash-Funktionalität jedoch in der Regel bereits beinhaltet. Aufgrund dessen „übersetzt“ die Herstellerfirma das SWF-Kürzel derzeit offiziell mit der umständlichen Formel „Macromedia Flash File Format“.

4.5.3.1 Dateistruktur

Anders als diese unzugängliche Terminologie erweist sich das Flash-Format selber bei genauerer Betrachtung jedoch als relativ einfach, „straightforward“,[2] und gut durchdacht: “It’s a nice balance between compactness of representation and speed of rendering”, wie Slashdot-Kolumnist Jon Katz treffend kommentiert [Katz00]. Obgleich das Flash-Format zugunsten einer hohen Kompressionsrate streng binärcodiert ist, setzt sich jede SWF-Datei dennoch stets aus so genannten (wenn auch binären) „Tags“ zusammen, die aufgrund ihrer Modularität stets die Abgeschlossenheit und Erweiterbarkeit des Formates garantieren.

Diese „Tags“, die ähnlich wie konventionelle HTML-Tags aufgebaut sind (und somit als Attribute so genannte „Records“ aufweisen), jedoch im Gegensatz zum XML-Prinzip in der Regel nicht „geschlossen“ werden (d.h. es existieren keine End-Tags), lassen sich wiederum in zwei Kategorien Unterteilen: Den so genannten „Definition Tags“, in denen sich grafische Figuren (*shapes*) sowie Bitmaps, Buttons, Schriftarten, Text, Sounds und Sprites *definieren* lassen, d.h. lediglich im Speicher abgelegt werden – und den „Control Trags“, die für die letztendliche Darstellung oder aber Modifikation eines (stets zuvor definierten) Objekts auf dem Bildschirm verantwortlich sind.

Die Reihenfolge sowohl der Definitions- als auch der Kontroll-Tags folgt im Rahmen des SWF-Formates übrigens keinerlei festgelegten Struktur (wie etwa dem „Painter’s Model“ bei SVG), sondern kann relativ willkürlich erfolgen: Maßgeblich für Tiefenschachtelung der einzelnen Objekt-Schichten ist ausschließlich das (vorgeschriebene) `depth`-Attribut, welches, ähnlich wie der `z-index` des dHTML-Grafikmodells die jeweilige Bildebene determiniert [vgl. Star01:19]. Auch die Sequenz der Flash-Kontrollanweisungen ist in diesem Zusammenhang völlig beliebig, da erst mit dem so genannten `ShowFrame`-Tag die zuvor spezifizierten und (zumeist mittels `PlaceObject`-Tag samt Transformationsmatrix) platzierten bzw. modifizierten Elemente tatsächlich am Bildschirm dargestellt werden.

Schnell deutlich wird bei Betrachtung der SWF-Dateistruktur daher die äußerst strenge *Frame-basiertheit* des Flash-Formates: Ebenso wie bereits bei der „gewöhnungsbedürftigen“ [vgl. Abbe01] Benutzerschnittstelle der Flash-Anwendung selbst steht durch die zentrale Rolle der Teilbilder (*frames*) der Animationscharakter des Formates deutlich im Vordergrund. So wird etwa eine Animation eines Objektes über einen bestimmten Zeitraum stets in verschiedene, endlich viele Teilanimationen entsprechend der Frame-Rate zerlegt: Das SWF-Format betrachtet nun nicht mehr den Gesamtzusammenhang der Animation, sondern repräsentiert stets nur noch die *diskrete* Änderung des einzelnen Objektes, die in dem jeweiligen Frame vorzunehmen ist. Bei genauerer Analyse des Dateiaufbaus wird daher schnell deutlich, dass dem `ShowFrame`-Tag

[1] Davis Michie: “The FLA-format is not really a format – it’s a memory-dump of what's inside the Flash application.” [Bryn99]
[2] Anm: Hierfür existiert leider keine aquädate deutsche Übersetzung: Etwa „direkt“ oder „unkompliziert“.

eine zentrale Rolle im Rahmen des Flash-Formates zukommt. Die (schematische) Darstellung einer beispielhaften Flash-Datei[1] setzt sich demnach in der Regel wie folgt zusammen:

```
FWS // File-Header (ist immer die Umkehrung zu 'SWF')
   flashVersion 3 fileSize 741 movieWidth 550 movieHeight 400 frameRate 12 count 2
// Beginnen mit 1. Frame:
   SetBackgroundColor rgb_hex ffffff // Weisser Hintergrund
   DefineShape id 1 // Form Definieren (Daten weggelassen)
   PlaceObject shapeId 1 depth 26 matrix // Objekt mit Transf.-Matrix platzieren
                                  [ a_fixed   b_fixed] = [00010000   00000000]
                                  [ c_fixed   d_fixed] = [00000000   00010000]
                                  [tx_fixed  ty_fixed] = [000010a4   00000410]
ShowFrame // 1. Frame anzeigen
   PlaceObject shapeId 1 depth 26 matrix // Objekt verschieben (= neu platzieren)
                                  [ a_fixed   b_fixed] = [00010000   00000000]
                                  [ c_fixed   d_fixed] = [00000000   00010000]
                                  [tx_fixed  ty_fixed] = [000012ea   00000690]
ShowFrame // 2. Frame anzeigen: Objekt wird diagonal nach rechts unten verschoben
...
```

Listing 4.5.3.1.1: „File-Dump" einer Beispiel-SWF-Datei (Ausschnitt)

Auf Basis dieser Struktur wird überdies auch das grundlegende Prinzip des Flash-Formates deutlich: Im Gegensatz zu anderen Grafik- und Animationsmodellen können in SWF-Dateien keine „ganzheitlichen Zusammenhänge" etwa über den Verlauf einer Animation oder logischer Objektgruppen abgebildet werden – diese werden stets in recht einfache Low-Level-Objekte sowie einzelne Frames einer Animation aufgegliedert. Dies dient zwar eindeutig einer sehr schnellen Darstellung der Flash-Daten (schließlich muss ein entsprechendes Anzeigesystem lediglich stets die einzelnen Anweisungen der jeweiligen Frames sequentiell ausführen), lässt jedoch hinsichtlich des konzeptionellen Grundprinzips der SWF-Struktur den Schluss zu, dass das Flash-Format an sich aus theoretischer Sicht ein eher „dummes Speicherformat" darstellt, welches die Abbildung *logischer* Zusammenhänge nicht erlaubt.

Aus diesem Ansatz ebenso ersichtlich ist somit auch der eigentliche Verwendungszweck des Formates als primäres Anzeige- und Exportformat: Da eine Fragmentierung der Animationsdateien im Rahmen einer Web-Exportfunktion stets „trivial" ist, die „Zusammensetzung" (bzw. Rekonstruktion) der einzelnen Flash-Elemente hingegen erheblich aufwendiger ausfällt, ist das Format zur Darstellung im Web zwar optimal geeignet, lässt sich jedoch weit weniger komfortabel weiterverarbeiten. Auf die hiermit verbundenen Schwierigkeiten weisen unter anderem [PMEB01, Prob00a] hin: Aufgrund der Komplexität der Rekonstruktion korrekter Animationszusammenhänge zog es das Entwicklerteam der Universität Nottingham um Steven Probets schließlich vor, die Fragmentierung der Animationsdaten schlichterdings beizubehalten und lediglich mittels JavaScript zu „emulieren" [vgl. Prob00b,c].

4.5.3.2 Grafikmodell

Weitaus „kniffliger" erweist sich an dieser Stelle jedoch das Flash und somit auch dem SWF-Format zugrunde liegende Grafikmodell [vgl. PMEB01]: Da Semantik und Syntax des Flash-eigenen „Vector Paintings"[2] noch der SmartSketch-Generation entstammt und sich somit an „progressive Illustrations-Ansätze aus den Anfängen der 90er Jahre" [Dowd00] anlehnt,[3] weicht die Implementierung vektorbasierter Formen und Stile erheblich von der Gewohnten PostScript-Notation ab: Da die Zeichenmethode des ursprünglich auf Pen-Computing-Geräte ausgerichteten SmartSketch Formen und Stile bereits *während* des Zeichnens

[1] Anm: Da die Struktur der Datei aufgrund der Binär-Eigenschaft des Formates nicht direkt dargestellt werden kann, behilft man sich an dieser Stelle etwa eines so genannten „File Dumpers", der die binären Informationen der einzelnen Tags und ihrer Inhalt „parst" und in menschenlesbarer Form ausgibt.

[2] vgl. [Dowd00]

[3] Anm: Schließlich war das SmartSketch-Prinzip ursprünglich auf Pen-Computing-Plattformen ausgerichtet.

automatisch erkennen sollte, sind beispielsweise Füllungen in Flash stets entlang einer Linie realisiert, d.h. statt eine Füllfarbe für das Gesamtobjekt zu bestimmen, können (und müssen) für jede Linie Füllfarben und Stilattribute für jeweils die rechte und die linke Kantenseite angegeben werden [vgl. Prob01]. Auch werden Kurven nicht wie aus PostScript gewöhnt mittels zweier „Anfasser" (*control points*) in ihrer Krümmung determiniert, sondern mittels des so genannten „quadratischen Bézier"-Verfahrens, welches zwar lediglich *einen* Kontrollpunkt erfordert, das Zeichnen auch simpler Formen jedoch erheblich erschweren kann:

> *[Flash's] use of quadratic Béziers makes it a bit of a pig to draw circles, arcs and all curved forms.*
> [Katz00]

4.5.3.3 Dateizugriff

4.5.3.3.1 Das Macromedia-SDK

Da eine mögliche Erstellung oder gar Weiterverarbeitung Flash-basierter Daten aufgrund der Binär-Eigenschaft des SWF-Formates, des unkonventionellen Grafikmodells [s.4.5.5.3.2] sowie der „Frame-Fragmentierung" einzelner, diskreter Grafikobjekte und Transformationen schnell aufwendig und komplex geraten kann, hielt sich unmittelbar nach der Erstveröffentlichung der überdies unvollständigen und teils fehlerhaften[1] SWF-Spezifikation der Enthusiasmus der Entwicklerschaft noch deutlich zurück: Lediglich zwei in ihrem Leistungsspektrum recht überschaubare C-Implementierungen – ironischerweise gleich beide unter dem Titel LibSWF [Haeb99, Agui00] – ermöglichten auf Basis der Spezifikation eine rudimentäre, für professionelle Belange jedoch unzureichende Erzeugung dynamischer Flash-Dateien. Um das SWF-Format jedoch auch für „Dritt-Entwickler" (*3rd-Party-Developers*) attraktiv zu machen, veröffentlichte Macromedia[2] aufgrund dessen Anfang 2000 das so genannte *SWF Software Development-Kit* (kurz: SWF-SDK), das neben einer schlüssigen Formatdokumentation den direkten (wenn auch nur schreibenden) Zugriff auf Flash-Dateien nun endlich auch mittels einer direkten Programmierschnittstelle (in diesem Falle C++) ermöglichte. Neben dem so genannten „Low Level Manager", der die einzelnen Definitions- und Control-Tags lediglich in einzelne Klassen „verpackte", umfasste das SWF-SDK darüber hinaus auch den so genannten „High-Level-Manager", welcher für die Erstellung Flash-basierter Grafiken und Animationen im Rahmen einer höheren Abstraktionsschicht eine konzeptionell durchaus gelungene Syntax bereitstellt. Unter Anwendung dieses Toolkits lassen sich somit bereits anhand weniger Codezeilen interessante, direkt ablauffähige Flash-Filme erstellen:

```
#include <HF3SDK.h>
int main() {
    HFMovie movie = HFMovie(); // erzeuge einen neuen Flash-Film (300*300 Pixel)
    movie.SetSize(6000, 6000);
    movie.SetFrameRate(3);

    HFRectangle* rect1 = new HFRectangle(0,0,2000,2000);   // drei Rechtecke
    HFRectangle* rect2 = new HFRectangle(1000,1000,3000,3000);
    HFRectangle* rect3 = new HFRectangle(2000,2000,4000,4000);

    rect1->SetSolidFill(Red_RGBA); // setze die Füllfarben
    rect2->SetLinearFill(Blue_RGBA, Black_RGBA);
    rect3->SetRadialFill(Yellow_RGBA, Violet_RGBA);

    rect1->SetDepth(1); // platziere Rechtecke übereinander
    rect2->SetDepth(2);
    rect3->SetDepth(3);

    movie.Frame(0)->addObject(rect1); // füge Rechtecke zu 1. Schlüsselbild hinzu
    movie.Frame(0)->addObject(rect2);
    movie.Frame(0)->addObject(rect3);
    movie.Frame(1)->RemoveObject(rect3); // r.3 dreht sich um eigenen Mittelpunkt
    rect3->Rotate(FloatToFixed(30.0));
```

[1] vgl. [Star01] p.12

[2] Anm: Hersteller des SDKs war eigentlich das wie Macromedia in San Francisco angesiedelte Softwarehaus *Middlesoft*. [vgl. BEng00]

```
    movie.Frame(1)->AddObject(rect3);
    movie.Frame(2)->RemoveObject(rect3);
    rect3->Rotate(FloatToFixed(60.0));
    movie.Frame(2)->AddObject(rect3);

    // Animation beginnt von vorn ...
    movie.WriteMovie("beispiel.swf");// schreibe die SWF-Datei und beende das Programm
    delete rect1; delete rect2; delete rect3;
    return(0);
}
```

Listing 4.5.3.3.1: Der High-Level-Manager im konkreten Beispiel [1]

Die etwas ungewöhnlichen Ausmaße und Koordinaten, die auch in [Listing 4.5.3.3.1] deutlich werden, rühren im Übrigen aus der Tatsache, dass Flash-intern sämtliche Größenangaben stets in so genannten „Twips“ ($^1/_{1440}$ inch) abgespeichert werden, einer Auflösungsunabhängigen Einheit. Bei der normal üblichen Bildschirmauflösung von 72 DPI entsprechen 20 Twips also einem Pixel. Ein 2000 · 2000 Twips großes Quadrat (wie etwa jedes der Rechtecke in unserem Beispiel) nimmt somit auf dem Bildschirm lediglich 100 · 100 Pixel ein.

Bei Ausführen des in [Listing 4.5.3.3.1] angegebenen Quellcodes wird nun ein lediglich 266 Bytes schlanker Flash-Film erzeugt – [Abb.4.5.5.3.1] zeigt eine Momentaufnahme des zweiten Schlüsselbildes, das dritte Quadrat ist bereits um 30 Grad rotiert:

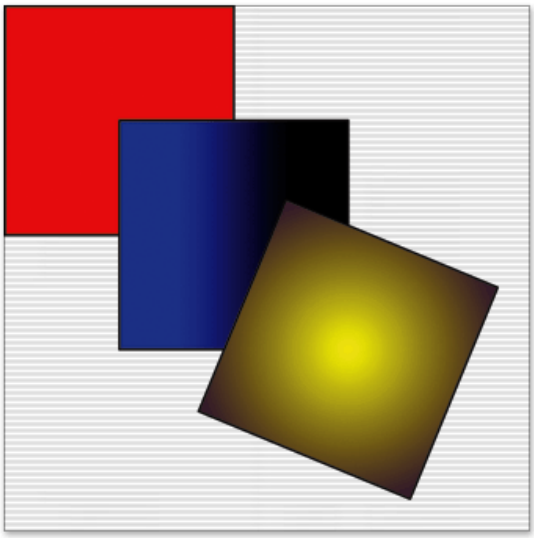


Abb. 4.5.5.3.1 Darstellung des aus [Listing 4.5.3.3.1] erzeugten SWF-Films (Momentaufnahme)

4.5.3.3.2 Third Party-APIs

Die kostenfreie Bereitstellung dieses – wie im Beispiel gezeigt – äußerst komfortablen Development-Kits löste im Gegensatz zur reinen Spezifikations-Veröffentlichung einen richtiggehenden „Run“ auf die Entwicklung Flash-fähiger Export-Module aus: Da die praktische Syntax nun die Konvertierung jeglicher Vektordaten in das Flash-Format extrem vereinfachte, fand sich bald ein entsprechender SWF-Filter in zahlreichen (wennauch statischen) Illustrationswerkzeugen wieder – eine schier epidemische Format-Verbreitung, die Macromedia durch die gezielte Veröffentlichung des SDK nicht nur gut geplant, sondern dank zusätzlichem, „aggressivem Marketing” [vgl. Raux01] auch überaus konsequent und erfolgreich erreicht hat.

4.5.3.3.2.1 Server-Side Flash

Neben der reinen Anwendung im Rahmen C++-basierter Windows- und Macintosh-Anwendungen, für deren Zwecke das SDK ja ursprünglich konzipiert war, erschien die Funktionalität der Programmierschnittstelle freilich auch für einen Server-Seitigen Einsatz im Internet zunehmend interessant: Insbesondere die Erzeugung *dynamischer* SWF-Dateien, deren Inhalte sich individuell verändern ließen, war aufgrund der bis dato eingeschränkten und überdies recht umständlichen Datenanbindung des Flash-Frameworks in diesem Zusammenhang von großem Interesse. Aufgrund der Tatsache, dass C++ aufgrund der Komplexität und „Gefährlichkeit“ der Sprache[2] auf nur recht wenigen Internet-Servern als Skriptsprache zur Anwendung kommt, wurde im Gefolge der SWF-SDK-Veröffentlichung eine schier unüberschaubare Vielfalt an Flash-Generatoren und Portierungen des Macromedia-SDK zur Serverseitigen Erzeugung beliebiger Flash-Filme veröffentlicht: Neben recht überschaubaren, C-basierten Lösungen, die lediglich auf den bereits zuvor bereitgestellten Funktionen aufbauen [BEng00, HeEz01] sowie einfachen,

[1] aus: [Star01] p.18

[2] Anm: Bei der Erstellung C++-basierter Serverprogramme besteht insbesondere die Gefahr unzulässiger Pointer-Befehle, die den gesamten Server zum Absturz bringen könnten.

reitgestellten Funktionen aufbauen [BEng00, HeEz01] sowie einfachen, aber teils „durchaus raffinierten" Server-Skripten, die insbesondere auf Basis formulargesteuerter Web-Eingaben schlichte, aber dennoch beeindruckende Flash-Filme zu „dynamisieren" vermögen [ToKe00, Wilk02], hat in der OpenSource-Szene insbesondere das *Ming*-Projekt[1] weitere Verbreitung erlangt, welches neben dem Hauptmodul PHP mittlerweile nahezu sämtliche Server-Sprachen (Perl, C++, Python) unterstützt und sich auch in Multimedia-Agenturen großer Beliebtheit erfreut.

4.5.3.3.2.2 Java spricht SWF

Wachsende Bedeutung kommt im Rahmen der „automatischen Flash-Generierung" jedoch auch der in weiten Teilen der Server-Welt Einzug haltenden Programmiersprache Java [s.3.5.3] zu: Da insbesondere aufgrund der Plattformunabhängigkeit und im Gegensatz zu C++ auch höheren Sicherheit eine Entscheidung „eher zugunsten von Java als von C++ ausfällt" [Star01], erscheint eine direkte Portierung des Macromedia-SDKs in diesem Zusammenhang natürlich äußerst reizvoll: Über das so genannte *Java Native Interface*,[2] so zumindest der Vorschlag von Benjamin Stark und Ralf Kunze von der Universität Osnabrück [vgl. Kunz00, Star01], ließen sich etwa die Schnittstelle zischen C++-Bibliothek und Java einmalig über die Methodenköpfe (*headers*) definieren. Über einen einfachen Aufruf direkt aus der Java-Umgebung heraus könnte auf diesem Wege die gesamte Funktionalität des SDKs, das ja ursprünglich nur in C++ vorlag, auch im Rahmen einer Java-betriebenen Serverlösung nutzbar gemacht werden. Der von Kunze und Stark im gleichen Zug angekündigte, vollständig Java-basierte Flash-Generator [vgl. Star01:47] wurde jedoch bedauerlicherweise nie veröffentlicht.

4.5.3.3.2.3 XML als Schnittstelle

Deutlich weiter vorangeschritten sind an dieser Stelle indes die ebenfalls Java-basierten Flash-Lösungen *JavaSWF2* [Main00] sowie *Saxess Wave* [vgl. Dabb01, Star01:46ff], die sich jedoch über die von Seiten der Programmiersprache bereitgestellten Möglichkeiten hinaus noch im Zwischenschritt der Funktionalität von XML [s.5.1] bedienen: So ist im Rahmen beider Frameworks sowohl die Umwandlung binärer Flash-Dateien ein XML-konformes Text-Pendant, als auch im Umkehrschritt die Erzeugung animierter Flash-Filme auf Basis eines zuvor formulierten XML-Dokumentes oder aber direkt aus der Java-Logik heraus möglich. Zur Verdeutlichung des letzteren Falles kann an dieser Stelle etwa ein triviales „Hello-World"-Beispiel herangezogen werden, das (nach Referenzierung einer externen Flash-Datei, die bereits die eingebettete Schriftart enthält) die zweizeilige Nachricht „Hallo Welt" im Flash-Player ausgibt:

```
public class HelloWorld
{
    public static void main( String[] args ) throws Exception
    {
        FontDefinition fontdef = FontLoader.loadFont("VerdanaFont.swf");
        Movie movie = new Movie();
        Frame frame = movie.appendFrame();
        Text text = new Text(null);
        Font font = new Font(fontdef);
        text.row(font.chars("Hello", 25), new Color(0,0,255),0,0,true,true);
        text.row(font.chars("World!",25), new Color(255,0,255),0,0,false,false);
        frame.placeSymbol(text, 200, 200);
        movie.write(args[0]);
    }
}
```

Listing 4.5.3.3.2: HalloWelt-Beispiel unter Verwendung des JavaSWF2-Frameworks [Main00]

[1] Anm: Nicht zu verwechseln mit MNG (ebenfalls "Ming" ausgesprochen): In diesem Falle stammt der Name aus der Comic-Welt: „Ming" hieß der Erzrivale des Comic-Helden *Flash* Gordon.

[2] vgl. Beth Stearns: "Java Native Interface," Sun Microsystems: http://java.sun.com/docs/books/tutorial/native1.1 [10.2.03]

Die *Wave*-Lösung der Kölner Firma Saxess erlaubt hingegen mittels einer speziellen XML-Dokumentstruktur (DTD), der so genannten SWF Markup Language (SWFML), die exakte Repräsentation einer Flash-Datei in XML-Form: Hierbei wird (wie übrigens bereits in [4.5.3.1] angedeutet) schlichterdings jedes SWF-„Tag" in ein „echtes" XML-Tag mit den entsprechenden Attributen umgewandelt. Unser „Hallo Welt"-Beispiel sähe in dieser XML-Fassung (die ja nichts anderes als eine Text-Darstellung der SWF-Datei selbst darstellt) daher folgendermaßen aus:

```
<?xml version="1.0" encoding="UTF-8"?>
<SWF w="300" h="300" color="ffffff"
      xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="../../swfml-1.0.xsd">
      <text ID="text1" x="200" y="200" font="Verdana" size="25" color="0000ff">
         Hallo
      </text>
      <text ID="text2" x="200" y="225" font="Verdana" size="25" color="ff00ff">
         Welt!
      </text>
      <PlaceObject IDREF="text1" depth="1"/>
      <PlaceObject IDREF="text2" depth="1"/>
      <ShowFrame/>
</SWF>
```

Listing 4.5.3.3.2: HalloWelt-Beispiel unter Verwendung des Saxess Wave-Frameworks [Dabb01]

Mit einer ganz ähnlichen, da ebenso der SWF'schen Tag-Semantik entlehnten Struktur arbeitet auch das *Spark*-Projekt [KuCu02], welches ebenfalls auf Basis der Java-Programmiersprache die Umwandlung von SWF-Dateien in deren XML-Form und vice versa erlaubt. Die Anwendung der strukturierten, textlichen XML-Konvention in Form eines logischen Zwischenschritts macht an dieser Stelle nicht nur insofern Sinn, als dass die binärkodierten Inhalte der Flash-Datei in eine strukturierte und menschenlesbare Form übergeführt und sodann selbst in einem simplen Texteditor weiterverarbeitet werden können, sondern dient darüber hinaus insbesondere im Rahmen einer Java-basierten Server-Lösung einem noch einer Verbesserung des „Work-Flows": Da sich speziell die Java-Schnittstelle hervorragend zur Verarbeitung XML-basierter Daten und Dokumente eignet [vgl. Star01], können beliebige, in die Flash-Datei zu integrierende Inhalte mithilfe einer sehr einfachen Java-Syntax in den so genannten „DOM-Baum" der XML-Datei eingehängt und auf diese Weise nach anschließender Konvertierung (die nun jedoch praktischerweise automatisch durch die eben genannten Frameworks [vgl. Main00, Dabb01, KuCu02] übernommen wird) dynamische Flash-Filme äußerst komfortabel erzeugt werden.

Neben der bereits in [2.3.4] diskutierten, „internen" (jedoch leider unter Verschluss gehaltenen) XML-Engine der ebenfalls Flash-Filme erzeugenden *iCreate*-Anwendung [Wana02], macht sich auch das wie die zuvor genannten Lösungen Java-basierte *JGenerator*-Projekt [Skav00] diesen erfreulichen Umstand zunutze: Auf optionaler Basis eines unter OpenSource-Lizenz verfügbaren *TomCat*-Servers des nicht-kommerziellen Apache-Projekts kann die Flash-Technologie im Rahmen einer Server-Lösung auch „on-the-fly" im Web nutzbar gemacht werden.

Da auch der Flash-Konzern Macromedia aufgrund der bald unüberschaubaren Vielfalt rasch aufkeimender Server-Lösungen auf Basis des SWF-Formates das enorme kommerzielle Potential des serverbasierten Ansatzes automatischer Flash-Erstellung schnell erkannte, setzte sich mit dem Macromedia *Generator*, der naturgemäß im Gegensatz zu der Mehrzahl der soeben genannten, durchweg[1] kostenfrei verfügbaren Lösungen die *Gesamtheit* der in der jeweiligen SWF-Spezifikation enthaltenen Funktionen abzudecken vermochte, jedoch alsbald eine kommerzielle Anwendung am Markt durch. Auch die etwas preisgünstigeren, jedoch ebenso kostenpflichtigen *Generator*-Alternativen *ASP Flash Turbine* der Lissabonner Softwareschmiede Blue

[1] Anm: Mit Ausnahme von Wanadus *iCreate* [s.2.3.4]

Pacific [NoDi00] sowie der so genannte „Swift Generator“ konnten der Dominanz des Macromedia-Konzerns auch auf diesem Gebiet nur wenig anhaben.

4.5.3.3.2.4 Im Schlepptau der Spezifikation

Die letztendliche Chancenlosigkeit der nicht-kommerziellen 3rd-Party-Alternativen begründet sich neben dem äußerst geschickten Marketing des Macromedia-Konzerns, welcher das „monopolistische“ [vgl. Zmoe00] Flash-Studio alsbald im „Bundling“ mit der Server-basierten Generator-Software anbot, letztendlich in der unaufhörlichen *Erweiterung* des SWF-Formates, der die unabhängigen Alternativ-Entwickler stets dem letzten Stand hinterherhinken lässt: So findet sich in den meisten der zuvor beschriebenen Lösungen lediglich ein nur rudimentärer Funktionsumfang wieder, der maximal die 3. oder 4. Generation des Flash-Formates abdeckt – im Rahmen multimedialer Web-Präsentation jedoch erst interessante Funktionalität wie ActionScript-Tags (oftmals die Grundlage jeglicher Flash-Interaktivität) oder Füllverläufe sucht man derweil zumeist vergeblich.

Mit der Veröffentlichung der extensiv erweiterten 6. Version des Flash-Formates, die als Bestandteil von Macromedias *MX*-Produktreihe gleich eine ganze Reihe neuer Funktionen [vgl. Cham02] wie etwa der Einbettung ganzer Video-Streams [vgl. BGTM02] ermöglichte, läutete das Unternehmen jedoch letztendlich das Ende der „serverseitigen 3rd-Party-Ära“ ein: Da die intelligente XML-Schnittstelle des neuen Flash MX eine direkte Anbindung Flash-basierter Anwendungen an dynamische XML-Daten ermöglichte, war zugleich ein Großteil der ebendiese Funktionalität in Form externer Programmlogik bereitstellenden Produkte hinfällig geworden. Da das Unternehmen mit der deutlich zeitversetzten[1] Herausgabe der entsprechenden Formatspezifikation zugleich die Aktualisierung der zugehörigen SDK-Schnittstelle einstellte und auch das betagte „Generator“-Projekt in den Ruhestand versetzte, wurde den diesbezüglichen Entwicklungsbemühungen zahlreicher unabhängiger Software-Projekte praktisch die Grundlage entzogen. Aufgrund dessen erstaunt es auch nur wenig, dass innerhalb der letzten Monate praktisch keine Bewegung mehr auf diesem Segment zu beobachten ist, da der MX-Ansatz alternative Schnittstellen im Prinzip „obsolet gemacht hat“ (Macromedia-Erklärung).

Hinsichtlich des SWF-Zugriffs auf Basis einer höheren Programmiersprache muss somit abschließend festgestellt werden, dass aufgrund zahlreicher Entwicklungsbemühungen in der Vergangenheit zwar unzählige, entsprechende Software-Komponenten bereits allgemein verfügbar sind, die sowohl Parsing als auch Generierung Flash-basierter Dateien in einem rudimentären Umfang auf sehr komfortable Weise ermöglichen – die zukünftige Entwicklung in dieser Richtung dürfte sich jedoch insbesondere aufgrund mittlerweile ausbleibender Bewegung in diesem Segment sowie der Einstellung des Macromedia-SDKs sowohl zunehmend aufwendig gestalten, als auch der Sinn diesbezüglicher Ansätze aufgrund der Veröffentlichung des Macromedia MX-Frameworks mittlerweile durchaus fraglich erscheint.

4.5.3.3.3 Konvertierungsaspekte

Auch die Konvertierung des SWF-Formates sowohl zur Weiterverwendung von Vektor- und Animationsdaten in anderen Formaten und Multimedia-Anwendungen, als auch zur möglichen Nutzung der Flash-Plattform als Ausgabemedium für zuvor umgewandelte Fremdformate erscheint in diesem Zusammenhang mitunter problematisch: So verkompliziert nicht nur die in [4.5.3.1] erläuterte „Frame-Fragmentierung“, sondern ebenso das recht unkonventionelle Grafikmodell des Flash-Formates [s.4.5.3.2] eine direkte Überführung der Flash-Daten etwa in PostScript-verwandte Syntax, weswegen die Aussage des SWF-Entwickers Jacek Artymiak,[2] die Programmierung eines Flash-Parsers sei „äußerst einfach“,[3] nicht zuletzt aufgrund der

[1] s.o.: Die Zeitdifferenz betrug knapp 10 Monate.
[2] vgl. [Arty02]
[3] vgl. ebenda.

ständigen (zuletzt im Oktober 2002 spezifizierten) Erweiterungen des SWF-Formates, an dieser Stelle doch zu relativieren ist.

Bei einer möglichen Konversion in umgekehrter Richtung,[1] die ja aufgrund der beeindruckenden Verbreitung des Flash-Players zunächst durchaus attraktiv erscheint, werden jedoch schnell die Grenzen des SWF-Formates deutlich: Insbesondere bei einer Nutzung SVG-basierter Daten [s.5.4] fällt schnell auf, dass das Flash-Format speziell aus grafischer Hinsicht dem Ursprungsformat um einiges nachsteht: So müssten im Rahmen eines entsprechenden Konvertierungsprozesses nicht allein sämtliche Bildanimationen entsprechend einer willkürlichen Bildrate in endliche Frames „fragmentiert" werden – auch zahlreiche Photoshop-ähnliche Bildeffekte, Filter, Muster und Schriftfunktionen, die das SVG-Format nativ bereitstellt, müssten bei einer Überführung ausschließlich aufgrund diesbezüglicher „Versäumnisse" des SWF-Formates entsprechende „Work-Arounds" entwickelt werden:

> *[Flash] can't even do simple geometry or variable operations without driving a programmer into work-around hell.*
>
> [Balo00:2]

4.5.3.4 Konsequenz

Obgleich daher auf Basis der „Ubiquität" des Flash-Mediums die Nutzung der entsprechenden Plug-In-Verbreitung zugunsten einer maximalen Erreichbarkeit im Web auch im Rahmen dieser Diplomarbeit (zum optimalen „Deployment" vektorbasierter Präsentationsdaten) per se nicht uninteressant erscheint, habe ich mich aufgrund dieser aus meiner Sicht konzeptionellen Schwächen des SWF-Formates (welches in der Tat erhebliche „Altlasten" mit sich herumzutragen hat)[2] letztendlich entschlossen, diesen Weg *nicht* zu beschreiten, da die Implementierung entsprechender Überführungsalgorithmen einschließlich unzähliger „Work-Arounds" (s.o.) sowohl aufgrund zahlreicher diesbezüglicher Vorarbeiten [vgl. Pmeb01, Prob00a,b,c][3] lediglich ein recht triviales „Puzzle-Stück" darstellen würde, als auch die verbleibende Programmierarbeit eher „handwerklichen" Charakter hätte und in meinen Augen daher für diese Diplomarbeit nur wenig geeignet erscheint.

4.5.4 Diskussion des Flash-Ansatzes

Nach dieser Erkenntnis gilt es überdies, sich die Zielsetzung dieser Diplomarbeit noch einmal in Erinnerung zu rufen: So geht es bei der Betrachtung Web-basierter Präsentationslösungen und –Formate ja nicht ausschließlich um die Analyse der technischen Funktionen und Eigenschaften eines Formates und seiner Darstellumgebung,[4] sondern ebenso um die Beurteilung seiner Web-Eigenschaften, d.h. der „Affinität" des Formates mit seiner medialen Umgebung – in diesem Falle also dem Hypertext-basierten WWW.[5] Bei Betrachtung der öffentlichen Diskussion über eben diese „Webfähigkeit" Flashs wird jedoch schnell deutlich, dass die Eigenschaften der Flash-Technologie insbesondere im Hinblick auf Navigation und Benutzungserlebnis, also der so genannten „Usability", in der Tat äußerst umstritten sind:

> *Flash has been identified as a key culprit in bad Web design, enabling pages of blinking text and galloping images that do little more than consume bandwidth.*
>
> [Beck02]

---

[1] Anm: Gemeint ist hier somit eine Überführung von Fremddaten in das Flash-Format

[2] Anm: Insbesondere erscheinen mir unter diesem Aspekt die „Frame-Fragmentierung", das unkonventionelle Grafikmodell [vgl. Dowd00, Prob00a,b] sowie die (soeben erläuterten) grafischen Mängel des SWF-Formates ausschlaggebend.

[3] s. hierzu auch die bereits verfügbaren Software-Komponenten in [4.5.5.3.2]

[4] Anm: Mit „Darstellumgebung" ist in diesem Falle der Player gemeint, der über die reine Formatdarstellung hinaus die Ausgabeeigenschaften (in diesem Falle) Flashs kennzeichnet: So sind Anti-Aliasing und die erfreulich performate Darstellung nicht Eigenschaften des Formates, sondern vielmehr des zugehörigen Players.

[5] s. hierzu die entsprechende Diskussion in [3.1]

4.5.4.1 Flash versus Usability

Wenn auch die durchaus beeindruckenden, technischen Eigenschaften der Flash-Technologie trotz einiger konzeptioneller Mängel des Formates [s.4.5.3.3.3] recht unzweifelhaft feststehen, so sind und waren vielmehr die mitunter kritischen *Folgen* des Formates Bestandteil einer äußerst kontroversen Diskussion: So stellte etwa Jakob Nielsen, Speerspitze der Usability-Bewegung, die bereits die zweifelhafte Entwicklung des HTML-Standards kritisch kommentierte [s.3.2.1] im Rahmen eines vielbeachteten Essays („Flash: 99% Bad“)[1] fest, dass Flash den Prinzipien des World Wide Web bereits im Grundsatz widerspricht:

> *Although multimedia has its role on the Web, current Flash technology tends to discourage usability for three reasons: it makes bad design more likely, it breaks with the Web's fundamental interaction style, and it consumes resources that would be better spent enhancing a site's core value.*
>
> [Niel00a]

Neben offensichtlichen Versäumnissen des Flash-Frameworks („Zurück“-Button, Suchfunktion, Texteigenschaft) griff Nielsen und mit ihm eine große Gefolgschaft über den „Flash-Mißbrauch“ [Curt01] empörter Web-Entwickler [Ragu99, McGr00a, Rose00] insbesondere die „fatalen Konsequenzen“ des Grundprinzips Flash-basierter Multimediaelemente an: Schon aufgrund der dem Animationscharakter Flashs zugrunde liegenden Linearität Flash-basierter „Filme“ verkomme das einst Textbasierte, interaktive Hyptertextmedium Internet zusehends zu „passivem Web-Fernsehen“. Großflächiger „Missbrauch“ der durch die Flash-Technologie bereitgestellten Funktionalität führte dann letztendlich unter den Augen entsetzter Usability-Experten zu einer „katastrophalen“ Situation im Web: Insbesondere die bereits überwunden geglaubte „Unsitte“ bunter Einstiegs- und Intro-Seiten („splash pages“) erfuhr mit dem Boom der Flash-Technologie eine unselige Renaissance, und hielten den Internet-Nutzer zumeist in Form endloser „Film-Vorspänne“ „von den eigentlichen Inhalten ab, für die er eigentlich gekommen war“.[2] Die (ebenso wie die zuvor in [3.4.1] diskutierten, animierten GIF-Elemente) überwiegend „sinnlose“ Verwendung bunt blinkender Flash-Animationen [vgl. McGr00a] war nicht nur „Kennzeichen einer narzisstischen, neuen Flash-Designer-Generation“, die anstatt einem konkreten Zweck zu dienen, sich „lieber in verträumten Spielereien der eigenen Kreationen erging“, [vgl. Lipm00] sondern senkte zugleich die Usability, also die Nützlichkeit und Bedienbarkeit einer „erschreckenden“ Anzahl von Websites empfindlich herab.

4.5.4.1.1 Der Sturm der Entrüstung

Aufgrund dieser äußerst negativen Konsequenzen Flash-basierter „Design-Eskapaden“ [Zmoel00] bestimmte während des unaufhaltsamen Siegeszuges der Flash-Technologie zugleich eine Flut äußerst Flash-kritischer Bücher und Essays die Diskussionskultur des Internet: So bezeichnete Chris MacGregor, einer der führenden Experten Flash-basierten Web-Designs [vgl. Beck02], das Animationsformat gar als „Krebsgeschwür des Internet“ [McGr00a] und befahl den in seinen Augen fehlgeleiteten „Flashern“, endlich aufzuhören, die Nutzer des Internet fortwährend zu „missbrauchen“ [vgl. McGr00b], während Informations-Designer Scott Ragus in *Flash* schlichterdings die reine „Inkarnation des Bösen“ zu erkennen glaubte [Ragu99]. Diese Erkenntnis wurde freilich sogleich von Beter Balogh aufgegriffen, der Flash mit der Figur des Teufels selbst personifizierte, welcher sich in einer „dämonischen Ansprache“ [3] höhnisch an die dieser Entwicklung chancenlos gegenüberstehende Gemeinde selbsterklärter „Usability-Experten“ richtet:

> *I am the cancer that has riddled the formerly healthy body of the Web… And you – wretched coders, huddling at your desks – did you think I would abide by your specs? Bah! I wipe myself with your white papers. Thin pages? Fast download times? Here's what I think of your pathetic download times…*

---

[1] vgl. [Niel00a]
[2] vgl. ebenda.
[3] vgl. [Balo00]

> *Within a year, every single commercial site will yield to my demands, clogging all major pipelines with gigabit after gigabit of worthless, empty, babbling 'content.'*
>
> [Balo00:1]

4.5.4.1.2 Die Kritik verebbt – gegen Bares

Nach einem regelrechten "Sturm der Empörung", in dem sich zahlreiche Web-Kolumnisten im Zuge des „Flash-Booms" ihre Unmut über die fatalen Folgen der „Flash-Epidemie" [Zmoe00] lauthals bekundeten, kehrte jedoch 2002 verdächtige Ruhe im „Lager der Usability-Experten" [vgl. Rieh01:7, Depe02:22] ein. Grund hierfür waren jedoch weniger die im Rahmen der Flash MX-Generation realisierten „Usabiltiy-Elemente" (Integration des „Back"-Buttons, verbesserte ‚Accessibility' für Sehbehinderte) sowie die langsam abebbende Flut professioneller, jedoch überladener Flash-Anwendungen, die trotz offensichtlicher Usability-Schwächen in der Vergangenheit noch mit Preisen überhäuft worden waren [vgl. Grue01], scheinbar „geläutert" von den „Design-Sünden" [vgl. Kenn00] jedoch zunehmend einsichtig hiervon abrückten – die „wahre" Ursache der urplötzlich verstummenden Kritik bestand vielmehr in wirkungsvollen Annäherungsversuchen an die einstigen Kritiker: So erhielt nicht nur der ehemals vehementeste Anti-Flash-Ideologe Jakob Nielsen im Rahmen der Flash MX-Veröffentlichung einen lukrativen „Bruderschaftsvertrag" von Flash-Herstellerfirma Macromedia,[1] auch Chris MacGregor[2] veröffentlichte statt der zuvor harschen Worte ein umfangreiches (und zudem wohldotiertes) Flash-Papier zur „Entwicklung benutzerfreundlicher Flash-Inhalte" [McGr02]. Selbst „Accessibility-Nazi" (SlashDot)[3] Joe Clark, der dem Unternehmen zuvor noch vorgeworfen hatte, „Behindertengerecheit vollständig zu ignorieren" [vgl. Clar00], schlug nun sanftere Töne an: „The new Flash MX authoring environment and the equally new Flash Player 6 solve a few accessibility problems." [Clar02]

Nicht zuletzt der Hinweis des Autors, für „maximale Usability" jedoch unbedingt den Umstieg auf das (empfindlich teurere) MX-Paket zu wagen,[4] weist jedoch darauf hin, dass nicht ausschließlich die „überwältigende Funktionalität von Flash MX", [vgl. Arty01] sondern ebenso finanzielle Beweggründe die ehemaligen Kritiker von den Vorzügen des Flash-Ansatzes letztendlich überzeugt haben dürften.

4.5.4.1.3 Fragliche Web-Tauglichkeit

Im Bezug auf die tatsächliche „Webtauglichkeit" der Flash-Software bleibt abschließend jedoch insbesondere im Hinblick darauf, dass bislang äußerst wenige Flash-Anwendungen auf die freilich aufwendig zu realisierenden Usability-Funktionen wie dem mühsamen Einpfegen zusätzlicher Meta-Daten tatsächlich zurückgreifen, sowie auch Suchmaschinen derzeit mit Flash-Inhalten nur wenig anfangen können, an dieser Stelle festzustellen, dass Flash, bei allen Verbesserungen seit der einst heftig kritisierten [vgl. Ragu99, Niel00] Erstversion immer noch „lediglich äußerst eingeschränkt" als webfähig bezeichnet werden kann.

Dies begründet sich, wie bereits im Rahmen der strukturellen Formatanalyse in [4.5.3.1] beleuchtet, nicht zuletzt in dem sowohl dem Anwendungsprinzip, der Authoring-Software als auch dem konzeptionellen Grundprinzip des SWF-Formates zugrunde liegenden Multimedia-Paradigma Flashs: So ist nicht nur (wie bereits in [4.5.2] beleuchtet) die GUI des Flash-Studios streng auf die Animationseigenschaft Flash-basierter „Filme" ausgelegt – auch die strikte Frame-Einteilung der SWF-Dateistruktur macht deutlich, dass eine Flash-Datei ohne Animationscharakter im Prinzip unvorstellbar ist. Im Gegensatz zur Mehrzahl der in [4.3] betrachteten, vektorbasierten Webformate findet sich aufgrund dessen derzeit nahezu kein einziges Beispiel

---

[1] vgl. hierzu die entsprechende Presserklärung des Macromedia-Konzerns: „Macromedia and Usability Guru Jakob Nielsen work together to improve Web Usability". San Francisco, 3. Juni 2002

[2] s.o.: Autor der Artikel "A Cancer on the Web called Flash" [McGr00a] sowie "Hey Flasher, Stop Abusing your Visitors!" [McGr00b]

[3] vgl. den entsprechenden SlashDot-Eintrag vom 5. Mai 2002: http://developers.slashdot.org/article.pl?sid=02/05/06/0310256 [11.2.03]

[4] "The changes have also automatically improved access to existing Flash content when viewed in the Flash Player 6, but to maximize Flash accessibility for your users you'll need to publish content from Flash MX.", aus: "Flash MX: Moving Toward Accessible Rich Media." (o.V.) *A List Apart*, Ausg. 143. New York, 26 April 2002 http://www.alistapart.com/stories/flash_mx_moving [11.2.03]

einer *statischen*, Flash-basierten Vektorgrafik im Internet. Trotz des auch von [Niel00] noch einmal bekräftigten Grundsatz Jared Spools, dass die überwiegende Mehrzahl der Web-Benutzer Animationen im WWW „eigentlich stets irritierend" findet [SSSA99:89f], „erzwingen" sowohl Flash-GUI als auch SWF-Format daher geradezu die zumeist unnötige Animationseigenschaft Flash-basierter Web-Anwendungen, denn:

> *Flash encourages gratuitous animation: Since we can make things move, why not make things move?*
>
> [Niel00]

4.5.4.2 Flash-Ästhetik

Dieser durch das Flash-Prinzip implizit vorgegebene „Zwang zur Animation" hat sich daher bislang auch in der Praxis in Gestalt einer speziellen Ästhetik niedergeschlagen, auf die unter anderem auch Christine Zmoelnig im Rahmen ihrer Dissertation "The Aesthetics of Macromedia Flash" [Zmoel00] hinweist. Der ästhetische Gesamteindruck des Flash-basierten Multimedia wird demzufolge in der Regel durch „stete Unruhe" gekennzeichnet: Selbst in prinzipiell eher statischen Präsentations-Situationen wie etwa einer Menü-Komponente oder eines einfachen Schaubildes neigen Flash-Anwendungen aufgrund des inhärenten „Animations-Diktats" zu permanenter Bewegung: „Immer scheint irgend eine Komponente des Bildschirms in Bewegung zu sein, nur selten ruht der Gesamt-Screen in sich selbst".[1]

Über diesen „verstörenden" [SSSA99:89f] Animationscharakter hinaus ist überdies die Einteilung der gestalterische Grundlage multimedialer Flash-Präsentationen in zwei „ästhetische Klassen" möglich: Insbesondere in der „ersten Phase" des Flash-Designs überwiegen demnach so genannte „self-indulgent sites" [vgl. Lipm00, Kais00], auf denen sich sowohl professionelle als auch grafisch unbewanderte Flash-Autoren „betrunken durch die von Flash offerierten Möglichkeiten" [McGr01] hemmungslos „austobten" und sich in selbstverliebten Flash-Spielereien[2] „zum Selbstzweck ergingen":

> *The result is scores of rococo splash pages that do little except showcase their designers' capacity for self-indulgence…*
>
> [Lipm00]

Die zweite, bedauerlicherweise noch deutlich spärlicher gesäte Kategorie ist die der „schlichten Schönheit": Entsprechend der von [Karv00] ausgeführten, ästhetischen Grundprinzipien der „Beauty of Simplicity" wird diese ästhetische Klasse insbesondere durch einfache, aber ob ihrer Schlichtheit überzeugende Screen-Gestaltung gekennzeichnet:

> *Graphic designers often complain that usability experts always want design that is too simple, that is, boring. Simplicity in this sense is a kind of "stripped" simplicity - the design is stripped naked of all fancy features, colours, and flashy, moving objects. Is this what users really want? Or could there be a second kind of simplicity that they actually mean, 'designed' simplicity – clear, and 'clean' […], but in a stylistic and beautiful way? We think it is this latter form of simplicity that is asked for.*
>
> [Karv00:89]

Obgleich diese Form der „simplistischen Ästhetik" auch zunehmend in Flash-basiertem Design Einzug hält, besteht der problematische Aspekt dieses Gestaltungsprinzips jedoch darin, dass sich derartige „Schlichtheit" in den Augen vieler Designer mit den „Grundsätzen" der Flash-Technologie widerspricht: So findet sich dieser Design-Ansatz in der Tat derzeit viel eher in HTML-basiertem Web-Design, als in den immer noch oft zu überladenen Flash-Animationen wieder:

> *Was Flash für die Webgemeinde zum Ärgernis macht, ist das Pech, dass es förmlich dazu verleitet, missbraucht zu werden.*
>
> [Curt01]

---

[1] frei übersetzt aus [Zmoe00]

[2] vgl. [Ragu99]

Insbesondere Semi-Professionelle Flash-Gestalter neigen ob der Funktionsvielfalt (und dem aufgrund dessen ebenso überbordenden GUI) der Flash-Anwendung immer noch verstärkt dazu, sich in den Realisierungen visuell opulenter Multimedia-Präsentationen zu ergehen, anstatt auf die Gestaltung schlichter, aber überzeugender Präsentationen zu setzen – ein Irrweg, an dem meiner Einschätzung nach auch das Anwendungsprinzip und das zugrunde liegende Flash-Format nicht ganz schuldlos sind. Auf die Tatsache, dass insbesondere ersteres maßgeblichen Anteil an den „Design-Sünden" und der immer noch mangelhaften Webtauglichkeit Flash-basierter Animationen trägt, weist unter anderem [True01] hin:

> *With the latest release of Flash, I have noticed a marked increase in the number of 'Flash for Fools in Five Minutes' books… The basics of Flash are relatively easy to grasp. It's these basics that most amateurs begin and end with, churning out those 'sites' that fuel the fire for these myth-conceptions.*[1]
>
> [True01]

4.5.5 Fazit

4.5.5.1 Flash – ein verdienter Marktführer?

Zusammenfassend muss im Hinblick auf Potential und Möglichkeiten des Flash-Formates daher zwar die enorme Verbreitung der Player-Komponente als auch die nicht unbeachtlichen Möglichkeiten des Formates selber, insbesondere hinsichtlich Sound- und Animationseigenschaften, lobend erwähnt werden; auch die Vektorbasiertheit des SWF-Formats, in Verbindung mit vollständigem (wenn auch optionalem) Anti-Aliasing und konsequenter Schrifteinbettung, tragen äußerst positiv zum Gesamteindruck bei. Bei oberflächlicher Betrachtung kann somit durchaus festgestellt werden, dass sowohl Authoring-Tool als auch das Flash-Format im Rahmen eines wirkungsvollen Deployments multimedialer Präsentationen durchaus überzeugen könnten – wären da nicht die doch erheblichen „Schönheitsfehler", die dieses Bild bei genauerem Hinsehen doch erheblich trüben: So stellt das empfindlich kostenintensive Flash-Paket des „monopolistischen" [Zmoe00] Herstellers Macromedia derzeit unbestritten das „einzig diskutable" Authoring-Tool der Flash-Technologie dar, da auf Basis der SWF-Spezifikation durchaus existierende, alternative Entwicklungsbemühungen[2] entweder „abgewehrt" werden konnten [s.4.5.2] oder aber aufgrund einiger kritischer Aspekte des SWF-Dateiformates [s.4.5.3.1-3] insbesondere nach Veröffentlichung von Flash MX nur wenig interessant erscheinen.[3]

4.5.5.2 Eignung in Bezug auf Web-Präsentation

Im Rahmen dieser Diplomarbeit erscheint die vollständig Flash-basierte Realisierung Web-Basierter Präsentationen überdies aus ästhetischer sowie technischer Perspektive fragwürdig: So erscheint sowohl das Anwendungsprinzip als auch die preisliche Positionierung der Flash-Anwendung im Hinblick auf die äußerst schlichten Anforderungen und Design-Kompetenzen, die etwa der durchschnittliche PowerPoint-Benutzer der Erstellung geschäftlicher Präsentationen entgegenbringt, sowohl wenig angemessen, als auch die Animations-basiertheit (Zeitkontinuität) sowie mitunter unumgängliche Programmieranforderungen dem eigentlichen Präsentationsprinzip eher entgegenstehen. Neben der überdies fragwürdigen ästhetischen Dimension Flash-basierter Präsentationen [s.4.5.4.2] sind es jedoch insbesondere die zweifelhaften *Usability*-Aspekte [vgl. Ragu99, Niel00] und der damit verbundenen, mangelnden „Web-Fähigkeit" des SWF-Formates, die das letztendliche Verdikt an dieser Stelle zu Ungunsten von Flash ausfallen lassen: Neben störenden „Randeffekten" des Formates wie etwa dessen „Non-Retrievability" (Flash-Daten, lassen sich etwa im Gegensatz zu SVG [s.5.4] nur sehr umständlich herunterladen und auch nur bedingt weiterverarbeiten) sowie fehlen-

---

[1] zitiert aus: [True01]. Hinweis: Mit dem Begriff der „Mythen" spielt Truese auf die Kritik unter anderem Jakob Nielsens [vgl. Ragu99, Niel00] an „unnützen" Flash-Animationen an.

[2] Anm: Insbesondere erscheint an dieser Stelle der OpenLGX-Ansatz relevant.

[3] Anm: Da derzeit keines der am Markt verfügbaren Alternativ Programme die erst letzten Oktober veröffentlichten Funktionen [vgl. Cham02] wie Video-Embedding [vgl. BGTM02] unterstützen, besitzt Flash MX einen derzeit uneinholbar scheinenden Marktvorteil.

der *Granularität* der Benutzer-Kontrolle[1] ist an dieser Stelle insbesondere der Aspekt der fehlenden *Texteigenschaft* des Formates relevant. Da Flash-Textinhalte im Rahmen von SWF nicht nur „recht umständlich" [Katz00] implementiert,[2] sondern überdies auch binär komprimiert werden, können semantische Inhalte Flash-basierter Präsentationen somit derzeit weder von Suchmaschinen indiziert, noch von einem Mensch gelesen werden. Die Lizenzvereinbarung des Macromedia-Konzerns geht sogar noch einen Schritt weiter: Demzufolge ist es verboten, Flash-Software „in irgend einer Weise zu dekompilieren oder in eine menschenlesbare Form zu überführen".[3]

4.5.5.3 Flash versus Lesbarkeit

Nicht nur dieser „erschütternde Wortlaut" [vgl. Cagl01], sondern insbesondere die dahinter stehende Grundhaltung Macromedias macht an dieser Stelle deutlich, dass das Unternehmen trotz offen gelegter SWF-Interna [vgl. Macr98,02b] und des (mittlerweile eingestellten) SDKs *keineswegs* primär um Offenheit und Transparenz bemüht ist, (erkennbar auch an den fehlenden Bemühungen um eine Standardisierung des Formates),[4] sondern das Flash-Format vielmehr immer noch eine in erster Linie „proprietäre" [vgl. Abbe00, Doug01] und „geschlossene" [Arty02] Dateistruktur repräsentiert.

Die nähere Betrachtung der SWF-Architektur [s.4.5.3.1] legt überdies weitere Schwächen des Formates offen: Neben der recht problematischen „Frame-Fragmentierung" und der hiermit verbundenen Schwierigkeiten bezüglich Rekonstruktion der Animationszusammenhänge und dem erschwerten *Parsing* der SWF-Dateien [s.4.5.3.3.3] erscheint überdies die unstrukturierte und willkürliche Ordnung von Objektdefinitionen und Kontrollanweisungen innerhalb der Flashdatei zumindest „strukturell fragwürdig". Das eigentliche Hauptproblem des Formates besteht jedoch letztendlich in der maßgeblich durch die geschlossene Binärstruktur des Formates gekennzeichneten, mangelnden Web-Fähigkeit Flashs, welche das in Grundzügen funktionell überaus interessante SWF-Format schlussendlich für einen Einsatz im Rahmen Web-basierter Präsentationen *ungeeignet* erscheinen lassen: So stehen das strikt animationsbasierte Präsentationsmodell Flashs sowie zweifelhafte Usability-Eigenschaften [vgl. Niel00] des Formates letztendlich unseren eingangs formulierten Anforderungen [s.Kap.1] bezüglich Web-Fähigkeit (und hier speziell der Anspruch einer auch konsequent web*gemäßen* Lösung) sowie dem eher auf dem Dexter-Modell beruhenden [s.2.2], konventionellen Präsentationsprinzip entgegen.

Insbesondere die zahlreichen, bereits existierenden Ansätze, das „geschlossene" SWF-Format in eine menschenlesbare Form zu überführen, weisen an dieser Stelle jedoch einen möglichen Ausweg aus diesem „Dilemma": So ermöglichen nicht nur unzählige 3rd-Party-Frameworks [KuCu02, Dabb01, Main00, Wana02] schon jetzt die beidseitige Konvertierung Flash-basierter Binärdaten in das strukturierte, textbasierte XML-Format – auch die Konzentration der MX-Generation des Flash-Autorenwerkzeugs auf XML-Funktionalität [vgl. Cham02] machen insbesondere die offensichtliche Notwendigkeit deutlich, präsentationsbezogene Informationen neben einer binären, stark komprimierten Variante überdies in einer logisch strukturierten, textbasierten Form hinterlegen und somit sowohl für Menschen als auch für Internet-Suchmaschinen besser durchsuchbar machen zu können. An dieser Stelle drängt sich freilich der unter anderem hierfür konzipierte *XML-Standard* zur Lösung dieses Konfliktes geradezu logischerweise auf.

---

[1] Für einzelne Teil-Seiten oder –Bereiche der Anwendung lässt sich, im Gegensatz zu Html, keine separate URL angeben (zweifelhafte „Notlösung": Pseudo- URLs mit „Einstiegs-Pointern")

[2] "[Flash's] text handling, by being user-agent-neutral, requires the programmer to jump through hoops..." [Katz00]

[3] frei übersetzt aus der Flash-Lizenzvereinbarung Macromedias (Abschnitt 2 (b) VI.)
http://www.macromedia.com/support/shockwave/info/licensing/license.html [12.2.03]

[4] vgl. [Cagl02] pp.10ff.

5 XML: Strukturierte Vektorgrafik für das WWW

5.1 Grundlegende Eigenschaften von XML

Da dies ja keine Diplomarbeit über XML an sich darstellt (s. hierzu etwa die ebenfalls im Fachbereich Digitale Medien entstandenen Arbeiten [Duck01, Rein02, Duff03, Meit03]), sondern diese Strukturkonvention für unsere Zwecke lediglich im Hinblick auf XML-basierte Vektor- und Präsentationsformate im Internet von Interesse ist, sollen an dieser Stelle lediglich die in diesem Zusammenhang wichtigen Aspekte kurz beleuchtet werden. Da aufgrund des epidemischen „XML-Booms“ [1] der letzten Jahre [vgl. PoWi00] überdies eine Fülle aufschlussreicher Artikel und Bücher [vgl. insb. Mach97, McLa01, Arm01] diesen Themenkomplex bereits umfassend abdeckt und verständlich erläutert, sollen sich die folgenden Ausführungen daher lediglich auf die im Rahmen dieser Diplomarbeit relevanten Aspekte beschränken.

5.1.1 Begriffsklärung

Dennoch erscheint insbesondere nach den Erkenntnissen aus [3.1] zunächst eine grundlegende Begriffsklärung zum Verständnis der späteren Anwendung der XML-Konvention im Rahmen vektorbasierter Präsentationen im Web unerlässlich: So muss zunächst einmal festgehalten werden, dass das Web im Gegensatz zur verbreiteten Volksmeinung nicht den „Nachfolger von HTML“ [Boei03], sondern vielmehr von dessen grundlegender Strukturkonvention SGML [SGML86] darstellt – daher auch der Begriff der „eXtensible Markup Language (XML). So wie SGML die Formulierung *beliebiger* Sprachen auf Basis von strukturierten „Tags“ und einer Dokumentdefinition (DTD) erlaubt, und HTML lediglich auf dieser Grundlage die Verwendung einiger bestimmter (da dem speziellen Zweck einer strukturierten Hypertext-Umgebung dienender), so genannter HTML-Elemente oder –Tags im Rahmen der HTML-DTD zulässt und somit eine Anwendung (genauer: ein Dokumenttyp) von SGML darstellt, so erlaubt auch die XML-Konvention die Erstellung verschiedener, beliebiger Ausprägungen bzw. Anwendungen. Im Gegensatz zur reinen SGML-Anwendung HTML stellt XML somit eine erweiterbare („eXtensible“) Sprache dar, die sich nun wiederum ähnlicher (teils sogar derselben) Konventionen von SGML bedient, diese jedoch zugleich erheblich vereinfacht: Während HTML aufgrund eingeschränkter Erweiterungsmöglichkeiten, wie bereits in [3.1] diskutiert, für komplexere Anwendungen ungeeignet ist, stand der Weiterverwendung des SGML-Standards bislang dessen „ungeheure Komplexität“ [vgl. Bosa97, Star01:25] und „abschreckende Syntax“ [Mach97] entgegen.

An dieser Stelle setzt nun die eXtensible Markup Language an: Durch gezielte Vereinfachungen und Einschränkungen des SGML-Umfangs erreichte das World Wide Web Consortium (W3C), welches den Standard bereits 1998 als so genannte *Recommendation* verabschiedete, zugleich eine deutliche Reduzierung der Komplexität SGMLs wie auch eine deutlich höhere Konsistenz:

> *Ohne dass die Idee des strukturierten Markups von SGML aufgegeben worden wäre, wurden alle komplexen und selten verwendeten Eigenschaften entfernt. Wie radikal die Einschnitte sind, mag man dem Umstand entnehmen, dass die formale Definition von XML auf 33 Druckseiten möglich ist. SGML benötigt mehr als 500. Trotz der Radikalkur ist XML aufwärtskompatibel zu SGML geblieben, gewissermaßen ein SGML ‚light’.*
>
> [Mach97]

[1] „In den USA ist ein wahrer XML-Boom ausgebrochen […] Software-Entwickler setzen alles daran XML-fähige Versionen ihrer Produkte auf den Markt zu bringen“ [vgl. PoWi00]

Besteht die Kernidee SGMLs, sämtliche Inhalte im Rahmen so genannter, beliebig verschachtelter „Tags“[1] abzulegen, natürlich auch in XML weiterhin, so wurden dennoch erhebliche Einschränkungen vorgenommen: Aufgrund der bislang schwierigen Verarbeitung SGML-basierter Dokumente entschloss man sich daher, die Formulierung von XML-Anwendungen deutlich strikter zu gestalten: So müssen im Rahmen von XML etwa sämtliche Tags auch wieder durch ein entsprechendes „Closing Tag“ geschlossen werden, ebenso wie neben leer stehenden Attributen (`<td nowrap>`)[2] auch inhomogene Schreibweisen bezüglich Anführungszeichen (" bzw. ') sowie Groß/Kleinschreibung (`<table>` oder `<TABLE>`) nicht mehr erlaubt sind.

5.1.2 Struktur und Darstellung

Auf Basis dieser Konventionen lassen sich nun per XML beliebige Datenstrukuren formulieren, die überdies frei („semantisch“) benannt werden können. Damit ein verarbeitendes System („Parser“) die Daten des so definierten Dokumententyps allerdings auch auswerten und auf seine formale Richtigkeit überprüfen kann, wurde im Zuge der Standardisierung XMLs zunächst ein entsprechendes Definitions-Modul aus dem Vorgänger SGML praktisch unverändert übernommen, welches eine Formulierung der Dokumentstruktur über die so genante „Dokumenten-Typ-Definition“ (Kurz: DTD) ermöglicht. Da die Syntax ebendieser DTDs, auf die an dieser Stelle jedoch nicht näher eingegangen werden soll, jedoch selber weder der SGML- noch der XML-Konvention genügt und darüber hinaus komplexere semantische Zusammenhänge[3] nicht ermöglicht, wurde diese „Altlast“ daher mittlerweile (wenn auch erst teilweise)[4] durch den W3C-Standard *XML Schema* [SpTh00] abgelöst.

Da aufgrund der Abgeschlossenheit wie auch der endlichen Verschachtelungs-Eigenschaft jedes gültige XML-Dokument einen *Baum* aufspannt, der als Wurzel, Zweige und Blätter die jeweiligen, hierarchisch strukturierten Tag-Elemente sowie deren „Kinder“ enthält, lassen sich beliebige XML-Daten, sofern sie denn der XML-Konvention genügen (und somit „wohlgeformte“ Dokumente repräsentieren) in einer solchen Baumdarstellung am Bildschirm anzeigen. Eine derartige Funktionalität stellt Microsofts Internet Explorer derzeit etwa schon von Haus aus bereit:

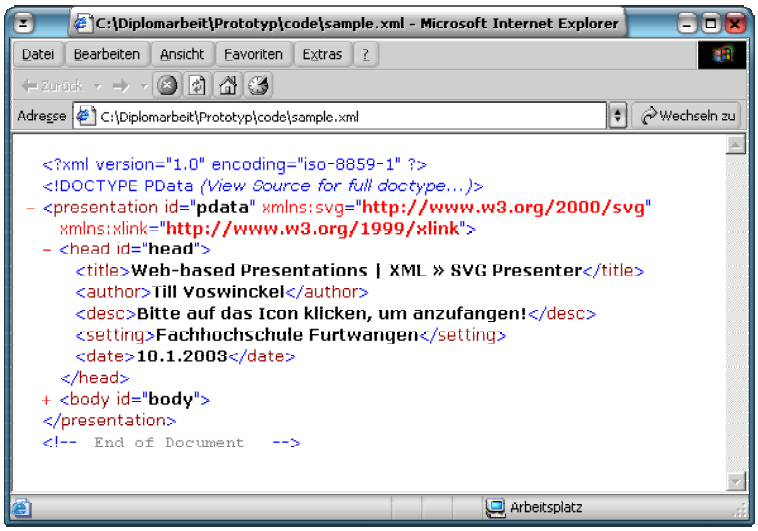


Abb. 5.1.2.1: Darstellung eines XML-Dokuments in Microsofts Internet Explorer (links).

Nun ist allerdings diese datenorientierte Sicht auf XML-Inhalte aus gestalterischer Sicht bzw. im Rahmen einer Präsentation nicht unbedingt besonders spannend oder gar „ästhetisch überwältigend“. An dieser Stelle besteht jedoch, ebenso wie bereits im Rahmen der SGML-basierten HTML-Sprache – die Möglichkeit, die Anzeige-Eigenschaften der XML-Daten mithilfe eines *Style Sheets* zu spezifizieren, um auf diese Weise eine individuelle Darstellung des XML-Dokuments zu erreichen.

[1] Syntax: <tag attribut="Wert">Textinhalt</tag>

[2] Anm: Im Rahmen von XML muss jedes Attribut nun auch einen Wert besitzen, bspw: <td nowrap="">.

[3] Anm: Etwa die Vorgabe, wie oft sich Elemente wiederholen dürfen etc. [vgl. Arm01]

[4] Anm: So unterstützt die Mehrzahl der derzeitigen XML-Parser die sog. „Validierung“ XML-basierter Dokumente derzeit etwa ausschliesslich die DTD-Syntax, auch wenn XML Schema „in Zukunft kommen“ soll [vgl. Star01:31]

5.1.3 Reine Stilfrage: CSS, DSSSL, XSL

Neben Anwendung der bereits aus HTML bekannten [s.3.2.3.2] Web-Standards *Cascading Style Sheets* [CSS96,98] sowie des auf sämtliche SGML-Dokumente anwendbaren (und jede XML-Datei ist aufgrund der Rückwärtskompatibilität [vgl. Mach97] ja zugleich SGML-konform), jedoch aufgrund seiner schier erschlagenden Komplexität im Prinzip „unzugänglichen" [vgl. ROHD98:191] Lisp-Dialekts DSSSL[1] wurde daher im Rahmen der Entstehung des XML-Standards eine „eigene" Style-Sheet-Sprache entwickelt, die so genannte eXtensible Stylesheet Language (XSL). Diese ist nicht nur in der Lage, XML-Dokumente zu verarbeiten und zur Präsentation aufzubereiten, sondern beruht ebenso selber auf der XML-Konvention, d.h. jede Xsl-Anwendung stellt selbst wieder eine Ausprägung bzw. (genauer: ein Dokumententyp) von XML dar. Im Rahmen dieser Entwicklungsbemühungen spaltete sich dieses Framework jedoch erneut auf in die so genannten XSL-*Formatting Objects* (XSL-FO), welche eine medienunabhängige (primär jedoch Dokumenten- bzw. Seitenbasierte)[2] *Formatierung* von XML-Elementen erlaubt, sowie die weitaus mächtigere Transformationssprache *XSLT*, die nun eine vollständige Verarbeitung und Umwandlung der XML-Daten in eine gänzlich andere (XML-)Datenstruktur ermöglicht. Im Zusammenhang mit der *Präsentation* von (XML-basierten) Daten kann so etwa eine XML-Dokumentstruktur in ein entsprechendes Präsentationsformat wie z.B. SVG [5.4] transformiert werden. Das besondere an der XSLT-Sprache ist nun, dass sie, obgleich selber lediglich eine XML-Anwendung, wie eine „echte" Programmiersprache verschiedene *Variablen* und *Kontrollstrukturen* zur Verarbeitung und Transformation der Quelldaten bereitstellt – in diesem Falle jedoch auf XML-Basis: So können mithilfe von XSLT-Elementen wie `xsl:template`, `xsl:for-each` oder `xsl-if` beliebig komplexe Transformationsanweisungen intelligent „programmiert" werden.[3] Ein in meinen Augen nicht zu unterschätzendes Problem stellt an dieser Stelle jedoch die Tatsache dar, dass hier mithilfe der Datenstruktur- und Dokumenten-Konvention XML versucht wird, Strukturen einer Programmiersprache nachzubilden. Dies ist jedoch nicht nur für Programmierer sehr umständlich und zunächst ungewohnt, sondern überdies auch aus konzeptioneller Perspektive problematisch:

> *We are sceptical of whether XSL's syntax is appropriate for a style sheet language. XSL uses XML's syntax – this makes XSL a Markup language. The strength of Markup languages is they are very effective for embedding meta-data (or instructions) within data. Style sheets though are almost entirely meta-data because styles sheets, in essence, are instructions to a browser on how to process and format a document… CSS's syntax seems more appropriate for style sheets: CSS is much easier to read and expresses presentation in a way that better corresponds to listing rules about presentation.*
>
> [MaMu99]

Die derzeit populärste XSLT-Anwendung stellt daher natürlich die Umwandlung von XML-Daten in das HTML-Format als Ausgabemedium dar, um so eine gezielte „Präsentation" der XML-Dokumente im Webbrowser zu erhalten. Im Rahmen des *Internet Explorers* kann so etwa für ein beliebiges XML-Dokument durch die Angabe einer entsprechenden XSLT-Datei, die wiederum die entsprechenden, so genannten „Processing Instructions" (Verarbeitungsanweisungen) zur Umwandlung der XML-Daten beinhaltet, stets eine individuelle Darstellung der Daten erreicht werden. Dies jedoch erscheint nun wiederum in den Augen mehrerer Kritiker, unter anderem [MaMu99], höchst bedenklich, da es die Möglichkeiten der XSL-Sprache wiederum auf die „verkrüppelte" Funktionalität der HTML-Darstellung reduziert:

> *We are concerned that this is positioning XSL as a language for producing HTML rather than as an independent presentation language. While this technique may be useful for making XML documents backward*

---

[1] "DSSSL, which uses Lisp syntax, is widely regarded as having usability problems and has not seen widespread adaption" [Bosa98].

[2] Anm: Da die XSL-Formatting Objekts primär Dokument-Attribute wie Schriftarten, Zeilenabstände, Seitenlayouts etc. umfassen, ist dieser Ansatz primär auf die Ausgabe von Print-Medien (derzeit besteht die populärste Aufgabe der FO-Prozessoren (FOP) etwa darin, aus XML-Daten mittels XSL-FO ausdruckbare PDFs zu generieren.)

[3] Anm: Aufgrund der Komplexität der XSLT-Sprache soll jedoch an dieser Stelle auf Details hinsichtlich Syntax und Semantik verzichtet werden.

> *compatible with HTML-only browsers, it limits the presentations of documents to the rather crippled presentation abilities of HTML.*[1]

Neben diesem problematischen Aspekt erscheint in meinen Augen überdies die derzeit noch mangelhafte XSL-Unterstützung der Web-Clients indes weitaus bedenklicher: So ist derzeit ausschließlich Microsofts *Internet Explorer* ab der Version 5.0 [vgl. Duck99:104] in der Lage, ein clientseitiges XSLT-Transformationsskript automatisch auszuführen und Daten somit entsprechend aufzubereiten – für alle anderen Browser kann von einer derartigen Funktionalität hingegen derzeit nicht ausgegangen werden. Aufgrund dessen muss bei Anwendung einer XSLT-baiserten Konvertierung die Umwandlung der Daten stets serverseitig und mithilfe eines zusätzlichen XSLT-Prozessors durchgeführt werden, was meiner Einschätzung nach jedoch einen nicht zu rechtfertigenden Aufwand darstellt.

5.1.4 Prozeduraler Zugriff: Die DOM-Schnittstelle

Da dieser Transformation derzeit ohnehin mithilfe eines Java- oder C++-betriebenen XSLT-Prozessors [vgl. Kay01, Star01:43] realisiert wird, erscheint es in meinen Augen hingegen weitaus sinnvoller, sich statt der umständlichen, deklarativen Verarbeitung der XML-Daten mithilfe eines, wie soeben diskutiert, durchaus nicht unproblematischen [vgl. MaMu99] XSLT-StyleSheets *direkt* der Möglichkeiten einer „richtigen" Programmiersprache zu bedienen. Das XML-Framework stellt zum komfortablen, prozeduralen Zugriff auf sämtliche Elemente eines XML-Dokuments hierfür das so genannte *Document Object Model* (kurz: DOM) bereit. Dieses Datenzugriffsmodell basiert auf der hierarchischen Baum-Eigenschaft eines XML-Dokuments und erlaubt nun die logische Abarbeitung einzelner XML-„Knoten" sowie deren im Rahmen der Baumhierarchie auftretenden „Kinder", „Geschwister" und „Eltern". Aufgrund des ausgesprochenen Komforts dieser „Baumschnittstelle" [Star01:32], verbunden mit der Mächtigkeit und logischen Syntax einer Programmiersprache wie Java [s.3.5.3] oder ECMAScript [ECMA97] steht dem Nutzer als auch dem Programmierer mit dem DOM-Modell zur Verarbeitung von hierarchisch strukturierten (XML-)Daten somit ein in meinen Augen unschlagbares Zugriffsmodell zur Verfügung.

Nach Feststellung dieser durchaus beeindruckenden Funktionalität sowie der komfortablen Schnittstellen, die uns das XML-Framework zur Formulierung und Verarbeitung textbasierter Datenstrukturen anbietet, stellt sich an dieser Stelle jedoch naturgemäß die Frage, wie ebendiese Technologie nun im Rahmen Webbasierter Präsentationssysteme nutzbar gemacht werden kann:

> *The beauty of XML is it provides a whole new way of structuring 'stuff' that goes beyond just organizing data. With XML, data is imbued with meaning and purpose... In fact, [it] is terribly exciting if you're inclined toward trying whiz-bang graphics on the Web.*
>
> [Gibb00a]

5.2 XML für Präsentations- und Vektorgrafik

5.2.1 Lesbare Grafiken

Neben der insbesondere für WWW-Anwendungen relevanten Umformulierung der (bislang lediglich *SGML*-konformen) HTML-Syntax entsprechend der XML-Norm, dem so genannten XHTML-Ansatz [vgl. Duck01:48ff], sowie die zum Austausch von beliebigen, strukturierten Datensätzen (etwa aus zwischen verschiedenen Datenbanken) interessante Möglichkeit, aufgrund der Erweiterbarkeit der XML-Sprache *eigene*, individuelle Datenstrukturen definieren zu können, steht im Rahmen dieser Diplomarbeit natürlich die Anwendbarkeit der XML-Funktionalität im Hinblick auf grafische *Präsentations*daten im Vordergrund: Aufgrund der in [4.5.4] identifizierten Problematik eines sinnvollen, logischen Aufbaus sowie der bislang

[1] vgl. [MaMu99]

nur über „Work-Arounds“ erreichbaren[1] *Menschen-Lesbarkeit* der Präsentationsdaten erscheint es daher aus mehreren Gründen sinnvoll, sich bei einer Formulierung dieser Daten des – wie in [5.1] gesehen – mächtigen XML-Frameworks zu bedienen: Zunächst einmal lassen sich insbesondere XML-basierte Grafiken (im Gegensatz beispielsweise zu den „geschlossenen“ und zumeist geschützten SWF-Dateien Flashs)[2] sehr einfach sowohl be- als auch zur weiteren Nutzung verarbeiten:

> *Man kann diese Dateien auch ohne Grafikprogramm leicht lesen, verstehen und bearbeiten […] Der Vorteil: Daten, die in der Grafik enthalten sind, lassen sich bequem weiterverwenden oder auch ganze Gestaltungselemente per Suchen/Ersetzen leicht austauschen…*
>
> [Nsw02:219]

Neben der „Human-Readability“ [vgl. Laga98] erleichtert die Text-Eigenschaft XMLs natürlich überdies die Durchsuch- und Indizierbarkeit, also die „Lesbarkeit“ XML-basierter Daten für die so genannten „Robots“ der Internet-Suchmaschinen:

> *Auch Suchmaschinen können [XML-Dateien] potenziell besser indizieren als etwa das binäre Flash-Format, insbesondere, wenn sie bereits spezielle Filter für XML besitzen.*
>
> [ibid]

Im Gegensatz zu reinen Textdaten oder dem „wirren“ Formatierungsgeflecht HTMLs besteht im Rahmen XML-basierter Dokumente indes die Möglichkeit, die Struktur und logischen Zusammenhänge Web-basierter XML-Dateien auch für Internet-Suchmaschinen transparent zu machen: Bei entsprechender Bereitstellung semantischer und syntaktischer Zusammenhänge in Form einer DTD- oder *Schema*-basierten Typendefinition wären vormals „sture“ Text-Bots daher in der Lage, derartige Strukturinformation mit in die Indizierungsdaten aufzunehmen und somit auch den Webbenutzern zur Verfügung zu stellen.

Ruft man sich noch einmal in Erinnerung, was wir im Zusammenhang mit der „semantischen Bedeutung“ und der dadurch bedingten Indizierbarkeit vektorbasierter Grafiken bereits in [4.1.1] festgestellt hatten, könnte auf dieser Basis eine Verknüpfung der Möglichkeiten XMLs mit den Vorzügen vektorbasierter Grafik [s.4.1] insbesondere im Kontext des World Wide Web die „optimale Lösung“ zur Erstellung Internetbasierter Präsentationslösungen darstellen, da die Inhalte der Präsentation – neben optisch und funktionell ansprechender Darstellung – überdies durch die Bereitstellung semantischer Strukturinformationen (mithilfe des XML-Frameworks) in Web-gemäßer Form bereitstünden und mit der Unterstützung entsprechender Internet-Suchmaschinen somit eine im abstrakten Sinne „intelligente“ Anwendung darstellen könnte.

So weit nun die Theorie – in der Praxis verfügt hingegen keines der bisher betrachteten Formate über die Gesamtheit dieser Eigenschaften: Während HTML trotz intensiver „Design“-Bemühungen [s.3.2.3] ein für Präsentationszwecke immer noch weitgehend ungeeignetes Format darstellt, so muss auch das im Hinblick auf seine Darstellungseigenschaften zweifellos interessante *Flash*-Format als wenig Web-tauglich und überdies konzeptionell problematisch [s.4.5.4] bezeichnet werden.

5.2.2 XML meets Vektorgrafik: Eine überzeugende Kombination

Aufgrund dessen rücken an dieser Stelle nun verschiedene Entwicklungsbemühungen in den Mittelpunkt unseres Interesses, welche die beeindruckende grafische Funktionalität der Flash-Technologie mit der HTML'schen Einfachheit[3] und der „intelligenten“ (s.o.) Struktureigenschaft XMLs zu verbinden suchen.

---

[1] Anm: Im Rahmen von Flash sind dies erstlinig die Möglichkeiten des „File-Dumpings“ [s.Abb.4.5.3.1.1] sowie die Abbildung in das (XML-basierte) SWFML-Format.

[2] Anm: An dieser Stelle bietet sich freilich eine Fülle so genannter „Flash-Cracker“ oder „Un-Protecter“ an, die den Bearbeitungsschutz der SWF-Datei aufheben können. Dies ist jedoch aus rechtlicher Hinsicht natürlich sehr bedenklich und soll daher im Rahmen dieser Diplomarbeit keine Rolle spielen.

[3] Anm: Dies ist insbesondere aus Erstellersicht relevant: Sowohl sind die meisten Menschen mit HTML vertrautl, als auch die Bearbeitung HTML-basierter Dokumente in einem Texteditor sehr einfach vonstatten geht. [vgl. NSW02:219]

Wenn wir nun die im Rahmen einer Präsentation anfallenden Datentypen betrachten, so wird deutlich, dass ein Präsentations-„Format“ gleich eine ganze Reihe unterschiedlicher Komponenten beinhaltet: Neben der Speicherung rein textlicher Inhalte (z.B. in Form von Stichworten) gilt es freilich ebenso, deren Struktur sowie die logischen Zusammenhänge (etwa Formatierungsangaben oder semantische Auszeichnung: Überschrift, Zitattext etc.) abzubilden. Darüber hinaus müssen sich, wie etwa die Analyse des PowerPoint-Formates zeigt, natürlich überdies etwaige Bilddaten, insbesondere natürlich vektorbasierte Schaubilder und Illustrationen, die die Präsentation zusätzlich enthalten mag, im Rahmen des Präsentationsformates abbilden lassen, wie auch eventuelle Bewegungsdaten der einzelnen Bildelemente (i.e. Animation) oder Übergänge.

5.2.2.1 XMLisiertes Flash – die Lösung?

Ein „trivialer“ Ansatz, die Gesamtheit dieser Komponenten im Rahmen einer XML-basierten Lösung könnte daher etwa die *direkte* Konvertierung des Flash-Formates, welches bekanntlich alle dieser Eigenschaften abzudecken vermag [s.4.5], in ein entsprechendes XML-Pendant darstellen. Allein die Tatsache, dass mit „Spark“ [KuCu02], *Saxess Wave* [vgl. Dabb01, Star01:46ff] sowie *JavaSWF2* [Main00], um nur einige zu nennen, gleich eine ganze Reihe von konkreten Implementierungen, die dieser Herangehensweise folgen, bereits am Markt existiert, macht die offensichtliche Nachfrage nach einem derartigen Ansatz deutlich. [Listing.4.5.3.3.2][1] veranschaulicht beispielhaft eine derartige XML-Struktur, die sich quasi direkt aus der Datei-Architektur des SWF-Formates ableitet.

Da die Analyse des Flash-Dateiformates in [4.5.3.1] jedoch gezeigt hat, dass die SWF-Architektur aufgrund der „fragmentierten“ Frame-Struktur sowie der ansonsten ungeordneten Definitions- und Kontrollelemente wahrhaftig nicht ganz unproblematisch ist, erscheint an dieser Stelle ein direktes „Mapping“ ebendieser Struktur in eine entsprechende XML-Syntax mehr als fragwürdig: So ist etwa aus der Struktur der resultierenden XML-Datei weder die „z“-Schichtung der einzelnen Elemente (da diese nicht durch die Reihenfolge der Objekte, sondern deren `depth`-Attribut determiniert wird), noch der Zusammenhang der Animation direkt ersichtlich. Die konzeptionellen Vorteile des SWF-Formates (Binärkomprimierung, schnelle Darstellung) würden überdies durch eine derartige Konversion wegfallen – das resultierende XML-Pendant wäre zugleich jedoch durch die nicht unerheblichen Nachteile der Flash-Dateistruktur gekennzeichnet.

Darüber hinaus lässt freilich zudem die Tatsache, dass Flash nach wie vor eine „proprietäre“ Technologie darstellt [vgl. Abbe00, Doug01] sowie vollständig ausbleibende Norm- oder Standardisierungsbemühungen seitens der Herstellerfirma Macromedia [vgl. Cagl01], an dieser Stelle den Schluss zu, dass ein derartiges, XML-basiertes Flash-Derivat wohl kaum „die Lösung unserer Probleme“ darstellen kann.

5.2.2.2 Die saubere Alternative: XML-Vektorformate from Scratch

Unsere Aufmerksamkeit soll sich in den folgenden Abschnitten daher vielmehr auf von Grund auf neu konzipierte Formate konzentrieren, die eine konsequentere Umsetzung der XML-Architektur verfolgen, gleichzeitig jedoch den spezifischen Anforderungen der Web-Umgebung Rechnung tragen und überdies insbesondere direkt im Hinblick auf eine mögliche, offizielle *Standardisierung* von Seiten des W3C-Konsortoums entwickelt wurden.

[1] s. hierzu in [4.5.3.3]

5.3 XML-basierte Vektorstandards im Web

Aufgrund der Tatsache, dass sich das Darstellungsprinzip sämtlicher Präsentationsanwendungen (PowerPoint, Harvard Graphics, Flash…) stets durch deren maßgebliche *Vektor*-Eigenschaft kennzeichnen lässt [s.4.4], erscheint daher eine Betrachtung der Entwicklung XML-basierter Internet-Vektorformate, auch im Rahmen dieser Diplomarbeit von Interesse: Insbesondere im Hinblick auf den derzeitigen WWW-Vektorstandard SVG [s.5.4], welcher weitgehend aus diesen Entwicklungsbemühungen hervorgegangen ist, gilt es daher zu untersuchen, inwieweit sich diese Vektorformate als „Trägermedien" für multimediale Präsentationen im Web eignen. Einen Hinweis darauf, dass einige dieser Ansätze bereits „deutlich besser als PowerPoint" [Fibi01:99] sind und somit überaus interessante Ansatzpunkte zur Entwicklung eines *alternativen Präsentationskonzepts* darstellen [s.Kap.6], gibt an dieser Stelle unter anderem die Diplomarbeit von Iris Fibinger, die sich etwa mit dem SVG-Standard auseinander gesetzt hat:

> *Generell spricht für diese Standards die Tool-Unabhängigkeit, die auf Power-Point natürlich nicht zutrifft… Unterstützung in Sachen Präsentationen mit SVG erhalten wir bereits heute von Software-Firmen, die spezielle Programme entwickelt haben, mit denen sich aus beliebigen XML-Dokumenten – z.T. „on the fly" – [ansprechende] Präsentationen erstellen lassen.*
>
> [Fibi01: 99]

5.3.1 HGML

5.3.1.1 Hintergrund

Der erste, extrem einfach gehaltene Ansatz zur Definition eines vektorbasierten Standards fußt indes auf dem bereits angesprochenen Konzept der bandbreitenschonenden Übertragung von polygonalen Bilddaten. Es überrascht auch unter Beachtung des zum Entwicklungszeitraum[1] aufkommenden Handy-Booms daher nicht, dass sich dieser Entwurf aufgrund seiner strukturellen Einfachheit primär auf die Anwendung im Mobilfunkbereich konzentriert, obschon der Einsatz auf „allen möglichen Internet-Endgeräten"[2] ausdrücklich erwähnt wird.

Bereits die Zusammenstellung der beteiligten Entwickler gibt Aufschluss über diese Orientierung: Im Frühjahr 1997 erstellt das Labor für Elektro- und Kommunikationstechnik an der Universität Plymouth in Zusammenarbeit mit dem britischen Mobilfunkbetreiber Orange PCSL erste Machbarkeitsstudien[3] über die Entwicklung eines möglichen vektorbasierten Formates für die bandbreitenschonende Grafikübertragung auf GSM-Handys: Aufgrund der äußerst geringen im derzeit verbreiteten GSM-Netz maximalen Bandbreite von nur etwa 9600 Bit pro Sekunde hat in diesem Bereich eine möglichst volumenarme Übertragung von Bilddaten oberste Priorität. Wegen der bereits im vorangegangenen Kapitel erwähnten Vorzüge polygonorientierter Grafikformate gegenüber Pixelgrafiken hinsichtlich der beanspruchten Datenvolumina [s.4.1.1] bietet sich natürlich besonders auf diesem Gebiet die Anwendung von vektorbasierten Bilddateien an. Schnell wurde den Entwicklern daher die Notwendigkeit eines möglichst universell einsetzbaren Vektor-Standards deutlich. Zur Begünstigung eines möglichen Standardisierungsprozesses entschieden sich die Forscher zudem für eine Formulierung der Grafikdaten in textlicher statt (wie im vorausgegangenen Kapitel erläutert) binärer Form. Der Arbeitstitel des geplanten Formates, das von Seiten der Universität wie auch des Mobilfunkbetreibers gleichermaßen propagiert wurde: Hyper Graphics Markup Language (HGML).

---

[1] Anm: Hierbei ist der grobe Zeitraum um Formatdefinition (Juli 1997) und Entwurfsvorlage (Juni 1998) gemeint.

[2] "HGML is applicable to all types of Internet device" [HGML98]

[3] vgl. [RFEP99]

5.3.1.2 Aufbau und Syntax

Da sich allerdings zu Beginn des Formatentwurfs auch die Entwicklung des Wunschformates XML noch in einem sehr frühen Stadium befand,[1] definierten die Entwickler sowohl eine „Tag-basierte" als auch eine weitere, „minimalistische" Notation, die der ersteren semantisch jedoch vollständig entspricht. Daher orientiert sich auch die *tag based* Syntax des Formates [HGML98] eher lose an der letztendlich verabschiedeten Konvention [XML98] - die Autoren sprechen an dieser Stelle trotz selbsterklärter XML-Fixierung [2] lediglich von „HTML-like Tags".[3] Die Idee der verkürzten, minimalistischen Notation wird zudem von den Entwicklern an späterer Stelle wieder etwas zurückgestellt.[4]

5.3.1.2.1 Grundsätzlicher Aufbau

Grundkonzept des Format-Prototypen ist in beiden Notationen die Repräsentation trivialer Polygontypen (auch „Primitives" genannt) durch eine zugunsten von Bandbreite und Performance möglichst reduzierte Auswahl an Grafik-Elementen. Eine einfache Linie würde demnach in HGML wie folgt abgebildet:

```
<line coords="10,10,20,30" color="red" style="solid" fill="yellow" psize="3">
```

Listing 5.3.1.1: Linien-Primitive in HGML (Tag-Notation)

```
LINE 10,10,20,30,3,1,3
```

Listing 5.3.1.2: Linien-Primitive in HGML (Minimalistische-Notation)

Die *Attribute* des verwendeten Elements (dies kann sowohl das vorausgestellte Beispiel als auch `circle`, `square`, `polygon` etc. sein) geben hierbei die zu spezifizierenden Eigenschaften des jeweiligen „Primitives" dar, wobei diese durch einen numerischen Wert (etwa „3" für die Kantendicke `psize`), Bezeichner („red") oder auch ein Variablen-Array (wie etwa die Koordinatenangaben) repräsentiert werden können.

Verwirrend erscheint in diesem Kontext der Attributvergabe die Verwendung einer `style`-Eigenschaft [vgl. HGML98] Da in der Spezifikation eingangs explizit von „HTML-like Tags" die Rede ist, überrascht an dieser Stelle die Einführung eines dem (zum Formlierungszeitpunkt bereits weit verbreiteten) HTML-Schlüsselwortes zur *Style-Sheet*-Definition. Da es sich hierbei ja ausdrücklich *nicht* um Stile gemäß der CSS-Konvention handelt, mutet dies, zusammen mit einem ebenso definierten `<setstyle>`-Tag, zumindest etwas seltsam an.

5.3.1.2.2 Multimedia: Fehlanzeige

Über die Formulierung einfacher Vektorformen hinaus bietet HGML, wie die Entwickler selbst bemerken,[5] jedoch keinerlei Multimedia-Funktionalität: Weder die Referenzierung externer Bitmaps, noch Scripting-Möglichkeiten oder, wie an anderer Stelle [Farm99] irrtümlich behauptet,[6] Animationseigenschaften werden durch die Spezifikation realisiert.

Etwas enttäuschend erscheint auch die Umsetzung von Text-Elementen: Hier fallen die vorhandenen Gestaltungsmöglichkeiten insbesondere hinsichtlich typografischer „Kontrolle" drastisch gering aus.[7] Da diese

---

[1] vgl. [HGML98]
[2] "HGML could now be defined using XML…" [ibid]
[3] ibid.
[4] "There was a natural leaning towards the HTML approach..:" [ibid]
[5] "HGML is only applicable to certain types of image…" [HGML98]
[6] "[…] The ability to animate [shapes] to paths." [Farm98]
[7] Lediglich die Auswahl aus 3 vordefinierten Schrifttyen (TimesRoman | Helvetica | Dialog) sowie einfache Stilauswahl sind spezifiziert [vgl. HGML98]

in der Funktionalität selbst hinter HTML weit zurückstehen, kann an dieser Stelle auch die Einschätzung der Wiener Informatik-Dozentin Shermin Voshmgir [Vosh99], HGML stelle die „grafische Alternative zu HTML" dar, nicht nachvollzogen werden.

Spezielle Besonderheit von HGML ist nach dem Konzept der Entwickler hingegen, *vorgefertigte Formen* (etwa *ARROW1*, *SEARCH*, *LOGO*, *BULLET1*…), die bereits auf dem jeweiligen Endgerät lokal abgespeichert sein müssen, mit entsprechenden Positionierungsangaben über die dafür bereitgestellten Tags `<clipart>` und `<theme>` ansprechen und darstellen zu können: Die Verwendung einer Reihe häufig verwendeter Symbole würde somit zu einer zusätzlichen Einsparung von Bandbreite führen. Ein Pfeil könnte beispielsweise dementsprechend statt der Definition mehrerer Polygone durch die einfache Angabe eines `clipart`-Tags realisiert werden:

```
<CLIPART name="ARROW1" coords="10,10" width="20" height="50">
```

Listing 5.3.1.3: Einfügen eines Pfeiles via „clipart"

Diese Eigenschaft wurde allerdings aufgrund einer recht irreführenden Formulierung im Rahmen der Spezifikation [HGML98] sowie entsprechender Presseverlautbarungen oft fehlinterpretiert – so waren selbst Mobilfunk-Experten [Bozw00] letztendlich der Ansicht, HGML sei ein Standard, in dem überhaupt keine Bilddaten selber, sondern lediglich Verweise auf lokale Cliparts enthalten seien.[1]

Ein weiteres, bemerkenswertes Feature bilden in der HGML-Spezifikation hingegen verschiedene Bildmanipulations-Tags, die es ermöglichen, vorhandene Elemente – wie etwa die zuvor bereits angesprochenen Clipart-Gruppen – beliebig oft platzieren (`<copy>`,`<paste>`), drehen oder anderweitig manipulieren zu können. Obgleich nur sehr rudimentär umgesetzt, kann diese Funktionalität durchaus als Inspiration für die Wiederverwendungs- (`<use>`,`<symbol>`) und Manipulations-Tags der später folgenden SVG-Spezifikation [vgl. das hierauffolgende Kapitel] betrachtet werden.

5.3.1.3 Umsetzung

Obgleich die Wissenschaftler der Universität Plymouth, zusammen mit Forschern von Orange PCSL, im Sommer 1997 mit dem im Mai entwickelten[2] Prototypen bereits erste Feldversuche unternahmen [RFEP99] sowie eine Java-Applet zur HGML-Darstellung in Internet-Browsern[3] entwickeln konnten, veranlasste erst die im Frühjahr 1998 publizierte Spezifikation des ebenfalls vektorbasierten Internet-Grafikformates PGML[4] die HGML-Initiatoren zu hektischer Aktivität [Heis98]: Im Juni 1998, sogar noch nach der Veröffentlichung der ebenfalls mit PGML konkurrierenden VML-Note [VML98] im Mai desselben Jahres, reichte das britische Konsortium schließlich die HGML-Spezifikation beim World Wide Web-Konsortium (W3C) als *Note* (Erster Antragsschritt zur WWW-Standardisierung) ein. Bedauerlicherweise sieht man dem Dokument doch sehr deutlich an, dass der Antrag offensichtlich mit recht heißer Nadel gestrickt wurde: Neben falscher Tag-Formulierung,[5] mangelhafter Strukturierung und irreführenden Formulierungen[6] lässt das Papier vor allem Umsetzung und Anwendung des Standards offen: So stellen u.a. [NeWi00] lapidar fest, dass – trotz des bereits 1997 veröffentlichten Prototypen und Java-Applets, „keine Umsetzung" des HGML-Formates erfolgt sei.[7]

---

[1] „[…] Never transmitting the picture …" [Bozw00]

[2] vgl. [HGML98]

[3] s. hierzu auch Kap. 3.5.3 (Java-Applets)

[4] [PGML98], vgl hierzu auch 5.3.2 (PGML)

[5] Die in der Spezifikation notierten Tag-Attribute werden irreführenderweise durch Kommas separiert, was jedoch weder der HTML- noch XML-Konvention entspricht.

[6] s.o. im selben Kapitel (*Vorgefertigte Formen*) sowie [Bozw00]

[7] vgl. [NeWi00]

5.3.1.4 Status

In der Tat erhielt die eingereichte Konvention trotz großer Ankündigung[1] nie die von den Entwicklern gewünschte oder andernorts [Farm98] prophezeite Anerkennung – geschweige denn Statusänderung seitens des W3C: Das Konsortium entschied sich,[2] wohl aufgrund der Unausgereiftheit der Spezifikation[3] wie sicherlich auch der parallelen Auseinandersetzung um die PGML- und VML-Einreichungen [vgl. Cagl02], dem Kommissions-Antrag der HGML-Entwickler den angestrebten *Working Draft*-Status (der einen möglichen nächsten Schritt in Richtung Standardisierung dargestellt hätte) zu verweigern.

Somit wurde sowohl den Entwicklern der Universität Plymouth wie auch den Mobilfunkbetreibern der Weg zur ersehnten massentaugliche Umsetzung und Verbreitung der HGML-Konvention versperrt. Dennoch arbeitete das Elektrotechnische Institut der Universität auch nach der sich abzeichnenden W3C-Ablehnung in den Folgejahren noch an weiteren Softwarekomponenten für das entwickelte Format, wie etwa einen in VisualBasic realisierten, prototypischen HGML-Editor.[4]

Nach dem schließlich, besonders in bandbreitentechnischer Hinsicht sehr „enttäuschenden" [Reus00] Bildformat WBMP des Mobilen Internet-Standards WML [WAP98] bemühte der seinerzeit seitens der Universität Plymouth für die HGML-Entwicklung verantwortlich zeichnende IT-Dozent Steven Furnell gar ein weiteres Forschungsprojekt [Film99] zur Analyse der möglichen Vorzüge HGMLs gegenüber dem letztendlich zur Anwendung gekommenen WBMP, welches sich gar die Erweiterung und Neu-Definition von HGML zur Aufgabe machte.[5]

5.3.1.5 Fazit

Auch letztgenannte Bemühungen konnten jedoch schlussendlich nicht verhindern, dass HGML zum heutigen Tage, auch angesichts der letztlich ausbleibenden Aktivität auf diesem Gebiet, als gescheiterter Formatentwurf bezeichnet werden muss. Im Rahmen dieser Diplomarbeit ist die Sprache aufgrund ihrer „Over-Simplifiedness" [vgl. Arci02] sowie, wie bereits ausgeführt,[6] fehlender Multimedia-Eigenschaften, hinsichtlich einer Anwendung als „Master-Format" zur Multimedia-Darstellung ohnehin nur von begrenztem Interesse und wird daher im Rahmen weiterer Überlegungen in diese Richtung wohl keine bedeutende Rolle einnehmen.

5.3.2 PGML

5.3.2.1 Hintergrund

Weitaus interessanter erscheint daher im Hinblick auf Multimedia-Eigenschaften und grundlegender Struktur das bereits im April 1998 als W3C-Note eingereichte Konzept der „Precision Graphics Markup Language", kurz: PGML [PGML98]. Im Gegensatz zum vorausgehend besprochenen Wireless-Vektorformat HGML,[7] für welches lediglich jeweils ein universitärer und industrieller Zweig verantwortlich zeichnete,[8] wurden die Bemühungen um die Standardisierung der PGML-Konvention von den vier im Grafik- und Internetbereich führenden US-amerikanischen Unternehmen Adobe, IBM, Netscape und Sun unterstützt.

---

[1] "The restrictions of cellular technology should be overcome…" [Rein98a]

[2] „Das Konsortium wird sich grundsätzlich einigen müssen, bevor ein Entwurf veröffentlicht werden kann." [Heis98]

[3] s.o.; 5.3.1.3

[4] vgl. [Film99]

[5] "The work should then seek to extend the HGML implementation…" [ibid]

[6] vgl. 5.3.1.2 – Multimedia-Eigenschaften.

[7] s. Kap. 5.3.1

[8] Universität Plymouth und Orange PSLC. [vgl. Kap. 5.3.1.1]

Die Federführung des Projektes lag jedoch beim kalifornischen Computergrafik-Giganten[1] Adobe: Aufgrund des überwältigenden Erfolgs[2] des Grafikstandards *PostScript* im Offline-Bereich[3] und des 1997 aufkommenden „Hypes" um den angekündigte SGML-Nachfolger XML[4] [vgl. CKR97, MACH97] bemühten sich die Entwickler der Adobe-Forschungslabors[5] um eine „XMLisierung" [Wine98a] des zum Zeitpunkt der Entwicklung bereits weit verbreiteten PostScript-Standards: Wie in der Spezifikation bereits einführend bemerkt,[6] orientiert sich das Format damit speziell bei der Definition von Polyogonelementen *semantisch* sehr stark an PostScript (und damit auch am Nachfolger PDF)[7] – die Syntax entspricht hingegen vollständig und übrigens strukturell weitaus „sauberer" umgesetzt als im Konkurrenzformat VML[8] oder dem Wireless-Pendant HGML[9] der praktisch parallel entwickelten und veröffentlichten[10] XML-Konvention.

Die trotz der noch erkennbaren Unausgereiftheit[11] und Entwicklungsdynamik des XML-Standards [vgl. MACH97] bereits derart konsequente und elegante Umsetzung der Konvention verdankt die PGML-Arbeitsgruppe unter anderem auch der Mitarbeit des Netscape-Entwicklers Kevin McCluskey am PGML-Entwurf [PGML98], dessen Unternehmen ja zeitgleich in der XML-Formulierung involviert war. Die Netscape-Kooperation versprach überdies noch die Möglichkeit einer baldigen Integration eines PGML-Viewers in Netscapes seinerzeit noch marktführendem Internetbrowser – ein aufgrund der Schlüsselfrage der Browserunterstützung nicht unwichtiger Umstand. Die darüber hinaus in PGML engagierten Unternehmen IBM und Sun Microsystems stellten zudem eine mögliche Schnittstelle zur boomenden[12] Internet-Programmiersprache Java[13] dar.

5.3.2.2 Semantik und Syntax

Wie bereits angesprochen,[14] baut die PGML-Semantik vollständig auf Struktur und vor allem *Imaging Model*[15] der PostScript- und PDF-Konventionen auf. Im Gegensatz dazu weicht PGML allerdings syntaktisch von der inversen polnischen Notation des PostScript-Formates[16] wie auch der linearisierten Fassung in PDF-Dokumenten [s.4.2.2] ab. Vielmehr am Objektkonzept der ebenfalls in der PGML-Spezifikation vorgesehenen Programmiersprache Java [vgl. PGML98, Bayn98] und der Attribut-basierten Notation in XML orientiert, kann au diese Weise etwa ein Rechteck-*Primitive* mit einem einfachen XML-Tag abgebildet werden:

```
<rectangle x="100" y="100" width="100" height="100"/>
```

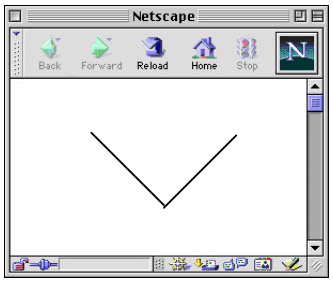
Listing und Abb. 5.3.2.1: Rechteck-Primitive in PGML.

Die Grafik-Attribute aller jeweiligen Elemente, die den Eigenschaften entsprechender PostScript und PDF-Objekte semantisch vollständig entsprechen, sind hierbei deutlich sinnvoller und durchdachter in die XML-

[1] vgl. [Piff02]
[2] ibid.
[3] s. hierzu auch Kap. 4.2.1
[4] s. Kap. 5.1
[5] Nabeel Al-Shamma, Robert Ayers, Richard Cohn, Martin Newell und der spätere SVG-Herausgeber Jon Ferraiolo, vgl. hierzu auch die Autoren- und Copyright-Anmerkungen in [PGML98]
[6] vgl. [PGML98]
[7] s. hierzu auch Kap. 4.2.2
[8] s. hierzu das folgende Kapitel 5.3.3 (VML)
[9] s. Kap. 5.3.1.4
[10] vgl. [XML98]
[11] "By December 1997, the W3C had published a formal XML specification, and the language emerged to… well, yawns." [Cagl02] p.8
[12] vgl. [Orch97]
[13] s. hierzu auch Kap. 3.5.3
[14] s. Kap. 5.3.2.1
[15] vgl. [PGML98] Kap. 2.4 und 3
[16] s. Kap. 4.2.1

Konvention eingearbeitet, als dies etwa im HGML-Format [HGML98] erkennbar ist. Darüber hinaus ist eine Definition von CSS-Stilangaben, die stets Vorrang vor ihren jeweiligen XML-Pendants (etwa bei Farbangaben) haben, gemäß der PGML-Spezikation möglich. Besonders gelungen und dem Nachfolge-Standard SVG,[1] der hier nach [Behm02] „völlig un-XMLisch“[2] vorgeht, an dieser Stelle sogar leicht überlegen ist meiner Ansicht nach in PGML die Realisierung von Freiform-Objekten über das `<path>`-Element gelöst: Statt mittels „unstrukturierter“ [Heis98] Koorinatenangaben ordnet PGML die einzelnen Knotenpunkte als Tochterelemente dem jeweiligen Pfadelement unter:

```
<path>
<moveto x="100" y="100"/>
      <lineto x="200" y="200"/>
      <lineto x="300" y="100"/>
      <closepath/>
</path>
```



Listing und Abb. 5.3.2.2: Polygonpfad in PGML (links) und seine Darstellung (rechts)

Ebenso angenehm fallen einige besonders im Internet sinnvolle und benötigte [vgl. Bayn98] Erweiterungen auf, die den angegrauten PostScript-Standard vorsichtig und zukunftsweisend [Rein98b] im Hinblick auf Web- und optische Funktionalität erweitern. Insbesondere seien hier, wie bereits seitens der Entwickler angesprochen,[3] die erweiterten Kontrollmechanismen hinsichtlich *Anti-Aliasing* und Transparenz genannt.

Weiteres Lob verdienen die auch unter Berücksichtigung der im selben Jahr veröffentlichten SMIL-Spezifikation [SMIL98] ausgereift, wenn auch hinreichend schlicht[4] erscheinenden Animationsmöglichkeiten nahezu sämtlicher PGML-Elemente,[5] ebenso wie Manipulations- und DOM-Funktionen mittels ECMAScript.[6]

Einzig die typografischen Gestaltungsmöglichkeiten fallen erneut etwas enttäuschend aus: zwar sind absolute Positionierung von Textelementen sowie Zuweisung von Schriftarten über CSS-Angaben möglich – „wirklicher“ Textfluss ist trotz des Anspruchs der Durchsuchbarkeit,[7] ebenso wie eine besonders von Grafikern erwünschte absolute Kontrolle über Schriftbild und –Größe in Ermangelung von *Font-Embedding* [8] jedoch nicht möglich. Lediglich in [Rein98b] wird eine Browserseitige Unterstützung des Schriftsystems *OpenType*[9] angesprochen,[10] welches eine Einbettung von Schriftarten zumindest im Rahmen einer HTML-Integration erlauben würde. Da es sich, trotz der beeindruckenden Animations- und Scripting-Funktionalität, bei PGML primär um eine Webbasierte Grafik-Beschreibungskonvention handelt, fehlen zudem wichtige Multimedia-Funktionen wie etwa eine mögliche Soundunterstützung oder bewegte Videosequenzen (wobei letztere sich allerdings über „Workarounds“, zumindest theoretisch,[11] realisieren lasen).

5.3.2.3 Umsetzung

Nicht nur hinsichtlich der inhaltlichen Konzeption, sondern auch zeitstrategisch bewies das hinter PGML stehende Konsortium[12] ein gutes Gespür: bereits wenige Wochen nach Verabschiedung des PGML zugrunde

[1] s. Kap. 6
[2] vgl. [Heis98] p.54
[3] vgl. [PGML98] Kap 2.3 Pkt. 2
[4] vgl. [Bayn98]
[5] ebenda, Appendix C.7
[6] vgl. [ECMA97]
[7] vgl. [PGML98] Kap 2.3 Pkt. 13
[8] Einbettung von Schriftarten zur ubiquitären Verwendung
[9] Dieses wurde leider nie vollständig implementiert [s.4.4]; vgl. hierzu [Wild99] pp.443f sowie [Meis97]
[10] “Font Mapping: CSS2 (Browsers will use OpenType)” [Reid98b]
[11] Hierbei kämen dann u.a. die Animations- und Scriptability-Eigenschaften der PGML-Spezifikation zum tragen.
[12] s. 5.3.2.1

liegenden Dokumentstandards XML durch das W3C [XML98] konnten auch die PGML-Entwickler den Spezifikationsentwurf für „PGML 1.0“ in Gestalt einer *Note* beim Web-Konsortium im April 1998 einreichen [PGML98]. Durch die hiermit eingenommene Vorreiterrolle des Formatentwurfs gegenüber der zeitgleich in Entwicklung befindlichen, ebenfalls text- und vektorbasierten Ansätze der Konkurrenz (namentlich VML,[1] HGML[2] und DRAWML[3]) besaß PGML neben einem zeitlichen nun auch einen diskussionsstrategischen Vorteil: Während die konkurrierenden Arbeitsgruppen durch die frühzeitige PGML-Einreichung zu fast hektischer Aktivität angetrieben wurden und einige Formataspekte teilweise gar unausgereift erschienen ließen,[4] konzentrierte sich auch die öffentliche Diskussion in der Internet-Community vorrangig auf die PGML-Spezifikation.

5.3.2.3.1 Export-Filter

Nach lobender Erwähnung durch zahlreiche Internetgrößen [Wine98c, Rein98b, Bayn98] kündigten, auch unter dem Eindruck der namhaften, hinter PGML stehenden Unternehmen, eine Reihe von Softwareherstellern Schnittstellen für den PGML-Ex- und Import an. Besonders die problemlose Anpassung gegebenenfalls bereits existierender PostScript-Exportalgorithmen zur Realisierung möglicher PGML-Filter erwies sich hierbei als überzeugendes Argument [vgl. Will01, Mose99]. Neben Adobe (v.a. mit Illustrator und Photoshop) selber waren dies unter anderem das im Consumer-Bereich führende CorelDraw,[5] wie auch überraschenderweise die mit einem eigenen Webformat-Entwicklung gescheiterten[6] Xara-Entwickler.[7]

Aufgrund der XML-Orientierung des PGML-Formates wurden überdies zahlreiche Implementierungen auf dem eher grafik-fremden Bereich des Software Engineering realisiert:[8] So stellt PGML etwa bis heute das grafische Export-Format des offenen UML-Modelers ArgoUML [Argo98] dar – obgleich seit geraumer Zeit Pläne existieren, das „veraltete“ PGML[9] durch den bereits vorhandenen SVG-Filter [10] zu ersetzen.

5.3.2.3.2 PGML goes Web: Erste PGML-Viewer

Zur Erleichterung einer erhofften Browser-Integration des vorgestellten Formates entwickelte Adobe zeitgleich mit der Spezifikations-Veröffentlichung eine relativ schlanke [11] ActiveX-Komponente namens „Coffe Browser“ zum Betrachten von PGML-Dateien [Doug98], die, da eine baldige Netscape-Integration allein schon durch die Mitarbeit Kevin McCluskeys[12] als gesichert galt [Ricc98], vor allem die Kompatibilität mit Microsofts an sich „PGML-feindlichem“[13] Internet Explorer sicherstellen sollte.

Um die Attraktivität des Formates für Softwareentwickler [vgl. Argo00, Muel00b] wie die Kompatibilität hinsichtlich alternativer Internet-Browserprogramme [Tetz95, Kirs98] zu erhöhen, wurde im Sommer 1998 mit der zusätzlichen Entwicklung eines PGML-Java-Viewers begonnen, der die Einsatzmöglichkeiten des Formates darüber hinaus auf mobile Endgeräte[14] erweitern sollte. Überraschenderweise beteiligte sich Java-Erfinder Sun Microsystems, zumindest auf dem Papier ja ebenfalls Mitglied des PGML-Konsortiums,

---

[1] s. 5.3.3

[2] s. 5.3.1

[3] s. 5.3.4

[4] vgl. [HGML98] sowie 5.3.1.4

[5] vgl. die entsprechende Pressemitteilung von Adobe: “Adobe Submits Proposal to Improve Quality of Web Graphics with IBM, Netscape, and Sun.” San Jose, 13. April 1998

[6] s. Kap.4.3

[7] vgl. Chris Dickman’s Kommentar in [Wine98b]

[8] vgl. hierzu auch [Muel00b]

[9] vgl. [Argo00] Kap. 2.5.3.2.2

[10] ebenda, Kap. 2.5.3.2.5 ff.

[11] vgl. [Walt98]

[12] s. 5.3.2.1

[13] Microsoft war Haupt-Initiator des konkurrierenden Vektorstandards VML [VML98] s.a. 5.3.3

[14] s. 5.3.1; vgl. hierzu ebenso [HGML98]

nicht direkt an der Entwicklung des Viewers: Lediglich Adobe war neben dem federführenden Programmiererteam bei IBM an der Entwicklungsarbeit beteiligt.[1]

Aufgrund der sehr elegant gelösten Entsprechung des PGML-Objektmodells mit der Programmiersprache Java [vgl. Bayn98, PGML98] überzeugten schließlich die „hohe Performance und geringe Größe" der Java-Umsetzung,[2] welche noch im Herbst desselben Jahres auf der XML-Konferenz in Chicago [Doug98] vorgestellt werden konnte [Walt98].

5.3.2.4 Status

An der W3C-Front verdüsterten sich hingegen bereits wenige Monate nach Veröffentlichung der PGML-Note die Aussichten auf eine zügige Standardisierung des eingereichten Formates: Aufgrund strategischer Überlegungen zum Schutz des marktführenden Flash-Formates[3] [Star01:12] beteiligte sich Adobe-Erzrivale[4] Macromedia an der Veröffentlichung des konkurrierenden Microsoft-Ansatzes zur XML-basierten Vektorgrafik: VML [VML98]. Dies führte in Verbindung mit der daraufhin hastig veröffentlichten HGML-Note [HGML98] schließlich zu einer Blockade-Situation innerhalb des Web-Konsortiums, welches sich aufgrund der verzwickten Konstellation zu keiner Entscheidung durchringen mochte [Heis98]. Letztendlich entschied sich das W3C jedoch nach langer und heftiger Diskussion[5] schweren Herzens, beiden Einreichungen die *Final*-Anerkennung zu verweigern (gleichbedeutend mit der Einstellung aller Bestrebungen zur Standardisierung des Formates), und strebte stattdessen einen von beiden Seiten akzeptierten Kompromiss an, der schließlich in das [im hierauffolgenden Kapitel behandelte] SVG-Format mündete.

Der Einfluss des PGML-Ansatzes ist jedoch besonders im Hinblick auf SVG nicht zu unterschätzen: So fand ein Großteil der Vorarbeit am nach dem W3C-Entschluss unverändert belassenen PGML-Entwurf, im Gegensatz zu anders lautenden Stimmen,[6] den Weg in die endgültige SVG-Spezifikation [SVG01], die wiederum in weiten Teilen von (zumindest strukturellen) PGML-Konzepten bestimmt wird. Dies liegt nicht zuletzt am Beitrag des Adobe-Technologiechefs Jon Ferraiolo, der bereits maßgeblich an der Formulierung des PGML-Entwurfs mitgearbeitet hatte und schließlich die Federführung der SVG-Spezifikation übernahm.

5.3.2.5 Fazit

Ursprünglich als reiner Diskussionsvorschlag eingereicht,[7] ist das bereits im Entwurfsstadium sehr ausgereift wirkende PGML-Format trotz allen Hohns[8] als durchaus bemerkenswerter Pionier der XML-basierten Vektorstandards zu betrachten. Neben den in SVG als PGML-originär erkennbaren Elementen ist der bleibende Beitrag von PGML daher in der für „damalige Verhältnisse" erstaunlich klaren syntaktischen Formulierung zu sehen, die selbst dem „Nachfolger" SVG in mancher Hinsicht überlegen ist. Allein die Tatsache, dass die Anzahl der Unterschiede und Verbesserungen bei genauerem Vergleich des frühzeitigen PGML-Entwurfs [PGML98] mit der erst gut drei Jahre später verabschiedeten SVG-Spezifikation [SVG01] überraschend gering ausfällt, lässt im Hinblick auf die für diese Diplomarbeit relevante multimediale Funktionalität lediglich den Schluss zu, dass durch die strategische W3C-Blockade seitens des VML-Konsortiums [s.o.] – zur großen Freude Macromedias – erhebliche Zeit „verschwendet" wurde. Eine Verwendung der PGML-Konvention wäre somit auch im Rahmen eines präsentationsbezogenen Prototypen,[9] die notwendige, frei-

[1] vgl. [Walt98]
[2] "[…] The Java viewer is only 300K in size…" [ibid]
[3] s. 4.4
[4] vgl. [Piff02]
[5] vgl. [Cagl02] p.10
[6] vgl. [KrMa00, NeWi00]
[7] "The important thing to remember… is that [it is an] experiment … contributed as a starting point for discussion". [Reid98b]
[8] "Have you heard of PGML? I don't think so." [Will01]
[9] s. ebenso Kap. 6

lich aufgrund des W3C-Kompromisses nicht zustande gekommene Browser-Integration und Massenverbreitung einmal hypothetisch angenommen, aufgrund der extrem hohen Java-Affinität [vgl. Bayn98] und überzeugenden Architektur des Formates, auch noch im direkten Vergleich mit dem letztendlich angewandten SVG[1] durchaus interessant gewesen.

5.3.3 VML

5.3.3.1 Ursprung

Bereits 1997 hatte der zum Jahresbeginn aufkommende XML-„Hype" [HoDu00] auch weite Teile des Software-Giganten Microsoft ergriffen, sodass führende Kräfte der Entwicklerschaft [Wu99] frühzeitig über eine Ablösung des ungeliebten und veralteten [2] WMF-Austauschformates nachdachten. Nach Bekanntwerden der ersten Details [Mach97] der noch im Entwurfsstadium befindlichen XML-Konvention [XML97] machten sich die Microsoft-Programmierer daher schon sehr bald an die Spezifikation eines möglichst XML-konformen Grafikformates, das erst später Kontur annehmenden sollte. Aufgrund der Neuausrichtung des Unternehmens auf speziell *Internet*-basiertes Authoring und Datenverarbeitung wurde im Verlaufe dieser Entwicklungsarbeit eine sehr enge Kopplung des geplanten Formates an das bestehende, populäre HTML [3] immer deutlicher. Der Arbeitstitel des Entwicklungsprojektes lautete daher schlicht „Vektor Markup Language", oder kurz: VML.

VML als „proprietäres, textbasiertes 2D-Vektorformat von Microsoft" zu bezeichnen [NeWi00], ist hingegen trotz der Schlüsselrolle des Softwareunternehmens nicht ganz zutreffend: An der Entwicklungsarbeit an VML beteiligt waren darüber hinaus noch die ohnehin mit Microsoft strategisch verbundenen[4] Unternehmen Autodesk, Hewlett-Packard und der schließlich von Microsoft selbst geschluckte [Marx99] Flow-Chart-Hersteller Visio. Zuletzt schloss sich auch Grafikproduzent Macromedia den Entwicklungsbemühungen um die VML-Spezifikation an – die verdächtige Überschneidung des bereits etablierten[5] Flash-Formates mit Komponenten des VML-Entwurfs [vgl. Micr98a][6] lässt jedoch an dieser Stelle den Schluss zu, dass diese Mitarbeit primär aus strategischen Gründen[7] denn durch „echtes Interesse" [Wu99] an einem Erfolg von VML motiviert war.

Die „Initialzündung" zur Veröffentlichung der Spezifikation gab paradoxerweise eine Initiative der unternehmerischen „Gegenseite": Nach der sehr frühen Veröffentlichung der Spezifikation des konkurrierenden Vektor-Formats PGML[8] als W3C-*Note* im April 1998 [PGML98] sah sich zunächst Macromedia genötigt [Star01:12], die Struktur des bislang geheim gehaltenen Flash-Internetformates SWF[9] zu veröffentlichen [Macr98]. Das Unternehmen drängte in der VML-Arbeitsgruppe zudem auf eine rasche Einreichung einer Spezifikation von VML an das Web-Konsortium, um hinsichtlich der öffentlichen Diskussion und eines erhofften Standardisierungsverfahrens mögliche Zeit- und Chancenverluste gegenüber der PGML-Konkurrenz zu vermeiden.[10] VML hingegen, wie etwa Microsoft-Produktmanager Steve Sklepowich ausführt, als „Kom-

---

[1] ibid.

[2] s. 4.3.2.1: Windows Metafile.

[3] s. 3.2

[4] vgl. Hierzu die entsprechende Presseerklärung von Autodesk [„Autodesk Named Microsoft Global Partner of the Year…", San Rafael, Cal. 16. Juli 2001] sowie die offizielle „HP/Microsoft-Allianz"-Seite von Hewlett-Packard [http://www.hp.com/solutions1/microsoft]

[5] s. 4.5

[6] "VML is a text-based interchange and delivery format for vector graphics, whereas Flash is a binary format for the delivery of vector-based graphics and animation" [Micr98a]

[7] gemeint ist an dieser Stelle der „Schutz" des Flash-Formates vor der von Adobe entwickelten PGML-Spezifikation [s.a. 5.3.2.4]

[8] s. 5.3.2

[9] s. 4.5.2

[10] s. 5.3.2.1

plementierung“ des zuvor eingereichten PGML-Entwurfs zu betrachten, [Wals98] führt selbst bei eingefleischten Gläubigern des Software-Konzerns zu ungläubigen Kommentaren:

> *We suspect, much as the Viet Cong used to ‚complement' the U.S. Marines.* [Khar98]

Da VML insbesondere aufgrund der Bemühungen Microsofts, das noch im Entwurfsstadium befindliche Format in die marktführenden Browser- und Office-Produkte zu integrieren, eine sehr hohe de-facto-Verbreitung genießt und somit zumindest aus rein quantitativer Sicht ein interessantes „Master“-Format zur Realisierung von Multimedia-Präsentationen darstellen könnte, soll der nun folgenden qualitativen, strukturellen Analyse des Formates entsprechend etwas mehr Raum beigemessen werden:

5.3.3.2 Aufbau und Syntax

5.3.3.2.1 Schwer durchschaubares Code-Geflecht

Bei der schließlich im Mai 1998 eingereichten Spezifikation [VML98] sticht als augenfälligstes Merkmal der VML-Architektur insbesondere die sehr enge Verstrickung des Vektor-Formates mit der Seitenbeschreibungssprache HTML hervor: So existiert, ganz im Gegensatz zur Aussage der Format-Entwickler, VML begünstige den „Austausch von Vektordaten zwischen verschiedenen Anwendungen“ [Wals98, Wu98], überhaupt gar keine separate VML-Formatdatei (mit einer zu erwartenden “.VML”-Erweiterung) zur ausschließlichen Beschreibung von Bilddaten – die Polygonalformen treten nur in Verbindung mit der stets benötigten *Host*-Sprache HTML auf. So ist es (wie etwa in PGML[1] und SVG[2] vorgesehen) bedauerlicherweise *nicht* ohne größeren Aufwand [Khar98, KrMa00] möglich, mittels VML in eine Webseite integrierte Bilddateien herunterzuladen oder zu manipulieren. Die von den Autoren versprochene „Einfachheit“ [Math89] beim Verwalten auf diese Weise gespeicherter Vektordaten [3] trifft somit nur auf gewiefte Informatiker zu, die sich zur Grafikbearbeitung primär auf textliche Quelleneditierung stützen. Ein gänzlich undokumentieres Feature des Microsoft-basierten VML-Workflows ist überdies die Applikations-übergreifende *Copy-&-Paste*-Funktionalität: So können einzelne VML-Elemente einer Webseite in Microsofts Internet Explorer [4] (übrigens ausschließlich Maus-basiert) ausgewählt und über die Zwischenablage in Microsoft Office-Produkten weiterverarbeitet werden.

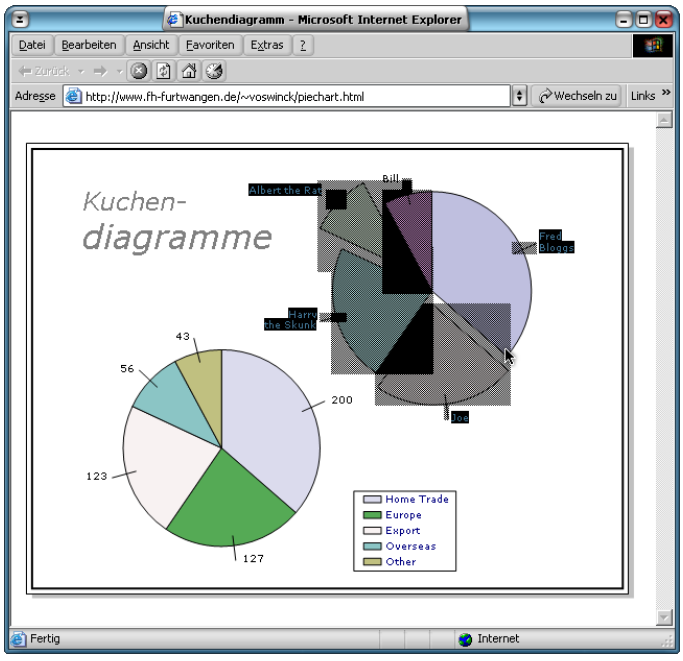


Abb. 5.3.3.1: Objektauswahl im Internet Explorer (r.)

Da der Umkehrweg sowie die Auswahl mehrerer Objekte hingegen nur auf sehr umständliche Art und Weise funktioniert [s. Abb. 5.3.3.1], kann hierzu abschließend festgestellt werden, dass den so euphorisch angekündigten [vgl. Wu98, Wals98] Möglichkeiten des Grafikaustausches aufgrund der benutzerunfreundlichen Umsetzung in der Praxis nur geringe Bedeutung zukommen kann. Die Erstellung einfacher VML-Grafiken von Hand, die aufgrund der XML-Basiertheit des Formates ja bereits mit einem schlichten Texteditor möglich ist, wird für in XML-Syntax ungeübte Anwender darüber hinaus durch die erzwungene *Namespace-*

[1] s. 5.3.2
[2] s. 5.4
[3] “[…] Shapes can be moved within the document and between documents very easily:” [Math98]
[4] Hinweis: Ab Version 5 Beta 2 [GoRi99], laut [NeWi00] sogar bereits ab Version 4 (dies ließ sich jedoch nicht verifizieren)

Verwendung erschwert: Da der VML-Parser ja stets in der Lage sein muss, grafische Vektorelemente (VML-Tags) von den umgebenden HTML-Tags zu unterscheiden, ist die Einführung eines separaten Namensraumes für XML im Kopf-Bereich der Webseite zwingend vorgeschrieben:

```
<head>
      <title>Titel der HTML-Seite</title>
      <xml:namespace prefix="v"/>
      <style>v\:*{behavior="url(#default#VML)"}</style>
</head>
```

Listing 5.3.3.1: Einführung des VML-Namespaces in der HTML-Datei

Alternativ hierzu kann die Definition des XML-Namensraumes natürlich auch als Attribut im *Root*-Element der HTML-Datei (in diesem Falle also `<html>`) erfolgen:

```
<html xmlns:v="urn:schemas-microsoft-com:vml">
```

Listing 5.3.3.2: Einführung des VML-Namensraumes im Root-Element

Das somit allen in der Seite vorkommenden VML-Tags voranzustellende „v"-Präfix erweist sich in der praktischen Anwendung nicht nur als relativ zeitraubend, sondern kann besonders für Anfänger des VML-Authorings eine sehr frustrierende Erfahrung darstellen, da die Darstellung von Vektorbildern mit fehlerhaft formulierten VML-Tags höchst „unbefriedigend" ausfällt. Von einer angeblich „interessanten Syntax" der VML-Spezifikation, die etwa [NeWi00] ausmachen, kann also meiner Ansicht nach an dieser Stelle nicht die Rede sein.

5.3.3.2.2 CSS-Syntax: Stilistische Verwirrung

Eine weitere, etwas verstörend wirkende Eigenschaft stellt überdies die Tatsache dar, dass sämtliche Layout-Eigenschaften nicht direkt als Tag-Attribute der jeweiligen grafischen Elemente realisiert sind, sondern vollständig über das Style-Sheet-Modell [CSS98] abgebildet werden. Dies wäre grundsätzlich zwar eine gute Idee – die Umsetzung dieser Implementierung läuft jedoch der Grundidee der Style Sheets, paradoxerweise gerade durch die sehr strenge Einhaltung der CSS2-Spezifikation, meiner Einschätzung nach völlig zuwider: So werden zwar Positionierungs- und Dimensionsdaten der einzelnen Elemente (wie etwa `style='top:10; left:10; width: 20; height: 30'>` mittels Style Sheets dargestellt, typische stilistische Attribute wie Linienstärken und Fülleigenschaften müssen hingegen, da etwa `fillcolor` oder `stroke`-Attribute im ursprünglich für HTML entworfenen CSS-Standard nicht spezifiziert sind, über direkte Elementeigenschaften der VML-Objekte realisiert werden:

```
<v:rect style="top: 10; left: 30; width: 150; height: 50"
 stroke="true" strokecolor="red" strokeweight="1" fill="true" fillcolor="blue"/>
```

Listing 5.3.3.3: Rechteck-Primitive *in VML mit CSS-Positionierungs und XML-Stilangaben*

Nun sollte sich dies jedoch, wie uns der Grundsatz der Style Sheets lehrt,[1] exakt umgekehrt verhalten: Während, wie [Eise02] bemerkt, Position und Größe eines Grafikelementes dessen *Struktur* ausmachen, bilden Farbgebungs- und sonstige Stileigenschaften den *Präsentations*-Aspekt der jeweiligen Abbildung [vgl. Mage00]. Die Auszeichnung der VML-Elementeigenschaften erscheint daher unter diesem Blickwinkel reichlich inkonsistent. Für komplexere Fülleigenschaften wie beispielsweise Verläufe ist darüber hinaus noch die Einbettung eines separaten `fill`-Objektes als *Tochter*element möglich:

```
<v:rect style="width: 150; height: 50" strokecolor="gray" fillcolor="gray">
```

[1] vgl. [Eise02] p.8

```
     <v:fill color2="white" method="sigma" angle="0" type="gradient"/>
</v:rect>
```

Listing 5.3.3.4: Definition eines linearen Verlaufes mittels `fill`*-Tochterelemen*

Abb. 5.3.3.1: Darstellung des in [Listing 5.3.3.4] *spezifizierten Rechtecks (rechts)*

Auf die einzelnen Variablen des Elements soll hier der Kürze halber nicht eingegangen werden [s. hierzu Micr98b, Reid00], da [Listing 5.3.3.4] lediglich der Illustration *child*-basierter Elementeigenschaften dient. Fest gehalten werden soll an dieser Stelle lediglich die Problematik der recht inkonsequenten Attributvergabe, da die Objekteigenschaften, wie bereits ausgeführt, auf Basis scheinbar recht uneinsichtlicher Kriterien[1] sowohl mittels CSS-Angaben, direkten XML-Attributen sowie zusätzlichen Tochterelementen [s. Listings 5.3.3.3 sowie 5.3.3.4] zu implementieren sind.

5.3.3.2.3 Grafik-Primitives: Linien und Pfade

Neben dem in [Listing 5.3.3.3] angegebenen Rechteck-Element werden auch in VML, wie bereits in den Spezifikationen der XML-basierten „Konkurrenz" (im Besonderen PGML[2]) enthalten, weitere *Primitives* [Safe01:1081] zum einfachen Erstellen grafischer Grundformen zur Verfügung gestellt, etwa `line`, `curve`, `arc`, `oval` etc. [vgl. VML98]. Das darüber hinaus komplexere Formen ermöglichende `path`-Element nimmt hierbei eine Sonderstellung ein, da es sowohl als diskretes Element als auch in Attributform[3] in Erscheinung treten kann.

Die `path`-Syntax weicht an dieser Stelle allerdings erheblich von der in [PGML98] vorgeschlagenen Möglichkeit der Pfad-Definition über als XML-Tochterelemente realisierte Knotenpunkte [s. Listing 5.3.2.2] ab: Wie in [VML98][4] spezifiziert, werden hier einzelne Pfadknoten in einem sich aus Positionsdaten und Befehlskürzeln [5] zusammensetzenden Array abgebildet,[6] wobei letzterer zugleich das Hauptattribut des Pfad-Elements darstellt.

```
<v:shape style="top: 0; left: 0; width: 200; height: 200"
   stroke="true" strokecolor="gray" strokeweight="1" fill="true"
```

Listing 5.3.3.5: Definition eines Sterns über das `path`*-Element*

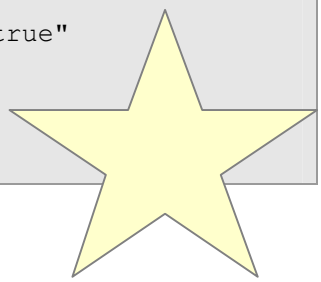


Abb. 5.3.3.2: Darstellung des in [Listing 5.3.3.5] *spezifizierten Sterns (rechts)*

Die hohe Komplexität[7] und Kompaktheit[8] dieser Notationsweise bedingt zeitgleich einen Verlust der „Lesbarkeit" des Codes, d.h. Struktur und vermutliches Aussehen des beschriebenen Pfades lassen sich nicht mehr ohne weiteres aus dem Quelltext ersehen. Dies erscheint jedoch unter Berücksichtigung einer damit einhergehenden signifikanten Speicherplatzeinsparung [vgl. Tard01] aufgrund der Tatsache, dass die VML-

[1] die Attributverleihung erfolgt strikt im Rahmen (jedoch, wie ausgeführt, nicht im Sinne) der CSS2-Spezifikation [CSS98]
[2] s. 5.3.2
[3] Hinweis: Im Bezug auf das ShapeType-Element, wie weiter unten beschrieben.
[4] vgl. [VML98] Textanker: #_Toc416858391
[5] etwa m (= moveto: Pfad-Startkoordinate) oder l (=lineto: linearer Folgeknoten); vgl. [ibid]
[6] Hinweis: Komma-Separierte Zeichenkette, die jedoch als Array geparst wird.
[7] vgl. [ibid]
[8] Die „Commands" (moveto, lineto, close, nofill…) werden zugunsten von Speicherplatz-Effizienz statt menschen-lesbar lediglich in abgekürzter Form (m, l, c, nf…) kodiert.

Syntax ohnehin deutlich weniger im Hinblick „Human Readability“ als etwa PGML[1] entwickelt wurde, als noch hinnehmbares Manko der VML-Pfadnotation.

5.3.3.2.4 Object Cloning: Shape und Shapetype

Aus [Listing 5.3.3.5] ist zugleich ein weiterer Grundbestandteil [vgl. HoDu00] der VML-Spezifikation ersichtlich: Das generische `shape`-Element, welches, um sich an dieser Stelle eines Bildes aus der Theorie der Objektorientierung zu bedienen, das „Mutter“-Element der enthaltenen Grafik-Elemente bildet. Da dieses „abstrakte“ Objekt per se keine darstellbare Form besitzt, setzt sich sein letztendliches Erscheinungsbild aus diskreten *Children* (d.h. den einzelnen Grundelementen der Form) zusammen, denen es gleichzeitig direkt im `<v:shape>`-Tag spezifizierte Attribute vererbt.

Besitzt ein `shape`-Element hingegen keine darstellbaren Tochter-Elemente, so muss dies nicht notwendigerweise heißen, dass auch keine Figur angezeigt wird: Über die so genannte *Shape Type*-Funktionalität lassen sich einzelne grafische Formen und Figuren in VML vordefinieren und beliebig oft wiederverwenden. Hierbei spezifiziert das `<v:shapetype>`-Element den Prototypen des Symbols, auf das später zurückgegriffen werden soll. Im Gegensatz zum `<v:shape>`-Objekt werden die mittels `<v:shapetype>` formulierten Gebilde jedoch nicht automatisch dargestellt – erst wenn eine Referenzierung der Figur über einen im HTML-Dokument eindeutigen Bezeichner (`id`) des *Shape Type*s erfolgt, wird die „prototypische“ Struktur des `shapetype`-Elements dupliziert und mit in der eigentlichen *Instanz* des Symbols optional anzugebenden Zusatzattributen auf dem Endgerät dargestellt.

Beispielsweise könnte so unser in [Listing 5.3.3.5] formulierter Stern statt als diskretes `<v:shape>`-Element in Form eines `<v:shapetype>`-Prototypen definiert und anschließend, etwa mit verschiedenen Füll-Attributen oder unterschiedlichen Größen versehen, mittels beliebig vieler Objekt-Instanzen einen ganzen Sternen-Himmel bilden:

```
<v:shapetype id="stern" ...
   path="m8,65l72,65,92,11,112,65,174,65,122,100,142,155,92,121,42,155,60,100xe"/>
</v:shapetype>
```

Listing 5.3.3.6: Definition eines Stern-*Prototypen als* Shape Type

```
<v:shape type="stern" style="left:0;  width:10; height:10" fillcolor="blue"/>
<v:shape type="stern" style="left:20; width:20; height:20" fillcolor="green"/>
<v:shape type="stern" style="left:40; width:30; height:30" fillcolor="yellow"/>
```

Listing 5.3.3.7: Instanzen des Stern - *Shape Types* [aus Listing 5.3.3.6]

Abb. 5.3.3.3: Darstellung der in [Listing 5.3.3.6] *spezifizierten Instanzen (r.)*

Unter Verwendung ebendieser *Shape Type*-Funktionalität haben, dies nur als Randbemerkung, Entwickler der Britischen Causeway Graphical Systems Ltd. einen interaktiven und den astrologischen Gegebenheiten exakt entsprechenden Sternenhimmel vollständig in VML realisiert: http://starmap.causeway.co.uk[2]

In [Listing 5.3.3.6] wird im Übrigen eine weitere, etwas stutzig machende Eigenschaft VMLs im Zusammenhang mit dem `<v:shapetype>`-Element deutlich: Anstatt, wie etwa im Nachfolgestandard SVG[3] bei der `<symbol>`-Definition realisiert [vgl. Eise02:53ff], die Pfadinformationen der Prototypen einfach genau entsprechend dem `<v:shape>`-Element als `path`-Tochterknoten einzuhängen, muss dies über ein separates `path`- bzw. `v`-Attribut direkt im `<v:shapetype>`-Tag spezifiziert werden. Dieser Umstand lässt die VML-

[1] s. 5.3.2
[2] URL am 25.2.03 verifiziert.
[3] s. 5.4

Syntax nicht nur reichlich inkonsistent erscheinen, er bedingt zudem die Tatsache, dass in VML, zumindest bei ausschließlicher Verwendung der *Shape Type*-Funktionalität, nur relativ einfache Grafiken, die sich durch einen einzigen Pfad realisieren lassen, für eine Wiederverwendung in Frage kommen. Nennenswerte Bandbreiteneinsparungen sind, auch durch die Tatsache, dass das ebenfalls im VML-Entwurf auftauchende `<v:group>`-Element, mit dem sich mehrere Grafikobjekte logisch zusammenfassen lassen, eine derartige Funktionalität nicht erlaubt, somit von der derzeitigen Spezifikation nicht zu erwarten.

5.3.3.2.5 Intelligente Tags

Zwei hingegen äußerst faszinierende Eigenschaften, die sowohl `shape` als auch `shapetype`-Elementen als optionale *child elements* zur Verfügung stehen, sind `formulas` und `handles`.[1] Beide Objekte ermöglichen bereits ohne Einsatz von JavaScript und DOM-Funktionen eine Variablen-orientierte Dynamisierung der einzelnen VML-Objekteigenschaften: So lassen sich via `formulas` verschiedene Element-Attribute statt durch fixe Werte mit variablen Eigenschaften belegen, die innerhalb der `<v:formulas>`-Tags zur Laufzeit berechnet werden. *Handles* erweitern diese Funktionalität noch um zusätzliche Interaktivitäts- und UI-Features. Da beide Elemente jedoch bereits aufgrund ihrer sehr komplexen syntaktischen Formulierung das manuelle Authoring extrem erschweren, soll an dieser Stelle von Details hinsichtlich einer möglichen Implementierung von `formulas` bzw. `handles`-Funktionalität, schon allein, da die allgemein verfügbaren VML-Export-Tools diese Funktionalität bereits vollautomatisch abdecken, abgesehen werden.

5.3.3.2.6 Typografie

Im Hinblick auf die textliche Gestaltung bietet VML derzeit zweierlei Alternativen: Zunächst einmal erlaubt – und fordert – die Spezifikation aufgrund der engen Verflechtung mit dem Seitenformat HTML eine verschachtelte Anwendung der HTML-Syntax für längere Fliesstexte. So lässt sich unter Verwendung des VML-eigenen `textbox`-Tags beliebiger HTML-Code einfügen, auf den sich wiederum die bereits in [3.2.3] beschriebenen Gestaltungsmöglichkeiten [2] anwenden lassen:

```
<v:rect style="top=400; left=1000; width=1000; height=400">
  <v:textbox >
    <p class="styleSheetClass"><b>HTML-</b>Fliess-<br>Text</p>
  </v:textbox>
</v:rect>
```

Listing 5.3.3.8: Einbettung von HTML-Code in ein VML-`textbox`-Element.

Trotz der erheblichen Nachteile [vgl. Smit01], die insbesondere bei flexibler Textpositionierung an dieser Stelle deutlich werden,[3] besitzt die Anwendung des HTML-Texthandlings bei längeren Texten sogar gegenüber dem nachfolgend beschriebenen SVG,[4] welches trotz interessanter typografischer Möglichkeiten bei größeren Textmengen enttäuscht [vgl. Cagl02],[5] mitunter einige Vorteile. Allerdings werden speziell bei anspruchsvollen typografischen Gestaltungswünschen, die VML bzw. das an dieser Stelle zum Tragen kommende HTML nicht vollständig zu erfüllen vermag, die mangelhaften schriftgestalterischen Möglichkeiten von VML offensichtlich [KrMa00].[6] Aufgrund dessen bietet sich an dieser Stelle ein zweiter, primär auf Microsofts WordArt [vgl. Edri00] aufbauender Ansatz der Textgestaltung in VML an:[7] Mithilfe des eigens

[1] vgl. [VML98] Textanker #_Toc416858392 bzw. …93

[2] Hiermit sind beispielsweise HTML-Font-Tags, Style Sheets [CSS98] der OpenType-Embedding gemeint.

[3] "[VML] does have some strange blind-spots, particularly if you want detailed control over text placement... Because Vml relies completely on the CSS syntax for text placement, it has no way of rotating captions, so writing your Y-axis caption vertically alongside the axis is a very tricky exercise" [Smit01]

[4] s. 5.4

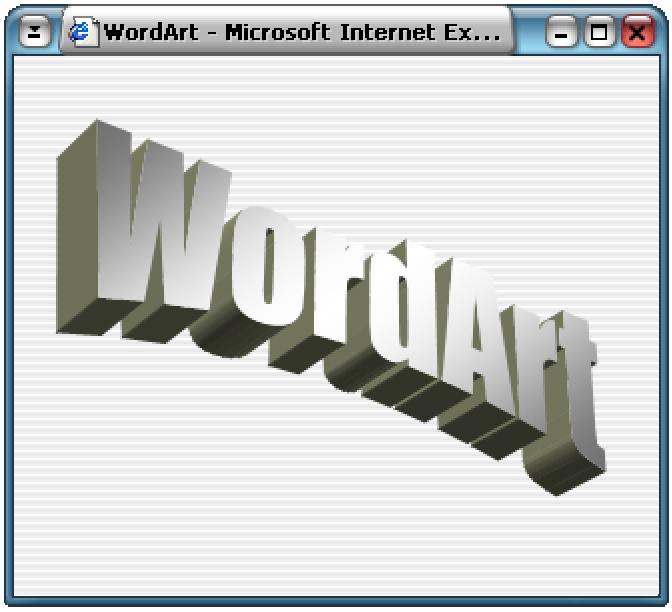
[5] "Text in SVG is something of a Disappointment" [Cagl02] p.241

[6] „VML selbst ermöglicht nur ansatzweise die grafische Manipulation von Texten" [KrMa00]

[7] „Eine Alternative ist, hierfür das in allen Office-Anwendungen enthaltene WordArt für Texte zu nutzen" [ibid]

hierfür bereitgestellten `textpath`-Elements [1] werden unter Verwendung des für diese Funktion benötigten, internen WordArt-Renderers die zugehörigen Elementdaten (neben insbesondere dem `string`-Attribut des Textpfades sind dies beispielsweise Füllungsgradienten und `<v:shadow>`-Elemente für die Schattenberechnung) zur Laufzeit in VML-Polygone umgewandelt und als solche im Viewer dargestellt.

Abb. 5.3.3.4: WordArt-Funktionen mit VML



Auch wenn hierbei, wie etwa [Smit01] kritisch anmerkt, besonders kleinere Textelemente durch diese Outline-Technik „auseinanderfallen“,[2] so lässt sich auf diesen „Umweg“ auch wieder die aufgrund der HTML-Einbettung ansonsten fehlende [3] Kontrolle über *Anti-Aliasing* der Textdarstellung herstellen.

Obgleich [Rein98b] hierzu bemerkt, eine Kontrolle über Polygon-Rendering und Anti-Aliasing sei durch die VML-Spezifikation [VML98] im Gegensatz zum PGML-Pendant [4] „nicht abgedeckt“,[5] ist seit der neuen VML-Rendering-Engine in Microsofts Internet Explorer 6 [6] der explizite Zugriff auf Aliasing und Rendering-Eigenschaften einzelner VML-Elemente sehr wohl möglich [vgl. Smit02]. Über das `antialias`-Attribut lässt sich eine (per *default* aktivierte) Kantenglättung auf Wunsch somit an- oder ausstellen.[7] Damit ist mit VML[8] eine sehr mächtige Kontrolle über Rendering-Verhalten der darstellenden Engine und somit dem letztendlichen Aussehen selber möglich, die sogar die derzeitigen gleichgearteten Möglichkeiten in SVG und Flash übertrifft, da letztere das manuelle Abschalten des Anti-Aliasing jederzeit ermöglichen.

Aus gestalterischer Sicht lässt sich zudem die bereits eingangs erwähnte, enge Integration VMLs in den umgebenden HTML-Kontext als „Vorteil“ [McLD01] betrachten: Wie etwa in [Reid00] erwähnt, ermöglicht die aus struktureller Sicht eher problematische[9] Verwendung des HTML-eigenen Tabellen-Konstrukts eine im Gegensatz zur Pixelgenauigkeit VMLs deutlich flexiblere Positionierung über prozentuale Größenangaben der Tabellen-Tags. Einzelne VML-Elemente lassen sich durch diese Technik etwa mithilfe des `vAlign`-Attributes der einschließenden `table`-Zellen horizontal und vertikal zentriert ausrichten oder in Bezug mit anderen tabellarischen HTML-Objekten [10] stellen.

Enttäuschend fallen hingegen die erweiterten Multimedia-Eigenschaften von VML selbst aus: So enthält [VML98] in Bezug auf die etwaige Animations-Möglichkeiten lediglich einen lapidaren Hinweis [11] auf die Option, einzelne Vektorobjekte via *Scripting*-Zugriff animieren zu können; auf eine etwaige Implementierung dieser Funktionalität findet sich in der Spezifikation hingegen keinerlei Hinweis. In der Praxis hinge-

[1] vgl. [VML98] #_Toc416858398
[2] “The snag is that the result is drawn as an outline, rather than written as text, so at small sizes it falls apart terribly” [Smit0]
[3] vgl. [Smit02]
[4] s. 5.3.2
[5] “not covered” [Rein98b]
[6] s. hierzu auch „Umsetzung“ im selben Kapitel
[7] mögliche Variablen: true | false.
[8] Hierbei ist eigentlich VML in Verbindung mit den Internet-Explorer-Versionen 6 und später gemeint.
[9] s. 3.2.3.1
[10] Dies ist über die Anwendung sowohl absoluter als auch prozentualer Dimensionsdaten der Tabellen-Elemente möglich. [s. hierzu auch 3.2.3.1]
[11] vgl. [VML98] Text-Anker #h3:introduction.requirements, Pkt. 4

gen liefern beispielsweise [Rich00] sowie dHTML-Entwickler Phil Richards [1] interessante Beispiele eindrucksvoll umgesetzter Animations-Engines, die über aufwendige JavaScript-Codes sogar (wenn auch triviale) 3D-Features aufweisen.

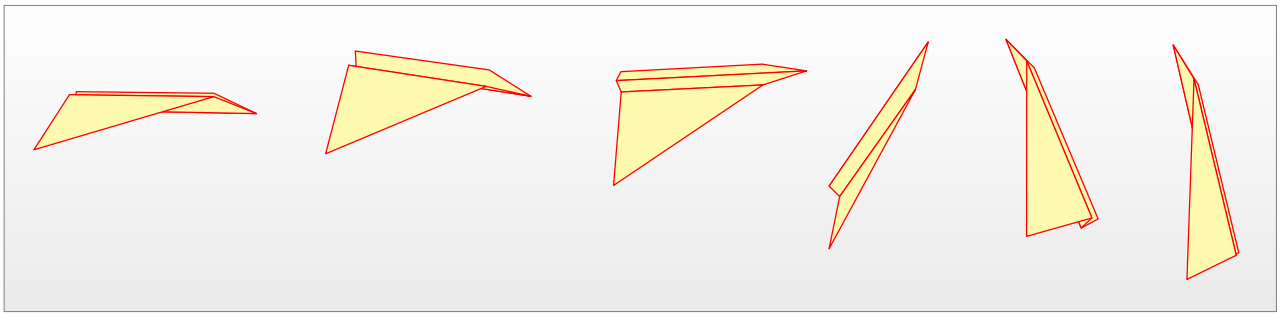


Abb. 5.3.3.5: Animation eines Papierfliegers mittels gescriptetem VML nach Phil Richards [2]

Die etwas umständlichen Erklärungen der VML-Entwickler zu diesem auffälligen Versäumnis [vgl. Wu98, Math98] legen an dieser Stelle den Schluss nahe, dass auf Animationseigenschaften vor allem aufgrund „politisch" motivierter Beweggründe verzichtet wurde: So lässt etwa [Micr98a] vermuten, dass insbesondere auf Druck des VML-Gruppenmitglieds Macromedia eine unter Umständen bereits spezifizierte Animationsimplementierung in VML wohl unter Rücksicht auf Macromedias primär animationsbasiertes Flash-Format [s.4.5] wieder entfernt wurde. Der ausdrückliche Hinweis [Wu98] auf die Verwendung von VML-Komponenten speziell „in Verbindung mit Animations-Formaten"[3] weist außerdem auf die strikte Auffassung der VML-Spezifikation als primär *statisches* Grafikformat hin. Dies erhöht freilich erneut die Komplexität der in VML integrierten Anwendungsschichten: Neben den bereits angeführten Komponenten HTML, CSS und JavaScript, sowie unter Verwendung des Office-Workflows u.U. darüber hinaus noch weiterer proprietärer Tag-Elemente,[4] kommt bei einer Animations-Implementierung im Sinne der Entwickler [vgl. Wu98] das hierfür vorgesehene HTML+TIME-Modell zum tragen [s.3.5.2]. Authoring und Integration insbesondere der VML-, HTML- und Animationskomponenten fällt aufgrund der verschachtelten Notation der verschiedenen Elemente in ein und derselben Datei allerdings sehr umständlich und komplex aus.

5.3.3.3 Umsetzung

Nach der formalen Anerkennung der VML-Mitgliedervorlage [VML98] durch das W3C im Mai 1998 und verschiedener euphorischer Kommentare bezüglich des eingereichten Formates [Wals98] wurde es aufgrund der bereits in [5.3.2.3] beschriebenen Patt-Situation zwischen insbesondere VML und PGML sowie weiterer, im gleichen Zeitraum eingereichter Vorschläge [HGML,[5] DrawML und Web Schematics][6] im Umfeld des Web-Konsortiums alsbald sehr ruhig um die eingereichte VML-Spezifikation. Tenor des W3C war im Sommer 1998, dem VML-Entwurf die formale Anerkennung als *Recommendation* zu verweigern und einzelne Komponenten der Vektorsprache, zusammen mit dem syntaktisch saubereren PGML-Format zu einem neuen Standard [SVG, s. 5.4] zu verschmelzen. [vgl. Rein98b]

Microsoft hingegen ließt sich, im Gegensatz zu den ebenfalls vor dem W3C gescheiterten PGML-Entwicklern [s. 5.3.2.4] von den bereits anlaufenden Vorarbeiten zu SVG „nicht beirren" [Behm02:52] und verfolgte trotz formaler Mitgliedschaft im SVG-Entwicklerteam weiterhin den „eigenen" VML-Ansatz. So

[1] Phil Richards: The *evolve* Home Page [http://www.p-richards.demon.co.uk/vml/vml_main.htm]
[2] s. ebenda: [http://www.p-richards.demon.co.uk/vml/paperdart1.htm]
[3] "VML does not define animation of graphics, but it can be used in conjunction with animation formats" [Wu98]
[4] hiermit sind die mittels Office-Export integrierten „w:" [Word], „p:" [PowerPoint] und „o:" [Office]-Tags und Attribute, die primär dem Re-Import der HTML-Dateien und somit der Rekonstruktion der originären Office-Formate dienen, gemeint.
[5] s. 5.3.1
[6] DrawML und Web Schematics: s. 5.3.4

arbeitete das Unternehmen noch im Sommer 1998 an einer Implementierung eines „Großteils“ [Smit00] des spezifizierten VML-Umfanges. Aufgrund der „Schlankheit“ des ursprünglich als „proof of concept“-Studie [Math98] entwickelten Viewers sowie der unternehmensstrategischen Entscheidung des Software-Riesen, an der „proprietären“ [NeWi00] VML-Spezifikation auch ungeachtet der voranschreitenden W3C-Bemühungen um SVG festzuhalten, wurden grafische VML-Rendering-Funktionalitäten[1] bereits in den Beta-Versionen der im Spätherbst 1998 veröffentlichten, auf ihrem Marktsektor jeweils führenden Microsoft-Produkte Internet Explorer 5 sowie sämtlicher Office97-Komponenten integriert.

Allein die Tatsache, dass der VML-Renderer in Microsofts marktführenden[2] Microsoft Browsers integriert ist und VML-Elemente daher in nahezu allen Microsoft Internet Explorer-Versionen[3] ohne zusätzliches Plug-In dargestellt werden können, verschafft VML einen deutlichen Vorteil gegenüber dem Hauptkonkurrenten PGML: Trotz der Mitarbeit Netscapes an der PGML-Spezifikation im Frühjar 1998 wurde eine bereits angekündigte Integration der PGML-Funktionalität in Netscapes Internet-Browser „Communicator“ nach dem Scheitern des Formates vor dem W3C-Konsortium auf Eis gelegt. VML ist hingegen aufgrund der massiven Verbreitung des Internet Explorers [vgl. Duck01] auf der absoluten Mehrzahl der heutigen Internet-Endgeräte darstellbar und genießt daher sogar noch weitergehende Unterstützung, als dies auch derzeit noch dem eigentlichen de-jure-Standard SVG vergönnt ist.[4]

5.3.3.4 Status

Zumindest laut [GoRi99] wurde das nicht-standardisierte Format daher von der Gemeinde der Internet-Entwickler „rasch angenommen“.[5] Da VML die erste XML-basierte Spezifikation zur Abbildung von Vektorgrafiken im Internet darstellt, die aufgrund der [bereits angesprochenen] „Par-force-Implementierung“ größere Verbreitung erlangen konnte, interessierten sich zunächst freilich primär universitäre Kreise für die textbasierte Funktionalität der Sprache: Im Gegensatz zum bereits weit verbreiteten, jedoch binären Flash-Format ließen sich nun strukturierte, textliche Daten (etwa aus mathematischen Formeln) in der Grafik abbilden und ebenso von Internet-Suchmaschinen indizieren [vgl. Lill96]. Es überrascht daher nicht, dass eine der ersten und zudem bekannteren Veröffentlichungen zum Thema, wie später übrigens auch bei SVG [vgl. NeWi00], im Bereich der computergestützten Kartografie angesiedelt ist [GoRi99].

Der großen Masse jedoch blieb das VML-Format, da im Consumer- wie auch im professionellen Internet-Produktionsbereich aufgrund der ausbleibenden Standardisierung nur wenig über Details und Anwendungen der Spezifikation veröffentlicht wurde, weitgehend unbekannt. Auch die wissenschaftliche Forschung in diesem Bereich verblieb daher, wie [McLD01] zurückhaltend formulieren, auf äußerst „beschränktem“ Niveau.[6] Neben einigen, wenigen prototypischen Webanwendungen, die die Funktionalität der Sprache selber, wie auch eigens in JavaScript hinzuprogrammierten 3D-Features[7] demonstrieren [vgl. Rich00], ist die Anzahl der in VML realisierten Grafiklösungen im Internet bis zum heutigen Tage verschwindend gering. [HaUl02] gibt an dieser Stelle einen Überblick über die interessantesten, bislang realisierten VML-Anwendungen – auch hier bemerkenswerterweise im besonderen Hinblick auf kartografische und GIS-bezogene Lösungen.

Diese „Übersichtlichkeit“ hinsichtlich der tatsächlichen VML-Implementierungen erscheint insbesondere unter Berücksichtigung der doch recht beachtlichen Verbreitung der „eleganten“ [Smit01] VML-

[1] D.h. die darstellende Komponente. Diese ist jedoch nicht in der Lage, die lexikalische Form der VML-Dokumente zu parsen [vgl. Math98]

[2] vgl. [Duck01]

[3] ab Version 5 Beta 2 [GoRi99], laut [NeWi00] sogar bereits ab Version 4 (dies ließ sich jedoch nicht verifizieren)

[4] s. 5.4

[5] “Therefore, the authors quickly adopted VML…” [GoRi99]

[6] “Research into this field has been limited” [McLD01]

[7] vgl. Phil Richards: The *evolve* Home Page [http://www.p-richards.demon.co.uk/vml/vml_main.htm]

Vierwerkomponente des Internet Explorer etwas paradox – demonstriert doch zugleich eine überbordende Fülle „nervzehrender“ Flash-Anwendungen [vgl. Zmoe00] die dem gegenüberstehende Popularität vektorbasierter Internet-Grafiken.

Dies mag meiner Einschätzung nach sicherlich in den mangelhaften bzw. gänzlich fehlenden, professionellen Erstellungs- und Editiermöglichkeiten für das Format begründet liegen: So stellt das einzige, gänzlich dem VML-Format gewidmete Editor-Werkzeug Microsofts „VML Generator“ aus dem Jahre 1998 dar, welcher jedoch aufgrund seiner dilettantischen Benutzeroberfläche, unzureichender Funktionalität und des wohl dadurch bedingten, mangelnden Zuspruchs seit 2001 nicht mehr weiterentwickelt wird.

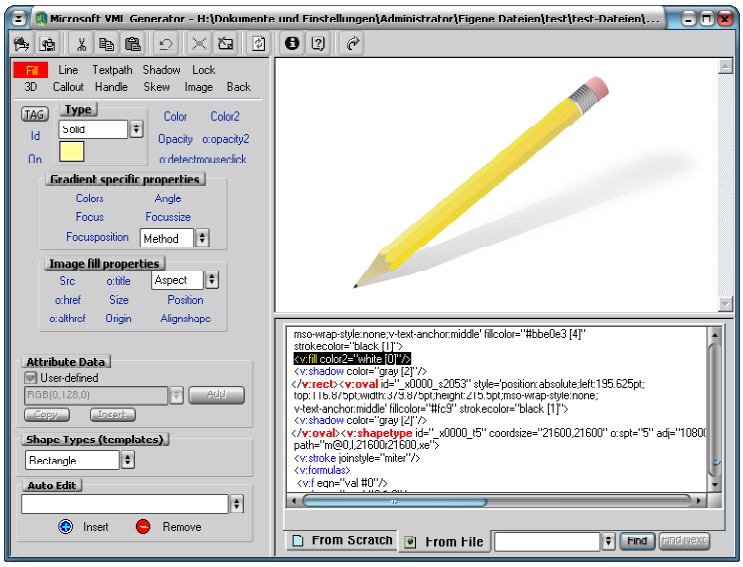


Abb. 5.3.3.6: VML Generator (rechts)

Darüber hinaus kommt VML, wie bereits in [5.3.3.3] besprochen, in sämtlichen Komponenten der Microsoft Office-Reihe[1] als „natives“ Grafikformat [GoRi99] zur Anwendung, speziell im Rahmen der (freilich, wie in [2.3.3] erkannt, reichlich „problematischen“)[2] Internet-Exportfunktion der Programme Word und PowerPoint.[3] Im Zusammenhang mit eben diesen Export-Filtern wird die Nachbearbeitung der VML-Codes jedoch zur schier unlösbaren Aufgabe: Aufgrund der sehr engen Verflechtung der Vektorelemente mit den umgebenen HTML-Komponten, sowie den im Zuge des Office-Exports noch hinzukommenden, ebenfalls nahtlos eingebetteten Office-Tags lässt sich der recht komplizierte, generierte Quellcode nur mit großer Mühe manuell nachbearbeiten. Aufgrund der hohen Komplexität des somit erzeugten Quellcodes wäre auch eine noch hinzukommende, in Ermangelung bislang verfügbarer Software-Tools ebenfalls manuell vorzunehmende Animierung bzw. Scripting der generierten Elemente mit einem besonders hinsichtlich der bestehenden Alternative „Flash“ in keinem Verhältnis stehenden Zeitaufwand verbunden, zumal die auch die derzeit verfügbare Dokumentation [VML98, HoDu00], im Gegensatz zur Aussage von [GoRi99][4] noch sehr spärlich ausfällt.

5.3.3.5 Fazit

Trotz all jener bislang beschriebenen Mängel[5] des Formates, und derer gibt es reichlich, gilt es doch, den herausstehenden Vorzug VMLs, namentlich die hohe client-seitige Verbreitung des Viewers, nicht aus den Augen zu verlieren. So führt etwa VML-Entwickler Adam Smith an, der Nachfolgestandard SVG sei zwar „erheblich besser, sauberer und vollständiger“, aber eben noch lange von der mit VML bereits erreichbaren Verbreitung entfernt:

> *Why wait for SVG when you can do VML today?* [Smit01]

---

[1] Die Entwickler selber geben hier alle Office Version seit Off. 97 als VML-unterstützend an [vgl. Math98], andere Quellen nennen hingegen erst die darrauffolgende Version 9 [Khar98] bzw. Office 2000 [KrMa00, Reid00]

[2] vgl. [Will02]

[3] s. 2.3

[4] “[…] It was from the start rather well documented…” [GoRi99]

[5] s. 5.3.3.2, 5.3.3.4

Diese verbreitungstechnische „Lücke“ lässt sich bereits jetzt auf recht einfache Art und Weise durch eine einstweilige, automatische Konvertierung der beiden XML-Ausprägungen stopfen: Da sich sowohl XML als auch SVG doch in gewisser Weise syntaktisch durchaus ähneln und beide zudem der XML-Konvention vollständig entsprechen, führt bereits ein recht einfach gestricktes XSLT-Konvertierungs-Skript eine qualitativ ausreichende Überführung der in VML abbildbaren SVG-Elemente in VML-Syntax durch. Entsprechend zu berücksichtigende Konvertierungs-Kriterien wurden überdies bereits zu einem ähnlichen, in Perl verfassten Skript öffentlich dokumentiert [Reid99].

Eine vollständige Abbildung ist hingegen lediglich in der umgekehrten Richtung möglich – bei der oben angeführten Konversion geht bei komplexeren Gebilden, bei denen etwa Filterfunktionen, Transparenz, Schrifteinbettung oder Animation zur Anwendung kommen, stets sichtbare Information verloren. Lösbar wären derartige Problematiken zwar durch entsprechende „Work-Arounds“[1] – die Zielgrafik büßt allerdings, wie bereits bei einem gleichartigen, wenn auch komplexeren Konvertierungsverfahren nach Flash der Fall [s.4.5.4], stets an Bildqualität und Speichereffizienz ein. Daher erscheint auch unter diesem Blickpunkt eine genauere Betrachtung des semantisch und syntaktisch überzeugenderen[2] SVG-Standards im hierauffolgenden Kapitel weitaus interessanter als derartige, umständliche (und überdies verlusftbehaftete) „Rückführungen“.

Schlussendlich haben die „Vorarbeiten an VML“ allerdings nicht, wie [NeWi00] ausführen, nicht nur „zur Entwicklung von SVG maßgeblich beigetragen“, sondern bilden, leider aufgrund des paradoxen Microsoft-Widerstands gegen den zum Teil selber mitverfassten SVG-Standard, bis auf weiteres überdies eine nicht zu vernachlässigende Alternative zum SVG-Format, denn:

> *Auf jeden Fall kann man davon ausgehen, dass VML Bestandteil der Microsoft-Welt bleiben wird.*
> [KrMa00]

5.3.4 Schematische Ansätze: WebSchematics und DrawML

5.3.4.1 Relevanz

Obgleich im Rahmen dieser Diplomarbeit hauptsächlich Ansätze zur diskreten, visuellen Darstellung speziell Präsentationsbezogener Grafiken von Interesse sind, sei an dieser Stelle noch ein bislang gänzlich unbehandelter Aspekt präsentationsbegleitenden Bildmaterials erwähnt: die Modellierung *schematischer* Grafiken, Diagramme und Schaubilder. Diesen liegt zwar im Vergleich zu den bislang angesprochenen, primär auf optische Ästhetik abzielenden Grafikkonzepten eine weit abstraktere Herangehensweise zugrunde, dennoch stellen sie speziell im Bereich wissenschaftlicher Vorträge, Dokumentationen und *Online Lectures* einen integralen Bestandteil der visuellen Präsentation dar und besitzen somit auch für multimediale, internetbasierte Anwendungen eine gewisse Relevanz.

Speziell im Kontext der XML-basierten Internet-Vektorgrafiken, die unter diesem Blickwinkel bereits angesprochen wurden [s. 5.3.1-3] wird der bei der Formulierung schematischer Grafiken deutlich höhere Abstraktionsgrad deutlich: Wurden im Rahmen dieses Kapitels bislang die relativ schlichte, sehr diskrete Polygon-Sprache HGML [s. 5.3.1], das immer noch rein grafisch orientierte XML-Pendant des PostScript-Formates PGML [5.3.2] und schließlich die bereits komplexere Berechnungen ermöglichende Vektor-Sprache VML [5.3.3] betrachtet, so konzentrieren sich die im Folgenden behandelten Ansätze weniger auf die grafische Präzision Webbasierter Vektorgrafik, sondern vielmehr auf deren relationale, syntaktische Aussagekraft, sowie logische und saubere Strukturierung. Im Vordergrund der Überlegungen steht daher nicht die Problematik der grafischen Funktionalität eines „Master-Formates“, sondern vielmehr der Wunsch, ein

[1] z.B. Schatten-Emulator, Schriften-Vektorisierer etc.
[2] vgl. KrMa00, Smit01

„sauberes Modell“ [vgl. WSML98] zur inhaltlich korrekten Formulierung komplexer Daten, Wechselbeziehungen sowie logischer Beschränkungen (sog. „*Constraints*“) zu erhalten.[1]

5.3.4.2 Parallelen zur XML-Konkurrenz

Dennoch wird bei Betrachtung der veröffentlichten Herangehensweisen an die Thematik deren enge Verknüpfung mit den zeitgleich erarbeiteten, diskreten Vektorgrafik-Sprachen PGML und VML mehr als deutlich. So werden die aufgrund der beiderseitigen Anlehnung and die zum Veröffentlichungszeitpunkt beider Ansätze heiß diskutierte[2] XML-Konvention wenig verwunderlichen, offensichtlichen Analogien nicht nur semantisch, sondern interessanterweise auch bezüglich struktureller Parallelen hinsichtlich der Objekt-Definition deutlich. Auch das Timing der Veröffentlichungen selber steht, wie bereits in [5.3.1-3] erkennbar, in enger Wechselwirkung mit der parallelen W3C-Diskussion um PGML und VML.

So kann bereits der erste, öffentlich beim World Wide Web-Konsortium eingereichte Diskussionsvorschlag [WSML98] ebenso wie zuvor die SWF- und VML-Veröffentlichungen [Macr98, VML98] als „hastige“, direkte Antwort auf den bereits im Mai eingereichten PGML-Entwurf [PGML98] gesehen werden. Es erstaunt daher nicht, dass sich dieses durch Wissenschaftler des britischen Rutherford Appleton Laboratory erarbeitete Strategiepapier mit etwaigen Details noch sehr zurückhält. Die Autoren[3] siedeln ihre „Web Schematics“ jedoch auf deutlich höherer Abstraktionsebene als den PostScript nahezu vollständig entsprechenden, rein visuellen PGML-Vorschlag an und machen zugleich deutlich, dass der Fokus ihrer Arbeit im Gegensatz zu den optisch präzisen PGML- und VML-Formaten auf einer vor allem *logisch* sauberen Definition schematischer Diagramme beruht.

5.3.4.3 Web-Schematics: Viel Theorie, wenig Konkretes

Aufgrund der wohl durch die frühe Einreichung der grafischen „Konkurrenz“[4] verursachte Hektik bei der Erstellung ihrer schematisch orientierten W3C-Note stellen die RAL-Forscher[5] David Duce und Robert Hopgood ihren eher zögerlichen Vorschlägen zur Konkreten Definition einer möglichen Markup-Syntax zunächst eine weit ausführlichere, generelle Diskussion bislang verfügbarer Ansätze dieses Arbeitsgebiets voran [vgl. WSML98]. So bilden etwa die bereits aus der Unix-Welt bekannten Anwendungen *Thot* [QuVa94], $L^AT_EX\ e2$ [vgl. GRM97] sowie insbesondere die *PIC*-Sprache des Dokumentensystems *troff* [Kern84] die Grundlage[6] des von Duce und Hopgood verfolgten Ansatzes, schematische Schaubilder nach dem *Markup*-Modell Internet-fähig zu machen. Obgleich die in [WSML98] veröffentlichte, rudimentäre Dokumenttypdefinition[7] Formulierungen sowohl absolut diskreter Grafikelemente wie auch in der Form unbestimmter, erst durch eine *Rendering-Engine* berechnete Objekte enthält, lässt die Spezifikation eine klare Aussage hinsichtlich deren Priorität schmerzlich vermissen.

So macht eine längere Diskussion hinsichtlich der Repräsentation einzelner, grafischer Elementattribute sowie auf der „abstrakten“ Seite die Einführung dem späteren, separaten MathML-Standard [vgl. Math99] ähnelnder, mathematischer Objekte die Uneinheitlichkeit der in [WSML98] vorgestellten Spezifikation deutlich. Die Abbildungen [5.3.4.1] und [5.3.4.2] veranschaulichen beispielsweise die „innere Zerrissenheit“ des offensichtlich zwischen diskreter Definiton präziser Grafikelemente und rein logischer Objektverknüpfung schwankenden Funktionsumfangs der „Web Schematics“:

---

[1] “The need for a clean model to underpin mark-up of schematic graphics is highlighted and this should be a key issue to be addressed in discussion of this document.” [WSML98]

[2] vgl. [CKR97, MACH97]

[3] David Duce und Bob Hopgood [WSML98,DuHo00, DHH02]

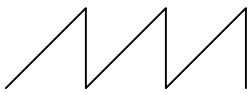
[4] vgl. [PGML98, VML98]

[5] Rutherford Appleton Laboratory, Oxon, England.

[6] vgl. hierzu [WSML98]

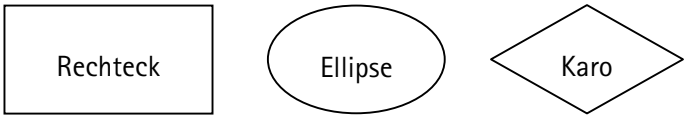
[7] Anm.: Im Gegensatz zu [Loth98] stellt die [WSML98] keine vollständige DTD bereit, sondern lediglich vage Attributbeschreibungen.

```
<POLYLINE coords = "((0,0),(1,1),(1,0),(2,1),(2,0),(3,1),(3,0))">
```



Listing und *Abb. 5.34.1: Sägezahn-Linie in* Web Schematics *und deren Darstellung.*

```
<BOX shape="rect">Rechteck</BOX>
<BOX shape="ellipse">Ellipse</BOX>
<BOX shape="diamond">Karo</BOX>
```



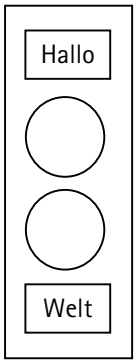
Listing und *Abb. 5.3.4.2: Der Form nach unbestimmte Grafikobjekte in* Web Schematics *und deren Darstellung.*

Darüber hinaus sind sich die Entwickler des Konzeptes selbst über eine aquädate Verwendung CSS-Basierter *Style Sheets* [vgl. CSS96] noch bei Veröffentlichung der Spezifikation augenscheinlich im Unklaren,[1] ebenso wie die im Entwicklungszeitraum bereits veröffentlichte XML-Konvention [XML97] unverständlicherweise zugunsten des veralteten SGML-Standards [SGML86] völlig ignoriert wird. Die offensichtliche Unausgereiftheit des daher implizit als „Diskussionsvorschlag" eingereichten Papiers wird überdies durch die Tatsache verdeutlicht, dass die als Verfasser des W3C-Antrags zeichnenden RAL-Wissenschaftler zum Veröffentlichungszeitpunkt, wohl auch aufgrund der Unvollständigkeit der Markup-Definition selber,[2] keine konkreten Realisierungsansätze oder gar diskussionswürdige Umsetzungen präsentieren können.

5.3.4.4 DrawML: Der überzeugende Nachzügler

Mit einer konkreten Implementierung kann hingegen der noch Ende desselben Jahres veröffentlichte, konkurrierende Ansatz *DrawML* des schwedischen Consulting-Unternehmens Excosoft aufwarten, das seinem überzeugend formulierten W3C-Antrag [Draw98] neben einer vollständigen, XML-konformen DTD sogar bereits fertig realisierte Java-Klassen beistellt. Obgleich der Herangehensweise der „Web Schematics" aus struktureller Sicht durchaus „nicht unähnlich" [vgl. DuHo00],[3] macht die nähere Betrachtung der Spezifikation eine weitaus konsequentere, Diagramm-Orientierte Umsetzung des Grafen-Prinzips deutlich.[4] So können einzelne Objekte sowohl logisch ineinander verschachtelt als auch komfortabel miteinander verknüpft werden, wie [5.3.4.3] veranschaulicht:

```
<drawml>
<shape type="rectangle" arrange="vertical">
             <shape type="rectangle">
                   <lmward>Hallo</lmward>
             </shape>
<shape type="oval"></shape>
<shape type="oval"></shape>
<shape type="rectangle">
<lmward>Welt</lmward>
</shape>
</shape>
</drawml>
```

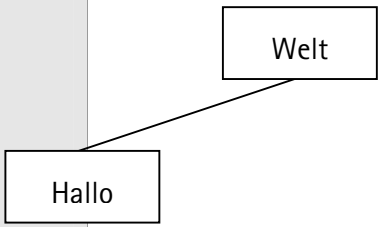


Listing und Abb. 5.34.3: Verrschachtele DrawML-Grafik und ihre Darstellung

```
<drawml>
    <shape type="rectangle" shape-id="d1">




    <line>
        <pos attach-to="d1" attach-point="north"/>
```



[1] "[…] It is not clear to the authors of this proposal what the correct approach should be." [ibid]

[2] "This is a topic that requires discussion within a workshop setting…" [ebenda]

[3] „[…] Similar to Web Schematics" [DuHo00]

[4] vgl. hierzu auch [Lill98b]: "The defining characteristic of DrawML is that the artistic layout of the diagram is secondary to its structural nature…"

```
        <pos attach-to="d2" attach-point="south"/>
    </line>
</drawml>
```

Listing und *Abb. 5.3.4.4: Verknüpfte* DrawML*-Elemente (links) und ihre Darstellung (rechts).*

Neben einer architektonisch durchaus interessanten Semantik und Syntax, die in [Draw98] deutlich wird, kann zudem meiner Ansicht nach auch die Verknüpfung XML-basierter Element-Attribute mit CSS-basierten Stileigenschaften, in [WSML98] und selbst noch im grafisch orientierten Microsoft-Entwurf [VML98] nur unzureichend umgesetzt, endlich auch in logischer Hinsicht überzeugen, da in DrawML eine nun auch syntaktische Trennung etwa stilistischer Farbangaben (via CSS) von den direkten XML-Objekteigenschaften erlaubt.[1]

5.3.4.5 DrawML und SVG

Trotz dieser beeindruckenden Funktionalität konnte sich der DrawML-Entwurf aufgrund des mit Dezember 1998 bereits reichlich späten Einreichungsdatums schlussendlich nicht uneingeschränkt „durchsetzten": Da nach Beilegung des VML/PGML-Konflikts [vgl. 5.3.2.4, 5.3.3.4] bereits im Sommer desselben Jahres die schlichtende SVG-Arbeitsgruppe ihre Arbeit an dem beide Vorschläge zusammenbringenden Standard [vgl. Rein98b] aufgenommen hatte, kündigte sich bereits im Herbst 1998 eine inhaltliche Auftrennung der Entwicklungsbemühungen an: So sollten diskrete, grafische „low-level"-Objekte zunächst die mit SVG0 bezeichnete, „schlichte" Grundlage des geplanten Formates bilden, während die so genannte SVG1-Arbeitsgruppe mit der Definition einer schematischen, sowie logisch abstrakteren Semantik betraut wurde.[2] Schon bald kristallisierten sich im Zuge dieser inhaltlichen Separation erste Grundzüge der verwendeten Komponenten heraus: Während klar war, dass durch PGML [vgl. PGML98] bereitgestellte Grafik-Objekte die für SVG0 erforderlichen *Primitive*-Anforderungen optimal erfüllten, sollten für die höhere Abstraktionsschicht des SVG1-Bereichs Konzepte sowohl Aspekte der in Microsofts VML-Spezifikation enthaltenen `forms`-Logik [vgl. 5.3.3.2] Anwendung finden [WSML98:11], als auch einzelne Ideen der „Web Schematics" aufgegriffen werden [vgl. Lill98a].

5.3.4.6 Enttäuschte Standardisierungsbemühungen

Obgleich Excosoft-Chef Jan Christian Herlitz im Rahmen der XML-Europakonferenz Anfang 1999 in Granada noch einen verzweifelten Versuch unternahm, das W3C-Konsortium von einer Integration des DrawML-Entwurfs in die SVG1-Abstraktionsschicht zu überzeugen [Herl99], fanden die zweifellos interessanten Ansätze DrawMLs wohl aufgrund des späten Veröffentlichungsdatums der Spezifikation [Draw98] weitaus weniger Anklang als die bereits erwähnten Ansätze der VML- und *WebSchematics*-Entwickler [vgl. Lill98b], und sind daher trotz der Mitwirkung Herliz' am späteren SVG-Standard [SVG01] nur ansatzweise in die so genannten „high-level"-Komponenten der SVG-Spezifikation eingeflossen. Auch der weiteren Forderung Herlitz' nach „permanenter Auftrennung" des SVG-Komplexes[3] kam die W3C-Arbeitsgruppe, die eine erneute Fragmentierung offensichtlich zu verhindern suchte, zu Lasten der in der 1. Version des Standards nur rudimentär umgesetzten, logisch-schematischen Funktionalität, nicht nach.

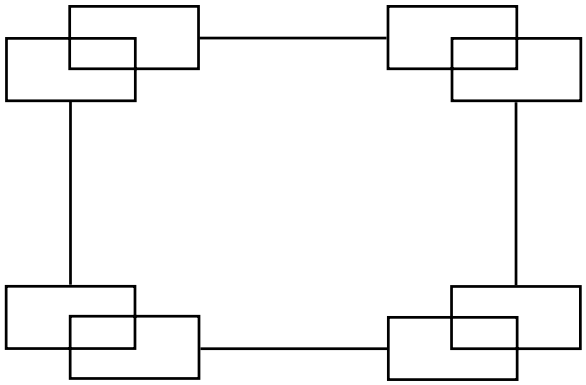
Die Enttäuschung auch der „Web Schematics"-Initiatoren hinsichtlich der augenscheinlichen SVG-Konzentration auf primär grafische Funktionalität [vgl.DuHo00] wird indes durch die fortschreitende Entwicklungsarbeit der RAL-Wissenschaftler an einer geplanten, eigenen WSML-Implementierung deutlich. In Ermangelung „wirklicher", nativ *WebSchematics* unterstützender Darstellungssysteme greifen die Forscher schließlich auf das freilich seitens des W3C ebenfalls nicht weiterverfolgte, jedoch aufgrund der Micro-

[1] "A color is specified as in the CSS standard…" [Draw98]
[2] vgl. [WSML98] p.11
[3] "[…] I am suggesting SVG0 and SVG1 to be permanently split…" [Herl99]

soft-Implementierung weithin verfügbare VML-Modell [s. 5.3.3] zurück: So stellen David Duce und Robert Hopgood schließlich noch 2000 einen vollständig in JavaScript realisierte Übersetzungs-Engine vor, die die mittlerweile weiterentwickelte, abstrakte WSML-Logik in diskrete VML-Vektorobjekte überführt.[1]

```
<pic>
<up>                <right>
      <box/>              <box/>
      <line/>             <line/>
      <box/>              <box/>
</up>               </right>
<left>              <down>
      <box/>              <box/>
      <line/>             <line/>
      <box/>              <box/>
</left>             </down>

</pic>
```



Listing und Abb. 5.3.4.5: Schematischer WSML-Code sowie die dazugehörige Darstellung in VML.

Überdies gelang es Duce und Hopgood im selben Jahr, einen Großteil der strukturellen Ansätze ihres *Web-Schematics*-Entwurfs in einen weiteren, ebenfalls XML-basierten Ansatz zur Formulierung schematischer Diagramme einfließen zu lassen. Dieses auf der Graphen-Theorie aufbauende Konzept [Mark01] erhielt aufgrund seines insbesondere akademischen Hintergrunds[2] seitens des Web-Konsortiums derart weitgehende Unterstützung, dass bereits der Prototyp der W3C-eigenen Browser-Konzeptstudie *Amaya* über einen funktionsfähigen GraphML-renderer verfügt [vgl. DuHo00:6].

5.3.4.7 Fazit

Abschließend bleibt jedoch zu bemerken, dass die Ansätze schematischer Elemente im Rahmen der SVG-Entwicklungsbemühungen unter dem Aspekt tatsächlicher Implementierugen noch auf einem relativ rudimentären Niveau verblieben sind. Vollständige und überzeugende Konzepte existieren zwar bereits [Draw98, Mark01], sind jedoch noch überwiegend auf recht provisorisch erscheinende Converter-Tools[3] angewiesen. Insbesondere die Transformation XML-basierter Markup-Sprachen zur Abbildung relationaler Diagramme[4] mittels der durch das im XML-Framework enthaltenen Übersetzungs- und Style Sheet-Anwendung XSL bzw. XSLT [s. 5.1] in die primär grafikorientierten, jedoch ebenfalls XML-konformen Vektorsprachen VML und SVG erscheint auf diesem Hintergrund interessant. Ebendiese durch XSL bereitgestellte Transformations-Funktionalität erscheint den Initiatoren der Schematics-Initiative jedoch offensichtlich „nicht mächtig" und flexibel genug [vgl. DuHo00:27], sodass eine wirkliche Integration bzw. darüber hinausgehende eigenständige Anwendung schematischer Graphen noch Bestandteil voranschreitender[5] Entwicklungsarbeiten ist.

Im Rahmen dieser Diplomarbeit kommt dem soeben behandelten, schematischen Ansatz aufgrund des sehr hohen Abstraktionsgrades und seines geringen Bezugs zur eigentlichen Grafikdarstellung, geschweige denn möglicher Multimedia-Funktionalität, daher ohnehin nur begrenzte Bedeutung zu – lediglich unter dem Aspekt eines „high-level"- bzw. minimalistischen Ansatzes, der strukturelle, relationale und inhaltliche Daten mit innerhalb einer deutlich eingeschränkten Syntax abzubilden vermag (etwa in Form von logischen Diagrammen oder Schaubildern), erscheinet diese Herangehensweise auch im Bezug auf multimediale Prä-

[1] vgl. [DuHo00] pp.23, 25
[2] vgl. hierzu [Mark01]
[3] vgl. [DuHo00] pp.23, 25
[4] vgl. [Draw98, Mark01]
[5] "Work in progress…" [DuHo00:23]

sentationen interessant [s. hierzu auch Kap. 6.1]. Wie die vorangegangene Betrachtung der unterschiedlichen Implementierungen deutlich gemacht hat, sind die bereits existierenden Vorschläge [WSML98, Draw98] nicht zu einer darüber hinausgehenden Grafikdarstellung[1] in der Lage und somit primär im Rahmen der Vorarbeiten zur unter diesem Aspekt weitaus interessanteren Vektor-Sprache SVG relevant, auf welche daher im nun folgenden Kapitel etwas detaillierter eingegangen werden soll.

5.4 Der Standard: SVG

Obgleich den soeben ausführlich betrachteten Entwicklung erster, „vorsichtiger" Schritte hin zu einem XML-basierten Web-Standard [s.5.3.1-4] im Gegensatz zu proprietären de-Facto-Standards[2] wie etwa *Flash* [s.4.5] ein quantitativer Erfolg oder gar nennenswerte Akzeptanz trotz existierender Implementierungen[3] zunächst verweht blieb,[4] so stellen diese Ansätze dennoch „äußerst wichtige Vorarbeiten" [vgl. NeWi00] zu einem Internet-Format dar, das als bislang einzige XML-Ausprägung mit Fug und recht als der einzige „echte" [Fibi01:99], offene *Standard* für Vektor-basierte Webgrafik bezeichnet werden kann: Die „Skalierbaren Vektorgrafiken" (*Scalable Vector Graphics*) – kurz: SVG.

Da die Entstehungsgeschichte dieses Standards durch die umfassende Betrachtung seiner „Vorgänger" in den vorangegangenen Abschnitten [s.5.3.1-4] mitunter bereits erwähnt wurde, soll im Folgenden zum besseren Verständnis der heutigen Architektur [s.5.4.2] die Entwicklung des SVG-Formates nur sehr knapp dargestellt werden:

5.4.1 Entwicklungsgeschichte

Bereits im Sommer 1998, als zusehends deutlich wurde, dass aufgrund der W3C-Blockadesituation aus „politischen Gründen" [vgl. Khar98] weder dem PGML-Entwurf des von Adobe angeführten Firmenkonsortiums [s.5.3.2], noch dem von Microsoft entwickelten VML-Antrag [s.5.3.3] den Vorzug gegeben werden konnte, veranlasste das Web-Konsortium die Gründung einer Expertenkommission, die einen „Kompromiss" zwischen den beiden Ansätzen herbeiführen sollten. Die Leitung der Arbeitsgruppe, im Übrigen „schon damals SVG genannt",[5] übernahm schließlich Jahres W3C-Mitarbeiter Chris Lilley, der als Chef der „Graphics Activity"-Abteilung bereits in [Lill96] einen fundierten Anforderungskatalog für vektorbasierte Webgrafik vorgelegt hatte [vgl. Lill02]. Als inoffizieller „Alterspräsident" des SVG-Gremiums, das daraufhin bereits im September desselben Jahres seine Arbeit aufnehmen konnte, konnte jedoch PGML-Entwicklungsleiter Jon Ferraiolo maßgeblichen Anteil an der Konzeption des entstehenden Formates nehmen.

Nachdem bereits im Februar 1999 das erste *Working Draft* (Konzept) der Scalable Vector Graphics veröffentlicht werden konnte und noch im selben Jahr gleich drei unterschiedliche SVG-Implementierungen (von IBM,[6] BlackDirt [Dele99] und Adobe[7]) folgten [vgl. Lill02], zogen sich die Beratungen und Diskussionen um den werdenden Standard entgegen des ursprünglichen Zeitplans[8] noch gut zwei Jahre in die Länge,

---

[1] Diese Herangehensweise würde, wie in [6.1] weiter ausgeführt, die „Maximal"-Komponente einer Präsentations-Lösung bilden.

[2] vgl. [NeWi00]

[3] s. hierzu etwa [5.3.3.3]

[4] s. jeweils Abschnitte 4-5 der Kapitel 5.3.1-4

[5] vgl. [Behm02] p.52

[6] „IBMs SVGView auf dem AlphaWorks-Server erhältlich", vgl. hierzu die „Kleine SVG-Geschichte" der *ix* (Heise-Verlag, 2002): http://www.heise.de/ix/raven/Web/xml/svg/tl [14.2.03]

[7] Erste Prototypen des Adobe SVG Viewers, vgl. [Adob99]

[8] „Der neueste offizielle Plan besagt, dass SVG gegen Ende 1999 zu einer Empfehlung des W3C (das bedeutet, fertiggestellt und bestätigt) wird." [ibid]

ehe mit [SVG01] endlich die „finale“ Version des Vektorformates als W3C -Recommendation und somit de-jure-Standard [vgl. NeWi00] verabschiedet werden konnte.

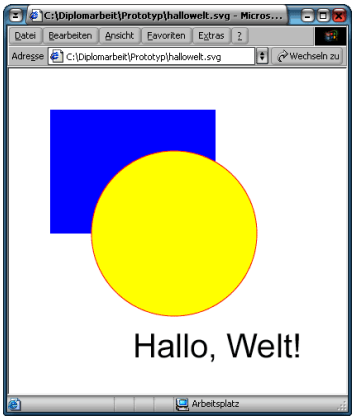
Diese doch erhebliche zeitliche Verzögerung, die zugleich den „kometenhaften Aufstieg“ des Flash-Formates [s.4.5], welches diese Lücke derweil nur zu gut zu füllen wusste [vgl. Gibs00] mit befördert haben mag, versucht Lilley indes mit „erheblichem Diskussionsbedarf“ und „intensiver Aktivität innerhalb der Arbeitsgruppe“[1] zu erklären: In der Tat stellt der letztendlich verabschiedete Standard weitaus mehr dar als nur eine „logische Fusion“ der vorhandenen Entwürfe PGML und VML: Neben weitgehenden Erweiterungen wie Verlaufsmuster, Filter und Bildmaskierung galt es im Rahmen der „konstruktiven Zusammenarbeit“ (Lilley) überdies, grundsätzliche Entwicklungskriterien und Aspekte wie die Trennung von Inhalt und Darstellung mittels CSS,[2] die Einbeziehung einer „intelligenten“ Abstraktionsschicht [s.5.4.3], sowie insbesondere die XML-Abbildung komplexer „Pfade“ zu klären [s.5.3.2], was schließlich erheblich mehr Zeit als ursprünglich anvisiert in Anspruch nahm:

> *Some features, such as the lack of declarative animation syntax in early drafts, was important enough to warrant a delay to the specification. Other feedback, while deemed important, could not be integrated into SVG 1.0 and was deferred for a later version.*
>
> [Watt02:1032]

5.4.2 Aufbau und Syntax

So präsentiert sich die letztlich veröffentlichte Spezifikation als zunächst leicht zugängliches Framework, welches bei genauerer Betrachtung hingegen eine äußerst umfangreiche Auswahl mächtiger und komplexer, grafischer Funktionen bereitstellt. Die Sprache an sich ist indes, nicht zuletzt aufgrund der „Legibility“-Eigenschaften XMLs [vgl. XML98],[3] „relativ leicht zu entschlüsseln“, [Pens00] erscheint jedoch im Gegensatz zum mitunter „wirren“ Code-Geflecht des VML-Formates [s.5.3.2] syntaktisch deutlich „sauberer“ und klarer formuliert: So lassen sich SVG-Dateien, anders als etwa VML oder das „proprietäre“ Flash-Format „auch ohne Grafikprogramm leicht lesen, verstehen und bearbeiten.“ [NSW02:219]

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
 "http://www.w3.org/TR/SVG/DTD/svg10.dtd">
<svg width="500" height="400"
 xmlns="http://www.w3.org/2000/svg">
    <desc>Hallo-Welt-Beispiel</desc>
    <rect x="50" y="50" width="200"
     height="150" style="fill:blue;" />
        <circle cx="200" cy="200" r="100"
         style="fill:yellow;stroke:red;" />
        <text x="150" y="350" style="font- size:40;">
            Hallo, Welt!
        </text>
    </svg>
```



Listing und Abb. 5.4.2.1: Wohlgeformtes Hallo-Welt-Beispiel (links) sowie dessen Darstellung.

Wie das „Hallo-Welt“-Beispiel zeigt, erschließt sich der „Inhalt“ einer SVG-Grafik, soweit man denn des Englischen mächtig ist, quasi direkt bei der Lektüre des Quellcodes. Da, wie bereits in [5.4.1] angesprochen, PGML-Autor Jon Ferraiolo federführend (um genau zu sein, als Herausgeber)[4] an der Formulierung der SVG-Spezifikation beteiligt war, konnte ein Großteil der „sprechenden“ (semantischen) Tag-Namen für die Auszeichnung der einzelnen Elemente sozusagen direkt übernommen werden.[5] Wie ebenso dem Bei-

[1] vgl. [Watt02] *Foreword* (nach XXIII)
[2] s. hierzu auch die entsprechende Diskussion in [5.3.2]
[3] “XML documents should be human-legible and reasonably clear” [XML98]
[4] vgl. hierzu [SVG98,01]
[5] Derzeit im Rahmen von Svg unterstützte Elemente: rect, line, polyline, circle, ellipse, polygon und path. [vgl. Behm02] p.54

spiel in [Listing, Abb.5.4.2.1] zu entnehmen ist, ist auch das SVG zugrunde liegende Grafikmodell durch den so genannten „Painter's Algorithm" [vgl. Watt02:33] gekennzeichnet: Einzelne grafische Objekte werden schlichterdings durch deklarativ darauffolgende Elemente überdeckt – ein zusätzlicher „z-Index", wie noch in Flash oder HTML benötigt, entfällt. Dieses (im Prinzip ja „logische") Prinzip „erhellt" nicht nur die syntaktische Struktur der SVG-Datei, sondern begünstigt überdies dessen Übertragung: Ähnlich wie beim „progressiven Rendering" von HTML-Dateien können SVG-Grafiken so sequentiell „gestreamt" und direkt gezeichnet werden, da später folgende Objekte ihre „Vorgänger" kurzerhand übermalen.

5.4.2.1 Deutlich sauberer als der Vorgänger VML

Im Gegensatz zum direkt an die PostScript-Syntax angelehnten PGML wird im Rahmen der SVG-Spezifikation hingegen eine verstärkte Konzentraion auf Cascading Style-Sheets [CSS98] deutlich: Im Gegensatz zum in dieser Hinsicht recht unlogisch vorgehenden VML-Format [s.5.4.3] vermag SVG an dieser Stelle jedoch inhaltlich zwischen mittels „direkten" XML-Attributen realisierten Positionierungs und Größenangaben (`x,y,height,width`) und im eher „stilistischen" Bereich angesiedelten Stilangaben (`fill,stroke…`) zu unterscheiden und ermöglicht somit eine deutlich sinnvollere Anwendung des Style-Sheet-Prinzips. Zwar sind auch im Rahmen von SVG „redundante" Deklarationen (beispielsweise die Determinierung eines Füllwerts sowohl als direktes Attribut als auch CSS-Angabe) möglich – wie bereits bei HTML werden die „direkten" XML-Attribute in diesem Falle jedoch stets durch deren CSS-Pendants überdeckt, was auch eine konsequente Anwendung umfassender Style-Sheet-Klassen möglich macht [vgl. Behm02:55].

Ähnlich wie in VML und Flash lassen sich nun überdies so genannte „Symbole" definieren, die über ein korrespondierendes `use`-Tag mit entsprechenden Transformationseigenschaften beliebig oft innerhalb einer SVG-Zeichnung zur Anwendung kommen und die Bilddaten über deren Wiederverwendbarkeits-Eigenschaft so überschaubar und kompakt halten kann.[1] Auch über die so genannten „Rendering-Eigenschaften" lassen sich, wie bereits im VML-Vorgänger die grafischen Darstellungseigenschaften (Anti-Aliasing, Qualitäts- oder Geschwindigkeits-optimierung) präzise regeln.

5.4.2.2 Kontrovers: Die verzweigten Pfade des *path*-Tags

„Nicht unumstritten" [vgl. Tard01] erscheint hingegen Umsetzung des zentralen `path`-Elements, welches die Definition beliebiger Linien- und Bézierpfade ermöglicht: Im Gegensatz zum PostScript-Verwandten PGML-Entwurf, der an dieser Stelle die „saubere" Einbindung entsprechender Tochterknoten vorsieht [s.5.2.3], geht SVG an dieser Stelle, ähnlich wie bereits in der VML-Spezifikation „im Grunde völlig unXML-Lisch vor" [Behm02:56]: So enthält stets das obligatorische `d`-Attribut des (im übrigen alleinstehenden)[2] die einzelnen Pfadknoten samt Kontrollpunkte als einzelne, überdies „unkonventionell" strukturierte Zeichenkette, was nicht nur die Lesbarkeit dieser Notation deutlich einschränkt, sondern überdies eine Interpretation der Pfaddaten über die DOM-Schnittstelle deutlich erschwert:[3]

```
<path d="M 50,10 L 10,50 L 90,50 Z" />  =  <path d="M 50,10 l -40,40 l 80,0 z" />
```

Listing 5.4.2.2: Definition eines Dreiecks (▲) in SVG mit jeweils absoluten (li.) und relativen Koordinaten (re.).

[1] vgl. hierzu [ibid] pp.56-57

[2] Anm: Dies heisst, dass der Pfad i.d.R. nicht „umfassend" ist (<tag>…</tag>), sondern lediglich einen einzelnen „Marker" (<tag/>) darstellt.

[3] s. hierzu die entsrpechenden Ausführungen von Jon Ferraiolo: "In recognition that path data into a single attribute makes DOM access difficult…", aus: *XML Developer Mailing List* (Thread „ What is wrong with SVG?") vom 6. März 2000: http://lists.xml.org/archives/xml-dev/200003/msg00199.html [16.2.03]

Obgleich auf die genaue Syntax SVG-basierter Pfaddaten aufgrund der nicht zu unterschätzenden Komplexität an dieser Stelle nicht eingegangen werden soll [s. hierzu Behm02, Watt02:231-246], so muss im Hinblick auf deren Erstellung dennoch festgestellt werden, dass diese praktisch nicht mehr „von Hand" zu verfassen sind: Allein die „Case-Sensitivity" des Daten-Strings, die etwa absolute von relativen Koordinaten unterscheidet, sowie die ungewohnte, durch Leerstellen separierte Schreibweise erschweren an dieser Stelle ein hinreichend komfortables *Editing*.

Diese „fragwürdige" [Tard01] Vorgehensweise begründet sich hingegen schlichterdings in den „erheblichen Platzeinsparungen", die mit der Anwendung dieser Notation einhergeht: So merkt SVG-Herausgeber Jon Ferraiolo in einem Statement[1] an, dass die PGML-Variante, die (wie bereits im Rahmen von [5.2.3] festgestellt) an dieser Stelle deutlich konsequenter vorgeht, „etwa doppelt so viel Speicherplatz" einnehme, was insbesondere in einer bandbreiten-sensiblen Web-Umgebung freilich nur schwer vertretbar erscheint. Aufgrund dessen, so Ferraiolo, habe man sich „nach langer Diskussion" daher für eine Anwendung der VML-Syntax entschlossen, da die PGML-Syntax ohnehin für das Gros der grafischen Grundformen SVGs, der so genannten „Primitives", praktisch unverändert übernommen wurde. Im Gegensatz zur allgemeinen Annahme [vgl. NeWi00], dass „die Vorarbeiten an VML" die SVG-Syntax massgeblich kennzeichnen würde, kann daher abschließend festgestellt werden, dass die Einflüsse des PGML-Ansatzes dennoch deutlich überwiegen.

5.4.2.3 Beeindruckende Funktionalität: Photoshop-ähnliche Filter

Über die reine Funktionalität der Abbildung grafischer *Primitives* hinaus beinhaltet die SVG-Spezifikation jedoch im Gegensatz zu den unter diesem Aspekt eher „bodenständig" wirkenden Vorgängern PGML und VML [vgl. KrMa00][2] deutlich leistungsvollere, beeindruckende Möglichkeiten: So lassen sich mithilfe der `linear`- und `radialGradient`-Tags nicht nur komplexe Transparenz- und Farbverläufe erzeugen sowie dank des komfortablen `pattern`-Elements interessante Füllmuster komponieren – das eigentlich „Sahnehäubchen" des grafischen Funktionsumfangs von SVG stellt unzweifelhaft die Photoshop-ähnliche Filter- und Maskierungsfunktionalität des Formates dar: Auf diese Weise können nun geradezu „überwältigende" Unschärfe-, Licht- und Verzerrungseffekte anhand sehr einfach zu realisierender Filter-Tags realisieret werden, für die zuvor selbst in Photoshop mitunter sündhaft teure Plugins benötigt wurden. Die letztendlich unschlagbare Eigenschaft SVGs besteht hierbei jedoch darin, dass sämtliche Effekte nicht nur beliebig kombinierbar sind, sondern überdies in Echtzeit und optimaler Qualität berechnet werden können: Selbst bei dynamisch veränderlichem Text oder vergrößerter „Zoom"-Darstellung bleibt die Darstellung der SVG-Grafik samt Effektfilters stets optimal.

Abb. 5.4.2.2: Vergrößerung eines schattierten Textelements: Die Darstellungsqualität bleibt stets erhalten

Im konkreten Beispiel (in dieser Form übrigens auch Bestandteil des *Presenter*-Prototypen)[3] wird so etwa über die Definition eines Unschärfeeffektes sowie eines „Offsets" (damit der entsprechende Schatten je um einen Pixel nach rechts unten versetzt dargestellt wird) und der letztendlichen Verknüpfung (*Merge*) dieser Operationen ein einfacher Filter definiert, der sich mittels einfacher URI-Referenz (für den Filter wurde ja zunächst eine eindeutige ID festgelegt) sämtlichen SVG-Grafikelementen zuweisen lässt:

```
<filter id="dropShadow" x="-50%" y="-50%" width="200%" height="200%">
```

[1] vgl. ebenda.
[2] „SVG ist vom Funktionsumfang her viel mächtiger ausgelegt als VML" [KrMa00]
[3] s. hierzu Kap. 6

```
    <feGaussianBlur in="SourceAlpha" stdDeviation="2" result="firstBlur" />
    <feOffset in="firstBlur" dx="1" dy="1" result="offsetBlur" />
    <feMerge>
        <feMergeNode in="offsetBlur" />
        <feMergeNode in="SourceGraphic" />
    </feMerge>
</filter>
```

Listing 5.4.2.3: Definition eines einfachen Schlagschatten-Filters unter Kombination zweier Operationen

Der in [Listing 5.4.2.3] spezifizierte Filter dient etwa als Grundlage für den ebenso in [Abb. 5.4.2.2] veranschaulichten Schlagschatten, welcher u.A. im Rahmen des *Presenters* –Prototypen [s.Kap.6] zum Tragen kommt.

Über diese *parametrische* Definition wird der eigentlichen Rendering-Engine des darstellenden Systems (wie etwa derzeit dem Adobe SVG Viewer Plug-In) zwar eine recht aufwendige Echtzeit-Berechnung der Filterdarstellung aufgebürdet – die Daten der SVG-Quelldatei bleibt hingegen äußerst schlank und lassen sich zudem relativ leicht entziffern [vgl. Pens00]. Auf diese Art und Weise lassen sich nun in komfortabler HTML-Manier ästhetische Filtereffekte realisieren, die zuvor noch nicht einmal im Rahmen komplexer Bildbearbeitungs-Tools wie *Photoshop*[1] [vgl. Blat97] in dieser Form möglich waren – und all dies unter direkter Anwendung nativer SVG-Syntax, die entsprechende, umständliche „Work-Arounds", wie sie etwa bei Erreichung desselben Effektes in Flash nötig sind, hinfällig macht.

5.4.2.4 Text und Typografie

Auch die Textbehandlung kommt bei SVG einiges komfortabler und konsequenter daher als die entsprechende Komponente der Flash-„Konkurrenz" [D'Amo00], die sich insbesondere aus Formatsicht diesbezüglich durchaus nicht unproblematisch verhält:

> *[Flash's] text handling, by being user-agent-neutral, requires the programmer to jump through hoops.*
> [Katz00]

Im Rahmen von SVG hingegen lassen sich Schriftarten, unter anderem mithilfe des mittlerweile in das Batik-Projekt [Croo01] übergegangenen [vgl. Schu02:62, Watt02:663], automatischen TrueType-Konverters *SVGFont* [Schw00], nicht nur relativ komfortabel direkt in den SVG-Code einbinden – das Text-Framework von SVG eröffnet überdies schier unbegrenzte, typografische Möglichkeiten: So lassen sich SVG-Schriftzeichen etwa an beliebigen Pfaden ausrichten und ohne vorherige Umwandlung in Polygone flexibel spationieren, drehen und wenden – die Texteigenschaft (und somit die Durchsuchbarkeit) bleibt stets erhalten. Lediglich die Umsetzung von Fliesstext, so merkt unter anderem Kurt Cagle an, stellt bislang eine „Enttäuschung" dar [Cagl02:241] - dies soll jedoch spätestens mit der 2.0-Version des Grafikstandards behoben werden, „vielleicht sogar bereits im Zuge der Version 1.2" [vgl. Watt02:38], die für Ende diesen Jahres Erwartet wird [vgl. Jack03a].

5.4.2.5 Objektorientiert: Animation mit SMIL

Über die Rolle eines statischen Vektorformates hinaus stellt die SVG-Formatspezifikation jedoch noch weitaus beeindruckendere Funktionalität zur überzeugenden Web-Präsentation bereit: So sind mithilfe einer vom Synchronisations-Standard SMIL [s.5.5] entliehenen *Animations*-Syntax nicht nur Bewegungen einzelner SVG-Elemente nach festgelegten Koordinaten oder entlang eines *motion paths* möglich – auch (nahezu) alle weiteren Eigenschaften eines SVG-Objektes lassen sich im Rahmen des `<animate>`-Tags auf diese Weise animieren: So ist der SVG-Renderer etwa in der Lage, Größe (`animateColor`), Farbe (`animateMotion`) oder auch eine beliebige Transformation (`animateTransform`) linear zu interpolieren – eine einfache

[1] Anm: In Photoshop besteht der zusätzliche Nachteil überdies

Angabe, *welches* Attribut *wie* animiert werden soll, genügt. Im direkten Vergleich mit dem Frame-basierten Flash-Ansatz wirkt diese Herangehensweise daher deutlich „sauberer" – die einzelnen Animationen und ihre Verknüpfungen lassen sich quasi direkt dem SVG-Quellcode entnehmen:

> *Durch die zeitbasierte Animation ist eine wesentlich einfachere Erstellung von SVG-„Filmen" möglich – außerdem sind die Dateien dadurch sehr platzsparend. Der Nachteil ist jedoch die erhöhte Rechenleitung auf der Clientseite.*
>
> [Kunz00]

Obgleich Flash, wie im Rahmen unserer Analyse des SWF-Formates [s.4.5.3.1] besprochen, von seinem Grundprinzip her primär Animationsbasiert ist und die SWF-Dateistruktur daher in erster Linie die sowohl kompakte als performant darstellbare [vgl. Katz00][1] Speicherung *linearer* Animationen ermöglicht, sehen die SVG-Experten Andreas Neumann, Peter Sykora und Andréas Winter [vgl. NeWi00, NSW02] daher insbesondere die Animationskomponente im Rahmen von SVG „deutlich eleganter" gelöst als beim Flash-Pendant:

> *Animationen setzt SVG zusammen mit SMIL etwas eleganter um. SMIL-Filmchen stützen sich auf feste Zeitvorgaben, während Flash auf Frames aufbaut, womit die Wiedergabedauer stark von der Rechenleistung des Clients abhängt. Flash kann zwar die wichtigsten Grafikeigenschaften animieren, SVG gewährt dem Nutzer aber mehr Spielraum, da es auf Wunsch jedes beliebige Attribut zum Leben erweckt – sogar mit wählbarer Beschleunigung.*
>
> [NSW02:220]

5.4.2.6 Do it yourself: Skripten mit ECMAScript

Auch die „Scriptability" des SVG-Formates erweist nach genauerer Betrachtung als dem proprietären Flash-Konkurrenten weitaus überlegen: So sind im Rahmen des Flash-Authoring-Tools zwar weitgehende Programmier- und Scripting-Möglichkeiten mithilfe der Flash-eigenen Skriptsprachen *FlashScript* und *ActionScript* durchaus gegeben – da beide Sprachen jedoch, wenn auch der JavaScript-Syntax entlehnt [vgl. Star01:11], nicht-standardisierte „Erfindungen" Macromedias darstellen, erscheint ihre Anwendung jedoch nicht ganz unproblematisch, da nicht nur die Eigenheiten der Programmiersprache von Grund auf neu gelernt werden müssen, sondern die Skript-Elemente überdies mit den JavaScript-Codebereichen der HTML-Umbebung inkompatibel sind.

Die Anwendung der standardisierten und in Form ihres Vorgängers JavaScript überaus bekannten und geläufigen „Programmier"-Sprache ECMAScript [ECMA97] im Rahmen von SVG stellt an dieser Stelle daher nicht nur den deutlich konsequenteren und aus Nutzersicht auch komfortableren Ansatz dar, sondern ermöglicht überdies den programmatischen Zugriff auf sämtliche Elemente und Objekteigenschaften des SVG-Dateibaumes selber, sowie, da sämtliche Komponenten über dieselbe *Script Engine* verbunden sind, darüber hinaus auch ihrer HTML- und Browserumgebung.

Aufgrund der ausgesprochenen „Mächtigkeit" insbesondere der neuen, objektorientierten Ausprägungen der ECMAScript-Sprache [s.6.5.1] stehen dem Programmierer von Präsentationssystemen somit schier unbegrenzte Möglichkeiten zur Verfügung: Neben einfachen SVG-Objektmanipulationen über die DOM-Schnittstelle [s.5.1] und somit maus-sensitiven *Roll-Overs* wie bereits aus HTML bekannt, ist die Realisierung umfangreicher Frameworks wie etwa einer eigenen Animations-Engine [s.6.5.2.1] oder der *client*seitigen Konvertierung separater XML-Strukturen [s.6.5.1] auf recht komfortablem Wege möglich. Darüber hinaus bietet sich – nicht zuletzt, um dem ohnehin im Rahmen der rechenintensiven *Viewer*-Anwendung stark belasteten Clientrechner zusätzliche „Arbeit" zu ersparen – im Rahmen von SVG die Möglichkeit an,

[1] „[SWF represents] a nice balance between compactness of representation and speed of rendering" [Katz00]

SVG-Dateistrukturen auch serverseitig zu erzeugen, was aufgrund der XML-Eigenschaft der Vektorsprache im Gegensatz zu Flash[1] auch mit recht einfachen Mitteln möglich ist.

5.4.2.7 SVG: The Power of XML

Die „schöne Tatsache, dass SVG XML ist“ [Behm02:53] beschert dem geneigten Programmierer an dieser Stelle jedoch noch weitere Vorteile: So ist nicht nur die Konvertierung, sondern ebenso die direkte Einbettung externer XML-Inhalte und somit eine unbegrenzte Erweiterung des Vektorstandards möglich: Ohne ein „unüberschaubares Tag-Chaos, wie wir es momentan bei HTML sehen“ [Bosa98][2] zu provozieren, erlaubt SVG einerseits über die *Namespace*-Eigenschaft [s. hierzu 6.5.1], andererseits über das so genannte `foreignObject` die nahtlose Einbindung beliebiger Fremdobjekte: So wird etwa die derzeit in der SVG-Spezifikation noch fehlende *Sound*-Funktion im Rahmen des Adobe SVG-Viewers momentan durch die `audio`-Erweiterung mit eigenem Namensraum provisorisch realisiert [vgl. Kunz00], während das *X-Smiles*-Projekt der technischen Hochschule Helsinki [VRKH02] die SVG-Elemente kurzerhand in einen SMIL-Kontext [s.5.5] einbettet und somit die synchronisierte Integration auditiver Elemente ermöglicht [vgl. Watt02:1045]

Auch die Integration weiterer, sich freilich parallel zum SVG-Fortschritt ständig weiterentwickelnder XML-Standards trägt zu einer stetigen Erweiterung des SVG-Funktionsspektrums bei: So ermöglicht die *XML Linking Language*, kurz „xLink“ [DMO01] in ihrer neuesten Fassung etwa weitaus komplexere Link-Funktionen, als sie derzeit etwa in Form der unidirektionalen `<a href…>`-Tags im Web bekannt (und somit natürlich im Rahmen von SVG verfügbar) sind. Da SVG nun die althergebrachte, native `href`-Verlinkung HTMLs durch den externen xLink-Standard ersetzt und seine Link-Komponente somit *modularisiert* hat, steht somit auch SVG-Anwendungen die erweiterte Funktionalität des xLink-Frameworks zur Verfügung, erkennbar beispielsweise an dem separaten xLink-Namensraum aller im Rahmen von SVG auftretenden URIs:

```
<image id="outline" xlink:href="images/outline.png" ... />
```

Listing 5.4.2.4: Definition einer URI-Referenz unter Einbeziehung des externen xLink-Namensraumes.

Zwar kommt in den allermeisten, derzeitigen Fällen der xLink-Anwendung immer noch die althergebrachte, unidirektionale Fassung „einfacher“ URLs zum Tragen – zumindest prinzipiell sind jedoch bereits jetzt komplexere Link-Funktionen [vgl. hierzu Mach97] im Rahmen dieses Frameworks möglich. Auch die direkte Integration weiterer Standards wie xPath, xPointer,[3] xQuery [vgl. Fibi01:4] sowie xForms [vgl. Watt02:1054], die an dieser Stelle jedoch nicht näher ausgeführt werden sollen, erweitert den SVG-Standard gleich um eine ganze Reihe mitunter recht umfangreicher, komplexer XML-Technologien.

5.4.3 Datei-Zugriff

Bei der Betrachtung der eigentlichen „Anwendungen“ der Vektorsprache kommt dem SVG-Standard freilich wiederum die bereits angesprochene XML-Eigenschaft zugute: Da SVG-Grafiken nichts anderes darstellen als leicht editierbare Textdateien, ist somit auch der beidseitige Dateizugriff ein Leichtes. Da mit der standardisierten XML-Schnittstellen XSLT und dem DOM-Ansatz [s.5.1] überdies sehr komfortable Frameworks zur Verarbeitung der SVG-Daten bereitstehen, erstaunt es daher nicht, dass bereits unzählige Ansätze [vgl. Lill99, Schu02] zur Verarbeitung und Konvertierung SVG-basierter Grafiken existieren.

---

[1] s. hierzu insbesondere [4.5.3.3]

[2] “People who don't understand this distinction tend to jump to the conclusion that an XML-aware application will allow them simply to sprinkle new tags throughout their HTML documents. Attempts to "extend" HTML this way will lead to an even worse mess than we've already got.” [Bosa98]

[3] Syntax: xlink:href="#xpointer(id("anObject")); [vgl. Watt02] p.79

Neben der mitunter problematischen XSLT-Schnittstelle [vgl. 5.1] erscheint – insbesondere im Rahmen einer möglichen, servergesteuerten Präsentationslösung – natürlich die Betrachtung Java- und somit DOM-basierter Ansätze interessant: So stehen mit dem SVG Toolkit des australischen CSIRO-Institutes [RoJa00] sowie dem ebenfalls auf OpenSource-Basis betriebenen *Batik* SVG-Toolkit [Croo01] des *Apache*-Projekts gleich zwei äußerst mächtige wie auch komfortable Frameworks zur Verfügung, die die Betrachtung,[1] Verarbeitung und Erzeugung von SVG-Grafiken ermöglichen. In Verbindung mit dem bereits erwähnten (und mittlerweile in das Batik-Projekt integrierten) Schrift-Converter *SVGFont* [Schw00] lässt die Arbeit mit SVG-Grafiken auf Java-Basis somit keinerlei Wünsche mehr offen. Angesichts der Tatsache, dass die Java-Plattform überdies die DOM-Schnittstelle mit am „saubersten" implementiert, erstaunt es freilich nur wenig, dass das SVG-Format neben einigen XSLT- und Perl-Implementierungen [Gard02] insbesondere durch Java-Server-Anwendungen von sich Reden macht, wie etwa auf Basis von JSP [vgl. Hunt02] oder den (mittlerweile in das „Wonder"-Projekt eingeflossenen) *SVGObjects* für Apples Server-Software „WebObjects" [vgl. LaCa03]

Auch im Hinblick auf mögliche Konvertierungs-Optionen kann SVG mit einer beeindruckenden Fülle bereits vorhandener Lösungen punkten: Neben bereits in zahlreichen, vektorbasierten Illustrationswerkzeugen (Illustrator, CorelDraw etc.) enthaltenen SVG-Export-Filtern stehen für alle erdenklichen Bildformate entsprechende Transformationswerkzeuge zur Verfügung. Aufgrund dessen erscheint auch die Tatsache, dass Macromedia dem Vektorstandard sowohl im Rahmen seines Vektor-Tools *Freehand* als auch mit *Flash* bislang die kalte Schulter zeigt [vgl. StLa02, Fest02] insofern mehr als verkraftbar, da auch aus dem für diese Zwecke mitunter „problematischen" [s.5.4.3.3] SWF-Format Flashs eine Konvertierung in ein entsprechendes SVG-Pendant „möglich" [vgl. Prob00a] wenn nicht gar „sehr einfach" [Arty02] realisierbar ist.

Neben dem SWF2SWG-Projekt der Universität Nottingham [Prob00c], welches neben automatischer Flash-Konvertierung überdies die Umwandlung des – dem SVG-Grafikmodell ohnehin näher liegenden – PostScript- bzw. PDF-Formates [s.4.2.1-2] auf Perl-Basis ermöglicht, dominieren auch in diesem Bereich primär Java-basierte Konversionslösungen: So stellt das von Carmen Delessio bereits 1999 entwickelte *WMF2SVG* – Package [Dele99] mit eines der ersten Entwicklungsprojekte dar, welches sich primär auf die erst entstehenden Vektorsprache SVG konzentrierte [vgl. Lill02]. Da das Java-Framework die Transformation der in der Windows-Welt verbreiteten WMF-Metafiles [s.2.3.3] zwar solide, aber (mangels DOM-Support) noch recht unstrukturiert und im Prinzip doch eher rudimentär vornimmt, findet sich in der derzeitigen Version des Batik-Projektes [Croo01] ein recht fortschrittlicher *Transcoder*,[2] der ebendiese Aufgabe weitaus konsequenter zu erledigen weiß. Neben einer Vielzahl weiterer, so genannter „Converter Tools" [vgl. Ferr00], die nahezu sämtliche, erdenkliche Vektorformate konversionstechnisch abdecken, bietet insbesondere der „Graphics 2D SVG Generator" des Java-„Erfinders" Sun Microsystems [Hard00] die Möglichkeit, auf Basis der Java-internen Darstellungsengine (Java2D) *beliebige* Zeichnungen in SVG zu überführen und umgekehrt. Insbesondere im Zusammenhang mit unzähligen weiteren, Java-basierten SVG-Toolkits [Pall00, Herm01] sowie den „klassischen" Frameworks CSIRO und *Batik* besteht unter Einbeziehung der bereits in [3.5.3] sowie [4.4] diskutierten, Java-basierten Viewer-Applets, die ja nichts anderes erledigen, als proprietäre Multimediaformate in ebendiese Java2D-Engine zu überführen, nun die Möglichkeit, quasi jedes nur erdenkliche Grafikformat zur Anwendung im Rahmen von SVG nutzbar zu machen.

Angesichts der schier unübersichtlichen Fülle weiterer Konversions-Tools, Server-Kits und Anwendungs-Frameworks, die samt und sonders auf SVG-basierte Anwendungen zugeschnitten sind (An dieser Stelle geben [Lill99] sowie [Schu02] diesbezüglich einen guten Überblick) kann somit zu diesem Thema abschließend zusammengefasst werden, dass SVG dank konkreter Anwendungs-Frameworks und hilfreicher Tool-

[1] Anm: Beide Toolkits beinhalten auch ein Java-basiertes Viewer-Programm [vgl. Schu02] p.62

[2] Package-ID: org.apache.batik.transcoder.wmf.tosvg. URL: http://xml.apache.org/batik/javadoc/org/apache/batik/transcoder/wmf/tosvg [17.2.03]

kits, die auch Fremdformate für den Vektorstandards nutzbar machen, gerade im Hinblick auf die Realisierung einer möglichen (eventuell Server-basierten) *Präsentationslösung fürs Web* (in dessen Rahmen u.U. die Konvertierung weiterer Formate wie etwa WMF [s.2.3.3] zum Zuge käme) keinerlei Wünsche offen lässt.

5.4.4 SVG aus ästhetischer Perspektive

Doch nicht nur in technischer Hinsicht [s.5.4.2-3], sondern ebenso aus „ästhetischer" Perspektive vermag das SVG-Format (oder genauer: das Gros der Anwendungen auf Basis von SVG) zu überzeugen. In diesem Sinne kann – sogar eher noch, als vielleicht bei Flash [s.4.5.4.2] oder zuvor PowerPoint [2.3.2.2] der Fall – nahezu ein direkter Rückschluss zwischen Formatarchitektur, dem zugrunde liegenden Erstellungsprinzip und den daraus resultierenden, konkreten Ergebnissen (aus gestalterischer Sicht) gezogen werden: So ist etwa die primär statische Natur (auf der das SMIL-basierte Animationskonzept lediglich *logisch aufsetzt*) mit dafür verantwortlich, dass im Rahmen des SVG-Frameworkds bislang die in Flash verbreitete, „sinnlose (Über-)Animation" [vgl. Niel00] oder „Selbstverliebtheit" [vgl. Lipm00, Kais00] SVG-basierter Anwendungen ausgeblieben ist – wohl auch mit bedingt durch das bisherige Fehlen eines primär animationsbasierten Authoring-Tools wie das „umstrittene" [Muel02a] Flash.

Im Gegensatz zu letzterem tragen überdies die in SVG einzigartigen Filter- und Maskierungseffekte [s.5.4.2] dazu bei, auf sehr komfortable Art und Weise ästhetisch und professionell wirkende optische Effekte wie etwa weiche Schlagschatten[1] oder Tiefenschärfe[2] zu erzeugen. Insbesondere in Kombination mit den optisch sehr ansprechenden *Rendering-Eigenschaften* des (mittlerweile als Standard-Plug-In [vgl. Schu02:60f] betrachteten) Adobe SVG Viewers [Adob02], welcher neben vollständigem Anti-Aliasing zudem über eine interpolierte Darstellung auch teiltransparenter Pixelgrafiken [s.Abb.6.3.2] wie etwa des PNG-Formates [s.3.3.2][3] verfügt, hinterlässt die Bildschirmpräsentation SVG-Basierter Grafiken allein aufgrund der positiven Darstellungs-Funktionalität einen abgerundeten, positiven Gesamteindruck.

Darüber hinaus macht sich im Rahmen von SVG mitunter die Tatsache, dass sich der Vektorstandard etwa im Gegensatz zu den Microsoft-Formaten VML [s.5.3] oder WMF [s.2.3.3] derzeit vorrangig im professionellen Grafikbereich ansiedeln lässt [vgl. LiWe01], aus

ästhetischer Perspektive durchaus positiv bemerkbar: So lassen sich ebenso die meisten auf Basis von SVG realisierten Anwendungen, wie auch die im Internet bereits zuhauf zur Verwendung bereitstehenden, im Gegensatz zu den meisten Flash-Bibliotheken[4] jedoch zumeist kostenfrei angebotenen SVG-Grafiken als „überwiegend professionell" [Brun01] charakterisieren – was etwa von der „indiskutablen Ästhetik" [Godi01:3] Microsoft-basierter Grafik-Cliparts. nicht behauptet werden kann.

Nicht nur die soeben beleuchteten, ästhetischen Aspekte, sondern freilich ebenso die bereits zuvor betrachteten technischen Eigenschaften SVGs, die im Verlaufe dieser Diskussion bereits mehrfach Parallelen, Schnittmengen, jedoch ebenso aufschlussreiche Unterschiede zum derzeitigen Marktführer Flash deutlicht gemacht haben, drängen an dieser Stelle nun einen direkten Vergleich des Vektorstandards mit dem bereits in [4.5] näher erörterten SWF-Format und dem damit verbundenen Authoring-System Flash geradezu auf: So stellen nicht nur unzählige, diesbezügliche Arbeiten [vgl. Kunz00, PMEB01, Arty02, Prob00a,b,

[1] s. hierzu Abb. 5.4.2.2 bzw. Listing 5.4.2.3

[2] Anm: Dies ließe sich etwa sehr einfach realisieren, in dem den weiter „hinten" stehenden Grafikschichten ein Unschärfe-Filter zugewiesen wird.

[3] Anm: Das PNG-Format [Bout96] stellt derzeit überdies das „Standard"-Pixelformat, welches im Rahmen von SVG zur anwendung kommt, dar

[4] Anm: Im Rahmen von Flash ergibt sich an dieser Stelle das zusätzliche Problem, dass die Grafiken z.B. zur weiteren Verwendung im internen FLA-Format statt lediglich dem „normalen" SWF-Flash-Format bereitstehen sollten (obgleich man diese freilich „aufknacken" kann, was jedoch einen Bruch des Urheberrechts darstellt). Nach diesem Kriterium existierten dann in der Tat nur noch sehr wenige Angebote, die verwendbare, kostenfreie Flash-Grafiken bereitstellen

Neum02, Scho02 etc.] die Eigenschaften der beiden Formate direkt gegenüber, sondern überdies – direkt oder implizit – stets dieselbe Frage: „So, which format really *is* the better one?“ [Appn02]

5.4.5 SVG versus Flash

Zunächst einmal erscheint es angesichts der expliziten Mitarbeit Macromedias am SVG-Standard [vgl. SVG01] zumindest etwas verwunderlich, wenn nicht gar paradox, dass sich die beiden Format derzeit in einer derartigen „Konkurrenzsituation“ [vgl. D'Amo00] gegenüberstehen: So sollte man zunächst doch annehmen, dass sich Macromedia ebenso wie die Mehrzahl der an der Entwicklung SVGs beteiligten Unternehmen, um entsprechend zugeschnittene Lösungen oder zumindest Veröffentlichungen bemüht. Nicht so bei SVG: Ähnlich wie die SVG-Koautoren Microsoft und Netscape, die bislang zur großen Enttäuschung der „SVG-Community“ keinerlei Anzeichen in Richtung einer Browserintegration haben erkennen lassen [s.5.4.7.1], arbeitet hat Flash-Hersteller Macromedia dem SVG-Standard sogar deutlich entgegen: So wies der Firmensprecher des Unternehmens anlässlich der jüngsten Fertigstellung des SVG „Mobile“-Subsets [FFJ03] statt der Ankündigung, eventuelle SVG-Filter in Macromedia-Produkte nativ zu integrieren, stattdessen lieber auf Kompaktheit und Verbreitung des eigenen Flash-Plugins sowie auf die „geringe SVG-bezogene Nachfrage“ der Kundschaft hin [vgl. Fest02].[1]

> *If you don't think the battle on the Web is Flash against SVG, take another look.* [Dumb01]

Da Macromedia somit offensichtlich die Vorzüge des eigenen, proprietären Formates „preist und vermarktet“ [Fest02], anstatt sich auf den neuen Vektorstandard auszurichten, muss sich der Konzern trotz der derzeit unbestrittenen Marktführerschaft dennoch den Vergleich mit dem XML-Pendant gefallen lassen: Und gerade hier, so die einhellige Meinung der zahlreichen Flash-Kritiker, stellt SVG ein Format dar, dass „viele Vorteile gegenüber Macromedia Flash vorweisen kann“ [Kunz00:10]:

> *SVG offers the features of SWF without the drawbacks of SWF's proprietary nature.* [Pens00]

Obgleich sich, wie das Zitat bereits andeutet, ein Großteil der Kritik an den eher „moralisch-rechtlichen Aspekten“, also der proprietären Natur des Flash-Ansatzes erschöpft, welcher dem OpenSource- und Standardisierungsgedanken SVG freilich diametral entgegensteht, so besitzt das SVG-Format dennoch auch darüber hinausgehende, konzeptionelle Vorteile, die den Vektorstandard nicht nur im Rahmen kartografischer Anwendungen [vgl. NeWi00, Bugg03] oder zur Informations-Visualisierung deutlich im Vorteil sehen, sondern insbesondere auch im Hinblick auf Web-basierte Präsentationen durchaus interessant erscheinen lassen: So stellt etwa die Tatsache, dass das Animation und Interaktionen in SVG „erheblich eleganter“ [NSW02:220] umgesetzt sind, als dies im SWF-Pendant der Fall ist, sowie die im Vergleich zu Flash deutlich komfortablere Abbildung von Objektabhängigkeiten und –Verknüpfungen einen nicht zu unterschätzenden Vorteil dar. Auch im Hinblick auf eventuelle Objektinteraktion hat der auf dem *Dexter*-Modell [s.2.2] aufsetzende SVG-Ansatz deutlich die Nase vorn, da im sich Gegensatz zum „Frame-fragmentierten“ Flash-Format „ganzheitliche“ Animations- und Interaktionszusammenhänge abbilden lassen [s.5.4.2]

Auch und insbesondere die bereits angesprochene Textbasiertheit der XML-Ausprägung SVG beschert dem Format zugleich erhebliche Vorzüge im Vergleich zur binärkodierten Konkurrenz:

> *A great benefit of SVG being text-based it that all text (no matter if a font has been applied to it or not) and graphics are completely searchable, something that has never been possible within Flash.* [ibid]

Auch die hiermit verbundene, sehr einfache Integration in die HTML-Umgebung (im Gegensatz zu den mittels `<object>`-Tag extern zu referenzierenden, binären Flash-„Filmen“ lassen sich SVG-Grafiken naht-

[1] “We have not had significant demand from our user base for SVG-related features…” [Fest02]

los in einen XHTML-Kontext einbetten) sowie die ausgesprochene Kompatibilität mit im Web-Umfeld bereits vertrauten Techniken [vgl. Scho02] wie Cascading Style-Sheets [CSS98], ECMAScript [s.6.5.1] und DOM-Zugriff [s.5.1] stellen insbesondere aus Erstellersicht einen erheblichen Vorteil dar: Wie bereits in Zusammenhang mit dem „trivialen" HTML-Editing gewöhnt, lassen sich SVG-Grafiken schnell und intuitiv mit einem Texteditor fabrizieren – eine direkte Abhängigkeit von einem teuren, professionellen Authoring-Tool wie Macromedias Flash besteht im Rahmen von Svg „glücklicherweise (noch) nicht". [vgl. Bern99:179, Kunz00:11, NSW02:219]

Nicht zuletzt stellt freilich auch die Tatsache, dass SVG ein „völlig offenes, textbasiertes" Vektorformat repräsentiert [vgl. Trip02], einen erheblichen Vorteil des Vektorstandards gegenüber dem „geschlossenen" [Beck02], binären Konkurrenten dar. Im Gegensatz zum Proprietären SWF-Animationsformat können SVG-Grafiken so entweder direkt nach Text- oder (entsprechend der in [4.1] formulierten Anforderungen) auch semantischem Inhalt durchsucht und entsprechend von „Web-Bots" indiziert werden: "Users can thus easily conduct text or Web searches of SVG graphics." [VaNi01]

> *Insgesamt kein Wunder, dass viele die Sprache auf dieselbe Stufe wie Flash stellen - oder gar darüber.*
> [Bem02:53]

Angesichts dieser doch nicht unbeachtlichen Vorzüge der SVG-Technologie erscheint auch die Forderung der Web-Community, den proprietären, ungeliebten [Scho02] de-Facto-Standard Flash endlich durch den „echten" W3C-Standard SVG „mittelfristig abzulösen" [D'Amo00] durchaus verständlich: Auch im Hinblick auf die erheblichen *Usability*-Probleme Flashs [s.4.5.4.1] sowie, teils dadurch bedingt, die nur *begrenzte* Web-Tauglichkeit des SWF-Formates [s.4.5.4] stellt das auf die Anforderungen des WWW regelrecht zugeschnittene SVG-Framework in der Tat eine überzeugendere Lösung für vektorbasierte Web-Grafik, und damit auch im Rahmen dieser Diplomarbeit, für eine entsprechende Präsentationslösung dar

5.4.6 Anwendungsbereiche des SVG-Standards

5.4.6.1 Computer-Aided-Engineering

Neben den „klassischen" *Multimedia*-Anwendungen, die derzeit noch von Macromedias Shockwave [s.3.5.4] und Flash dominiert werden, bietet sich der vielseitige Vektorstandard jedoch im Gegensatz zum Animationsbasierten SWF-Pendant noch für weitere Arbeitsgebiete an: Im Gegensatz zur Ansicht von WebCGM-Verfechter Dieter Weidenbrück [vgl. LiWe01], der an einer „Existenzberechtigung" des veralteten CAD-Standards in Form eines nur rudimentär „aufgemöbelten" Web-Plugins verzweifelt festhält, sieht dies die Mehrzahl der Experten mittlerweile ganz anders:

> *Aus technischer Sicht ist die strikte Trennung zwischen Anwendungsgebiet und Datenformat jedoch nicht nachvollziehbar. Technische Illustrationen sind mit einem geeigneten Potenzial von XML-Definitionen mathematisch ebenso gut beschreibbar wie mit WebCGM. Die Weiterentwicklung der XML-Grafik-Applikationen könnte daher zukünftig beide Anwendungsgebiete abdecken*
> [KrMa99]

Auch die Autoren der SVG-Spezifikaton selber, einst aus politischen Gründen auf die WebCGM-Linie Weidenbrücks eingeschwenkt,[1] haben mittlerweile das „universelle Potential" des eigenen Standards erkannt und schwärmen nun, der Existenz WebCGMs vollständig ungeachtet[2] stattdessen von der „Möglichkeit, mit SVG die Distribution CAD-basierter Illustrationen zu revolutionieren" [vgl. Watt02:913ff]. Bereits existierende, leistungsstarke Converter-Module wie etwa IBMs „CGM to SVG Transcoder" [vgl. Ferr00] demonst-

[1] s. hierzu [LiWe01]: SVG-Entwickler Chris Lilley betätigte sich hier als Co-Autor.

[2] „Although there have been a few attempts […] prior to SVG, […] they have generally been proprietary" [Watt02] p.913

rieren überdies schon jetzt vollständige Konvertierungsmöglichkeiten, die keine Wünsche mehr unerfüllt somit den separaten, ohnehin selten angewandten WebCGM-Standard zusehends unnötig erscheinen lassen

5.4.6.2 Internet-Kartografie

Die klassische Parade-Anwendung des Vektorstandards stellt jedoch zweifellos die Web-basierte *Kartografie* dar: Trotz anfänglicher Begeisterung für das diesbezüglich rudimentäre VML-Format [vgl. GoRi99] konzentriert sich mittlerweile das Hauptinteresse der Online-Kartografen nahezu ausschließlich auf die Fähigkeiten des SVG-Standards [vgl. NeWi00, Bugg03]: Das hierfür eigens eingerichtete *Carto.net*-Projekt verzeichnet bereits über ein Dutzend diesbezüglicher Diplomarbeiten und stellt der Öffentlichkeit unter www.carto.net/papers[1] sämtliche Ausarbeitungen, die sich mit SVG-basierter Kartografie auseinandersetzen, zur Verfügung.

5.4.6.3 SVG für Präsentationen: "Are Powerpoint's days numbered?" [Ogbu00]

Neben diesem „unverzichtbaren Arbeitsbereich" macht [Fibi01] jedoch zugleich unmissverständlich klar, dass das Potential von SVG sich an den eben genannten Anwendungen „bei weitem nicht erschöpft": Neben allgemeiner grafischer Funktionalität,[2] sowie den eben angesprochenen Bereichen der Web-Kartografie und Technischer Zeichnungen wird insbesondere auf die Bedeutung des Internet-Vektorstandards im Bezug auf *Multimediale Präsentationen* hingewiesen:

> *Hier wird wohl noch eine Weile lang Microsoft PowerPoint vorherrschen… Dennoch: Wollen wir einem echten W3C-Standard den Vorzug geben, dann können wir sagen: SVG ist besser. Denn SVG bietet uns deutlich bessere graphische Möglichkeiten sowie bessere Animations- und Interaktionsmöglichkeiten.*
>
> [Fibi01:99]

Genau an dieser Stelle setzt freilich diese Diplomarbeit an: Aufgrund der bereits in [5.4.2-4] gewonnenen Erkenntnisse über die technischen und ästhetischen Vorzüge des SVG-Standards scheint sich das Vektorformat für eine Anwendung im Rahmen Web-basierter Präsentationen förmlich aufzudrängen: So stellt es nicht nur aus „Verarbeitungstechnischer Sicht" [s.5.4.3] sowie ästhetischer Perspektive einen durchaus überzeugenden Ansatz dar, sondern kann neben beeindruckenden, grafischen Möglichkeiten [s.5.4.2] überdies mit optimaler „Web-Fähigkeit" punkten, da sich das rein textbasierte XML-Format auf Basis webkonformer Standards (CSS, DOM, ECMAScript…) scheinbar nahtlos in die HTML-basierte WWW-Umgebung einpasst [vgl. Scho02], aber dennoch mit der Funktionalität PowerPoint mehr als nur „mithalten" kann:

> *SVG potentially gives presentations the same capabilities as Microsoft PowerPoint: Style, images, animations, and multimedia.*
>
> [Ogbu00]

Der in meinen Augen gravierendste, konzeptionelle Vorteil SVGs stellt jedoch die grundsätzliche Struktur des Formates dar, insbesondere im Vergleich zur derzeit für Präsentationszwecke zumeist herangezogenen [s.2.3.4] Flash-Technologie: Während das strikt lineare, Timeline- und Animationsbasierte Authoring-Paradigma Flashs wie auch die entsprechende Dateistruktur des SWF-Formates insbesondere die Erstellung „konventioneller", strukturierter Präsentationen eher erschwert [s.4.5], entspricht das objektorientierte Modell des SVG-Ansatzes dem Präsentations-Gedanken schon viel eher. Da hier zunächst von der statischen Vektorform ausgegangen wird, die sich sowohl entsprechend des Hypertext-Modells, auf Basis verschiedener programmatischer Skriptmöglichkeiten wie auch SMIL-basierter Animationsfunktionalität erweitern lässt, wird sowohl die Erstellung als auch die (noch aus dem Web-Umfeld gewöhnte) Nutzung derart entstehender SVG-Anwendungen drastisch vereinfacht; Überdies können Präsentationsdaten nun *logisch* struk-

[1] Online-Verfügbarkeit zuletzt am [17.2.03] erfolgreich überprüft.
[2] Vgl. [Fibi01] pp.98ff.

turiert werden.[1] Kein Wunder also, dass nicht nur [Fibi01] feststellt, SVG sei dem verleideten [s.2.3.2-3] PowerPoint „deutlich überlegen“,[2] sondern *Sun*-Entwickler Uche Ogbuji den SVG-basierten Ansatz gar als „wahren PowerPoint-Killer“ [Ogbu00] identifiziert haben will:

> *SVG provides an ideal framework for presenting manipulable, interactive content into a multimedia information repository.*
>
> [ChHu01:483]

Nach dieser Erkenntnis stellt natürlich das zentrale Problem in diesem Zusammenhang die Frage da, auf welche Art und Weise die doch sehr umfassende und, wie in [5.4.2] gesehen, mitunter auch komplexe Funktionalität SVGs nun auch für Web-Präsentationen nutzbar gemacht und auf konkrete Präsentationsanforderungen angepasst werden kann. Um sowohl eine *Über*-Linearität (wie etwa in PowerPoint und Flash i.d.R. gegeben) wie auch den Hypertext-typischen „Spaghetti-Effekt“ [s.3.2.2] zu „umschiffen“, und überdies sowohl Autor als auch Nutzer bzw. Publikum vor übermäßiger Komplexität zu bewahren,[3] gilt es an dieser Stelle zunächst, das gewaltige Leistungsspektrum der SVG-Spezifikation hinsichtlich der „präsentationsrelevanten“ Anforderungen einzuschränken [s.6.1], und zudem einen „Kompromiss“ zwischen linearer und „verschachtelter“ Präsentationsführung und –Navigation[4] zu finden [s.6.4]

Aufgrund dessen scheint zunächst eine Betrachtung bereits vorhandener SVG-Präsentationslösungen relevant: Aufgrund der zuvor ausgeführten, ausgesprochenen *Eignung* des Vektorstandards hinsichtlich Web-basierter Präsentationen existiert freilich derzeit eine „schier unüberschaubare Flut“ [Sue02] SVG-basierter Präsentationssysteme [HSL00, Fisc02, Herm02]. Diese orientieren sich jedoch bedauerlicherweise samt und sonders an dem „ungeliebten“ [s.2.3], Slide-basierten „PowerPoint-Paradigma“ und schränken das prinzipiell weitreichende Präsentationspotential SVGs [vgl. Ogbu00] auf die doch recht begrenzten Möglichkeiten linearer, mit Stichwortlisten versehener „Bildschirm-Folien“ à la PowerPoint [vgl. Park01] relativ drastisch ein. Aufgrund der bereits im 2. Kapitel erarbeiteten, massiven Kritik und deutlichen Problematik, die mit diesen „sturen Schema“ [Sear98] einhergeht, soll im Rahmen dieser Diplomarbeit freilich ein anderer Weg beschritten werden – mit den entsprechenden, gedanklichen Grundlagen und der eigentlichen Realisierung werde ich mich jedoch im 6. Kapitel auseinandersetzen.

5.4.7 Problembereiche und Lösungsansätze

5.4.7.1 Präsentationsmodus

Zuvor sollten jedoch ebenso noch eventuelle Schwächen des derzeitigen SVG-Frameworks zur Sprache kommen, um eventuelle „böse Überraschungen“ im Rahmen der Prototyp-Konzeption zu vermeiden: So merkt etwa Iris Fibinger im Rahmen ihrer Diplomarbeit[5] an, dass „der einzige echte ‚Noch’-Nachteil“ des SVG-Formates gegenüber PowerPoint darin bestünde, dass es „derzeit keinen SVG-Viewer mit ‚Präsentationsmodus’“ gebe. [Fibi01:99] In diesem Punkt liegt Fibinger jedoch nicht ganz richtig: Zum einen ließe sich dieser Effekt im Rahmen der neuen SVG-Integration des PDF-Formates [Foss03] erreichen [s.4.2.2.5], welches eine *FullScreen*-Funktion[6] bereits im „Header“ automatisch „mitführt“ [vgl. BCM01], zum anderen ist der Vollbildschirm-Präsentationsmodus jedoch ebenso in einer WWW-Umgebung problemlos realisier-

[1] Anm: Wiederum im Gegensatz zum SWF-Format, wo der Flash-Autor i.d.R. keinen Einfluss auf die (zumeist ungeordnete) Struktur der eigentlichen Internet-Exportdatei (.SWF) hat.

[2] [vgl. Fibi01] p.99

[3] Anm: Schließlich stellt der Kernpunkt des Erfolges von PowerPoint ja auch dessen (zunächst gegebene) *Schlichtheit* dar.

[4] s. hierzu speziell den „Linearen Präsentationsmodus“ in [6.5.2.2]

[5] vgl. [Fib01]

[6] Syntax: obj << ... /PageMode /FullScreen ... >> endobj

bar: So ließe sich etwa mittels ECMAScript die Öffnung eines separaten, mittels „`fullscreen=yes`"-Attribut oder alternativer Maximierung[1] „triggern" und so die gewünschte „Bildschirm-Füllung" erreichen.

5.4.7.2 Komplexität

Weitaus schwerer wiegt an dieser Stelle hingegen die meiner Einschätzung nach erheblich zu hohe Komplexität, die die Entwickler der meisten bereits am „Markt" befindlichen[2] Präsentationslösungen [HSL00, Fisc02, Herm02] den Anwendern bei der Erstellung entsprechender Präsentationen aufbürden: So ist nicht nur in bislang keinem der Fälle eine grafische Benutzerschnittstelle vorhanden – zumeist werden überdies (zumindest rudimentäre) Programmierkenntnisse oder Vertrautheit mit XSLT-Prozessorumgebungen [s.5.1] vorausgesetzt. Die zumeist sowohl aus technischer als auch ästhetischer Hinsicht unbedarfte PowerPoint-Zielgruppe [vgl. Pirn01] dürften derartige Anforderungen und Benutzerunfreundlichkeit jedoch eher abschrecken.

Daher erstaunt es auch nur wenig, dass sich die Anwendung des SVG-Formates und speziell SVG-basierter Präsentationen derzeit noch fast ausschließlich im „akademischen Elfenbeinturm" konzentriert: Potentielle Anwenderschichten im kommerziellen Bereich wagen sich aufgrund bislang fehlender, komfortabler Authoring-Tools nur zögerlich an den unbekannten Standard heran. Da die entsprechende Viewer-Software überdies bisher im Vergleich zur Konkurrenz eher magere Verbreitungszahlen erreicht, stützt sich die überwiegende Mehrheit „professioneller" Nutzer hingegen überwiegend „pragmatisch" [vgl. Gibs00] auf zwar fragwürdige [s.4.5.4], aber dennoch etablierte Alternativen:

> *If SVG tools for artists appear and browsers offer native support for the format, SVG has a shot at success. Until then, my money's on Flash, since even nonartistic types like me can already make cool effects in Flash […] without the hassle of tagging.*
>
> [Powe00]

Derweil ist sich die Branche zwar einig, dass SVG konzeptionell deutliche Vorzüge gegenüber dem teils „verhassten" [Muel02] Flash-Konkurrenten aufweist – „Da das Format aber auf die Werbebranche zugeschnitten ist, konnte es sich bislang recht deutlich durchsetzen." [NeWi00]. Aufgrund dessen erscheint nicht nur angesichts des „übermächtigen Lokalmatadors Flash" [vgl. Gibs00] die Frage angebracht, in welchem Verhältnis insbesondere die für Web-Produzenten wichtigen Verbreitungszahlen der entsprechenden Plug-Ins zueinander stehen und wie auf dieser Basis um die „Erfolgschancen" des SVG-Formates denn bestellt ist.

5.4.7.3 Verbreitung

Und in der Tat sehen die Perspektiven von SVG, entgegen der Ansicht von [Kunz00][3] realistisch betrachtet diesbezüglich „nicht mehr sonderlich rosig aus" [Arah01]:[4] Im Gegensatz zum unbestrittenen Marktführer Macromedia, der für das proprietäre Flash-Plug-In Verbreitungszahlen von „bis zu 98%" angibt [vgl. Macr02], rechnen selbst optimistische Schätzungen hinsichtlich des derzeit führenden SVG-Plug-Ins *Adobe SVG Viewer*[5] mit einer Download-Rate im lediglich zweistelligen Millionenbereich: „Bei 500 Millionen In-

---

[1] Anm: An dieser Stelle würden entsprechende Koordinaten des maximalen Bildschirmbereichs zur Anwendung kommen (nur bei Netscape-Browsern erforderlich, IE unterstützt den FullScreen-Modus)

[2] Anm: Hierbei muss freilich hinzugefügt werden, dass sämtliche SVG-basierte Präsentationstools (mit Ausnahme des kommerziellen *ShowCaster* [vgl. LiJa03], der SVG halbherzig lediglich als optionales Export-Medium unterstützt) durchweg auf *OpenSource*-Basis und *kostenlos* angeboten werden.

[3] „Da Netscape und Microsoft an diesem Projekt mitarbeiten, ist gewährleistet, dass dieser Standard ebenso wie Flash sehr schnell in die Browser integriert werden könnte und somit sehr schnell viele User vorhanden sind, die SVG Grafiken betrachten können" [Kunz00]

[4] "Suddenly the future for SVG isn't looking quite so bright. Flash offers greater functionality, better compression and a larger installed user base" [Arah01]

[5] Anm: Neben dem *ASV* existieren jedoch noch die Java-betriebenen Viewers „Squiggle" (Batik) sowie die Viewer-Komponente es CSIRO SVG Toolkits [RoJa00] noch der erst jüngst vorgestellte IBM SVG Viewer [vgl. LiJa03]. Details hierzu siehe [Lill99, Behm01]

ternet-Nutzern", so rechnet etwa [Adam02] vor, ergebe das eine Verbreitung von „immerhin rund 10 Prozent"[1] – ein geradezu lächerlicher Anteil im Vergleich mit der fast vollständigen Abdeckung des Flash-Pendants. Neben den bereits im Rahmen von [4.5] dargelegten, „aggressiven Marketing" von Seiten Macromedias [vgl. Raux01] trägt zu diesem verhältnismäßigen Ungleichgewicht sicherlich auch die Tatsache bei, dass das Adobe-Plugin derzeit deutlich umfangreicher [2] und überdies Performance-technisch noch eine stufe behäbiger daherkommt als das ohnehin in dieser Hinsicht „sehr langsame" [Saul02] Flash-Pendant.

Aufgrund der schon in [3.5.4] diskutierten, ohnehin problematischen Anwendung von Plug-Ins an sich, sowie insbesondere angesichts der Tatsache, dass SVG von Seiten des W3C bereits seit geraumer Zeit als normativer, von zahlreichen, mächtigen Vertretern anerkannter Standard [vgl. SVG01] offiziell verabschiedet ist, weisen unter anderem die Siemens-Entwickler Jürgen Krüger und Christian Märtin energisch darauf hin, dass „Standards wie SVG eigentlich *ohne* Plug-Ins auskommen sollten" [KrMa00]:

> *You may find it odd that it takes a proprietary plug-in to support an open standard, but such is the state of the Web. Once the SVG standard is finalized, we suspect that browser makers will begin investigating ways to support it natively*
>
> [Zeld01]

Doch trotz der hier geäußerten Hoffnung Jeffrey Zeldmans ist es bedauerlicherweise auch mit der von [Kunz00] prophezeiten *Browser-Integration* des SVG-Standards bislang nicht weit her: So gibt es derzeit „weder von Microsoft eine Absichtserklärung, noch eine funktionierende Umsetzung seitens Netscape für seine aktuelle Browser-Version" [NeWi00]:

> *So if even if IE7 includes SVG, it looks like it will be a long time before SVG will be able to be taken for granted on PC browsers.*
>
> [Dumb01]

In der Tat lassen die fehlenden Bemühungen sämtlicher Browserhersteller, trotz Beteiligung am SVG-Konsortium, derzeit „erheblich zu Wünschen übrig" [KrMa00]: "All told, the browser situation doesn't look that great." Und doch scheint es Bewegung auch in dieser Richtung zu geben: So hat insbesondere in der Vergangenheit die „Croczilla"-Komponente [vgl. Frit02] des unabhängigen Browser-Projektes *Mozilla* erhebliche (wenn auch bislang unvollständige) Entwicklungsbemühungen hinsichtlich einer SVG-Integration in den Webbrowser unternommen. Wenn auch die von Alexander Fritze initiierten Arbeiten „mittlerweile in Stocken zu geraten scheinen" [Schu02:62], und ein offizieller, nativer SVG-Support daher „in näherer Zukunft unrealistisch erscheint",[3] so ist der Netscape-Sprössling dennoch der derzeit „heisseste Kanndidat" im Rennen um offizielle SVG-Unterstützung. Indes lässt sich mittlerweile selbst an der bislang „sturen" Microsoft-Front SVG-bezogene Aktivität ausmachen: So lässt etwa W3C-Gruppenleiter Chris Lilley durchklingen, dass der Software-Gigant in jüngster Zeit „nicht zu unterschätzende" Bemühungen in dieser Richtung unternommen haben:

> *Despite the absence of any announcements by Microsoft of support for SVG, there are indications at the time of writing that Microsoft has some exploratory work on SVG underway. Whether that work will lead to a decision on the part of Microsoft to support SVG natively in IE remains to be seen.*
>
> [Watt02:33]

Aufgrund dieser Erkenntnisse kann an dieser Stelle zusammenfassend festgestellt werden, dass "nativer Browser-Support zum jetzigen Zeitpunkt noch äußerst dürftig ausfällt".[4] Dies erscheint in meinen Augen

---

[1] vgl. [Adam] p.16

[2] Anm: Adobes SVG-Plug-In "wiegt" mit gut 2 MB derzeit erheblich mehr als die vergleichsweise schlanke *Flash*-Viewer-Komponente (250 KB)

[3] "For various reasons, official Mozilla builds will not include SVG support for the near future" [Frit02]

[4] "At the time of writing, however, native browser support for SVG is poor." [Watt02] p.33

insofern unverständlich, wenn nicht gar unfassbar, als dass die von Microsoft und Netscape mit unterstützte Entwicklung des Vektorformates bereits vor gut fünf Jahren begonnen hat und überdies schon seit zwei Jahren als offizieller „Web-Standard" verabschiedet ist – eine „halbe Ewigkeit" im Kontext des „schnelllebigen Internetzeitalters" [Doer00].

5.4.7.4 Authoring-Tools

Mit eines der Hauptprobleme des SVG-Formates, sich auch als *de-Facto*-Standard [NeWi00] im Web durchzusetzen, besteht sicherlich jedoch in dem noch derzeitigen Fehlen eines „Killer-Tools" zur Erstellung SVG-basierter Inhalte: Im Gegensatz zu Flash-„Eigner" Macromedia, welcher als „Monopolist" [vgl. Zmoe00] zugleich über das unbestrittene, zentrale Authoring-Tool des proprietären SWF-Formates verfügt, gestehen sogar die Entwickler des SVG-Standards eine diesbezügliche, „bedauerliche Lücke" [vgl. Watt02:27ff] kleinlaut ein. Selbst das diesbezüglich wohl fortgeschrittenste Produkt „WebDraw" des *Paint Shop Pro*-Herstellers Jasc [vgl. Trin02] kann in meinen Augen bei weitem nicht an die Funktionalität des Flash-Studios heranreichen: Zu unausgereift erscheinen insbesondere Quellcode-Kopplung und Benutzerschnittstelle im Vergleich zur übermächtigen Profi-Konkurrenz.

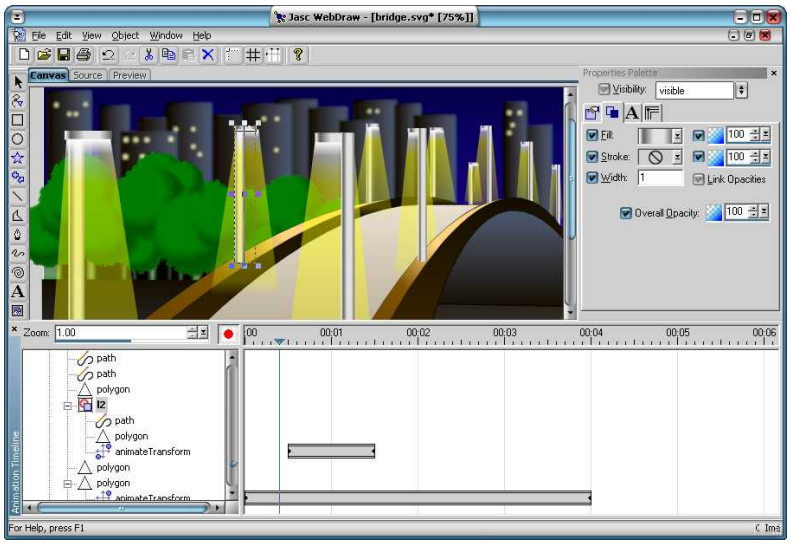


Abb. 5.4.7.3.1: User-Interface des SVG-Werkzeugs Jasc WebDraw. (rechts)

Neben den durchaus zahlreichen, aber vom Funktionsumfang insgesamt eher dürftigen Bearbeitungswerkzeugen mit SVG-Unterstützung [vgl. hierzu Lill99, Behm02] ruhen die Hoffnungen der SVG-Community daher erstlinig auf den angekündigten Produkten des federführenden SVG-Unternehmens Adobe: Insbesondere dem als Flash-Konkurrent zunächst „gefloppten" *LiveMotion 3* sowie die künftige Version des *ImageStylers* ist dank SVG-Support in den Folgeversionen ein diesbezüglicher „Killer-Status" durchaus zu wünschen [vgl. Arah99][1]

Darüber hinaus eröffnet die SVG-Architektur – ebenso wie bereits zuvor das Dokumentorientierte Seiten-Markup HTML – jedoch eine konsequente Separation der verschiedenen Erstellungsschichten, die ein komplexes, „allumfassendes" Killer-Tool zusehends unnötig erscheinen lassen [vgl. Fren02]. So lassen sich im Rahmen des SVG-Frameworks die verschiedenen, erstellerischen „Arbeitsschritte" sauber trennen: (Vektor-orientierte) Grafik-Designer müssten sich (dank ubiquitärem SVG-Filter in Adobe- und Corel-Produkten) lediglich um den Export zuvor erstellter Zeichnungen ins SVG-Format kümmern, während separate SVG-„Optimierer" (ähnlich der zuvor verbreiteten „HTML-Programmierer") sich lediglich um die Anpassung und „Nach-Komprimierung" des Codes kümmern brauchten.[2] Client- und serverseitige Skriptprogrammierung, CSS-Stilfragen sowie XSLT-Umwandlungen könnten in diesem Zusammenhang dann wiederum an weitere, spezialisierte Mitarbeiter übertragen werden.

Dem derzeitige Fehlen eines „alles erschlagenden" und damit „durchaus auch problematisch erscheinenden" [Fren02] Autorenwerkzeuges kann unter diesem Aspekt in der Tat auch Positives abgewonnen werden – stellt es doch nicht zuletzt im Rahmen unserer Zwecke [s.Kap.1]# eine *Chance* dar, ein benutzerfreundli-

[1] "SVG could offer the vector and restyling benefits of ImageStyler without the need for a bitmap middleman" [Arah99]
[2] vgl. hierzu Appendix J der SVG-Spezifikation [SVG01]

ches, insbesondere auf die Erstellung Web-basierter Präsentationen gemünztes, Programm bzw. User-Interface für das WWW *selber* zu erstellen – eine Idee, auf der diese Diplomarbeit aufsetzt: Durch die Umsetzung eines komfortablen „maximal einfachen GUIs" entstünde zwar nicht notwendigerweise „der PowerPoint-Killer auf SVG-Basis schlechthin" [Ogbu00] – zumindest aus UI-Sicht würde jedoch in diesem Bereich eher „eine Lücke gefüllt" [vgl. Fibi01:99], als dies in dem diesbezüglich sicherlich gesättigten Flash- oder PowerPoint-Segment der Fall wäre.

Da jedoch insbesondere aus dieser Ecke, wohl sicherlich aus „Angst"[1] vor einem Erfolg des SVG-Standards, vermehrt strategisch motivierte, zynische Kommentare laut geworden sind [vgl. Powe00, Gibs00, Arty01, Fest02], gilt es jedoch zunächst, neben der bereits diskutierten [s.5.4.7.1-3], berechtigten Kritik insbesondere an den externen Umständen (Verbreitung, Tools) des viel versprechenden Formates die oftmals geäußerten „Mythen", die den Standard mitunter noch immer umranken, zu „entzaubern".

5.4.7.5 Entzauberung der SVG-Mythen

5.4.7.5.1 Mythos Nr. 1: Riesige Dateien

Der sich insbesondere von Seiten der Flash-Verfechter gegen den Vektorstandard richtende Hauptvorwurf stellt sicherlich die Behauptung dar, dass SVG aufgrund der Texteigenschaft „schlicht und einfach zu platzverschwendend" sei, um den Bandbreitenbeschränkungen des Internet gewachsen zu sein:

> *Compared to SWF, SVG is a bandwidth hog.* [Arty01]

Sicherlich nimmt die XML-basierte Codierung zunächst mehr Speicherplatz ein, als dies etwa im Rahmen des binärkodierten Flash-Format der Fall ist – dennoch muss zugleich erwähnt werden, dass das Vektorbasierte Speicherverfahren sowie insbesondere die im Vergleich zum SWF-Pendant deutlich kompaktere Animationsdarstellung erheblich dazu beiträgt, dass sich die Datenrate SVG-basierter Bilddateien doch „deutlich im Rahmen hält"[2]. Insbesondere im direkten Vergleich mit dem binären PowerPoint-Format fallen entsprechende SVG-Pendants „wesentlich kleiner" aus [Fibi01:99]. Aufgrund des nicht zu verachtenden Rendering-Aufwandes merkt überdies [Powe00] an, dass insbesondere im Internet der zur Darstellung benötigten Rechenzeit mittlerweile „erheblich größere" Bedeutung zukommt,[3] zumal diese Debatte mit „universell verfügbaren Breitbandanbindungen" ohnehin zunehmend hinfällig wird [vgl. Rieh01:169].

Dennoch besteht im Rahmen der SVG-Spezifikation zur Einsparung von Speicherplatz und Bandbreite die Möglichkeit, Vektordaten statt in „reiner Textform" stattdessen mit GZIP-Komprimierung[4] abzuspeichern, die der Kompressionsrate des entsprechenden Flash-Pendants mit 10% nur noch unerheblich nachsteht.

5.4.7.5.2 Mythos Nr. 2: SVG - ein stummer Standard

Der zweite, nicht minder oft vernehmbare Vorwurf gegen SVG lautet hingegen, dass das Vektorformat „keinerlei Soundunterstützung" bereitstelle und aufgrund dessen einen „stummen Standard" darstelle, der im Hinblick auf mangelnde Video- und Audioeigenschaften nur bedingt als „Multimedia-fähig" bezeichnet werden könne. Auch diese Aussage kann jedoch zweifellos dem bereich der Legendenbildung zugeordnet werden, denn „natürlich bietet SVG auch Soundunterstützung" [Kunz00]: Wie bereits im Rahmen von [5.4.2] erläutert, kommt an dieser Stelle die MP3-basierte Sounderweiterung von Adobe zum Tragen. Gleichzeitig muss jedoch ebenso festgestellt werden, dass SVG in der Tat primär ein *Vektor*standard ist – der

---

[1] "Macromedia is just plain afraid of SVG," aus: Simon St. Laurent: "What I'd like is…" O'Reilly Network Weblogs – Feedback-Thread zu [StLa02]

[2] vgl. hierzu Appendix J der SVG-Spezifikation [SVG01]

[3] Anm: Aufgrund der Encodierung „in Reinform" hat an dieser Stelle freilich SVG die „Nase vorne."

[4] Anm: In diesem verlustlosen,auch in Form des ZIP-Formates bekannten Verfahren kommen insbesondere Erkenntnisse der Lempel-Ziv-Komprimierung [s.3.3.1.1] zum Tragen.

allerdings gerade „in Verbindung mit anderen XML-Formaten seine multimedialen Fähigkeiten voll ausspielen kann“ [Cagl02:12]: So stellt insbesondere die bereits im Rahmen des *X-Smiles*-Projekts [VRKH02] vollständig unterstützte Verknüpfung von SVG mit dem Synchronisationsstandard SMIL [s.5.5] eine ideale Kombination dar, die neben Webgemäßer Vektorgrafik und Sound überdies die synchronisierte Einbindung *gestreamten* Video-Materials [s.3.5.1] konsequent ermöglicht.

5.4.8 Abstrakte Daten und SVG

Ebendiese „anderen“ XML-Formate sind nun auch Ansatzpunkt einer weiteren, unumgänglichen Überlegung hinsichtlich Web-basierter Präsentationen: So ist es derzeit, wie ja bereits in [5.3.4] diskutiert, im Rahmen von SVG nur begrenzt möglich, über den reinen Textinhalt hinaus abstraktere, logische Verknüpfungen zwischen einzelnen SVG-Elementen schematisch abzubilden. Da sich die W3C-Arbeitsgruppe, wie dem Standard [SVG01] leicht zu entnehmen ist, letztendlich dazu entschlossen hat, die unter dem Arbeitstitel „SVG1“ konzipierte, höhere Abstraktionsschicht des Vektorformates zugunsten erweiterter, grafischer Möglichkeiten zu vernachlässigen [vgl. Herl99], lässt auch die aktualisierte Version 1.1 des Standards [FFJ03] auch an dieser Stelle noch einige Wünsche offen:

> *Although the SVG format is a huge step forward for many kinds of images, we can do even better for diagrammatic illustrations. In particular, SVG does not provide for flexible layout given different viewer requirements and browser capabilities, such as screen format and font preferences.*
>
> [Badr01:489]

Überdies ermöglicht die XML-Eigenschaft SVGs zwar die generelle Durchsuchbarkeit des Formates – ebenso wie beim „indirekten Vorgänger“ HTML verbleibt jedoch das von [Bosa97] als „HTML-Dilemma“ [vgl. Star01:24] bezeichnete Problem, dass mit dem indizierten Text (zumindest im Moment) *nicht* zugleich dessen semantische „Bedeutung“ berücksichtigt werden kann, weiterhin bestehen. Im Gegensatz zum HTML-Format, welches zumindest im Rahmen von Überschriften-Tags (`<h1>`,`<h2>`…) eine rudimentäre, semantische Formatierung vorsah und zuletzt ausschließlich durch „Aufweichung“ und „Missbrauch“ [Muen98] des Standards an derartiger Strukturierung einbüßte [s.3.2], ist im Rahmen des primär *visuell* orientierten Grafikstandards SVG eine derartige Strukturierung hingegen nur in sehr eingeschränkten Umfang möglich.

An diesem Punkt setzt hingegen die *Erweiterbarkeits*-Eigenschaft des SVG-Formates an: Da sich der Vektorstandard, ebenso wie jede andere XML-Ausprägung, durch speziell ausgezeichnete Fremd-Elemente beliebig „ausbauen“ lässt, legen nun gleich mehrere Lösungsalternativen eine dementsprechende, semantische Erweiterung der im Rahmen von [SVG01] spezifizierten Syntax nahe: Zum einen wird durch die Einführung separater `<constraint>`-Tags im Rahmen der so genannten „Constraint SVG“-Ausprägung (CSVG) die Definition logischer Variablen und Verknüpfungen zwischen einzelnen SVG-Elementen möglich [vgl. Badr01] – mit der Problematik der „semantischen Bedeutung“ der SVG-Inhalte setzt sich hingegen insbesondere der kanadischen IT-Spezialist Philip Mansfield[1] auseinander: So legen [MaFu01] einerseits grundsätzlich die Einführung so genannter „Grafischer Stylesheets“ nahe, zum anderen setzt Mansfield mit dem Programm *Catwalk* [vgl. Mans01, Fibi01:14] den zuvor theoretisch formulierten Anspruch sogleich in die Tat um: Dieser auf XSLT fußende Ansatz ermöglicht die direkte, automatische Konvertierung beliebiger XML-Dokumente, seien es nun Mathematische (MathML) oder Chemische Formeln (CML), in visuelle SVG-Grafiken.

Die durchaus beachtlichen Möglicheiten, die der *Catwalk*-Ansatz durch die Visualisierungs-Option beliebiger XML-Strukturen damit aufzeigt, legen jedoch zugleich eine weitere *Trennung* von semantischer Struktur und deren letztendlicher Darstellung in Form des SVG-Formates nahe, der *XML Style Sheet Language* (XSLT, deren „problematische“ Syntax [s.5.1] in diesem Zusammenhang derweil elegant umschifft wird) sei dank.

[1] URL: http://www.schemasoft.com [18.2.03]

Zur vertieften Diskussion dieser logischen Separation, die an dieser Stelle allerdings bereits zu weit in Fragestellungen bezüglich einer konkreten Realisierung eintauchen würde, sei an dieser Stelle lediglich auf [6.1] sowie [6.5.1] verwiesen, wo diese Aspekte im Rahmen der Konzeption eines reellen Prototypen vertieft behandelt werden sollen.

5.4.9 Fazit

Zunächst jedoch kann im Vorfeld dieser Überlegungen an dieser Stelle noch einmal zusammenfassend festgestellt werden, dass das Konzept XML-basierter, skalierbarer Vektorgrafiken im Sinne des SVG-Standards aus struktureller Sicht einen konsequent „sauberen" sauberen Ansatz darstellt, der beeindruckende grafische Möglichkeiten mit webfähiger, standardisierter Internettechnik vereint und somit im Gegensatz zur diesbezüglich problematischen Konkurrenz[1] eine durchaus beachtliche „Alternative" [vgl. Behm02] hinsichtlich tatsächlich webgemäßer Multimedia-Präsentation aufzeigt.

Im Rahmen dieser Diplomarbeit, die sich ja mit der Untersuchung multimedialer Internetformate insbesondere im Hinblick auf Web-basierte Präsentationen beschäftigt, erweist sich der Vektorstandard somit für unsere Zwecke als nahezu *ideal* geeignet, da das SVG-Format aus konzeptioneller Sicht sicherlich mit den interessantesten Ansatz der Darstellung und Verarbeitung multimedialer Daten im Internet repräsentiert und sich somit in der Tat nicht nur den „Flash-"[2] wie auch „PowerPoint-Killer schlechthin" [vgl. Ogbu00, Fibi01:99] darstellt, sondern durchaus noch darüber hinausgehende Möglichkeiten hinsichtlich multimedialer Präsentationen eröffnet [vgl. Cagl02:12][3]

Dennoch verbleibt an dieser Stelle die Befürchtung, dass dich dieses hochinteressante Format, trotz unbestrittener Vorzüge [s.5.4.2-6], eventuell nicht gegen die derzeit „übermächtige Konkurrenz"[4] durchsetzen könnte: Zu erdrückend erscheint momentan angesichts jeweils nahezu vollständiger Marktabdeckung das Ungleichgewicht zugunsten der beiden „quasi-Monopolisten" [Zmoe00, Stew01] Microsoft und Macromedia. „Doch SVG zieht nach" [NSW02:220]: Aufgrund steigender Verbreitungszahlen, optimistischer Anzeichen von Seiten der Browser-Hersteller [s.5.4.7.3] und der Hoffnung auf Adobes SVG-„Killer-Tools" *ImageStyler* und LiveMotion 3, so ist sich zumindest [Adam02] sicher, wird „der Markt spätestens mit SVG 2.0 kräftig ins Schwanken kommen"[5] – auch wenn es bis dahin, wie [Jack03a] deutlich macht, durchaus noch eine Weile dauern kann.

Da derweil jedoch insbesondere „mutige Pioniere" gefordert sind [vgl. Cagl02, Watt02], die sich trotz bislang nicht vollständiger Plug-In-Abdeckung an die Implementierung des noch jungen Standards machen, fiel mir letztendlich die Entscheidung nicht mehr sonderlich schwer, die in meinen Augen zukunftsweisende SVG-Technologie zur Umsetzung eines *eigenen*, Web-basierten Präsentationsansatzes zu nutzen – zu „kleinlich", ängstlich und rückwärtsgewandt erschienen mir die Argumente selbsternannt „pragmatischer" Web-Designer zugunsten einer „konventionellen" Anwendung des „kleineren Übels" Flash [vgl. Gibs00].[6] Da der Vektorstandard, wie im Rahmen dieses Kapitels ausführlich diskutiert, geradewegs auf die Bedürfnisse und Anforderungen Web-basierter Präsentation zugeschnitten ist, erscheint mir die Anwendung des SVG-Formates auch rückblickend im Hinblick auf die Zielsetzung dieser Diplomarbeit als der konzeptionell „sauberste", ideale Weg.

---

[1] Anm: Hiermit sind selbstredend in erster Linie Flash [s.4.5] und PowerPoint [s.2.3] gemeint.

[2] vgl. [Cagl02] p.12

[3] "Most people [...] will try to judge it against existing applications: SVG vs. Adobe PostScript, SVG vs. Macromedia Flash, SVG vs. Microsoft PowerPoint. All of these are valid comparisons, but to view SVG this way is to miss its true applicability." [Cagl02] p.12

[4] „Natürlich stellt sich anhand eines mächtigen Konkurrenten wie Flash die Frage, ob man SVG im Web heutzutage überhaupt einsetzen kann…" [Adam02] p.16

[5] vgl. [Adam02] p.14

[6] "Until the big-name players, such as Microsoft, develop fully-baked software that incorporates the SVG standards I would rather use working ubiquitous products… This forces the designs to be established proven, older solutions" [Gibs00]

5.5 SMIL

Vor der Diskussion der eigentlichen, „konkreten" Schritte im Hinblick auf die konzeptionelle Annäherung an einen möglichen Prototypen für ein Web-basiertes Präsentationssystem [s.Kap.1] gilt es jedoch zunächst, die Syntax des oftmals als „Multimedia-Präsentationsformat schlechthin" [vgl. BoEi02:113] bezeichneten Web-Standards SMIL [SMIL98,01] zum besseren Verständnis noch einmal kurz zu beleuchten: Angesichts der Tatsache, dass Teilkomponenten dieser *Markup-Sprache* etwa im Rahmen der „SMIL-basierten Animationssyntax SVGs" [s.5.4.2] oder im Zusammenhang mit dem „Textunterstützten" Streaming-Ansatz [s.3.5.1] bzw. der Hypertext-Zeitwerweiterung HTML+SMIL [s.5.5.2] bereits mehrfach zur Sprache kamen, dessen zugrunde liegenden Prinzipien bislang hingegen nur rudimentär erläutert werden konnten, soll dies im Kontext XML-basierter Multimedia-Formate dieses Kapitels (und ein ebensolches *ist* SMIL ja) an dieser Stelle „noch schnell" nachgeholt werden.

5.5.1 Entwicklung und Kontext

Zunächst jedoch erscheint eine kurze Reminiszenz zur korrekten konzeptionellen und entwicklungshistorischen Einordnung des Formates angebracht: So entstammt die Grundlage und Notwendigkeit der Entwicklung des SMIL-Frameworks etwa der bereits im Rahmen von [2.2] sowie [3.5] beleuchteten Erkenntnis, dass bisherige Hypermediamodelle, und in diesem Zusammenhang speziell das HTML zugrunde liegende *Dexter*-Hypertextmodell [HaSc94]

„insbesondere den *Zeitfaktor* multimedialer Präsentationen bislang nur unzureichend zu berücksichtigen vermögen"[1] [vgl. Hard99, HORB99]. Das schließlich auf dieser Erkenntnis fußende, *Amsterdam Hypermedia-Modell* [HBR94] des niederländischen Informatikinstitutes CWI [s. hierzu auch 2.2] bezieht schließlich auch zeitkontinuierliche Elemente [vgl. Stei99:11] in diesen Multimedia-Kontext mit ein, bleibt jedoch, wie Kommunikationsforscher Dietrich Boles von der Universität Oldenburg anmerkt „weit hinter den Möglichkeiten moderner Multimediasysteme zurück" [vgl. Bole98]. Dennoch gelang es den holländischen Forschern recht erfolgreich, den nicht unbeträchtlichen Einfluss ihres führenden Institutes auch beim World Wide Web-Konsortium geltend zu machen: Der schließlich (nach zweijähriger W3C-Entwicklungsarbeit) 1998 verabschiedete SMIL-Standard [SMIL98] trug daher nicht nur die deutliche Handschrift der CWI-Ingenieure, sondern kann darüber hinaus sogar als „Quasi-Implementierung" des AHM-Modells bezeichnet werden, da die grundlegende Architektur des Formates sich doch relativ deutlich an den prinzipiellen, AHM-spezifischen Überlegungen orientiert. [vgl. Hard98:51ff]

5.5.2 SMIL 1.0: schlicht und kompakt

Der grundlegende Ansatz der „Synchronized Multimedia Integration Language", kurz SMIL, besteht nun darin, zwar ähnlich wie HTML auf einer überschaubaren, simplen Tag-Auswahl einfaches Authoring *auf XML-Basis* zu ermöglichen, dies im Gegensatz zum Hypertext-Ansatz jedoch im Hinblick auf „Fernsehmäßige, multimediale Präsentationen" [vgl. SMIL98]. Die erste, Mitte 1998 standardisierte Fassung der Multimedia-Sprache war daher „ausgesprochen simpel" [Arci02] ausgelegt, und gliederte sich zu diesem Zeck in verschiedene funktionale Komponenten, die jeweils unterschiedliche Aspekte multimedialer Präsentation berücksichtigen: Zum einen ist im Rahmen des so genannten „Spatial Layout" eine räumlich präzise Positionierung verschiedenster Multimedia-Elemente möglich: So lassen sich Bilder, Text und Video mit beliebigen absoluten oder auch relativen Koordinaten in ihrer Position sowie Größe exakt bestimmen, auf Wunsch auch kacheln oder strecken und überdies (mittels so genanntem „Z-Index") übereinander „schichten" [vgl. Keit00].

[1] "Most hypermedia models and systems do not incorporate time explicitly. This prevents authors from having direct control over the temporal aspects of a presentation." [Hard99]

Die (im Übrigen recht einfache) Syntax in diesem Zusammenhang näher zu erläutern, würde an dieser Stelle sicher zu weit führen – die „wirklichen interessante Neuerung“ [Rutl99], die im Gegensatz zu den großteils bereits bekannten Positions- und Skalierungsaspekten mit der Veröffentlichung des SMIL-Konzepts einhergeht, manifestiert sich ohnehin zweifellos in dem Aspekt *temporaler* Synchronisation:

> *Die meisten SMIL-Konstrukte beschreiben den zeitlichen Ablauf einer Präsentation, und auch der größte Teil einer in XML definierten Struktur der meisten SMIL-Dokumente ist dem Timing gewidmet. Das ist insofern angemessen, als der zeitliche Ablauf zu den Innovationen gehört, die SMIL Webautoren bietet.*
> [Rutl99]

5.5.2.1 Synchronisation und Intelligenz

So lassen sich im Rahmen der SMIL-Spezifikation auch prinzipiell *diskrete* Multimedia-Objekte [vgl. Stei99:10] wie etwa statische JPEG-Bilder oder Textelemente mit konkreten Zeitkriterien (Beginn, Dauer etc.) versehen und so als kontinuierliche Komponenten etwa mit Video- oder Audioelementen koppeln. Neben dieser Zeit-basierten Koordinationsmethode ermöglicht SMIL überdies die Synchronisation der einzelnen Medienelemente über zentrale `<par>`, `<seq>` und `<excl>`-Tags, die wahlweise die *parallele*, *sequentielle* (lineare) oder auch *exklusive*[1] Wiedergabe einzelner Medien oder auch (wiederum anhand derselben Syntax) beliebig verschachtelter Elementgruppen erlaubt. [vgl. Reid01:3].

Darüber hinaus ist jedoch auch die Event- bzw. Link-basierte Verknüpfung einzelner Elemente möglich: So können beispielsweise Anfang bzw. Ende der Wiedergabe eines Video-Elements konditionell über den Zustand weiterer Objekte (z.B. beim Ende eines anderen Mediums) gesteuert werden, was die Synchronisation im Gegensatz zum „starren“, zeitbasierten Koordinationsschema, in dessen Rahmen etwa Puffer- und Bandbreitenprobleme nur wenig berücksichtigt werden können, erheblich flexibler gestaltet.

Auf ebendiese Problematik geht unter anderem auch die letzte, so genannte „logische Synchronisation“ [vgl. Keit00] des SMIL 1.0-Modells ein: So ermöglicht das `<switch>`-Tag, ähnlich der gleichnamigen Kontrollstruktur bei prozeduralen Programmiersprachen[2] die „intelligente“ [Rutl99] Spezifikation entsprechend angepasster Alternativen, beispielsweise im Hinblick auf unterschiedliche Bandbreiten (`system-bitrate`). Leider erschöpfen sich diese Möglichkeiten neben der Bitrate auf einige, wenige Kriterien (deren entsprechende Argumente überdies von den SMIL-Clients nicht immer zufrieden stellend übermittelt werden), sodass auch dieser „innovative Ansatz“[3] in der Praxis als „allenfalls rudimentär umgesetzt“ [vgl. Arci02] bezeichnet werden muss.

5.5.2.2 No reason to SMILe: Probleme und enttäuschende Resonanz

Insgesamt erscheint die erste Auflage des SMIL-Standards dennoch, trotz einiger Schwächen[4] im Hinblick auf multimediale Präsentationen als in dieser Hinsicht dem (diesbezüglich ungeeigneten)[5] HTML-Format sowohl bezüglich „räumlicher“[6] als auch temporaler Kriterien deutlich überlegen. Da insbesondere die Streaming-gerechte Synchronisation, die das SMIL-Modell bereitstellt, in Verbindung mit der Integration weiterer Multimedia-Elemente die bislang gravierenden *Usability*-Probleme der Streaming-Branche [s.3.5.1] teils zu mindern vermochte, stellte sich insbesondere RealSystems durch eine recht frühe Integration des Standards in den G2-Player an die Speerspitze der SMIL-bewegung. Dem Software-Riesen Microsoft

---

[1] Anm: Dies bedeutet, dass die Wiedergabe „konkurrierender“ Medien während der Ausführung des <excl>-Tags werden unterbrochen bzw. pausiert wird.

[2] Anm: In C, C++, Java, JavaScript etc. existiert beispielsweise ein ebenso strukturiertes switch-Element.

[3] vgl. [Rutl99] p.58

[4] Anm: So sah der Standard etwa zunächst keinerlei *Animation* der Elemente vor [vgl. Arci02].

[5] s. hierzu Kap. 3

[6] Anm: Dieser Aspekt konzentriert sich auf die Vorzüge SMILs bezüglich Positionierung, Skalierung und Anordnung (Kachelung, Streckung etc.) von Elementen.

hingegen erschienen die Möglichkeiten der ersten *Recommendation* [SMIL98] noch „zu dürftig“ und „nicht ausgereift“ [Cove99], weswegen dem Standard der Weg in Microsofts WindowsMedia-Suite zunächst noch verwehrt blieb [vgl. Phil98:30]: Insgesamt, so merkt auch [Arci02] an, blieb das ursprüngliche SMIL-Format nicht nur wegen „mangelnder Interaktions-Eigenschaften“ [Bole98], sondern insbesondere aufgrund „eklatant fehlender“[1] Grundfunktionen wie Animation (z.B. Bewegung) einzelner Medienelemente oder interessanter Übergangseffekte (*Transistions*) hinter den Erwartungen der „Multimedia-Community“ zurück [vgl. Arci02].

Obgleich als „Multimedia-Format“ [BoEi02:113] in „Konkurrenz zum ungeliebten Flash“ [McCa02] bezeichnet, bietet SMIL aufgrund seiner primären Synchronisations- und Koordinationsfunktion [vgl. Rutl99] *sekundärer Medien* jedoch „direkt“ keine „eigenen“ Gestaltungsmöglichkeiten: So sucht man in der SMIL 1.0-Spezifikation [SMIL98] etwa nach Grafik-*Primitives* oder Typografie-Funktionalität vergebens: Insbesondere *Real* reagiert an dieser Stelle zwar mit einer proprietären Texterweiterung [vgl. Keit00, McCa02], von deren Verwendung [Arci02] aufgrund deren „Inkompatibilität“ jedoch dringend abrät.[2] Auch „intelligente“ Positionierungseigenschaften des Layout-Modells werden selbst von den AHM-Autoren selbst im Rahmen von SMIL 1.0 noch „schmerzlich vermisst“ [vgl. RHO99:171][3]

Aufgrund der daraus resultierenden, syntaktischen „Schlichtheit“ des Formates war zunächst hingegen eine manuelle, „einfache Erstellung überzeugender Multimedia-Präsentationen von Hand“ [Rutl99] noch durchaus möglich. Da ebendiese Tätigkeit „auf Dauer jedoch deutlich zu mühsam ist und erhebliche Zeit in Anspruch nimmt“ [Cove99], wurde schon recht bald die Forderung nach leistungsfähigen Autoren-Werkzeugen nach dem Vorbild des Marktführers *Flash* laut[4] – ein Ruf, der zwar „nicht ungehört verhallte“ [vgl. Zeld01], aber bislang dennoch nicht das erhoffte „Killer-Tool“ hervorbringen konnte.

Aufgrund dessen erstaunt es nur wenig, dass nach einem kurzen, aber kräftigen „SMIL-Hype“, der insbesondere im akademisch-technischen Bereich eine Vielzahl von Java-Implementierungen zur Folge hatte [MLL98, Bult98 ChYu98], aufgrund der soeben dargelegten Schwächen und Versäumnisse der Spezifikation alsbald schnelle Ernüchterung einsetzte: So blieben nicht nur die technischen Möglichkeiten SMILs hinter den Erwartungen zurück, auch die User-basierte Akzeptanz SMIL-basierter Anwendungen war insbesondere für deren Produzenten „äußerst ernüchternd“:[5]

> *After initial enthusiasm in multimedia circles developing kiosks and similar applications, [SMIL] virtually disappeared from people's attention, in favour of other technologies*
>
> [Arci02]

5.5.3 SMIL 2.0

Daher wich die bereits drei Jahre später veröffentlichte, *zweite* Fassung des SMIL-Standards [SMIL01] konzeptionell erheblich von den Grundprinzipien des Vorgängers ab – offensichtlich auf Druck des Microsoft-Konzerns, der eine *direkte* Implementierung des SMIL 1.0-Standards 1998 noch abgelehnt hatte [vgl. Phil98:30, Cove99], sich jedoch durchaus für einzelne Komponenten des Frameworks sowohl im Hinblick

---

[1] vgl. [Arci02]

[2] “Unless you have to produce SMIL 1.0 specifically for either platform, you should avoid such extensions for the sake of portability” [Arci02]

[3] “While this was able to represent all the placement […], the coding would have been more efficient if a mechanism for centering visual media in their assigned regions was available to SMIL processing. Without it, any centering would have to be explicitly calculated for each media object of different size.” [RHO99] p.171-172.

[4] “While SMIL’s simple syntax enables the easy text-only authoring of smaller presentations, the larger-scale maintaining of SMIL-based websites will require a more powerful tool” [Bult98]

[5] So zumindest der Kommentar Henner Henkels (Entwicklungschef der NetBroadcasting-Firma *surver:net* im Hamburg) über den ausbleibenden Erfolg der frühen SMIL-Aktivitäten (http://surver.net/broadcasting/smil [20.2.03]) seines Unternehmens (Interview am 1. September 2002 in Hamburg)

auf Browser- als auch Streaming-Integration[1] interessierte. Hauptmerkmal der so genannten „SMIL Boston"-Spezifikation [vgl. Schm00] war somit eine *Modularisierung* des Gesamtsystem in insgesamt 9 Komponenten, welche jeweils auch „einzeln" (im Gegensatz zu SMIL 1.0, das auf ein funktionelles Gesamtsystem ausgelegt war) implementiert werden konnten [vgl. Arci02]. Neben der bereits aus der ersten Version bekannten Funktionalität war die „SMIL-Suite" überdies um elementare Module erweitert worden: So eröffnete der SMIL2.0-Standard etwa mithilfe der neuen Animations- und *Transition*-Funktionalität [vgl. Acri02] speziell Web-basierten Präsentationen entsprechend der in [Kap.1] gegebenen Anforderungen neue Möglichkeiten und macht das Format somit auch im Rahmen dieser Diplomarbeit äußerst interessant:

> *A typical function of presentation authoring systems (e.g. Microsoft PowerPoint) supports simple transitions for bullet points on presentation slides… [The SMIL2.0] timing model for the web supports the creation of multimedia resentations that synchronize video and audio with sophisticatedanimated graphics, using an author-friendly syntax.*
>
> [Schm02]

5.5.3.1 Modularisierung: Zweischneidiges Schwert

Insbesondere die konzeptionelle Modularität des neuen SMIL-Modells trug nun zunächst im Zuge der Formulierung anderer Formate erstmals Früchte: So konnte das Animationsmodul des Standards [ScCo01] etwa in den (bereits in [5.4.2] diskutierten) SVG-Standard integriert werden; mit dem so genannten HTML+TIME-Modell [vgl. SYS98, Schm00] bemühte sich derweil insbesondere Microsoft um eine Adaption der Synchronisations- und Animationskomponenten im Rahmen von dHTML, was aufgrund der ausschließlichen Unterstützung des eigenen „Internet Explorer" jedoch aus Kompatibilitätssicht freilich nicht unproblematisch erscheint [s. hierzu 3.5.2]

Währenddessen trug jedoch das eigentliche, „integrale" Gesamt-Modell des SMIL-Standards im Zuge dieser Entwicklung erheblichen Schaden davon: Mit der dank Modularisierung und Funktionalitäts-„Explosion" gewaltig angewachsenen Komplexität des Frameworks stieg bedauerlicherweise auch die „allgemeine Verwirrung" [Arci02] hinsichtlich der Frage, was „SMIL denn nun sei": Das „schlichte"[2] Multimedia-Format im Sinne von SMIL 1.0? Eine Animationssprache? Oder eine Kollektion unabhängiger Module?

> *[This] confusion about terminology [lead to] serious image consequences for SMIL in the mind of many creative professionals.*
>
> [Arci02]

Doch nicht nur aufgrund dieser Unsicherheit schmolz die anfängliche Begeisterung [Cove99, Reid01, McCa02] hinsichtlich des „neuen" SMIL2.0-Standards schnell dahin: Auch ein offensichtlicher „Mangel an Business- oder gestalterischer Orientierung" [Arci02], die bei Betrachtung der Spezifikation, als auch der verschiedenen konkreten Implementierungen des Standards deutlich wird, ließ das Format für Geschäftskunden oder „die breite Masse des WWW" [vgl. Rutl99][3] schnell uninteressant werden: So weisen nicht nur die bislang „rein akademischen" Authoring-Tools wie das von den CWI-Forschern „selbstlos" (für rund 600 Euro) vermarktete GRiNS-System [Bult98], wie auch die bislang nicht sonderlich ästhetisch wirkenden, erzielten Lösungen darauf hin, dass „Design" im Rahmen des allzu technisch daherkommenden Smil bislang „keine große Rolle zu spielen scheint". [Arci02]

---

[1] Anm: Hierbei ist die Unterscheidung zwischen dem (D)HTML/HTTP-basierten, Browser-fixierten Ansatz und der massgeblich auf dem (dann SMIL-erweiterten) ASF-Format Ansatzes der „Windows Media" Streaming-Bemühungen kennzeichnend.

[2] vgl. [Arci02]

[3] „SMIL verspricht für interaktives Multimedia das zu leisten, was HTML für Hypertext getan hat: es in jedes Wohnzimmer zu bringen" [Rutl99]

5.5.4 Enttäuschende Vergangenheit, hoffnungsvolle Zukunft [Arci02]

Aufgrund dessen lässt sich an dieser Stelle zusammenfassen, dass der Standard (selbst in dessen neuer Fassung) zwar insbesondere in Form des „weitverbreiteten Real-Players" [vgl. Zeld01] Anwendug findet,[1] der derzeitige Funktionsumfang des Formates jedoch allenfalls in *Nischen* erfolgreich genutzt werden kann.[2] Im Rahmen auch *ästhetisch* (ergo emotional) *überzeugender* Web-Präsentationen erscheint das Format hingegen bislang nur wenig interessant, da insbesondere die direkten, gestalterischen Möglichkeiten SMILs derzeit recht bescheiden ausfallen. Dies könnte sich freilich mit einer vollständigen[3] Integration des diesbezüglich erheblich relevanteren SVG-Standards grundlegend ändern: Durch die *Kombination* der beiden Formate, so merkt unter anderem auch [Cagl01] an, ließen sich „endlich" überzeugende Präsentationen realisieren, die sowohl die grafisch beeindruckenden Möglichkeiten der *Scalable Vector Graphics* [s.5.4] nutzen, jedoch ebenso die Multimediale Funktionalität[4] des SMIL-Standards zur Anwendung bringen könnten. Technisch aufgrund der XML-Eigenschaft *beider* Formate längst möglich, scheitert der „Durchbruch" dieser „winning combination" derzeit freilich noch an der Browser-Unterstützung: So existiert mit „X-Smiles" [VRKH02] bereits jetzt eine funktionsfähige, Java-basierte Implementierung Form eines „akademischen" Prototypen [vgl. Watt02:1045] – bis auch verbreitete Browsertypen wie RealOne oder Microsofts MediaPlayer bzw. Internet Explorer diese hochinteressante Kombination unterstützen, dürfte hingegen noch einige Zeit ins Land ziehen.

Abschließend muss daher bedauerlicherweise festgestellt werden, dass SMIL, obgleich ursprünglich als „*der* Multimedia-Präsentationsstandard schlechthin" [vgl. BoEi02] angekündigt, zwar derzeit im Bezug auf Synchronisation und textuelle „Zusatz-Features" Streaming-basierter Videoübertragungen vereinzelt durchaus zur Anwendung kommt, im Hinblick auf dessen eigentlich interessantestes Potential, nämlich die Realisierung Web-basierter Präsentationen [s.1.1] bislang „der breiten Masse" [vgl. McCa02] großteils unbekannt geblieben ist und insbesondere aus gestalterischer Sicht eine derzeit traurige Randexistenz fristet. Aufgrund der „nicht sonderlich berauschenden Vergangenheit" des Formates [Arci02], sowie angesichts des nicht zu unterschätzenden *Potentials* hinsichtlich einer (auch Browser-unterstützten) SVG-Integration hoffen und vertrauen Multimedia-Experten daher inständig auf eine mögliche Trendwende und, vielleicht, den „zukünftigen Durchbruch dieses Standards" [vgl. Reid01]

5.6 Fazit

Nach Abschluss dieser recht umfassenden Betrachtungen präsentationsrelevanter Formate und Lösungen ergeben sich an dieser Stelle natürlich gleich mehrere Fragestellungen: Einerseits natürlich die Frage nach dem hinsichtlich unserer Kriterien [s.1.1] am ehesten geeignet scheinenden oder gar „optimalen" Präsentationsformat; Andererseits natürlich Frage, ob sich denn alternativ zu allen im Rahmen dieser Diplomarbeit bereits betrachteten Ansätze (ohne Anspruch auf Vollständigkeit) nicht noch *weitere, zukünftige* Formate eventuell bereits jetzt ankündigen, deren „Konturen" momentan zwar noch nicht erkennbar sind, welche aber eventuell für die Zukunft bestimmend sein werden; Und nicht zuletzt natürlich die sich auf Basis dieser Untersuchungen förmlich „aufdrängende" Frage nach der „logischen Konsequenz": Welche Schlussfolgerungen lassen sich aus diesen Erkenntnissen ziehen? Was könnten die „nächsten Schritte" sein, die sich aus diesen Resultaten ergeben?

---

[1] "Given the number of RealPlayers out there, SMIL can already reach almost as many web users as Flash does" [Zeld01]

[2] Anm: Speziell im Zusammenhang mit *Untertitel*-Synchronisation kann Smil derzeit bescheidene „Erfolge" feiern

[3] Anm: Dieser Vermerk ist insbesondere deshalb relevant, da der SVG-Standard die Animations-Teilkomponente SMILs [ScCo01] bereits längst beinhaltet – im Gegensatz hierzu ist mit einer „vollständigen" Integration jedoch die Integration des kompletten SMIL-Standards gemeint.

[4] Anm: Insbesondere sind an dieser Stelle freilich Sound, Animation und Übergänge (*Transitions*) relevant [vgl. Arci02].

5.6.1 Rekapitulation der Erkenntnisse

Nun, zu den ersten Fragen lässt sich rückblickend auf unsere Betrachtungen sicherlich folgendes feststellen: Nach der differenzierten Untersuchung insbesondere natürlich der bislang in den von der Fragestellung berührten Bereichen führenden Formate *PowerPoint* (für konventionelle „Offline"-Präsentation), HTML (Dokumenten- und Informationsbasierte Web-Präsentation) sowie *Director* (Offline-Multimedia) und *Flash* (primär grafische Web-Animation) muss, ohne an dieser Stelle bereits den letztendlichen Erkenntnissen vorgreifen zu wollen [s. hierzu Kap.6] festgestellt werden, dass diese trotz unbestrittener Vorzüge (insbesondere hier natürlich der jeweils „großflächigen" Verbreitung) aus verschiedenen Gründen dennoch im Hinblick auf Web-basierte (und –gemäße) Präsentationen entsprechend unserer im 1. Kapitel formulierten Anforderungen „eher problematisch" erscheinen.

Die diesbezüglich eindeutig interessantesten Formate, welche zugleich auch die von Seiten des Web-Konsortiums verabschiedeten Standards in diesem Bereich darstellen, sind nun einerseits das Vektorformat SVG, andererseits der Synchronisationsstandard SMIL. Da sich beide Formate jedoch kontinuierlich noch stark weiterentwickeln und insbesondere die hinsichtlich unserer Fragestellung hochinteressante *Kombination* der beiden Ansätze derzeit nur von einer Minderheit der Web-Clients unterstützt wird, ist die Erkenntnis, dass insbesondere der SVG-Standard aus konzeptioneller Sicht das für unsere Zwecke wohl „am ehesten geeignete Format" darstellt, natürlich keine „pragmatische", sondern vielmehr eine theoretische und zukunftsorientierte Schlussfolgerung. Ebenso gilt diese Feststellung für den aspekt „ganzheitlicher Lösungen": So existiert für das noch recht junge SVG-Format zwar bereits eine ganze Reihe so genannte „Präsentationslösungen" [Arm01, Behz02, HSL00, Sue02, Herm02] – diese verfügen jedoch im Gegensatz zu entsprechenden Tools der „etablierten" Formate bislang über keinerlei komfortable Benutzerschnittstelle – von Usability oder Design-Orientierter GUI-Hilfestellung[1] ganz zu schweigen.

5.6.2 Konsequenz und Umsetzung

5.6.2.1 Die Master-Lösung: Möglich, nötig und sinnvoll?

Um damit auf die ursprüngliche Zielsetzung dieser Diplomarbeit zurückzukehren: Natürlich ist nicht nur die Feststellung der Eignung verschiedener Formatansätze unter diesem Gesichtspunkt relevant – auch die Frage, „was man damit machen kann", gilt es auf Basis dieser Erkenntnis an dieser Stelle zu beantworten: So könnte etwa ein möglicher Ansatz (dem auch eine frühe Konzeptstudie dieser Diplomarbeit zugrunde lag) aus einer (möglicherweise serverbetriebenen) „Master-Lösung" bestehen, die auf Basis „interner Präsentationsdaten" in der Lage ist, mithilfe entsprechender Konvertierungsmodule stets „formatoptimierte" Ergebnisse zu erzielen. Nach Auswertung der im Verlaufe der verschiedenen Formatuntersuchungen gewonnenen Erkenntnisse erscheint dies aufgrund der Tatsache, dass das Gros der soeben betrachteten Formate unterschiedliche Zwecke verfolgt[2] sowie konzeptionell und strukturtechnisch durchaus inhomogen aufgebaut ist, derzeit nur wenig sinnvoll. Die Umsetzung einer solchen Lösung stellt überdies aufgrund der im Verlaufe der Untersuchung festgestellten, universellen Verfügbarkeit jeweils geeigneter Konvertierungsmodule (zumeist auf der gemeinsamen Basis *Java*) im Großen und ganzen lediglich eine handwerkliche „Puzzle-Arbeit" dar,[3] die zwar arbeitsintensiv, aber aus konzeptioneller oder gar „wissenschaftlicher" Sicht keine sonderliche „Herausforderung" für diese Diplomarbeit bedeuten würde.

---

[1] Anm: Hiermit ist etwa ein (grafisches) „Framework" gemeint, dass den Nutzer etwa diskret dazu „zwingen" (bzw. ihm dabei helfen) könnte, Präsentationen nach vernünftigen, ästhetischen Kriterien zu produzieren.

[2] z.B. SMIL zur Koordination und Synchronisation beliebiger Multimedia-Elemente (ohne direkte schrifttechnische oder grafische Gestaltungsfunktion) bzw. andererseits SVG, welches ein konkretes Vektorformat darstellt.

[3] Die einzelnen Filter-/Konvertierungsmodule der jeweiligen Formate müssten einfach nur „zusammengesetzt" und angepasst werden.

Überdies würde insbesondere eine (aus „pragmatischer", also verbreitungstechnischer Sicht im Prinzip ja am allerehesten sinnvolle)[1] *Konvertierung* des konzeptionell „sauberen" SVG in das SWF-Format sowohl ein „Mapping" einer „hohen" XML-Vektorsprache in das eher animationsorientierte, jedoch wenig webfähige Binärformat SWF und damit einhergehend einen deutlichen Verlust Web-gemäßer Strukturinformation darstellen, als auch das „Squeezing-In" zahlloser SVG-Features (wie etwa Filter, Schlagschatten etc.) in das diesbezüglich „wenig intelligente" Flash-Format in eine wahre „Workaround-Hölle" [Balo00] ausarten könnte.

5.6.2.2 Zielformulierung

Aufgrund dessen habe ich mich (wie bereits in [5.4] ausgeführt) entschlossen, eine „konzeptionell saubere", zukunftsgerichtete Lösung auf Basis des SVG-Formates neu zu entwickeln. Ziel der Übung ist somit die Umsetzung eines funktionellen Prototyps, der auf dieser Grundlage die Erstellung und Darstellung „dramaturgisch strukturierter, webgemäßer Präsentationen" [s.1.1] ermöglicht. Obgleich angesichts möglichst niedriger Anforderungen hinsichtlich User-Kompetenz [vgl. Scha90] das zentrale Designkriterium dieser Präsentationslösung naturgemäß *Schlichtheit* lautet, sollen im Zuge der (konzeptionellen) Umsetzung jedoch zunächst nicht Usability-Aspekte (im Sinne der Bedienbarkeit etwa eines PowerPoint-GUIs), sondern die „maximal triviale" *Struktur* der entsprechenden Daten-Hierarchie und, daraus abgeleitet, des Präsentationssystems selber im Vordergrund stehen – schließlich konzentriert sich diese Arbeit primär auf Aspekte der *Ästhetik* und *Informatik* im Sinne von konzeptionell „sauberen" Ergebnissen, denn auf personenbezogene Ergonomie.

[1] Anm: Ein derartiges Modul wäre freilich allein deshalb interessant, weil sich die grafisch beeindruckende Funktionalität des (wenig verbreiteten) SVG auf das weit verbreitete Flash-Darstellungssystem abbilden ließe.

6 Konzeption des Prototypen

Nach der im vorangegangenen Kapitel diskutierten[1] Erkenntnis, dass sich insbesondere der XML-basierte Vektorstandard SVG im Hinblick auf eine mögliche, web-basierte Präsentationslösung hervorragend eignet, stellt sich natürlich angesichts eines „konkreten Realisierungswunsches" die Frage, wie dieses Format denn nun im Rahmen einer strukturell überzeugenden und konzeptionell möglichst „schlichten" Lösung [s.5.6] *nutzbar* gemacht werden kann: Eine *direkte* Anwendung erscheint aufgrund des derzeitigen Fehlens eines überzeugenden SVG-Präsentations-Werzeugs (so wie es etwa PowerPoint oder Flash darstellen) derzeit nicht möglich: Eine textbasierte Erstellung über den SVG-Quellcode wäre an dieser Stelle sicherlich zu aufwendig, zu komplex und der entsprechenden (PowerPoint-)Zielgruppe [vgl. Scha90] definitiv *nicht* angemessen.

Neben dieser hinderlichen Komplexität erscheint auch eine bereits in [5.4.8] diskutierte Eigenschaft des SVG-Standards im Hinblick auf die gewünschte Realisierung problematisch: So ist es selbst bei konsequenter Anwendung des CSS-Standards [CSS98] im Rahmen von SVG derzeit nicht möglich, einen entsprechenden *semantischen Kontext* bzw. eine logische Bedeutung in der SVG-Datei enthaltener Textinformationen abzubilden und somit auch für Internet-Suchmaschinen oder weitere, „intelligente" Verarbeitungssysteme nutzbar zu machen: Die primär grafisch-visuelle Ausrichtung des Standards und die derzeit fehlende Integration entsprechender Vorschläge [Herl99, Badr01, MaFu01] standen der Möglichkeit, hierarchische Strukturen, logische Verknüpfungen oder semantischen Kontext im Rahmen von „Basic SVG" abzubilden, bislang im Wege. [s.5.4.8]

6.1 Formatseparation

Der hochinteressante *Catwalk*-Ansatz Philip Mansfields, welcher die Einbeziehung externer XML-Strukturen sowie deren direkte Umwandlung in visuelle SVG-Grafiken ermöglicht [vgl. MaFu01, Mans01, Fibi01:14] weist uns an dieser Stelle jedoch den rettenden Ausweg aus diesem Dilemma und legt zugleich eine *Auftrennung* des gewünschten Präsentationskonzeptes in zwei verschiedene Komponenten nahe: Während das SVG-Format einen Grossteil der bereits im 1. Kapitel angesprochenen Anforderungen hinsichtlich Transparenz, visueller und multimedialer Funktionalität sowie „Scriptability" äußerst zufrieden stellend zu erfüllen vermag, bedarf es zur Formulierung zusammenhängender Inhalte auf einem höheren Abstraktionsniveau nun der Einführung eines *weiteren* Formates, das auch die hierarchische, dramaturgische Strukturierung verschiedener Präsentationsdaten erlaubt. Dem unter [5.3.4] besprochenen Grundgedanken der „high-level"-Definition abstrakter Diagrammdaten durchaus nicht unähnlich, konzentriert sich unser Interesse in diesem speziellen Fall somit auf ein Format, dass einerseits zur Wahrung maximaler Schlichtheit[2] möglichst trivial formulierbar („Minimal-Komponente"), jedoch zugleich ebenso in der Lage sein sollte, die wichtigsten inhaltlichen Aspekte eines Großteils derzeit noch mit herkömmlichen Methoden erstellter Präsentationen abzubilden.

Aufgrund der bereits in [5.1] festgestellten, überzeugenden Funktionalität der XML-Sprachkonvention [vgl. XML98] hinsichtlich textbasierter Internetdaten lag bei der Recherche nach einem entsprechend der soeben formulierten Anforderungen möglichst geeigneten „Minimal"-Format der Fokus naturgemäß auf der näheren Betrachtung XML-konformer Präsentationsformate. Und in der Tat weisen zahlreiche Veröffentlichungen namhafter Grafiksoftware- und IT-Unternehmen auf ein breites Interesse an einem derartigen, „trivialen" XML-Präsentations-Framework hin. Neben Sun Microsystems, die bereits im Rahmen des derzeit wohl meistgelesenen XML-Tutorials [Arm01] eine PowerPoint-verwandte Slide-Dokumentenstruktur nahe le-

[1] s. Kap. 5, insb. 5.4 sowie 5.6

[2] Anm: Der treffendere Begriff wäre an dieser Stelle das englische "Simplicity" (bzw. *Einfachheit*)

gen,[1] stellt interessanterweise auch Macromedia für seine stark XML-orientierte MX-Architektur ein Framework bereit,[2] mit dessen Hilfe sich XML-basierte Daten auf ein zuvor (freilich wiederum mit Flash MX) erstelltes *Template* anwenden und somit dynamische Flash-Präsentationen erzeugen lassen [Behz02].

```
<slideshow
 title="Sample Slide Show" date="08/12/2002" author="Yours Truly">
    <!-- TITLE SLIDE -->
    <slide type="all">
      <title>Wake up to WonderWidgets!</title>
    </slide>
    <!-- OVERVIEW -->
    <slide type="all">
        <title>Overview</title>
        <item>Why <em>WonderWidgets</em> are great</item>
        <item/>
        <item>Who <em>buys</em> WonderWidgets</item>
    </slide>
</slideshow>
```

Listing 6.1.1: Beispiel-Daten gemäß der Slideshow-DTD von Sun Microsystems' XML-Tutorial [Arm01]

```
<Presentation>
  <slide type="0">
    <title>Inkless Pens</title>
    <description>...description>
    <owner>Bob Quill</owner>
    <background>bg1.swf</background>
  </slide>
  ...
  <slide type="1">
    <line>NEXT STEPS:</line>
    <line></line>
    <line>Need to raise $17 million. )</line>
    <line>($1M development / $16M marketing)</line>
    <line>Partnership with distributors.</line>
    <line>Hire off-shore developers.</line>
  </slide>
  <slide type="2">
    <picture>graph1.swf</picture>
  </slide>
  ...
</Presentation>
```

Listing 6.1.2: Beispiel-Daten der Macromedia-DTD. Abb.6.1.1-2: Darstellung der Daten aus [Listing 6.1.2]

Insbesondere auf Basis des für unsere Zwecke interessantesten Formates SVG existieren „unzählige" [LiJa03] Ansätze seitens der W3C-Entwicklerschaft, konventionelle Präsentationsdaten mittels XML-basierter SVG-Syntax abzubilden. Das enorme Potential des SVG-Standards hinsichtlich Internetbasierter Präsentationen haben hierbei zweifellos die Web-Konsortium sehr aktiven Entwickler Vincent Hardy, Paul Sandoz und Ana Lindstrom-Tamer von SVG-Mitglied Sun Microsystems als erste erkannt und in ihrem Präsentationstool „SVG Slide Toolkit" bereits vor der Standardisierung des SVG-Formates konsequent verarbeitet [HSL00]. Interessanterweise fußt auch diese Implementierung jedoch nicht ausschließlich auf SVG-eigener Syntax: Entsprechend der bereits ausgeführten Überlegungen haben auch die Sun-Entwickler die Funktionalität des Präsentations-Toolkits in sowohl einen visuellen SVG-Teil als auch einen abstrakteren Definitionsteil separiert, wobei letzterer naturgemäß ebenfalls entsprechend der XML-Konvention formuliert ist:

```
<!DOCTYPE slideshow SYSTEM "slides.dtd">
<slideshow>
       <title>Slideshow Title</title>
```

[1] DTD-URL: http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/sax/samples/slideshow1b.dtd [27.1.03]
[2] XML-URL: http://www.macromedia.com/desdev/mx/flash/articles/flash_presentations/presentation.xml [27.1.03]

```
      <metadata>
            <speaker>Speaker Name</speaker>
            <jobTitle>Job Title</jobTitle>
            <organization>Organization</organization>
            <presentationDate>YYYY-MM-DD</presentationDate>
            <presentationLocation>Presentation Location</presentationLocation>
            <occasion>Occasion</occasion>
      </metadata>
      <slideset>
            <title>Item Drill Down Test</title>
            <slide id="slide9">
                  <title>Item and flat list</title>
                  <item>Item with flat list
                        <flatlist>
                              <item>Item 1.1</item>
                              <item>Item 1.2</item>
                        </flatlist>
                  </item>
            </slide>
            ...
      </slideset>
</slideshow>
```

Listing 6.1.3: Beispiel-Daten gemäß der Slideshow-DTD von Sun Microsystems' SVG Slide Toolkit [HSL00]

Neben den zahlreichen, weiteren SVG-Slide-Tools, die nach ähnlichem Prinzip wie dem oben genannten funktionieren, ist an dieser Stelle noch der Ansatz der University of Queensland, „JackSVG" erwähnenswert [vgl. Sue02]. Bei diesem relativ neuen SVG-Präsentationsprogramm wird jedoch anstatt der bei nahezu sämtlichen anderen Lösungen zur Anwendung kommenden XML-Transformationssprache XSLT[1] ein *Perl*-Server benötigt, um die abstrakte XML-Logik, die die eigentlichen Präsentationsdaten abbildet, in darstellbare SVG-Syntax zu verwandeln. Ebendiese „interne" XML-Syntax des JackSVG-Ansatzes ist jedoch, nicht zuletzt aufgrund ihrer Überschneidungen mit der (mittlerweile übrigens in das *Batik SVG Toolkit* [Croo01] übergegangenen)[2] *Sun*-Konkurrenz [HSL00] interessant – so lassen sich nicht nur strukturelle Parallelen (wie etwa die obligatorische Unterteilung in einzelne „Slides") erkennen, auch die Syntax deckt sich mitunter in frappierender Weise:

```
<presentation version='1.0' ...>
    <metadata name="title">Introducing JackSVG</metadata>
    <metadata name="subtitle">Presenting presentations</metadata>
    <metadata name="version">1.0</metadata>
    <metadata name="author">Hoylen Sue</metadata>
    <metadata name="event">JackSVG</metadata>
    <metadata name="date">10 Dec 2002</metadata>
    ...
    <group>
        <title>JackSVG</title>
        <section>
            <title>What is it?</title>
            <slide>
                <title>Not another SVG Presentation Tool!</title>
                <point>
                    <para>Everyone creates a SVG slide generator</para>
                         <point>
                             <para>It's the "Hello World" of SVG!</para>
                        </point>
                        ...
                    </para>
                 </point>
            </slide>
        </section>
    </group>
```

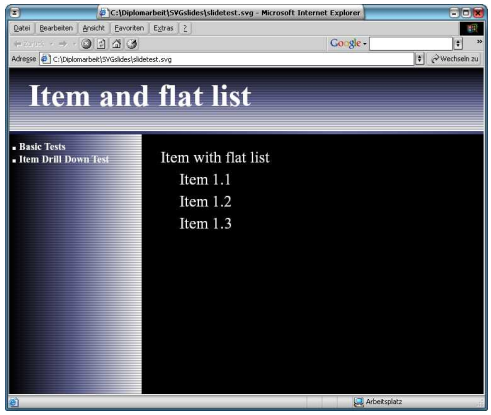
[1] s.5.1
[2] vgl. [Lill99]

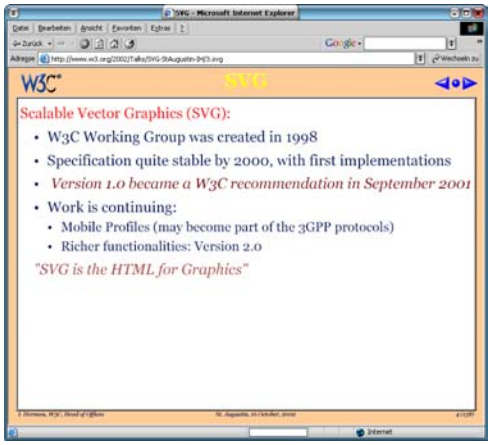
```
</presentation>
```

Listing 6.1.4: Beispiel-Daten gemäß der Slideshow-DTD von JackSVG *der Universität Queensland* [Sue02]

Dennoch kann die Lösung der University of Queensland als dem SVG-Slide-Tookit durchaus überlegen eingeschätzt werden – nicht zuletzt, weil in diesem Ansatz überdies das nicht uninteressante Feature der so genannten *Skins* zum tragen kommt [vgl. Grip03], mit dessen Hilfe sich, den PowerPoint'schen Templates nicht unähnlich, jeweils individuelle „Look-and-Feels" [1] der einzelnen Präsentationsdarstellungen realisieren lassen.

Die wohl ausgefeilteste Definition einer Slide-basierten Präsentations-DTD stellt jedoch zweifellos die *SlideML*-Spezifikation des Zürcher Informationsdienstleisters *Bitflux* dar [Fisc02]: Die umfangreiche Spezifikation, die im Hinblick auf eine „universelle Präsentationslösung" der OpenSource-Initiative „OSCOM" entwickelt wurde, bildet das „PowerPoint'sche Präsentationsmodell" recht exakt auf einer soliden XML-Grundlage ab und bezieht sogar das „Dublin Core"-Modell [vgl. OyGo01] zur Definition der „Header"-Daten mit ein. Da die Spezifikation überdies intensive Standardisierungs-Bemühungen in Richtung des W3C erkennen lässt, gilt es für die Zukunft sicherlich, diesen interessanten Ansatz nicht aus den Augen zu verlieren.





Als eines der wenigen diesbezüglichen Anwendungen, die auf eine eigene, „minimale" XML-Syntax verzichten, erscheint an dieser Stelle noch der Ansatz des W3C-Niederlande-Chefs Ivan Herman von Interesse, welcher stattdessen ein recht komplexes XSLT-Regelwerk anbietet [vgl. Herm02]. Neben der erfreulichen Aktualität dieses Frameworks[2] sei trotz der erhöhten Komplexität durch die reine XSLT-Anwendung die hiermit einhergehende, Flexibilität dieser Herangehensweise erwähnt, die im vergleich zum restriktiven Ansatz der beiden oben genannten Alternativen jedoch eine weitaus höhere gestalterische und technische Kompetenz voraussetzt.

Abb. 6.1.2.1-3: Präsentationsausgabe der SVG-Slide-Tools [HSL00], [Sue02] sowie [Herm02] (v.o.n.u)

Die Tatsache hingegen, dass sich auch diese Ansätze durchweg an der strikten Slide- und „Bulletpoint"-basierten Strukturierung PowerPoints orientieren, enttäuscht jedoch die Hoffnung, durch Separation des Inhaltes von der Darstellung zugleich eine strukturelle Alternative aufzeigen zu können – im Gegenteil: Die DTDs einiger „trivialer" Ansätze [Arm01, Behz02] stehen hinsichtlich ihrer Strukturierungsmöglichkeiten gar noch hinter denen PowerPoints zurück. Während die in [2.3.1] eingeführte *Gliederungs*-Funktionalität des Microsoft-Programms durchaus verschachtelte, hierarchische Bullet-Strukturen erlaubt, reduziert ein Grossteil der vorgefundenen XML-Ansätze die Schachtelungstiefe dieser Stichwort-Bäume sogar noch auf praktisch rein lineare Stichwortlisten:

[1] Details hierzu siehe [Grip03]

[2] Die letzte Änderung erfolgte noch im Januar dieses Jahres. [vgl. Herm02]

> *[Every] template insists on a heading followed by bullet points, so that the user is shepherded toward a staccato, summarizing frame of mind…*
>
> [Park01:1]

Die unter [2.3.2.1] sehr ausführlich diskutierte, inhaltlich-intellektuelle Kritik an PowerPoint-diktierter Dramaturgie und Strukturierung [vgl. Stew01] könnte somit durch die eben besprochenen Ansätze, auch wenn sich hierdurch die eingangs geforderte Trennung von Inhalt und Darstellung erzielt würde, nicht behoben werden, da die inhaltliche Struktur der jeweiligen Dokumenttypen Grundgedanken und Architektur des PowerPoint-Vorbilds lediglich haarklein nachzubilden versucht. Auch die primär „Designorientierten" Präsentations-Alternativen wagen es nicht, die scheinbar etablierte PowerPoint-Architektur zu durchbrechen. So bietet etwa Apples jüngst veröffentlichte Präsentationssoftware „Keynote" [Appl03] mitunter durchaus interessante Ansätze, zumindest die gestalterischen Aspekte PowerPoint-basierter Präsentationen durch professionelle „Themes" etwas aufzuhellen [s.Abb.6.1.4], ordnet sich zugleich jedoch dem durch PowerPoint vorgegebenen Anwendungsprinzip und Präsentationsansatz nahezu sklavisch unter. Die *Keynote* zugrunde liegende, ebenfalls XML-basierte Dateistruktur APXL[1] stellt darüber hinaus gar noch nicht einmal eine „Minimal"-Komponente wie zuvor definiert (und durch die soeben beleuchteten XML-Alternativen auch implementiert) dar, sondern bettet, ebenso wie PowerPoint, auch sämtliche „diskrete" Bildelemente direkt („*inline*") in die Präsentationsdatei ein – und diese, wie die XML-Community „entsetzt" feststellen musste [vgl. Ogbu03], unverständlicherweise „noch nicht einmal auf Basis des SVG-Standards".[2]

Abb. 6.1.4: Die Präsentationssoftware Keynote von Apple.

Eine „echte Alternative" zur problematischen PowerPoint-Architektur stellen nach diesen Betrachtungen somit weder die derzeit verfügbaren, strukturellen XML-Ansätze [Arm01, Behz02, HSL00, Sue02, Herm02], noch die aus „ästhetischen Beweggründen" entwickelten Konkurrenzprodukte [Appl03] dar: Obgleich insbesondere die SVG-basierten Alternativen interessante Herangehensweisen hinsichtlich Struktur und Format aufweisen, sowie Apples „Keynote" durch optische Eleganz durchaus zu beeindrucken weiß, kranken sowohl die (zumeist recht „akademischen" und somit praktisch nur durch IT-Kräfte benutzbaren) XML-Varianten wie auch das Consumer-Pendant von Apple an erheblichen Usability-Problemen [vgl. Frie03]. Des weiteren macht auch die Analyse der streng Slide-basierten XML-Struktur[3] bzw. des entsprechenden Anwenderprinzips (bei „Keynote") den fehlenden Impuls der soeben angesprochenen Konkurrenzprodukte deutlich, sich *wirklich* von den etablierten, „leidigen" [Sear98] Konventionen der durch PowerPoint geprägten Präsentationskultur zu emanzipieren und eine auch in dieser Hinsicht „echte Alternative" anzubieten.

Diese Erkenntnisse haben im Verlaufe dieser Diplomarbeit daher zur Entwicklung eines eigenen, entsprechend der eingangs ausgeführten Überlegungen inhaltlich separierten Präsentationsansatzes geführt, der sowohl aus Präsentationssicht zumindest die meisten der an moderne, Internet-basierte Präsentationen gestell-

[1] Syntax-Beispiel: http://paulboutin.com/presentation.apxl [19.2.03]
[2] vgl. [Ogbu03]
[3] s. Listings 6.1.3, 6.1.4

ten Anforderungen[1] erfüllen, jedoch zugleich auch technisch und inhaltlich eine wirkliche Alternative zur (wie in diesem Kapitel aufgezeigt) relativ uniformen Präsentations-„Norm“ aufzeigen sollte.

6.2 Strukturierung und Verzeichnis-Metapher

Den wohl interessantesten Hinweis auf einen möglichen „Ausweg“ aus diesem strukturellen Dilemma liefert frappierenderweise ein bereits in [2.1] angesprochener Ansatz aus den frühen 80er Jahren: In [Wine88] weist etwa Software-Pionier Dave Winer auf die Bedeutung strikt hierarchisch gegliederter Datenstrukturen für geschäftliche Präsentationssoftware hin:

> *This very early piece of software met the two criteria that make something an outliner: you can control the level of detail in the display of information, and you can reorganize according to the structure of the information.*
>
> [Wine88]

Winers dazugehörige Software MORE, obgleich Mitte der 80er Jahre ein „großer Erfolg“,[2] verschwand zwar in den 90er Jahren „merkwürdigerweise wieder von der Bildfläche“ [Wine99] – seine Ausführungen zur Konzeption und Erstellung gesamter Präsentationen ausschließlich über strukturierte Gliederungs-GUIs inspirierten mich jedoch zur Umsetzung einer „verrückten Idee“: Aufgrund der Tatsache, dass die gesamte Präsentation – bereits bei relativ linearer Ausführung nach dem „Headline-Bullet-Staccato“-Prinzip [vgl. Park01:1] – einer (wenn auch zunächst recht flachen) Baumstruktur folgt, ließe sich dieser Gedanke, insbesondere bei zunehmender „Verschachtelung“ der Präsentationsdaten, fortspinnen zu einem in seiner Gesamtheit Baum-orientierten Präsentationsansatz. Hierbei, so meine ursprüngliche Idee, könnten nicht nur die einzelnen „Bullets“ (i.e. Stichworte) als *Blätter* des Baumes, sondern ebenso Überbegriffe, Kapitel, und schließlich die Präsentation selber (als „Wurzel“ des Baumes) durch interne bzw. auch externe Baumknoten realisiert werden.[3]

Da der Fokus meiner Arbeit jedoch auf multimedialer Präsentation liegen sollte und somit die eigentliche Visualisierung derart strukturierter Daten im Mittelpunkt meines Interesses steht, führten meine Überlegungen alsbald zu der wohl naheliegendsten Umsetzung einer Baum-Darstellung, nämlich der durch die weite Verbreitung des Windows-Explorers hinlänglich bekannten Visualisierung hierarchischer Datei- und Verzeichnsdaten in der gleichnamigen „Tree“-Ansicht:

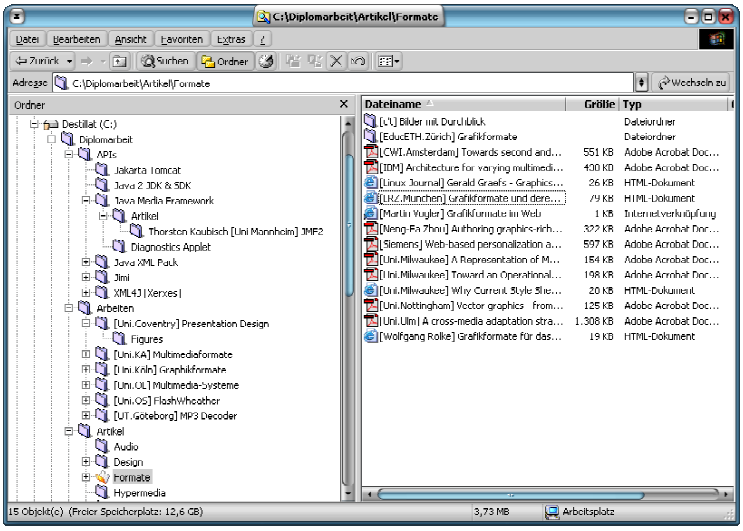


Abb. 6.2.1: Windows-Explorer mit Tree-View (Linkes Teilfenster)

> *The expanding and collapsing file system viewer first appeared in the Macintosh System 7, and now has become a common feature of all file system browsers.*
>
> [Wine99]

Aufgrund der Bekanntheit dieser Baum-Ansicht und der daraus abgeleiteten, allgemeinen Vertrautheit mit der zugehörigen Verzeichnis-Metapher [vgl. Fors98:53f] lag aus meiner Sicht somit eine Adaption dieser

---

[1] s. 1.1

[2] “Initially, the sales […] were incredible” [Wine88]

[3] vgl. hierzu die Baum-theoretischen Überlegungen in [Walt00] pp.32ff.

Metaphorik auf die Visualisierung allgemeiner Präsentationen durchaus nahe. Umso interessanter erscheint es daher, dass sich der „File-Begriff“ auch in den früheren Ausführungen Winers wiederfindet – wenn auch in einem leicht veränderten Zusammenhang:

> *The result [was an] outliner including a database, called High Files – because I thought of it as a high-level filing system…*[1]

6.3 Umsetzungsspezifische Überlegungen

6.3.1 *Icons* als illustratives Präsentations-Element

Während Winer den eigentlichen Visualisierungsaspekt dieses Gedankens jedoch eher als „unbeabsichtigte Randerscheinung“ betrachtet,[2] konzentrierte ich mich im Folgenden freilich auf die unter diesem Aspekt noch zu beantwortenden Fragen, insbesondere: Wie ließe sich die eher funktionelle Verzeichnis-Ansicht des Explorer entsprechend den Forderungen Seth Godins [Godi01] nach mehr „Emotionalität“ in Präsentationen entsprechend erweitern? Die Lösung hierzu lag freilich wiederum durch eine Eigenschaft GUI-basierter Betriebssysteme auf der Hand, nämlich die Verwendung illustrativer *Icons* zur Visualisierung der Ordnerinhalte (bzw. hier: Der Inhalte der eigentlichen Präsentationsdaten), sowie optionale „Fullscreen“-Bildintegration zur Ermöglichung von Schaubildern und „emotionaler Illustration.“ Wie bereits in [Fors98] erwähnt,[3] sind insbesondere *Icon*-Elemente zu verblüffenden Steigerungen sowohl der kognitiven Aufmerksamkeit als auch der emotionalen Affirmativität des Benutzers fähig, da durch die Einbindung dieser (wenn auch kleinformatigen) Bildelemente offensichtlich die Gefühlsbetontheit und „Explorativität“ des Dargestellten zum Präsentationseindruck insgesamt positiv beiträgt.

Dennoch gehen insbesondere mit der Übertragung dieser Icon-Eigenschaft aus dem eher GUI-fixierten Bereich der Verzeichnisverwaltung auf ein Präsentationssystem, speziell unter Berücksichtigung der besonderen „Wichtigkeit des Icon-Designs“ [vgl. Fors98:59] weitere Probleme einher: Einerseits schien mir die „ViewBox“, also die Größe des im Windows-*Explorer* zur Verfügung stehenden Ansichtsbereichs auf die einzelnen Ordner sowie die Ausmaße der zugehörigen Icons, zwar für Dateiverwaltungs- und somit logischerweise auch im Rahmen meiner angestrebten Lösung zu Bearbeitungszwecken der Gliederungs-Datenstruktur durchaus geeignet – *nicht* jedoch für den eigentlichen Zweck meines geplanten Ansatzes, nämlich der visuellen *Präsentation* der Daten. Auch bei einer (theoretischen) Verwendung der von Windows wie auch aller anderen Systeme neben der Miniatur-Ansicht[4] bereitgestellten, „großen“ Symbole wäre sowohl die Bildauflösung der Icons (mit 32·32 Pixeln) als auch die entsprechende Schriftgröße der Beschriftungen (und somit der eigentlichen, textlichen Inhalte der Präsentation) zur vollformatigen Bildschirmpräsentation völlig unzureichend, geschweige denn im Rahmen eines *Beamer*-projizierten Vortrages auch aus größerer Entfernung lesbar.

Abb. 6.3.1: Verschiedene Icon-Auflösungen (16x16 bis 128x128)

Aus diesem Grund boten sich im Rahmen eines SVG-basierten Präsentationskonzepts[5]

[1] vgl. [Wine88]

[2] “It only made sense that we should go the next step, and print the outline in presentation form – [but] I've always felt that graphics products … were too low-level to be useful to word and concept people” [Wine88]

[3] vgl. [Fors98] pp. 53ff.

[4] Anm: In der Regel sind dies 16x16 Pixel

[5] s. 5.4.4

an dieser Stelle gleich zweierlei Lösungsalternativen an: Sowohl die für die neueren Betriebssysteme MacOS X und Windows XP angebotenen, hochauflösenden[1] (jedoch freilich weiterhin Pixel-orientierten) Icon-Sets verschiedener Anbieter,[2] als auch entsprechende Konverter-Algorithmen des französischen ESSI-Instituts [Vant02] bzw. der Universität Tsukuba [Mori99, Mwt99], welche pixelbasierte Bilddaten in Bézier-Kurven überführen und automatisch in der Form vektor-basierten SVGs ausgeben können. Der letztgenannte Algorithmus scheint somit auf den ersten Blick hinsichtlich einer konsequent Vektor-orientierten Präsentationslösung am ehesten geeignet – aufgrund der leider noch recht fehlerhaften Implementierung[3] dieser Vektorisierungs-Ansätze kommen jedoch im Rahmen des *XML » SVG Presenter*s noch direkte Pixel-Daten in Form PNG-basierter[4] Icon-Dateien zum Einsatz. Dieser Umstand ist erstlinig der großen Auswahl frei verfügbarer (und somit von den potentiellen Nutzern des *Presenter*s direkt nutzbarer), durchaus ästhetischer Icon-Dateien [MTH97], welche der illustrativen bzw. gestalterischen Funktion, die den Icons im Rahmen der Präsentation zukommt, mehr als genügen, sowie der bereits in der 3. Auslieferung des *Adobe SVG Viewer*s im Gegensatz zu der kümmerlichen Pixelgrafik-Ausgabe des *Flash-Players* als hervorragend zu bezeichnenden Interpolations- und Anti-Aliasing-Funktionalität hinsichtlich der Darstellung auch pixel-basierter Rastergrafiken zu verdanken.

6.3.2 Veichnis-Metapher: Vertraut oder (noch) ungewohnt?

Neben dieser rein darstellungs-technischen Frage sind im Rahmen der Realisierung einer angedachten Präsentationslösung freilich überdies Überlegungen hinsichtlich der vermutlichen *Ungewohntheit* des (Ziel-)Publikums gegenüber der übertragenen Metaphorik relevant: So kann zwar von einer gewissen Vertrautheit mit der Verzeichnis-Methapher des Windows-Explorers an sich ausgegangen werden,[5] die Übertragung dieser Symbolik auf ein Präsentationsumfeld, welche ja Grundlage des *XML » SVG Presenter*s darstellt, könnte an dieser Stelle hingegen durchaus Schwierigkeiten bereiten. Insbesondere die Tatsache, dass im Gegensatz zum vertrauten Explorer-Prinzip inhaltliche Daten und Stichpunkte der Präsentation[6] *direkt* als Endknoten des Baumes auftreten statt, wie aus der Verzeichnisumgebung gewohnt, innerhalb von Dokument-Dateien (die dann wiederum in den einzelnen Ordnern enthalten sind)[7] könnte im Rahmen einer Präsentation für Verwirrung sorgen. Versuche an verschiedenen Test-Zuschauergruppen haben jedoch gezeigt, dass sich die überwiegende Mehrheit des Publikums nach anfänglicher Ungewohntheit schnell in die „neue" Umgebung einfindet, wohl nicht zuletzt deswegen, weil die Vertrautheit mit dem bereits bekannten Explorer-Prinzip diese Unsicherheiten offensichtlich überwiegt.

Des Weiteren konnte insbesondere die Entwicklung zweier unterschiedlicher Präsentationsmodi zur „Entschärfung" dieses wohl nur in der Theorie befürchteten Effekts beigetragen: Indem die Präsentationsstruktur sowohl mit der gewohnten Benutzungs-Metapher[8] *interaktiv* wie auch, was sich etwa im Rahmen einer Vortragssituation empfiehlt, relativ linear durchschritten werden kann (die einzelnen Unter-Hierarchien werden in letzterem Falle einfach automatisch geöffnet und wieder geschlossen), erschließt sich dem Benutzer (im Interaktionsfalle) bzw. dem Vortrags-Publikum die Bedeutung der übertragenen Symbolik fast automatisch:

---

[1] Icon-Auflösung: 128x128 Pixel

[2] s. hier etwa [MTH97]

[3] Anm: Relativ „buggy" (sowohl [Vant02] als auch [Mori99, Mwt99] neigen zu häufigen Abstürzen) wie auch „ignorant" hinsichtlich Transparenz etc.

[4] s.3.3.2

[5] s. 6.1 (Anfang)

[6] Ausnahme: Zitate und Bilder (diese werden als eigenständige Tochter-Elemente realisiert)

[7] Anm: Dies wird wiederum zumeist in einer eigenen Ansicht realisiert (vgl. etwa Abb. 6.2.1, rechtes Teilfenster)

[8] Gemeint: „Ausklappen" des Baumes durch das „+"-Symbol [vgl. Fors98:53]

> *Eine einfache Metaphorik, die ohne Erläuterungen zu verstehen ist… [Insbesondere] gelungen [sind ebenso] die „aufgeklappten" Ordner mitsamt Unter-Verzeichnissen…*
>
> [Fors98:53f]

6.3.3 Übersichtlichkeit

Als weiteren Vorzug des hierarchischen Aufbaus der Präsentationsdaten ist überdies die außerordentliche *Übersichtlichkeit* und *Transparenz* zu nennen, die mit deren Explorer-artigen Darstellung [s.6.2] einhergeht: So bleibt aufgrund der dynamisch expandierenden Teilbäume stets deutlich, wo sich der entsprechende Abschnitt im Zusammenhang der Gesamtpräsentation befindet. Überdies wird hierdurch ebenso transparent gemacht, wie viele Abschnitte den Zuschauer noch erwarten – ein nicht abreißen wollendes „Durchklicken" endloser Folien-Litaneien kann hierdurch somit zu verhindert werden. Ebenso erspart die baumartige Präsentationsstruktur die aufgrund des eben angesprochenen Effekts in PowerPoint oft üblichen „Übersichts-Diagramme", welche zumeist am Rande des eigentlichen Präsentationsbereiches manuell eingefügt werden müssen, um dem Zuschauer eine zumindest simulierte Transparenz hinsichtlich des inhaltlichen Aufbaus der Präsentation bieten zu können. Auch die obligatorische „Einstiegsfolie" Slide-basierter Präsentationen (mit einer groben „Übersicht" des folgenden Vortrags)[1] entfällt im Rahmen der hierarchischen Baumdarstellung: Durch den konsequent strukturierten Aufbau hierarchischer Präsentationen ergibt sich die „unterste" Ebene (die dann idealerweise stets die einzelnen „Haupt-Kapitel" enthält) von selbst. Vorraussetzung dieser Vorgehensweise ist jedoch naturgemäß eine entsprechend logisch strukturierte Präsentation, da sich bei Nicht-Einhalten dieser Grundprinzipien die soeben genannten Vorteile freilich nicht ergeben.

6.4 Die Rolle von SVG

Nach Klärung des grundlegenden Prinzips der geplanten Alternative (Strikte Trennung von inhaltlicher Struktur und Darstellung [6.1] zur „Forcierung" klar strukturierter, ästhetischer Präsentationen; Baum-Metapher [6.2] samt Icon-Symbolik [6.3], etc.) möchte ich an dieser Stelle vor Beschreibung der eigentlichen Realisierung noch kurz auf die nicht unwichtige Rolle des *Scalable Vector Graphics*-Formates (SVG) bei der Verwendung im Rahmen des für diese Diplomarbeit entstandenen Prototypen eingehen:

Nachdem die genauere Betrachtung des derzeit dominierenden *de-Facto*-Standards für Präsentationen, *PowerPoint* [s.2.3], insbesondere dessen gravierende Schwächen[2] und die Notwendigkeit einer Alternative deutlich macht,[3] zeigt die im 3. Kapitel näher beleuchtete Entwicklung des Standard-Web-Formats HTML, dass diese ursprünglich als rein wissenschaftliche Dokumentstruktur entwickelte Markup-Sprache zwar zunehmend in Richtung grafischer und multimedialer Funktionalität gedrängt wurde,[4] jedoch nichtsdestotrotz für Präsentationszwecke immer noch weitgehend ungeeignet erscheint [s. 3.6]. Das sich an dieser Stelle als quasi-Standard für Web-basiertes „Multimedia" etablierte *Flash-* bzw. SWF-Format [s.4.5] vermag diese „Lücke" zwar durch beeindruckende Animations-Eigenschaften scheinbar zu füllen, erweist sich im Umkehrschluss jedoch bedauerlicherweise als „von seinem Grundprinzip her nicht Web-geeignet",[5] da das proprietäre, binäre Dateiformat Web-gemäßer *Usability* entgegensteht, „schlechtes Design" fördert [Niel00a] und auch den Erstellungsprozess Firmenpräsentations-orientierter Webseiten unvorteilhaft beeinflusst [vgl. Dack99, Fren02].

---

[1] Bsp: „Einführung; Darlegung des Problems; Aspekt 1... Aspekt n, Zusammenfassung, Diskussion."

[2] s. 2.3.2-1-3

[3] vgl. hierzu auch [Wine88, Sear98, Stew01, Park01]

[4] s. 3.2.

[5] vgl. [Niel00a]

6.4.1 SVG als optimales Carrier-Format

An dieser Stelle kommt wiederum der XML-basierte W3C-Standard SVG ins Spiel: Mit beeindruckender grafischer und multimedialer Funktionalität (Filter, SMIL-gemäße Animation, Skripting…) und zugleich Web-konformer Text-Eigenschaft[1] verbindet die Vektorsprache elegant die Vorteile von HTML mit denen des *Flash*-Formates, ohne jedoch zugleich deren soeben erwähnte Nachteile[2] mit sich zu bringen. Daher drängt sich das Format logischerweise für Web-basierte Präsentationen förmlich auf: Aufgrund der offensichtlichen Mängel des insbesondere im Internet-Bereich mangelhaften *PowerPoint* [s.2.3.4] gehört es mittlerweile nahezu zum „guten Ton", sozusagen als „obligatorische Einstiegs-Anwendung" eine eigene Präsentationslösung auf SVG-Basis zu zimmern:

Not another one! Everyone creates a SVG slide generator… It's the "Hello World" of SVG! [Sue02]

Aus diesen Überlegungen begründet sich folglich auch der Grundansatz und auch der Titel des im Rahmen dieser Diplomarbeit entstandenen Prototypen eines „alternativen Präsentationskonzepts": *XML » SVG Presenter*. Bereits der Name weist somit bereits auf die Verwendung eines internen, XML-basierten Formates zur Strukturierung der Präsentationsdaten, [s.6.1, 6.5.1], die primär Darstellungs-orientierte Präsentationseigenschaft des Prototypen,[3] sowie auf die Anwendung des SVG-Formates als im Hinblick auf Web-basierte Präsentation „optimales" Ausgabemedium hin

Dennoch erstaunt diese "Schwemme" SVG-basierter Präsentationstools nicht zuletzt aufgrund der noch relativ bescheidenen Verbreitung des entsprechenden SVG-Viewers – so gehen selbst die optimistischen Schätzungen der SVG-Autoren selber [vgl. Watt02] lediglich von einer „deutlichen Minderheit" [Adam02:16] der Web-Benutzer aus, die SVG-Grafiken wirklich zu betrachten in der Lage ist.[4] Eine genauere Betrachtung der momentanen Zielgruppe macht jedoch die wachsende Begeisterung an SVG-basierten Präsentations-Möglichkeiten durchaus plausibel: So kann derzeit davon ausgegangen werden, dass die Mehrzahl sowohl der Plug-In-Anwender als auch der Ersteller SVG-basierter Präsentationen im Gegensatz zum eher Consumer-orientierten *Flash*-Format überwiegend im technisch versierten, akademischen Bereich anzusiedeln ist. Aufgrund dessen erscheinen auch die hohen Anforderungen hinsichtlich technischer und gestalterischer Kompetenz derer, die sich derzeit einer SVG-orientierten Präsentationslösung bedienen,[5] nur wenig verwunderlich.

6.4.2 "Yet another SVG Presentation Program"? [LiJa03]

Warum also nun „lediglich noch eine weitere Spielart" dieser SVG-basierten Präsentationsansätze im Rahmen dieser Diplomarbeit erstellen? Nun – sämtliche derzeitigen Herangehensweisen wagen es, wie bereits in [6.1] ausgeführt, bedauerlicherweise nicht, sich über die durch *PowerPoint* definierte, uniforme Präsentations-Norm auf „Slide"-basis [s.2.1-2] hinwegzusetzen, und zeigen daher zwar aus technischer Sicht, aber eben *nicht* aus ästhetischer oder inhaltlich-struktureller Perspektive eine wirkliche Alternative zum etablierten PowerPoint-Standard auf. Bereits der Ausspruch „Everyone creates a *SVG slide generator*", den Hoylen Sue der australischen Queensland-Universität in [Sue02] äußert, lässt insbesondere hinsichtlich der Wortwahl aufhorchen: So ist nicht etwa von einem alternativen Präsentationskonzept die Rede, sondern ausdrücklich von einem „Slide Generator", also einem simplen Tool, welches in der Erzeugung einzelner, be-

[1] Anm: Dies fördert wiederum die „Usability" SVG-basierter Anwendungen: Elemente lassen sich direkt kopieren und extrahieren, einzelne Inhalte sehr leicht von Internet-Suchmaschinen indizieren.

[2] s. hierzu auch 3.2 sowie 4.5

[3] Daher *Presenter* (und nicht etwa „Presentation Editor", da der Fokus auf der Darstellung und nicht der Erstellung der Präsentation liegt [s.6.6]

[4] „Native browser suppoert for SVG is poor…" [Watt02:33]

[5] s. hierzu auch [6.1]

züglich des optischen Erscheinungsbilds PowerPoint augenscheinlich[1] möglichst ähnlicher „Bildschirm-Folien“ (so die Microsoft-Terminologie) dem verleideten PowerPoint-Standard somit unfreiwillig nacheifert. Wie bereits ausführlich dargelegt, ist diese Eigenschaft leider allen derzeit verfügbaren PowerPoint-„Alternativen“ gemein.

Obgleich die strikte Orientierung an einzelnen „Slides“ (die jeweils ja wiederum einzelne „Screens“ darstellen und somit den gesamten Bildschirmbereich umfassen) durchaus den gedanklich naheliegendsten Schritt darstellt, da es sich ja schließlich bei der OnScreen-Präsentation um ein primär Bildschirm- bzw. Beamer-fixiertes Ausgabesystem handelt, zeigt in meinen Augen jedoch erst die im Rahmen dieser Diplomarbeit versuchte Emanzipation von diesem Prinzip durch primäre Orientierung an der gesamten, hierarchischen Struktur der Präsentationsdaten eine diesbezüglich konsequente Alternative auf. Den groben Rahmen hierfür haben ja bereits die in [2.3.1] sowie [6.1] angesprochenen Vorüberlegungen Dave Winers und David Searls' [Wine88, Sear98] abgesteckt – die eigentliche Umsetzung dieses strukturorientierten Prinzips ist jedoch erst im Rahmen der Einführung und Adaption des SVG-Standards hinsichtlich „web-tauglicher“ Präsentationen sowohl interessant als auch tatsächlich konsequent möglich geworden.

6.5 Umsetzung

6.5.1 Transformation der internen XML-Struktur

Zum besseren Verständnis der dieser Realisierung zugrunde liegenden Techniken sei an dieser Stelle noch einmal die bereits in [6.1] diskutierte Auftrennung der Präsentationslösung in das die Darstellung erst ermöglichende, mächtige „Maximal“- sowie ein ebenso XML-basiertes „Minimal“-Format erwähnt. Ohne erneut die hinter dieser Separation stehenden Grundprinzipien neu aufzurollen, sei an dieser Stelle nun der eigentliche Aufbau des „trivialen“ XML-Formats kurz erläutert: Wie bereits ausgeführt,[2] sollten im Rahmen einer Formulierung der eigentlichen Präsentationsdaten zwar die „üblichsten“ (d.h. in der Praxis am ehesten benötigten) Funktionen und Elemente einer Präsentation abgedeckt, jedoch zur Verbesserung der Usability und Wahrung inhaltlicher Konsistenz zugleich eine größtmögliche formattechnische „Schlichtheit“ erzielt werden.

Die Dokumenten-Typdefinition (DTD) der umgesetzten Lösung[3] fällt somit hinreichend einfach aus: Nach einem Kopfteil (`<header>`), in der sich Titel, mehrere Verfasser, Beschreibung, Datum, Ort etc. spezifizieren lassen, bildet im Hauptteil (`<body>`) das `<item>`-Objekt das wichtigste Element der Präsentation: Ein „Item“ kann somit sowohl ein einzelner Gedanke sein, als auch Überbegriff (Ordner) für andere Gedanken: Verschiedene *Items* lassen sich somit in Kapitel, Stichworte, Aufzählungen oder linear aufgereihte Argumente ordnen. Längere Texte oder Bildelemente können in Form von `<image>`-[4] bzw. `<text>`-Elementen einzelnen *Items* zugeordnet werden, deren (stets benötigtes) Bezeichnungs-Attribut dann Überschrift bzw. Beschreibung des untergeordneten Inhaltes darstellt.

Da neben der reinen, Benutzer-orientierten Strukturierung auch die optisch ansprechende Darstellung eine wichtige Rolle spielt [vgl. Karv00], kommt hier die den Betriebssystem-GUIs entliehene Icon-Metapher [Fors98:53ff] ins Spiel: *Items* werden nicht lediglich in textlicher Form, sondern zudem durch ein neugierig machendes, ästhetisch gestaltetes Bildelement ergänzt und illustriert, was sowohl die optische Orientierung für den Benutzer (da der Mensch ja primär ein visuell orientiertes Lebewesen ist) erheblich erleichtert, als auch die „Explorativität“ der Präsentation erhöht, da eine ähnliche Metaphorik bereits aus den grafischen

[1] s. hierzu die Präsentationsausgabe der SVG-Slide-Tools [HSL00], [Sue02] sowie [Herm02] in Abb. 6.1.2.1-3

[2] s. 6.1

[3] URL: http://www.fh-furtwangen.de/~voswinck/pdata.dtd [28.1.03]

[4] Aufgrund eines noch im aktuellen Adobe SVG Viewer (3.0) akuten Bugs ist in der momentanen Implementierung das <image>-Element als direktes SVG-Objekt (mit entsprechendem Namensraum: <svg:image…>) realisiert.

Dateisystem-Explorern bekannt ist.[1] Daher lassen sich neben einem voreingestellten Ordner-Icon (*default*) für jedes Item alternative Icon-Elemente spezifizieren. Besitzen einige *Items* weitere Unter-Elemente („Children"), so wird dies durch die intrinsisch verständliche[2] *plus*-Metapher verbildlicht.

Abb. 6.5.1.2: Plus-Symbol (rechts)

Da die Darstellung beispielhafter Präsentationsdaten gemäß der soeben beschriebenen Dokumentstruktur, welches den syntaktischen Aufbau dieser DTD verdeutlicht, an dieser Stelle etwas zu viel Platz einnehmen würde, entnehmen Sie das entsprechende Listing bitte [Anhang A].

Nach Definition der Struktur des XML-Formats sei an dieser Stelle noch erwähnt, dass im Vorfeld der eigentlichen Darstellung der Präsentation freilich zunächst eine interne Übersetzung der „trivialen" Präsentationsdaten in das daraufhin zur Anzeige verwendeten SVG-Format erfolgen muss.

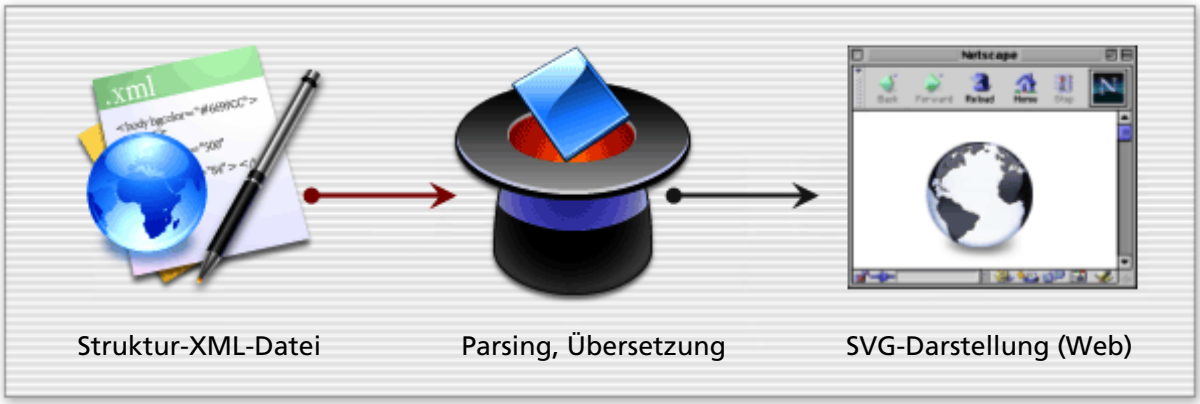


Abb. 6.5.1.3: Übersetzung der internen („trivialen") XML-Daten in sichtbaren SVG-Code.

Während nun im Rahmen der beispielhaften Sun-Lösung naturgemäß das XML-Framework der eigenen Programmiersprache Java zum Tragen kommt [vgl. Arm01], benötigt der *Flash*-basierte Ansatz von Macromedia [Behz02] freilich die XML-optimierte MX-Engine und somit einen aktuellen *Flash*.6-Viewer zur korrekten Übersetzung der internen Präsentationsdaten. Nahezu sämtlichen, auf SVG basierenden Lösungen liegt wiederum die hierfür naheliegend erscheinende Style-Sheet-Sprache XSLT zugrunde [vgl HSL00, Herm02], die beide XML-basierte Formate (i.e. die Präsentationsdaten sowie den sichtbaren SVG-Code) ihrerseits selbst mithilfe der XML-Konvention übersetzt.[3] [Sue02] bedient sich derweil der Sprache *Perl* zur Erstellung der SVG-Ausgabe mithilfe der gelieferten XML-Daten, was freilich die vorhandene Installation eines Perl-Servers voraussetzt.

Und eben dieses Server-Tool (auf dessen Einzelheiten an dieser Stelle nicht eingegangen soll), wird meiner Einschätzung nach bei einem Großteil der potentiellen Anwender *nicht* installiert sein, was einen in meinen Augen unnötigen Aufwand bezüglich des Parsing und der Konvertierung der XML-Daten bedeutet. Auch die Anwendung des XSLT-Mechanismus ist freilich der zwar aus theoretischer Sicht „sauberste Weg", aber dennoch nicht ganz problemlos: So unterstützt zumindest derzeit lediglich ein Bruchteil der heute verfügbaren Web-Browser die automatische Verarbeitung mitgelieferter XSLT-Anweisungen – für die meisten

[1] s. hierzu auch die bereits ausgeführten Überlegungen in [6.3]
[2] vgl. [Fors98] p.53
[3] s. hierzu auch 5.1

Anwendungen, einschließlich des Adobe SVG Viewer, ist jedoch die (wiederum reichlich problematische) Installation und Verwendung eines separaten XSLT-*Prozessors* (wie etwa des Java-basierten *Saxon*)[1] vonnöten, der die entsprechende Übersetzung schließlich durchführen kann:

> *At present, unfortunately, one must [an] XSLT processor. This restriction will become obsolete when XSLT 2.0 processors become more widely available…*
>
> [Herm02]

Diese in meinen Augen äußerst unerfreuliche Abhängigkeit von jeweils externen Prozessoren, Server-Tools oder *Viewern*, die, wie soeben erläutert, bedauerlicherweise bei *sämtlichen* bereits bestehenden, XML-basierten Präsentationsansätzen besteht, wollte ich konsequenterweise bei der Realisierung des *XML » SVG Presenter*s vermeiden. Aufgrund dessen habe ich mich, in Anlehnung an [Duck01:105],[2] bei der Umsetzung des *Presenter*s zur Anwendung der Client-seitigen Programmiersprache ECMAScript [vgl. Wild99:377] entschlossen. Diese durch die *European Computer Manufacturers Association* (ECMA) 1997 standardisierte Implementierung der zuvor unter dem Namen *JavaScript* (zuvor jedoch in sehr inhomogener, inkompatibler Erscheinungsform)[3] insbesondere in Browser-Umgebungen[4] bekannte Skriptsprache [ECMA97] besticht in ihrer neuesten Version insbesondere durch konsequente Objektorientierung und DOM-Unterstützung.[5] Konkret bedeutet dies unter anderem, dass man nun ähnlich wie in der Programmiersprache *Java* „Objekte" definieren und verändern kann [vgl. Lind02]

```
// Definition eines Integer-Objekts (bspw. im Kopfteil einer ECMAScript-Datei):

function Integer(value) { this.int = value; }
function Integer.prototype.set(value) { this.int = value; }
function Integer.prototype.value() { return this.int; }

// Im Hauptteil: Anwendung des Objekts:

var eineZahl = new Integer(5);
var zweiteZahl = new Integer(2);
zweiteZahl.set(eineZahl.value() + 3); // zweiteZahl = 8!
```

Listing 6.5.1.1: Instanziierung und Zugriff auf ein Integer-*Objekt mittels ECMAScript (In* script.js *des Prototypen)*

Speziell die DOM-Implementierung ECMAScripts ermöglicht wiederum einen für unsere Zwecke äußerst interessanten Zugriff auf einzelne Elemente eines XML-Dokuments. Ohne an dieser Stelle sonderlich weit in die Details der DOM-Funktionalität eindringen zu wollen,[6] sei an dieser Stelle zum Verständnis dieses Prinzips lediglich erwähnt, dass im *Document Object Model*, wie der Name bereits andeutet, das gesamte Dokument, auf das im Rahmen der DOM-Aktivität zugegriffen werden soll, als objektorientierte Hierarchie aufgefasst wird. Dies bedeutet, dass sich das gesamte Dokument in baumartig organisierte Objekte, die so genannten *Knoten* (oder „Nodes") unterteilt. Auf jeden einzelnen dieser Baum-Knoten kann nun wiederum mithilfe der DOM-Syntax direkt zugegriffen werden – und zwar nicht nur auf die einzelnen Eigenschaften des Node-Objekts selber, sondern sogar auf dessen übergeordnete Knoten, (*Parents*) ebenso wie sämtliche Tochterelemente (*Children*) und „Geschwister" (*Siblings*), d.h. die Baumknoten derselben Hierarchie-Ebene:

```
// Zugriff auf verschiedene Objekte einer Gruppe (Hier: MouseOver-Funktion)
...
```

---

[1] vgl. [Kay00,01]

[2] „Eine Weitere Möglichkeit, XML auf dem Webclient zu parsen, ist der EInsatz eines Javascripts. Der Vorteil hierbei ist, dass das XSL-StyleSheet nicht innerhalb des XML-Dokumentes festgeschrieben sein muss. Und sich somit abhängig vom Endgerät verschiedene XSLT der XML-Datei zuweisen lassen…" [Duck01] p.105

[3] vgl. [Wild99] p.377

[4] Anm: So ist etwa Microsofts JScript eine konkrete Implementierung des ECMAScript-Standards.

[5] DOM: *Document Object Model* (Objektmodell eines XML-Dokuments) vgl. [Arm01]

[6] vgl. hierzu [Arm01]

```
var svgGruppe = evt.getTarget().parentNode; // Betroffenes Objekt der Mausbewegung
var gradientenObjekt = svgGruppe.firstChild; // Erstes (unterstes) Objekt
var textObjekt = grad.nextSibling.nextSibling; // "Übernächstes" Objekt
textObjekt.setAttribute("class","white"); // Textfarbe weiß setzen
gradientenObjekt.setAttribute("visibility","visible"); // Verlaufs-Balken anzeigen
...
```

Listing 6.5.1.2: Zugriff auf XML-Objekte (hier: SVG-Elemente) mittels ECMAScript und DOM (In script.js*)*

Mithilfe dieser Funktionalität kann nun, wie am Beispiel [Listing 6.5.1.2] aufgezeigt, im Rahmen des Adobe SVG-Viewers auf einzelne Objekte eines SVG-Dokuments zugegriffen werden. Dies wiederum stellt sozusagen den „Grundstein" der dem *XML » SVG Presenter* zugrunde liegenden *Engine* dar, da nahezu sämtliche[1] SVG-Elemente der Präsentationsdatei auf Basis des „trivialen", internen XML-Formates zur Laufzeit berechnet, in den „virtuellen Baum" des SVG-Dokuments eingehängt und somit dynamisch angezeigt werden.

Bevor allerdings auf dieses grafische Framework zugegriffen werden kann [s.6.5.2] müssen die zunächst in Form *externer* XML-Dateien[2] vorliegenden Präsentationsdaten, wie bereits eingangs erwähnt [s.Abb.6.5.1.3], jedoch „geparst" und in *interne* Datenstrukturen transformiert werden: So werden etwa die Inhalte der *Header*-Sektion, die Angaben über Autor(en), Titel der Präsentation, Datum etc. liefern, in ein hierfür vorgesehnes „Header"-Objekt im Code-Bereich überführt:

```
// Header-Objekt:
function Head(aTitle) {
       this.title = aTitle;
       this.authors = new Array();
       this.date = new Date();
}
function Head.prototype.addAuthor(anAuthor) { this.authors.push(anAuthor); }
function Head.prototype.defineSetting(aSetting) { this.setting = aSetting; }
...
// XML-Parsing-Teil:

var presentationData = new Node(document.getElementById("pdata"));
var head = presentationData.filter(addNameSpace("head")); // Parse Head
var presentation = new Presentation(head.filter(addNameSpace("title")).text());
var authors = head.filter(addNameSpace("author")); // Autor(en) ausfiltern
if(authors.count()!=1) // Wenn es mehrere Autoren gibt...
 for(var i=0;i<authors.count();i++) //... dann füge für jeden einzelnen Autor..
     presentation.head.addAuthor(authors.node(i).text()); //... dessen Name ein!
else presentation.head.addAuthor(authors.text()); // Ansonsten nur den einen!
presentation.head.defineSetting(head.filter("setting").text()); // Ort angeben
...
```

Listing 6.5.1.3: Definition des Head-Objekts und Einfüllen der geparsten XML-Daten (script.js)

In gleicher Weise werden auch die einzelnen *Items* in entsprechene „Item"-Objekte in ECMAScript überführt – wobei, entsprechend der vorhin erläuterten Dokumenten-Typ-Definition, jedes einzelne Item natürlich wiederum eine Reihe[3] weiterer („Child-")Items beinhalten kann. Auf die detaillierten Parsing-Algorithmen im Detail einzugehen, würde den Rahmen dieser Diplomarbeit jedoch etwas sprengen, weswegen zum besseren Verständnis des Übersetzungs-Verfahren an dieser Stelle lediglich abschließend zusammen gefasst werden soll, dass sämtliche, zuvor in der DTD spezifizierten Präsentationsdaten in ähnlicher wie der soeben beispielhaft aufgezeigten Weise in gleichartig strukturierte, logische Objektstrukturen überführt werden. Dies dient im Vorfeld der eigentlichen Präsentationsdarstellung nicht nur der Verifizierung der syntaktischen Korrektheit der in der zugehörigen XML-Dateien angegebenen Präsentationsdaten, sondern darüber hinaus auch dem Umstand, dass die internen ECMAScript-Variablen weitaus schneller als die

[1] Anm: Bis auf die Hintergrundgrafik

[2] Beispiel: http://www.fh-furtwangen.de/~voswinck/sample.xml [28.1.2003]

[3] in ECMAScript natürlich: einen *Array*!

jeweils nur über die DOM-Schnittstelle verfügbaren Daten-Knoten durchsucht und verarbeitet werden können.

Zugleich sei jedoch ebenfalls der nachteilige Umstand erwähnt, dass aufgrund der konsequenten Verwendung sowohl des objektorientierten ECMAScript-Ansatzes als auch der DOM-Schnittstelle erhebliche softwaretechnische Anforderungen auf den Benutzer des *XML » SVG Presenter* zukommen: Da beide Funktionen erst seit relativ kurzer Zeit standardisiert und verbreitet sind, benötigt der *XML » SVG Presenter* neben dem aktuellen SVG-Viewer von Adobe in der Version 3.0[1] zudem eine fortschrittliche Browser-Umgebung wie den Internet Explorer 6,[2] da ansonsten weder Objektorientierung noch DOM-Funktionalität unterstützt werden. Diesen zugegebenermaßen recht hohen Software-Anforderungen steht jedoch eine durchweg konsequente und überdies zukunftsorientierte SVG-Anwendung gegenüber: Da diese Diplomarbeit sich ja primär an theoretisch und aus langfristiger Perspektive interessanten Technologien orientiert und somit nicht auf „pragmatische" Kriterien wie clientseitiger Ablauffähigkeit (i.e. Verbreitung) festgelegt ist, sind diese Einschränkungen aufgrund der nichtkommerziellen Prototyp-Eigenschaft dieser Konzeptstudie durchaus hinnehmbar. Darüber hinaus sei an dieser Stelle zudem noch erwähnt, dass der *XML » SVG Presenter* eben aufgrund dieser Funktionalität etwa im Gegensatz zu sämtlichen weiteren SVG-basierten Ansätzen auf keinerlei externe XSLT-Prozessoren [vgl. HSL00, Herm02], oder sonstiger externe (Server)Tools [Sue02] angewiesen ist, sondern eine auch offline funktionsfähige *Standalone*-Anwendung darstellt.

Bezüglich der eigentlichen Umsetzung des Prototypen erscheint an dieser Stelle noch ein weiteres Kuriosum der SVG-spezifischen Zielumgebung erwähnenswert, welches wiederum die Implementierung eines „Work-Arounds" erforderlich gemacht hat: So unterstützt etwa die derzeitige Version (3.0) des Adobe SVG Viewers *nicht* das zum Parsen externer XML-Dateien unter Windows angebotene *ActiveX*-Objekt des Redmonder Software-Riesen (MSXML), sodass der Aufruf

```
var myDom = new ActiveXObject("Msxml.DOMDocument");
```

… im Rahmen der Adobe-Umgebung zu keinerlei Ergebnis führt – der Zugriff auf externe XML-Dateien wäre somit lediglich bei Einbettung etwa in eine XHTML-Umgebung des Microsoft Internet Explorer unter Windows möglich. Da diese doppelte Einschränkung jedoch sowohl inkonsequent als auch recht umständlich erscheint, habe ich mich bei Realisierung eines entsprechenden „Work-Arounds" auf die bereits eingangs erwähnte DOM-Eigenschaft der SVG-Implementierung selber besonnen: Zwar ist diese gemäß der SVG-Spezifikation eigentlich nur zur Manipulation SVG-nativer Elemente vorgesehen, lässt sich jedoch freilich ebenso zur Analyse weiterer, nicht SVG-konformer Objekte „missbrauchen". So kommt in der nun realisierten Version des *XML » SVG Presenter*s die „Namensraum"-Eigenschaft von XML zum Tragen:[3] Demnach lassen sich „externe", also der eigentlichen SVG-Syntax fremde XML-Elemente in denselben Dokument-Code einbinden – stets versehen mit einem separaten, so genannten „Namespace"-Deskriptor, der das entsprechende Tag eindeutig einem anderen *Namensraum* und somit einer anderen DTD zuordnet. Der interne Parser der ECMAScript-Engine kann diese „eingefügten" Elemente sodann als separate XML-Objekte identifizieren und entsprechend verarbeiten. In der derzeitigen *Presenter*-Anwendung sieht dies etwa konkret so aus:

```
<svg xmlns="http://www.w3.org/2000/svg"
     xmlns:pd="http://www.fh-furtwangen.de/~voswinck">
   <title>XML >> SVG Presenter</title>
```

[1] URL: http://www.adobe.com/svg/viewer/install [28.1.2003]

[2] Ggf. Auch Version 5.5 – dies ist jedoch Mindestanforderung, da frühere Browser kein Objektorientiertes EcmaScript unterstützen!

[3] s. hierzu auch [5.1]

```
   <!-- XML-Präsentationsdaten -->
   <pd:presentation id="pdata">
      <pd:head id="head">
         <pd:title>Web-based Presentations | XML &#xbb; SVG Presenter</pd:title>
         <pd:author>Till Voswinckel</pd:author>
         ...
      </pd:head>
      <pd:body id="body">
        ...
      </pd:body>
   </pd:presentation>

   <!-- Weitere (SVG-native) Daten... -->

</svg>
```

Listing 6.5.1.4: Einbindung der XML-Präsentationsdaten in das SVG-Dokument.

Auf diese Weise kann die DOM-Eigenschaft des SVG-ECMAScript-Parsers auch zur Analyse der XML-basierten Präsentationsdaten sozusagen „missbraucht" werden [s. hierzu etwa die entsprechenden „`addNameSpace`"-Verweise in Listing 6.5.1.3] und somit auch die „triviale" Formatkomponente des *XML » SVG Presenter*s zur Weiterverwendung in ECMAScript, der Überführung in SVG und deren letztendlichen Darstellung im Rahmen der SVG-Viewer-Umgebung konsequent zur Anwendung kommen. Die Speicherung und Analyse der Präsentationsdaten in Form einer *externen* XML-Datei stellt zwar die aus struktureller Sicht „sauberste" Lösung dar, wird jedoch nach [Watt02:1042] erst im Zuge der Einführung einer DOM Level-3-Implementierung [vgl. DOM02] verfügbar sein. Obgleich auch die `parseXML()`-Funktion des Adobe SVG-Viewers ähnliche, externe Parsing-Funktionen bereitstellt [vgl. Quin02], allerdings derzeit ausschließlich im Rahmen der letztgenannten Viewer-Umgebung ablauffähig ist und dadurch eine architektonisch „einengende", proprietäre Lösung darstellen würde, erfüllt bis dahin somit das soeben dargelegte „Work-Around" diese Funktion in durchaus zufrieden stellendem Umfang.

6.5.2 Grafik-Framework

Freilich kommen die überzeugenden, skript-bezogenen DOM-Funktionen der SVG-Umgebung im Rahmen der *Presenter*-Realisierung nicht ausschließlich zu derart missbräuchlichen Zwecken, sondern natürlich auch durchaus in deren ursprünglichen Sinne zur Anwendung: Im großen und ganzen lässt sich das zur Echtzeit durch ECMAScript vorgenommene grafische „Rendering", d.h. bei SVG vielmehr das Verändern des sichtbaren SVG-Dokumentenbaumes,[1] in zwei Kategorien unterteilen:

- Ein *Amimations*-Framework (`AniObject` sowie `AniTracker`) sowie
- Anzeige der Präsentations-Baumhierarchie durch "Auf- und Zuklappen" der einzelnen Unter-Ebenen (`expand` und `collapse`)

Vor Betrachtung der einzelnen Komponenten stellt sich jedoch zunächst einmal die berechtigte Frage, inwieweit denn die Anwendung der ECMAScript-Engine für derartige Zwecke überhaupt notwendig und sinnvoll ist – schließlich stellt die SVG-Spezifikation [SVG01] selber bereits mit `<set>` und `<animate>`[2] durchaus mächtige und überdies der standardisierten SMIL-Semantik entsprechende Funktionalität sowohl zur Animation (was ein Animations-Framework ja prinzipiell unnötig machen würde) als auch zum Verändern, Anzeigen und Verstecken einzelner Elemente bereit. Diese separate Realisierung dieser Funktionen durch ein gesondertes Script-Framework liegt freilich im Detail der soeben angesprochenen SMIL-Funktionalität begründet: So ist es zwar durchaus möglich, Animationen und Objektveränderungen von

[1] Genauer bedeutet dies: Nicht das Script selber erzeugt die Ausgabe, sondern stets die SVG-Umgebung selber – bei Änderung des SVG -Dokumentenbaumes aktualisiert die Viewer-Enginge automatisch die Darstellung.

[2] Dies umfasst freilich ebenso die Derivate <animateMotion>, <animateColor> sowie <animateTransform>

den Zuständen und Manipulationen anderer Objekte abhängig zu machen,[1] etwa mittels `<animate begin="anderesElement.click"...>` und auch mehrere Animations-Elemente zu verschachteln – die parallele Änderung bzw. Animation mehrerer Attribute eines Elements ist jedoch *nicht* möglich: So lässt sich etwa eine Animation wie die in [Abb.6.5.2.1.1] gezeigte derzeit nicht mittels konventioneller SVG- bzw. SMIL-Syntax realisieren.

Auch die „dynamische" Anzeige der nächsten Hierarchie-Ebene des Baumes bereitet im Rahmen der von SVG angebotenen Funktionalität einige Schwierigkeiten: Da bei Aktivierung eines „Eltern"-Knotens (etwa durch Anklicken des „plus"-Symbols) ja nicht nur die Tochterelemente entsprechend sichtbar gemacht, sondern ebenso alle unterhalb dargestellten Symbole entsprechend weiter „nach unten" [2] verschoben werden müssen,[3] sind an dieser Stelle gleich die Attribute[4] einer enormen Anzahl sichtbarer und bislang unsichtbarer Objekte zu modifizieren. Dies wiederum kann bei ausschließlicher Realisierung in Script-freiem SVG sehr schnell unübersichtlich geraten – wenn es nicht gar völlig unmöglich ist.

Aus diesem Grunde habe ich mich auch hier zur Entwicklung verschiedener „Utility Classes" entschlossen, also grundlegender, hilfreicher Objekte und Funktionen, die auch komplexere Operationen wie etwa die eben angesprochene „Massen-Manipulation" von Elementen oder auch Animations-Eigenschaften ermöglichen sollten – und zwar naturgemäß wiederum mittels clientseitigem ECMAScript. Aufgrund dessen erscheint an dieser Stelle auch die Anwendung der Script-Engine beim bereits in [6.5.1] erläuterten *Parsing* der internen XML-Daten in einem anderen Licht: Da die Präsentationsdaten aus der XML-Struktur somit zur Animation bzw. Anzeige der einzelnen Objekte ohnehin in der Form ECMAScript-konformer Variablen vorliegen sollten,[5] macht auch der Skript-basierte Parser,[6] insbesondere im Vergleich mit den externe Prozessoren benötigenden SVG-Alternativen [HSL00, Herm02, Sue02] durchaus Sinn und trägt zudem zur erfreulichen[7] *Standalone*-Eigenschaft des *XML » SVG Presenter*s bei.

6.5.2.1 Animations-Framework

Im Zuge der Realisierung eines, wie soeben dargelegt, für die Umsetzung des *Presenter*s notwendigen Animations-Frameworks, gilt es freilich zunächst zu überlegen, welche Grundeigenschaften eine „Animation" erfüllen sollte und wie sich dies im Rahmen der ECMAScript-Syntax realisieren lässt: Zunächst sei festgehalten, dass eine diskrete Animation jeweils die Eigenschaft *eines* Objekts in einem bestimmten Zeitfenster von einem Ausgans- zu einem zuvor festgelegten Zielwert dynamisch verändern soll. Dies allein ließe sich in SVG- bzw. SMIL-Syntax ja noch sehr einfach realisieren:

```
<ellipse cx="200" cy="100" rx="48" ry="90">
    <animate attributeName="rx" begin="3s" dur="6s" from="48" to="198" ... />
</ellipse>
```

Listing 6.5.2.1.1: Animation des x-Radius einer Ellipse in SVG.[8]

[1] vgl. [Watt02] pp.428ff.

[2] Aus der Perspektive der Darstellungsfläche also: (Positive) Verschiebung entlang der y-Pixelkoordinate

[3] Darüber hinaus sind freilich noch einige Weitere Aktionen durchzuführen: Die Linien sind zu verlängern etc.

[4] In der Regel bezieht sich dies auf die x-, y- und visibility-Eigenschaften.

[5] Anm: Schliesslich wird die ECMAScript-Engine ja ebenso zur Steuerung der Animation verwendet und benötigt daher natürlich auch Daten (in Form von Variablen), in welcher Form diese Anweisungen zu geben sind.

[6] s. 6.5.1

[7] s. ebenda.

[8] Anm: Im *Animate*-Tag wurden die für diesen Sachverhalt irrelevanten und ohnehin „obligatorischen" Attribute attributeType="XML" sowie fill="freeze" entfernt.

Der in [Listing 6.5.2.1.1] dargestellte SVG-Code würde beispielsweise die `rx`-Eigenschaft (also den Radius zur X-Achse) der übergeordneten Ellipse in einem Zeitrahmen von 6 Sekunden (Zeitversatz: 3 Sekunden) vom Startwert 48° zeitkontinuierlich dem Zielwert annähern.

Weitaus komplizierter gerät dieser Sachverhalt jedoch, wenn verschiedene Attribute ein- und desselben Objekts parallel verändert werden müssen: So erfordert etwa die einfache „Vergrößerung" eines Objektes wie etwa in [Abb.6.5.2.1.1] die zeitgleiche Manipulation gleich 4 verschiedener Eigenschaften des Elements: So muss sowohl die Breite und die Höhe des Objekts entsprechend des angegebenen Skalierungsfaktors geändert, wie auch das Objekt in x- und y-Richtung verschoben werden, damit es „zentriert" bleibt und nicht einfach nach „rechts-unten" skaliert wird:

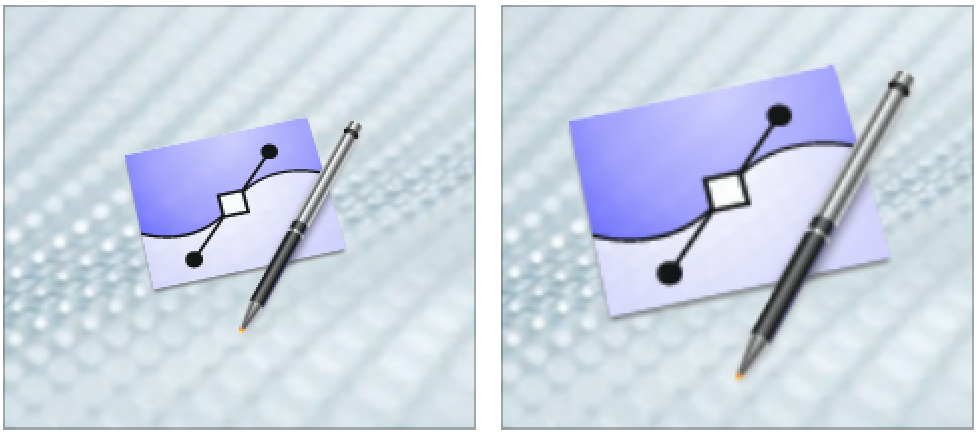


Abb. 6.5.2.1.1: „Zoom"-Funktion eines Bildsymbols bei gleichzeitiger Manipulation der x-, y-, height- und width-Attribute

Da dies, wie bereits eingangs ausgeführt, nicht mehr im Rahmen der durch SVG bzw. SMIL direkt bereitgestellten Syntax möglich ist, setzt sich das daraufhin für den *XML » SVG Presenter* entwickelte Animations-Framework aus zwei maßgeblichen Objekten zusammen: Dem Animations-Objekt (`AniObject`), welches sowohl das zu animierende Objekt wie auch die bereits erwähnten Animations-Eigenschaften (Animations-Dauer, zu verändernde Eigenschaft), wie auch einen Animations-*Tracker* (`AniTracker`), der die parallele Animation mehrerer solcher `AniObjects` ermöglicht. Der ECMAScript-Sprachumfang stellt hierfür etwa die Befehle `setTimeout` und `setInterval` bereit, mit deren Hilfe nun der `AniTracker` innerhalb beliebiger Zeitabstände die entsprechend im *Tracker* befindlichen AnimationsObjekte entsprechend ihrem Zielwert annähert. Ist eine Animation demnach abgeschlossen, so wird das entsprechende AniObject aus der Tracker-Liste gelöscht – ist auch diese leer, so ruft sich der Tracker nicht mehr selbst auf (rekursives Abbruchskriterium) und stellt seine Aktivitäten ein (so genannter *idle*-Zustand). Die Skalierungsfunktion aus [Abb. 6.5.2.1.1] lässt sich beispielsweise mithilfe dieser beiden Objekte nun relativ einfach realisieren:

```
svgObject = evt.getTarget(); // Zu animierendes Objekt ermitteln
...
var newX = getVal(obj, "x") - (oldWidth  * (factor-1)) / 2;
var newY = getVal(obj, "y") - (oldHeight * (factor-1)) / 2;
xAni = new AniObject(svgObject, "x", newX, duration);
yAni = new AniObject(svgObject, "y", newY, duration);
widthAni  = new AniObject(svgObject, "width",  oldWidth  * factor, duration);
heightAni = new AniObject(svgObject, "height", oldHeight * factor, duration);
...
at.add(new Array(xAni, yAni, widthAni, heightAni));
if(at.isIdle()) at.track();
```

Listing 6.5.2.1.2: Skalierungsfunktion mithilfe von AniObject *und* AniTracker.

Neben der freilich eher als „Spielerei" aufzufassenden „Zoom"-Funktion, die beispielsweise zur Vergrößerung von Icon-Dateien (im Prototyp momentan etwa eingangs verwendet) dienen könnte, kommt das Animations-Framework im Rahmen der eigentlichen Präsentation etwa bei den optischen „Fades" (bei Auswahl eines Items wird die Schrift weiß und der teiltransparente, schwarze Verlaufsbalken langsam eingeblendet) der Item-Selektion zum Einsatz:

Abb. 6.5.2.1.2: Einblendung des Verlaufsbalkens unter Verwendung von AniObject *und* AniTracker.

6.5.2.2 Darstellungs-Framework

Da, wie bereits eingangs erwähnt, nahezu sämtliche Ausgabe-Elemente entsprechend der geparsten XML-Daten erst zur Laufzeit berechnet und dargestellt werden können, wird zur tatsächlichen Präsentation freilich noch eine Sammlung grundlegender Funktionen für die korrekte Erzeugung der entsprechenden SVG-Elemente benötigt. Im Rahmen des *XML » SVG Presenter*s wird diese Aufgabe maßgeblich durch die zwei Kernfunktionen `expand()` und `collapse()` übernommen, die wiederum auf eine ganze Reihe grafischer Basic-Funktionen[1] zurückgreifen: Da im Normalfall[2] zu Beginn der Bildschirmpräsentation lediglich das so genannte *root*-Item, also die Wurzel des Präsentations-Baumes, via `folder()`-Funktion direkt dargestellt wird [s.Abb.6.5.2.1.2], fällt insbesondere der `expand()`-Methode die Hauptaufgabe zu, die Daten der bei Aktivierung jeweils auftretenden Teilbäume (und somit die Kind-Elemente des jeweiligen Knotens) in Form weiterer Item-Symbole korrekt darzustellen sowie alle unterhalb dargestellten Symbole entsprechend der y-Koordinate entsprechend „nach unten“ zu verschieben:

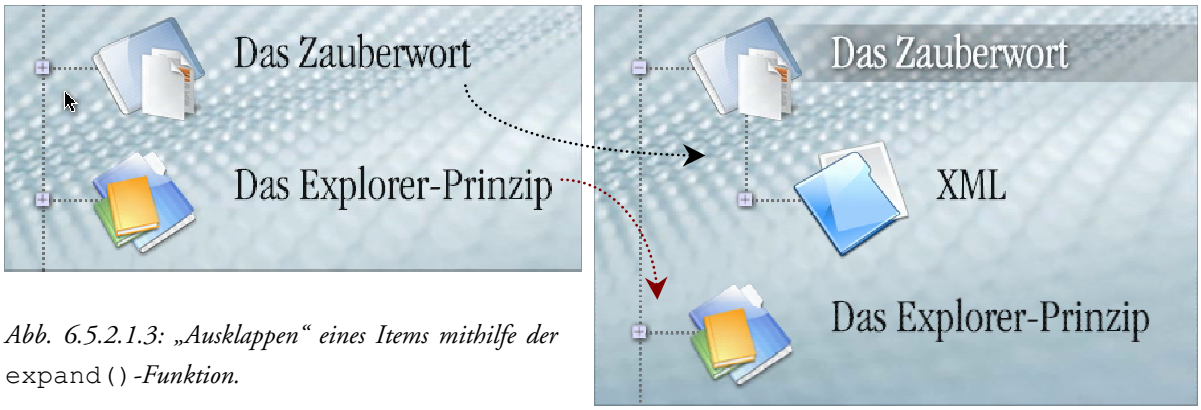


Abb. 6.5.2.1.3: „Ausklappen“ eines Items mithilfe der `expand()`*-Funktion.*

Die Beschreibung einzelnen der `expand()`-Funktion zugrunde liegenden Algorithmen und Methoden würde an dieser Stelle sicherlich zu weit führen – vereinfachend sei daher lediglich erwähnt, dass wiederum eine Anzahl „grafischer“ Funktionen die Basis für die Darstellung der verschiedenen sichtbaren Objekte (Icons, Textfelder, Rahmen etc.) darstellt, welche nun ihrerseits für die Erzeugung entsprechender SVG-Elemente[3] und deren korrekte Integration in die DOM-Hierarchie des SVG-Baumes verantwortlich zeichnen.

Die dem gegenüberstehende `collapse()`-Methode, die logischerweise die (rekursive) Funktion des *Schließens*[4] der entsprechenden Teilbäume übernimmt, ist naturgemäß zwar nicht auf die eben angesprochenen

[1] vgl. hierzu die Skript-Datei des Prototypen: folder, gradient, plus, hLine, vLine… [script.js]
[2] Anm: Alternativ hierzu kann bei Fehlen des *root*-Items auch direkt mit mehreren Items begonnen werden (*Direkt-Liste*).
[3] Hierfür steht im DOM-unterstützten ECMAScript wiederum die createElement-Funktion (bzw. createTextNode bei Textelementen) bereit.
[4] … oder „zuklappen“ (daher der Name collapse)

grafischen Funktionen angewiesen (da sie ebendiese Elemente einfach wieder löscht), muss jedoch ebenso wie ihr `expand`-Pendant entsprechende Verschiebungsoperationen durchführen, um die korrekte Positionierung der Baumknoten wiederherzustellen [vgl. hierzu Abb.6.5.2.1.3]

Abschließend erscheint im Rahmen der Umsetzungsbeschreibung hinsichtlich der soeben erläuterten „Auf- und Zuklapp"-Algorithmen noch der bereits in [6.3] theoretisch angesprochene *lineare* Präsentationsmodus erwähnenswert: Durch Aufruf der `nextItem()`-Funktion lassen sich sämtliche Baum-Knoten der Präsentation hintereinander „durchklappern" – bei Auftreten bereits expandierter „Unterverzeichnisse" oder noch ungeöffneter Teilbäume ruft diese Methode einfach die entsprechenden `expand()`- bzw. `collapse()`-Funktionen auf, und erzielt somit einen kontinuierlichen, linearen Präsentationsablauf. Entgegen der Befürchtung, diese Funktionalität könne aufgrund ihrer fehlenden Interaktivität bzw. Transparenz[1] zur Verwirrung des Publikums beitragen, hat sich nach mehreren Testläufen nicht bestätigt, da der Zuschauer gerade durch die hierarchische Baum-Eigenschaft des *Presenter*s die Orientierung offensichtlich nur in seltneren Fällen verliert.

6.5.3 Schwächen der SVG-Umgebung und Work-Arounds

Trotz der bereits in [5.4] sowie unter [6.4] unter dem Aspekt der Realisierung des *XML » SVG Presenters* diskutierten, beeindruckenden Möglichkeiten und Vorteile von SVG hinsichtlich Web-basierter Präsentationen sei an dieser Stelle jedoch noch auf einige weniger vorteilhafte Aspekte hingewiesen, die jedoch weniger die Formatspezifikation an sich, als vielmehr die momentan als Standard geltende [vgl. Schu02:60] SVG-Umgebung des *Adobe SVG Viewers* betreffen. Da jedoch neben der „reinen Theorie" einer konsequenten, sauberen Lösung natürlich im Rahmen der *Presenter*-Realisierung auch eine möglichst funktionelle und ästhetische Darstellung der in diesem Zusammenhang erstellten Präsentationen von Interesse ist,[2] haben diese im Folgenden behandelten, derzeitigen Mängel des *SVG Viewers* die Programmierung einer ganzen Reihe teils umständlicher „Work-Arounds" erforderlich gemacht. Neben der bereits erwähnten, eher programmiertechnisch relevanten „Namespace"-Lösung im Zusammenhang mit der fehlenden MSXML-Unterstützung [s.6.5.1] beziehen sich diese insbesondere auf die visuellen *Rendering*-Eigenschaften der SVG-Ausgabe des Viewers.

6.5.3.1 Phantomlinie

So muss derzeit etwa noch die Darstellung gestrichelter Linien, die ja im Rahmen des *XML » SVG Presenters* vermehrt zur Anwendung kommen, innerhalb des *Adobe SVG Viewers* als relativ fehlerhaft bezeichnet werden, da neben der eigentlichen Liniendarstellung stets eine nicht-vorhandene „Phantomlinie" mit dargestellt wird:

Abb. 3.5.1: Gestrichelte Linie in SVG mit Phantomlinie.

Fallt dies bei normaler Anwendung wie etwa in [Abb.3.5.1] noch nicht weiter ins Gewicht, verschlimmert sich die Darstellung dieser recht unansehnlichen Phantomlinie noch aus unerfindlichen Gründen bei dynamischer Erzeugung desselben Linienobjekts mittels EcmaScript. Da jedoch ebendies der Ausgangspunkt für den *XML » SVG Presenter* darstellt, musste im Rahmen der Umsetzung des Prototypen freilich eine alternative Lösung zur aus theoretischer Sicht ja naheliegendsten Verwendung des „gestrichelten" Linien-

[1] Phänomen: Man sieht nicht, an welcher Stelle die Maus die Teilbäume eröffnet etc.

[2] Dieser Umstand ist vergleichbar mit der Gegenüberstellung „Sauberen HTML-Codes" mit entsprechend von der diesbezüglichen Erwartung abweichenden Darstellung im Webbrowser (Browser-Orientierung statt Code-Orientierung) s. hierzu auch 3.2.3

Elements[1] gefunden werden. Aufgrund dessen kommt in der momentanen Version des Prototypen an dieser Stelle ein entsprechendes Rectangle-Objekt[2] mit Schachbrett-Muster zum Einsatz:

```
<pattern id="chess" patternUnits="userSpaceOnUse" width="4" height="4" x="0" y="0">
      <rect x="0" y="0" width="2" height="2" fill="dimgrey" />
      <rect x="2" y="2" width="2" height="2" fill="dimgrey" />
</pattern>
```

Listing 6.5.3.1: Verwendetes Schachbrett-Muster in SVG.

Im Rahmen der automatischen Erzeugung eines entsprechenden SVG-Elements kann nun auch unter ECMAScript-Zugriff einfach eine interne Referenz verwendet werden, um für das „emulierte" Linien-Objekt eine alternierende Musterung zu erhalten:

```
document.createElement("rect").setAttribute("fill","url(#chess)"); ...
```

Listing 6.5.3.2: Erzeugung eines Rechtecks mit entsprechendem Schachbrettmuster in ECMAScript.

Nachteil dieses „Work-Arounds" ist freilich, dass nun sowohl für horizontale als auch vertikale Linien (sonstige Linientypen sind überdies nicht mehr möglich) separate Erzeuger-Funktionen bereitgestellt werden müssen (`hLine()`, `vLine()`). Des Weiteren kann es natürlich auch im Rahmen dieser „Alternativ-Lösung" zu unerwünschten Darstellungsfehlern kommen.

6.5.3.2 Performance

Als weiterhin unvorteilhaften Umstand muss darüber hinaus auch die derzeit noch deutlich zu Wünschen übrig lassende Performance des *Viewer*-Systems von Adobe bezeichnet werden: Neben dem bereits negativ ins Auge fallenden Umfang des Plug-Ins (allein die Installationsdateien belaufen sich bereits auf 2.3 MB) ist insbesondere die Darstellung von *Animationen* (auch unter Verwendung der SVG/SMIL-eigenen Syntax) nur in den seltensten Fällen zufrieden stellend. Aufgrund des nicht zu unterschätzenden Rechenaufwands insbesondere hinsichtlich der Photoshop-ähnlichen Echtzeit-Filter, des bikubischen Anti-Aliasing[3] und des komplexen Objektmodells erscheint diese „Trägheit" zwar durchaus nachvollziehbar, muss jedoch in meinen Augen nicht kommentarlos hingenommen werden. Auch die teils erheblichen Performance-Probleme des konkurrierenden *Flash*-Viewers [s.4.5.1], der trotz des aus Rendering-Sicht weitaus günstigeren Dateimodells teils auch leistungsstärkere Systeme in die Knie zwingt, weisen wie beim diesbezüglich erheblich schwächeren SVG-Viewer auf eine bislang noch fehlende Anwendung Hardware-naher Algorithmen wie etwa der *Overlay*-Technik hin, wie sie etwa zur Darstellung von Audio/Video-Dateien bereits zur Anwendung kommen.

Neben den soeben ausgeführten Problemen, die allein mit Verwendung des als Standard geltenden[4] SVG-Viewers von Adobe verbunden sind, seien abschließend jedoch noch zwei weitere Kritikpunkte benannt, die sich wiederum primär an den derzeitigen Mängeln der SVG-Spezifikation selber festmachen: So stellt nach den bereits angesprochenen Schwächen der Sprache hinsichtlich paralleler Animationseigenschaften zudem der Umgang mit größeren Textmengen, die per se die Anwendung von Fließtext-Objekten erfordern, im Rahmen des derzeitigen SVG-Sprachumfangs eine „einzige Enttäuschung" dar [Cagl02:241].[5] Zum momentanen Zeitpunkt sind daher unter Anwendung des SVG-eigenen `<text>`-Elements lediglich einzeilige Texte möglich. Aufgrund dieser trotz der ansonsten recht beeindruckenden typografischen Möglichkei-

[1] Die Syntax, die Svg hierfür anbietet, ist der so genante „dasharray", der wiederum das eigentliche Linien-„Muster" determiniert.
[2] Zu Deutsch: Rechteck. Syntax: <rect x=".." y=".." width=".." height=".." etc./>
[3] Bem: Zur „ästhetischeren" Einbindung pixel-orientierter Grafiken, vgl. [Abb.6.3.2]
[4] vgl. [Schu02] p.60
[5] "Text in SVG is something of a Disappointment" [Cagl02] p.241

ten in Rahmen von SVG doch recht verblüffenden Nachlässigkeit der SVG-Entwickler war zur Realisierung des für die *Presenter*-Konzeptstudie vorgesehenen Zitat-Elements[1] [s.6.5.1] die Umsetzung eines weiteren, relativ umständlich zu programmierenden „Work-Arounds" erforderlich, welches die Fließtext-Eigenschaften längerer Zitate sozusagen „von Hand emuliert":

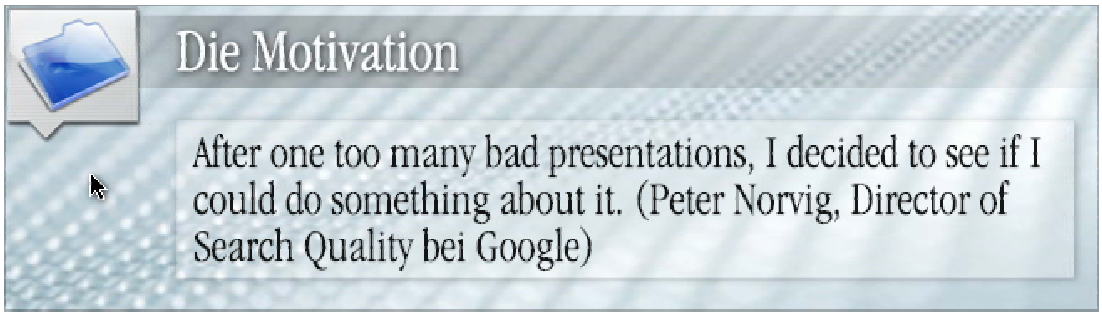


Abb. 6.5.3.3: Das <text>-Element für (auch längere) Zitate emuliert Fließtext.

Für diese Problematik haben die Entwickler der SVG-Spezifikation jedoch bereits Abhilfe angekündigt: So soll etwa die künftige Version des SVG-Standards 2.0 ein so genanntes `flowText`-Element beinhalten, das die soeben angemahnte Fließtext-Funktionalität endlich beinhalten soll [vgl. Watt02:1039ff]. Die seitens des W3C jüngst herausgegebene „SVG Roadmap" [Jack03a] macht jedoch deutlich, dass auch durch die Entwicklung der modularen Versionen 1.1 [FFJ03] sowie 1.2 der Veröffentlichungstermin der 2. SVG-Generation relativ weit nach hinten gerückt ist. Somit ist auch bezüglich des *XML » SVG Presenter*s davon auszugehen, dass die soeben dargelegte „Notlösung" zur Emulation von Fließtext noch einige Zeit bestand haben wird, auch wenn maßgebliche SVG-Entwickler „nicht ausschließen" wollen, ein entsprechendes, diesbezügliches „Hotfix" noch in der Ende dieses Jahres erwarteten Version 1.2 des Grafikstandards zu implementieren:

> *It seems likely that at least some automatic text wrapping functionality will be added in SVG 1.2*
>
> [Watt02:38]

6.6 Mögliche Erweiterungen und Verbesserungen

Neben den im vorangegangenen Kapitel diskutierten Schwächen sowohl der Viewer-Umgebung als auch der SVG-Sprache selber sollen natürlich auch die unter Betrachtung der derzeitigen Version des *XML » SVG Presenter*-Prototypen deutlich werdenden Mängel und Verbesserungspotentiale der hier vorgestellten Präsentationsläsung nicht unerwähnt bleiben: So beinhaltet die hiermit vorgestellte Konzeptstudie des *Presenters* bislang lediglich die Formatdefinition sowie den Prototypen der Präsentations-Komponente – zur Erstellung der Präsentationsdaten im XML-Format wurde bislang jedoch lediglich auf grafische XML -Editing-Tools [vgl. ShSi99, Tren02] zurückgegriffen, die eine sinnvolle Manipulation bzw. Neu-Erstellung einer Präsentation gemäß der durch die „Presentation Data"-Typdefinition vorgegebenen XML-Konvention erlauben:

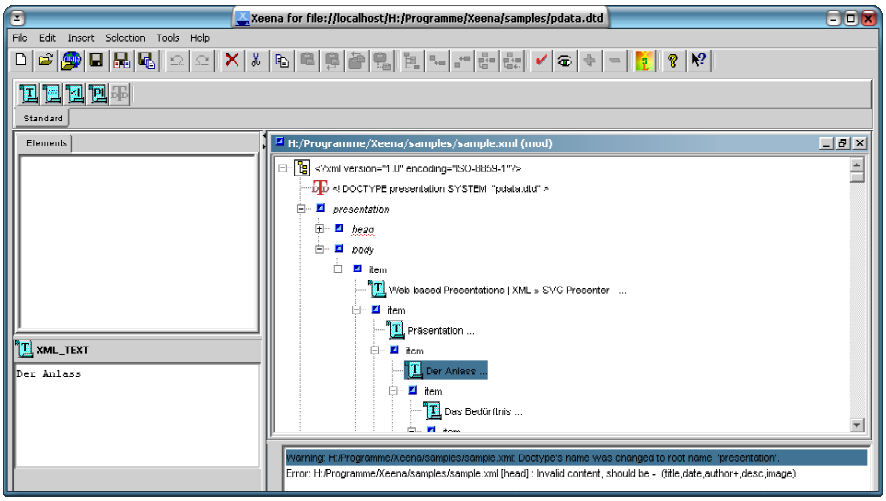


Abb. 6.6.1: Editierung der Präsentationsdaten mit Xeena [ShSi99]

Denkbar ist nun darüber hinaus freilich auch eine konsequente Umsetzung einer entsprechenden Editing-Komponente im Rahmen des *XML » SVG Presenters*, die anstatt der bislang zur

[1] Syntax: <text> … </text>

Anwendung kommenden, reinen Betrachtungs-, bzw. Präsentationsschnittstelle über zusätzliche Bearbeitungsmechanismen verfügt. Aufgrund der außerordentlichen Schlichtheit des *Presenter*-Formates ließe sich somit auch die Anzahl der umzusetzenden möglichen Befehle auf lediglich 3 so genannte *UI-Kommandos* reduzieren:

- Hinzufügen und Löschen einzelner Knoten.[1]
- Bearbeitung des Textinhaltes der einzelnen Knoten.
- Änderung des Icon-Elements einzelner Knoten.

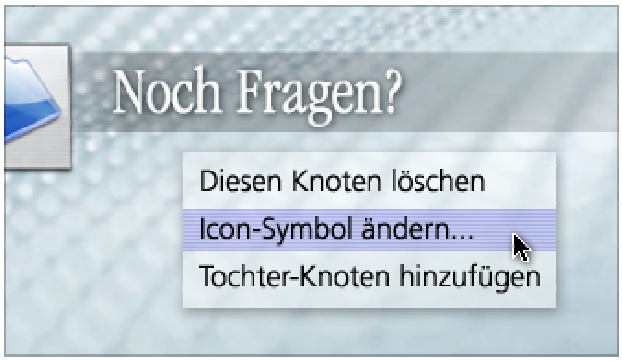
Im Zusammenhang einer Anwendungs-Suite, wie sie etwa PowerPoint darstellt, wären überdies programmatische Schritte wie das Speichern und Laden kompletter Präsentationen sowie das Umschalten vom Bearbeitungs- in den Präsentationsmodus etc. von Interesse.

Aufgrund der komfortablen SVG-Schnittstelle lässt sich vorab bereits feststellen, dass die Umsetzung dieser Schritte, zumindest aus konzeptioneller Sicht, geradezu „trivial" ausfällt und daher in der vorangegangenen Diskussion des Presenters bislang unerwähnt blieb. Die Editierung einzelner Textobjekte wäre bei entsprechender SVG-Erweiterung demnach entweder direkt möglich [s.Abb.6.6.2], oder aber – einiges umständlicher, wenn auch nicht sonderlich anspruchsvoll – über die `event.getCharCode()`-Routine auch mittels ECMAScript sehr leicht zu implementieren:

Abb. 6.6.2: Auswahl eines Knoten-Textabschnitts zur Bearbeitung

Auch das Löschen bzw. Hinzufügen von Knoten stellt unter diesem Aspekt keine Herausforderung dar – bei Aktivierung eines denkbaren „Bearbeitungsmodus" würde demnach das Anklicken einzelner Item-Objekte (besser: Kontextmenü via rechter Maustaste) wahlweise zur Löschung oder aber dem Hinzufügen weiterer (Child-)Knoten führen.

Abb. 6.6.3.1.1: Kontext-Sensitives Knoten-Auswahlmenü in SVG



Adobe's *SVG Viewer*-Implementierung erlaubt an dieser Stelle sogar die Modifikation des *Viewer*-eigenen Kontextmenüs [vgl. Adob02] über die DOM-Schnittstelle: So lassen sich sämtliche GUI-Elemente über das `window.contextMenu`-Zugriffsfeld äußerst leicht realisieren, da sämtliche Menü-Elemente ebenfalls in Form einer XML-Baumstruktur organisiert sind und sich somit sehr einfach ändern, löschen und zudem individuellen ECMAScript-Funktionen (via onactivate-Attribut) zuweisen lassen [vgl. Quin02]

[1] Dies umfasst somit sowohl <item>-, als auch <image>- und <text>-Elemente

Abb. 6.6.3.1.2: Modifiziertes, Viewer-natives Konextmenü

Eine weitere Option könnte hierbei überdies die Änderung des *Icons* über ein zusätzliches GUI-Element darstellen – die meiner Einschätzung nach sinnvollste Variante ließe sich etwa in Form eines Selektions-Menüs realisieren, das dem Benutzer wiederum eine Palette zur Verfügung stehender Bildelemente zur Auswahl stellt:[1]

Abb. 6.6.3.2: Icon-Auswahlmenü (rechts)

Die wohl am aufwendigsten zu realisierende Komponente einer derartigen Bearbeitungs-Schnittstelle stellt jedoch zweifellos die „Speicherung" der veränderten Baumstruktur dar: Obgleich sämtliche Daten durch die Echtzeit-gesteuerte Manipulation der Präsentationsdaten in Form einer verschachtelten Objektstruktur bereits im Arbeitsspeicher als lokale ECMAScript-Variablen vorliegen, erweist sich eine derzeit noch fehlende Schnittstelle zum *Abspeichern* dieser Daten als problematisch. Zwar bietet die aktuelle Adobe-Implementierung provisorische Linderung durch die (nicht-standardisierte) `postURL`-Funktion – wie bereits beim „Overloading" [2] des *Viewer*-Eigenen Kontextmenüs [s.Abb.6.6.3.2] besteht hier jedoch ein Konflikt hinsichtlich allgemeiner Kompatibilität, da auch diese Funktion lediglich auf Versionen des *Adobe SVG Viewers* anwendbar ist. Eine aus längerfristiger Perspektive interessante Alternative stellt hier die *Load-and-Save*-Spezifikation der DOM-Arbeitsgruppe dar [vgl. DOM02], die eine strukturell „Saubere" Implementierung erlaubt. Da diese Funktionalität jedoch erst *Working-Draft*-Status besitzt, wird offensichtlich noch einige Zeit verstreichen müssen, bis diese Schnittstelle standardisiert und auch in gängigen SVG-Viewern vollständig nutzbar sein wird [s. hierzu auch Watt02:1042]. Bis dahin ist die Realisierung einer derartigen Speicher-Funktion zwar durchaus technisch möglich, gestaltet sich jedoch aufgrund der soeben angesprochenen Problematik derzeit noch recht aufwendig. Aus diesem Grunde wurde bislang von einer Implementierung dieser Funktion (und damit auch der zuvor erläuterten,[3] „trivialen" GUI-Schnittstellen) abgesehen, da entsprechende, „saubere" Lösungswege erst in einiger Zeit verfügbar sein werden und ein umständliches, arbeitsaufwendiges „Work-Around" bis zu diesem Zeitpunkt nur wenig sinnvoll erscheint.

6.6.1 GUI-Komponente und Bearbeitungs-Komfort

Langfristig jedoch wird an der Realisierung einer Bearbeitungs-Schnittstelle freilich kein Weg vorbeiführen: Da sich der *Presenter* ja vorwiegend als Alternative zum augenscheinlich mangelhaften *PowerPoint* [4] positioniert, reicht der Bedienkomfort etwa mittels konventioneller XML-Editoren [ShSi99, s.Abb.6.6.1] im Hinblick auf die technisch und gestalterisch vorwiegend als „unbedarfte" einzustufende, potentielle Anwenderschaft [vgl. Scha90, Park01, Pirn01] natürlich bei Weitem nicht aus: Die im Rahmen der Konzeptstudie

[1] Die Auswahl externer oder lokaler Bilddateien wäre zwar technisch durchaus möglich, aber sowohl aus ästhetischer Perspektive (der Nutzer könnte hierbei wiederum unansehnliches Bildmaterial „hochladen") als auch Umsetzungstechnisch problematisch und zudem weitaus umständlicher zu realisieren.

[2] Mit dem „Überladen" ist die Modifikation bzw. Ersetzung der ursprünglichen Kontextmenü-Elemente durch Bearbeitungsspezifische Kommandos des *XML » SVG Presenters* gemeint.

[3] s. Abb. 6.6.2, 6.6.3.1.1, 6.6.3.1.2, 6.6.3.2

[4] s. hierzu Kap. 2.3

erzielte *Einfachheit* des internen Präsentationsformats sollte sich daher freilich auch in einer entsprechenden Benutzerschnittstelle (deren einzelnen Elemente ja soeben bereits skizzieret wurden) widerspiegeln, um in Konkurrenz zum auch UI-technisch überbordenden, „verwirrenden“ [1] PowerPoint-Monopolisten treten zu können – ein Versäumnis, das sich im übrigen auch die eher technisch versierten, XML-basierten Alternativen [HSL00,Herm02, Behz02, Sue02] zu „Schulden“ kommen lassen, da momentan keines dieser Lösungen über irgend eine Form einer Benutzerschnittstelle, geschweige denn eines ansprechenden GUIs verfügt und auch „Keynote“ [Appl02] diesbezüglich deutliche Schwächen aufweist [vgl. Frie02].

Einen deutlichen Vorteil würde der UI-orientierte Ansatz des *XML » SVG Presenters* zudem auch hinsichtlich benötigter Software-Werkzeuge genießen: Im Gegensatz zu sämtlichen derzeit verfügbaren Lösungen wäre auch die *Editing*-Komponente des *Presenters* ohne weitere Anforderungen direkt im SVG-Viewer-Umfeld anwendbar und zugleich auch (zukunftsorientiert) web-tauglich, da der Anwender sämtliche Änderungen direkt online vornehmen kann. Ein vergleichbares Modell existiert (jedoch vornehmlich auf *interaktive* Web-Anwendungen denn auf Präsentationen bezogen) bereits in Form des *ImpactBuilders*,[2] der die direkte Modifikation *Flash*-basierter Templates in einer Web-Umgebung erlaubt:

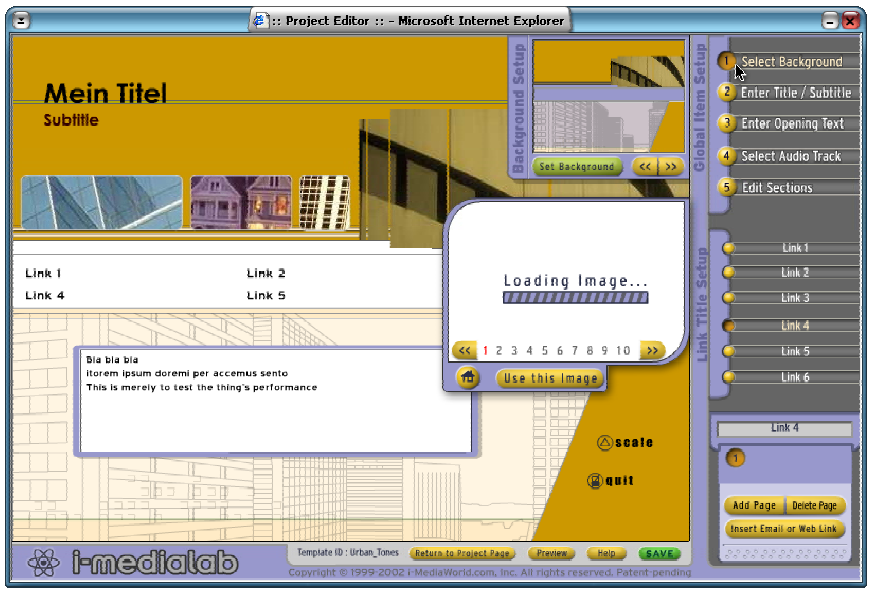


Abb. 6.6.4.1: Online-Bearbeitung eines Flash-Templates in ImpactBuilder Pro

Dass sich ebendieses Konzept des Online-Editing insbesondere bei Anwendung des SVG-Standards geradezu aufdrängt, beweist auch eine Implementierung des SVG-Entwicklers Adobe: Das vollständig online-betriebene SVG Draw [GBBW02] stellt beispielsweise ein Zeichenprogramm dar, welches beeindruckende (wenn auch rudimentäre) Illustrations-Funktionalität bietet, zugleich jedoch selbst vollständig in SVG (mit ECMAScript) realisiert wurde:

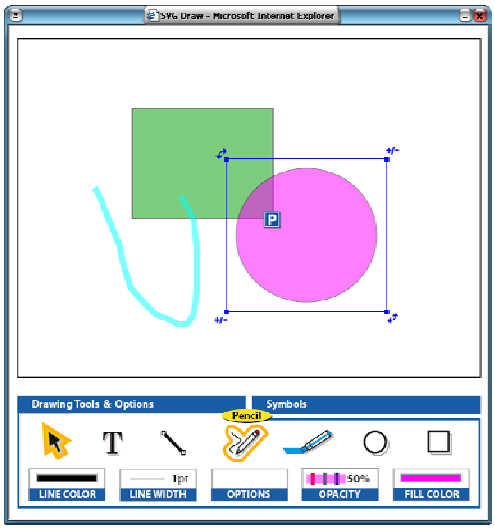


Abb. 6.6.4.2: Erstellung einer SVG-Zeichnung im selber SVG-basierten Adobe SVG Draw [GBBW02]

Die Web-Tauglichkeit des SVG-Formates wird somit nicht nur unter dem Blickwinkel des „Deployments“ (also der eigentlichen Darstellung) der Präsentation deutlich, sondern ebenso im Rahmen deren Erstellung: Beide Prozesse sind nun mithilfe des *XML » SVG Presenter*-Frameworks ohne zusätzliche Hilfsmittel sowie *direkt online* möglich und weisen überdies den Weg in Richtung Internet-basierter (und aufgrund der SVG-Eigenschaft) überdies ebenso web-*tauglicher* Präsentationen, die dank des aufgrund seiner „Restriktivität“ geradezu zwangsweise *schlichten* XML-Formates darüber hinaus „simplistisch“[3] zu bearbeiten sind

[1] Die Benutzerschnittstelle des Programms lässt die Anwender durch die „schier erschlagende Fülle an Optionen“ [Treu95] eher „verwirrt und überwältigt“ [FlRi01:7] zurück. [s.2.3.2.2]

[2] URL: http://www.impactbuilder.com [30.1.2003]

[3] Direkte Übersetzung des englischen „Simplistic“ (für das wiederum kein passendes, deutsches Wort existiert)

und sich somit auch stets innerhalb einen gewissen ästhetischen Grundrahmens bewegen.[1]

Speziell im Hinblick auf die Darstellungs-Eigenschaft des *Presenters* gilt es jedoch noch einen weiteren Aspekt der Bildschirmpräsentation zu beachten: Den bereits in [6.3] theoretisch erwähnten Vorteil der *Übersichtlichkeit* aufgrund der Baumstruktur der Präsentation, nach dem ähnlich wie im Windows-Explorer stets die aktuelle Position sowie die noch kommenden Elemente der Präsentation transparent bleiben. Falls jedoch aufgrund der erhöhten Anzahl auftretender Sub-Elemente die Sicht auf über- bzw. untergeordnete Elemente versperrt (indem diese beispielsweise nach oben oder unten und somit u.U. außerhalb des Bildschirmrands „geschoben" werden) würde ebendiese Eigenschaft jedoch verletzt, da die zur Orientierung notwendigen Objekte nicht mehr sichtbar wären. Im Rahmen eines *interaktiven*, Screen-Basierten „Durchlaufens" der Präsentation würde dies jedoch noch durch die Möglichkeit des „Aus-Zoomens" kompensiert, da der Nutzer stets zugriff auf die „Magnification"-Funktion des SVG-Viewers besitzt. So kann etwa durch einfaches Auszoomen des Anzeigebereiches die Sicht auf alle relevanten Objekte wiederhergestellt werden.

Einer eher vortragsorientierten Präsentation stünde diese Möglichkeit jedoch *nicht* offen, da sowohl die Schrift in der „verkleinerten" Ansicht schwerer zu lesen wäre, als auch das „Zoomen" eher zur Verwirrung der Zuschauer beitragen würde, als dass es der Übersichtlichkeit dienlich wäre. Eine unter diesem Aspekt gangbare Lösungsmöglichkeit könnte somit etwa darin bestehen, dem aktivierten Element voranstehende *Items* (die ja bei linearem Vortrag ihrerseits bereits aktiviert wurden) aus dem Darstellungsbaum rekursiv zu entfernen, damit lediglich die jeweils relevanten Über-Kapitel zur besseren Transparenz sichtbar bleiben:

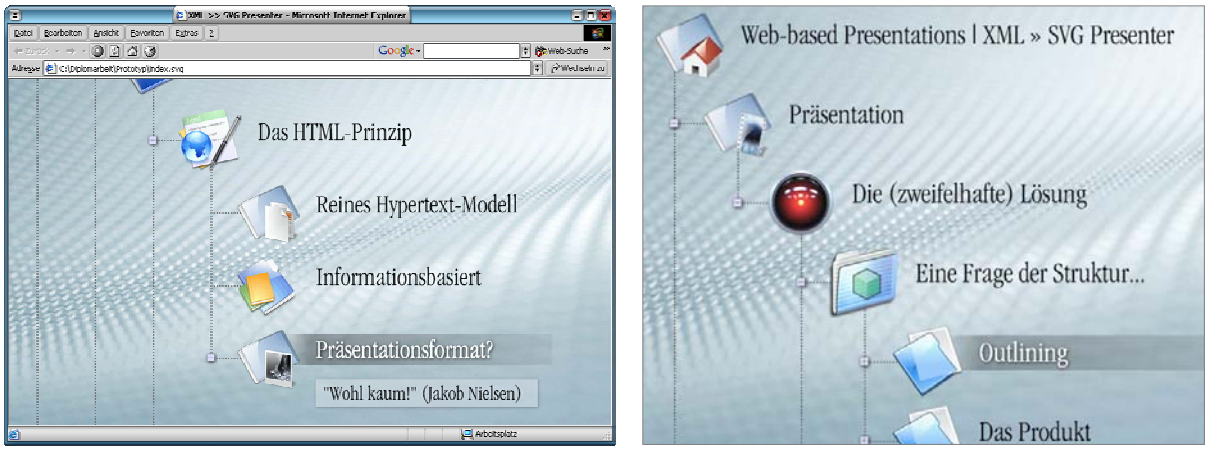


Abb. 6.6.5: Eleminieren redundanter Items zur Wiederherstellung der Übersichtlichkeits-Eigenschaft

Natürlich ist diese Lösungsmöglichkeit lediglich unter bestimmten Vorraussetzungen und auch nur in begrenztem Umfang interessant: Bei zunehmendem Umfang der Präsentationsdaten selber und somit zugleich ansteigender Verschachtelung stieße überdies nicht nur das eben skizzierte „Work-Around", sondern ebenso der gesamte *XML » SVG Presenter* selbst an seine Grenzen, da das gesamte System sich derzeit lediglich für konsequent strukturierte Präsentationen von begrenzter Komplexität anbietet. Derzeit beschränkt sich der Umfang des *Presenters* etwa auf Vorträge von bis zu 30 Minuten –darüber hinausgehende Präsentationsdaten wären derzeit bei gleich bleibender Darstellungsgröße hingegen nicht mehr sinnvollerweise zur Bildschirmpräsentation geeignet. Ein „Trick", der sich wiederum an dieser Stelle anbieten würde, könnte etwa die Hyperlink-Eigenschaft SVG-orientierter Objekte darstellen: So könnten einzelne Äste des Präsentationsbaumes wiederum auf weitere, ihrerseits jedoch separate Präsentationen bzw. –Teilbäume verweisen, um die Darstellbarkeit der Präsentationsdaten wiederherzustellen. Dies würde freilich wiederum über die Eigenschaften *dramaturgisch strukturierter* Präsentationen hinausgehen, da sich diese Herangehensweise eher an

[1] PowerPoint'sche „Design-Katastrophen" [vgl. Kap. 2.3] können hierdurch verhindert werden.

das Hypertext-Prinzip anlehnt und, wie bereits in [Kap.3] ausführlich diskutiert,[1] somit als nur „begrenzt Präsentationstauglich" zu bezeichnen wäre. Eine interessante Erweiterung (wenn auch aus eben diesen Gründen bislang nicht implementiert) stellt diese (sehr einfach zu realisierende) Funktion jedoch durchaus auch im Hinblick auf Web-orientierte Präsentationen dar.

6.6.2 Header-Informationen: Dublin Core

Schließlich die letzte Erweiterung, die im Rahmen insbesondere eines *Web*-basierten Präsentationsansatzes interessant wäre, könnte die Integration des so genannten „Dublin Cores" [2] [vgl. OyGo01] in das bislang (ebenso wie der body-Teil) „trivial" gehaltene Header-Segment der Präsentations-DTD selber darstellen: Ähnlich wie in der SlideML-Spezifikation [Fisc02] nahegelegt, könnten den Internet-Suchmaschinen auf diese weise Standardisierte, umfassende Meta-Informationen zur Verfügung gestellt werden, die eine entsprechende Indizierung „normativ" begünstigen würde. Aufgrund der Tatsache, dass im Moment jedoch das *Gros* der Suchmaschinen weder das (zukunftsweisende) Dublin Core-Modell noch die Indizierung XML-basierter SVG-Dateien unterstützt [vgl. NSW02], stellt auch diese mögliche Erweiterung eher ein „Nice-to-have-Feature" für zukünftige Versionen dar. Da auch der pdata-Hauptteil derzeit „maximal trivial" gehalten ist, würde eine derartige Integration überdies ein seltsames Ungleichgewicht zwischen header- und body-Komponente des Definitionsteils darstellen, weshalb an dieser Stelle erst einmal von einer derartigen Erweiterung abgesehen und der *Presenter*-Prototyp zunächst in seiner derzeitigen Form belassen wurde.

6.7 Fazit

Mit dieser Diplomarbeit liegt in Form des XML » SVG Presenters nun ein funktioneller Prototyp eines Präsentations-Frameworks vor, welches, abweichend von bisherigen („Slide-basierten") Präsentationskonzepten, auf der Grundlage des W3C-Vektorstandards SVG, die Erstellung in Gestalt einer hierarchischen Struktur dramaturgisch aufgebauter Präsentationen in Web-gerechter Form ermöglicht.

So trägt die (zur Erzielung möglichst ästhetischer Ergebnisse) gestalterisch „restriktive", wie auch zur einfachen Erstellung der Präsentationsdaten äußerst *schlichte* Dokumentenstruktur zum schnellen und leichten *Authoring* möglichst überzeugender, dramaturgisch strukturierter Präsentationen bei. Die hierbei zur Anwendung kommende Verzeichnismetapher garantiert einerseits publikumsbezogene Vertrautheit, wie auch eine verbesserte Übersichtlichkeit und Navigation der Präsentation im Sinne des *Usability*-Konzepts. Zu einer möglichst intuitiven, visuellen Erstellung *Presenter*-basierter Präsentationen sind überdies verschiedene GUI-Module angedacht, deren Grundkonzepte im vorangegangenen Kapitel erläutert wurden.

Die hiermit vorliegende Designstudie stellt natürlich derzeit keinen „ernst gemeinten „Konkurrenten" zu etablierten Business-Lösungen wie PowerPoint dar, soll aber eine konzeptionelle Alternative in Form eines prototypischen „proof-of-concept"-Modells aufzeigen: Da, wie in den vorherigen Kapitel dieser Diplomarbeit beleuchtet, derzeitige „Lösungen" weder dem Konzept des *World Wide Web* (im Sinne eines Informationsmediums), noch den Möglichkeiten und strukturellen Anforderungen überzeugender Präsentationen jeweils vollständig Rechnung tragen, ist dieser Prototyp somit primär als Versuch eines praktischen *Beweises* zu werten, dramaturgisch aufgebaute, durch ästhetische Darstellung auch emotional überzeugende Präsentationsansätze mit den Erfordernissen des Web vereinen zu können, und zu diesem Zwecke überdies die Vorzüge des hierfür derzeit am ehesten geeigneten Formates SVG zu veranschaulichen.

---

[1] s. hierzu insbesondere 3.2.1 sowie 3.2.2

[2] Anm: Dieses Modell definiert einen Satz normierter Metadaten, die (zumeist im Rahmen von Datei-*Headern*) in Webdokumente eingepflegt werden können

7 Zusammenfassung und Ausblick

Vor einem kurzen Ausblick, was zukünftige Entwicklungen gerade im Hinblick auf Web-basierte Präsentationstechnik noch bringen mögen, empfiehlt sich an dieser Stelle freilich noch ein kurzer Rückblick zur Zusammenfassung der im Verlaufe dieser Diplomarbeit gewonnenen Erkenntnisse, insbesondere um deren Relevanz hinsichtlich der Konzeption des *Presenter*-Prototypen zu verdeutlichen.

7.1 Ein Blick zurück…

So erscheint insbesondere rückblickend die Existenz auch heutigen (etwa im Rahmen des 1. Kapitels formulierten) Ansprüchen gerecht werdender, sehr früher Ansätze verblüffend: Insbesondere das *HyperCard*-Modell [s.2.1]erfährt überraschenderweise gerade im Zusammenhang mit hochaktuellen Themen, wie etwa „optimaler Web-Usability" [Niel99:71] oder aber des „Prinzips der Schlichtheit" im Rahmen des SVG-Formates [vgl. Cagl02:12f] auch in der jüngeren Vergangenheit lobende Erwähnung. Nicht zuletzt deswegen finden sich grundlegende Prinzipien ebendieses-Ansatzes auch wieder im Rahmen des vorliegenden Prototypen wieder:

> *The HyperCard principle is one I personally think has been forgotten (or deliberately displaced) by software vendors. Too many software tools have sought to create expensive "solutions" that are incredibly complex and only manageable by other tools (which of course those vendors provide).*
>
> [Cagl02]

7.1.1 Bisherige, problematische Standards

7.1.1.1 Das Prinzip PowerPoint

Dennoch konnten auch in diesem Falle die konzeptionell überzeugenden Aspekte dieses, wie auch des ebenfalls sehr frühen[1] Präsentations-Ansatzes Dave Winers [vgl. Wine88] eine markttechnische Verdrängung durch zwar nicht unbedingt technisch überlegene, aber aus unternehmensstrategischer Sicht weitaus geschickter vermarktete Produkte nicht verhindern [s.2.3.1]: Insbesondere *Microsofts* unangefochtene Monopolstellung am Markt „konventioneller" Präsentationssoftware erscheint angesichts des in dieser Diplomarbeit diskutierten, kritischen Anwendungsprinzips PowerPoints (bezogen sowohl auf den „fragwürdigen" [vgl. Sear98, Stew01] Inhalt als auch der „eigenwilligen" [Godi01:3] Ästhetik der erzielten Ergebnisse) nur unter Berücksichtigung der erfolgreichen, aggressiven Marketing- und Verdrängungsstrategie des Unternehmens verständlich. Darüber hinaus macht auch eine nähere Betrachtung des zugrunde liegenden Formates unbefriedigende Zugriffs-Schnittstellen sowie die recht begrenzte „Web-fähigkeit" PowerPoint-basierter Präsentationen deutlich.

Die entwicklungshistorische Untersuchung des der WWW-Suite zugrunde liegenden HTML-Formates macht indes deutlich, dass auch die Hypertext-Markupsprache zwar grundsätzlich aufgrund logischer Auszeichnungselemente eine konsistente Strukturierung wissenschaftlicher Dokumente erlaubt, trotz existenzieller Erweiterung jedoch bereits vom Grundprinzip her dem Konzept primär visueller, gezielt „gestalteter" Präsentationen eher entgegensteht. Intensive Bemühungen so genannter „Designer" [vgl. Rieh01] drängten das Dokumentenformat zwar durch die Einführung visueller Auszeichnung, JavaScript, DHTML und insbesondere gezielten „Missbrauch" [Muen98] der HTML-Syntax verstärkt in die Richtung eines „grafischen Mediums" [s.3.2.1] und führten so dessen ursprüngliche Strukturierung (trotz verzweifelter „Gegenmaßnahmen" wie CSS oder XHTML) völlig *ad absurdum* – an der grundsätzlich *beschränkten Eignung* des HTML-Formates in Bezug auf Multimedia-Präsentationen konnte dies jedoch nur wenig ändern.

[1] Anm: Winer veröffentlichte seine Konzepte zum strukturierten *Outlining* bereits Anfang der 80er Jahre [vgl. Wine88]

7.1.1.2 Das Web als bislang zweifelhafter Carrier für Multimedia-Präsentationen

An dieser Stelle setzte im Verlaufe der Entwicklung nun die so genannte Plug-In-Schnittstelle wie auch die *Applet*-Plattform der Programmiersprache Java an, die eine *Portierung* konventioneller Multimedia-Ansätze und –Formate in die Browserumgebung ermöglichten – aufgrund dürftiger Plug-In-Verbreitung und verschiedenen, problematischen Aspekten beider Ansätze jedoch mit nur mäßigem Erfolg. Lediglich die „übergewichtige" [3.5.3] *Director*-Portierung *Shockwave* sowie das, dank Vektoreigenschaft, weitaus schlankere Flash erlangten an dieser Stelle zufrieden stellende „Plug-In Penetration" [vgl. Macr02b]. Aufgrund dessen bietet sich das kompakte, proprietäre SWF-Format zwar für überwiegend Animationsbasierte Web-Präsentationen durchaus an, muss jedoch aufgrund der problematischen, binären Dateistruktur und mangelnder Internet-Usability [vgl. Niel99] als „nur begrenzt Web-gemäß" bezeichnet werden. Insbesondere das „Frame-Diktat" der Formatarchitektur sowie der darauf aufbauende „Animatimationszwang" des Flash-Anwendungsprinzips haben in der Vergangenheit überdies zu „selbstverliebten" [Lipm00] und speziell im Amateurbereich [vgl. True01] „ästhetisch problematischen" [Zmoe00] Flash-Anwendungen geführt, was wiederum eine Reihe kritischer Beobachter [Ragu99, Niel99, Fren02] zu dem Schluss verleitet, dass die Flash-Technologie dem „Web-Gedanken" grundsätzlich entgegenstehe [s.4.5.4].

7.1.2 Die Zukunft im Web: XML

7.1.2.1 XML-basierte Vektor- und Multimediaformate

Die darauffolgende Betrachtung speziell XML-basierter Vektorformate machte zwar deutlich [s.5.3.1-3], dass entsprechende, textbasierte Ansätze zwar die Vorzüge des vektorbasierten Flash mit Web-spezifischen Eigenschaften des HTML-Formates durchaus zu verbinden wissen – die entsprechenden, beim W3C als Standard-Vorschläge (*Notes*) eingereichten Formatentwürfe (PGML, VML…) wurden jedoch bedauerlicherweise, insbesondere aufgrund der strategischen Blockadehaltung der Microsoft/Macromedia-Allianz zunächst auf Eis gelegt. Das aus diesen Bemühungen erwachsene SVG-Format konnte aufgrund dessen erst spät [Svg01] als Standard verabschiedet werden, überzeugt bei näherer Betrachtung jedoch durch beeindruckende, grafische Funktionalität, gepaart mit den „web-gemäßen" Vorzügen der XML-Konvention. Daher stellt das W3C-Vektorformat speziell in Verbindung mit dem (leider nur mäßig erfolgreichen) SMIL ein durchaus interessantes Modell zur Realisierung Web-basierter Multimedia-Präsentationen dar.

7.1.2.2 Formatseparation und SVG-Präsentationsschwemme

Neben der faktisch bislang enttäuschenden Verbreitung der jeweiligen Format-Clients [s.5.4.7.3] stellt an dieser Stelle hingegen die nur rudimentäre Repräsentation abstrakter, logischer Verknüpfungen bzw. eine bislang ungenügende Trennung von Darstellung und Inhalt das Hauptproblem SVG-basierter Präsentationen dar. Zur Lösung dieser Problematik unterstützt das SVG-Framework via XSLT-Stylesheets, oder besser, mithilfe einer prozeduralen DOM-Verarbeitung jedoch die *Separation* dieser beiden Komponenten unter Hinznahme eines weiteren, internen Präsentationsformates – ein Ansatz, den nicht nur der dieser Diplomarbeit zugrunde liegende Prototyp verfolgt, sondern ebenso eine „schier unüberschaubare" [Sue02] Anzahl weiterer, SVG-basierter Präsentationslösungen [HSL00, Fisc02, Herm02]: Da sich das SVG-Format für Web-basierte Präsentationen geradezu anzubieten scheint, haben sich die Entwicklung diesbezüglicher Ansätze zu einem wahren „Hello-World des SVG"[1] entwickelt.

Ein höchst problematischer Aspekt dieser „epidemisch hervorsprießenden" [Sue02] Präsentationslösungen stellt angesichts der technisch in der Regel doch eher unbewanderten PowerPoint-Anwenderschaft in meinen Augen jedoch die ausgesprochen hohe Komplexität der jeweiligen, internen Präsentationsformate dar: So setzt das Gros der zugehörigen Dokumentstrukturen ein extrem hohes, technisches Kompetenzniveau

[1] vgl. [Sue02]

zur Bearbeitung der jeweiligen Präsentationsdaten voraus, von dem jenseits des akademischen Anwendungsgebeites jedoch keinesfalls ausgegangen werden kann. Überdies zeigen die derzeitigen Formatentwürfe durch eine sehr enge Annäherung an das eher problematische [s.3.2.1], „Slide-basierte" PowerPoint-Prinzip keinerlei konzeptionelle Alternative zur Erstellung Web-basierter Präsentationen auf.

7.1.3 Der XML » SVG Presenter

Hier setzt nun der im Rahmen dieser Diplomarbeit entwickelte *XML » SVG Presenter* an, der an dieser Stelle strukturell einen anderen Weg beschreitet: Angelehnt an das in groben Zügen bereits Anfang der 80er Jahre von Dave Winer [Wine88] formulierte Konzept dramaturgisch gegliederter, hierarchisch strukturierter Datenstrukturen, („Outlining") werden im Rahmen der so genannten „Minimal-Komponente" (als Präsentationsdatenformat) keine linearen, PowerPoint'schen Bullet-Charts, sondern vielmehr *verzweigte Baum-Hierarchien* erstellt. Dies entspricht vom Grundsatz her freilich nicht nur dem XML-Prinzip in geradezu idealtypischer Weise, sondern setzt sich überdies über das „ermüdende Slide-Stakkato" [Park01] bisheriger Ansätze hinweg. Da die interne Dateistruktur der Präsentationsdaten überdies radikal vereinfacht und somit auf eine wenige, „triviale" Elemente reduziert wurde, gestaltet sich die Bearbeitung der intuitiv verständlichen Präsentationsdaten zwar geradezu „trivial" – gleichzeitig können jedoch alle elementaren Anforderungen (Überschriften, Stichworte, Schaubilder, Zitate…) im Rahmen dieses ansonsten restriktiven Frameworks abgebildet werden.

Das Darstellungs/GUI-Konzept lehnt sich derweil an die Verzeichnis-Metapher moderner Dateiverwaltungssysteme [vgl. Wine99] an: So lässt sich die durch die interne Dateistruktur „erzwungene" Hierarchie einerseits auf Basis einer bereits durchaus vertrauten Bildmetaphorik konsequent abbilden. Durch dieses Darstellungsparadigma bleibt nicht nur die Übersichtlichkeit der dargestellten Präsentationsstruktur, sowie beim Interaktiven „Erkunden" eine explorative „Neugierde" gewahrt – im so genannten „Tree-Walking" ist überdies ein lineares „Durchschreiten" der gesamten Hierarchie nach vertrautem Muster möglich.

Durch die automatische, clientseitige Umwandlung der Präsentationsdaten wird zugleich eine stets „ästhetische" Darstellung der entsprechenden Inhalte förmlich „erzwungen", da über die Auswahl explorativer Bildsymbole („Icons") die Gestaltungsmöglichkeiten explizit gering gehalten werden: Zur Illustration der dargelegten Sachverhalte sowie der Integration „emotionaler Bildinhalte" [nach Godi01] ist lediglich eine Vollbild-Darstellung einzelner, externer Bildquellen möglich. Da der Nutzer jedoch keinerlei Einfluss auf gestalterische Kriterien (Positionierung, Größe…) nehmen kann, besteht (überspitzt formuliert) auch nicht die Möglichkeit, dass der ästhetische Gesamteindruck „verpfuscht" wird.

Zusammenfassend liegt mit dem *XML » SVG Presenter* somit ein prototypisches Präsentationsframework vor, welches die Formulierung hierarchisch strukturierter, primär textorientierter Präsentationen erlaubt. Die „triviale", interne Dateistruktur ermöglicht sowohl ein intuitives „Authoring" wie auch eine stets ästhetische (da automatisch zur Laufzeit erzeugte) Bildschirmdarstellung und führt somit das Hauptaugenmerk des Erstellungsprozesses überdies auf die dramaturgische (da baumartig verschachtelte) Strukturierung argumentativer Präsentationsdaten zurück („Outlining-Ansatz").[1]

7.2 Ausblick

Auf Basis der Verlaufe der dieser Diplomarbeit gewonnenen Erkenntnis, dass die offenen, XML-basierten Multimedia Standards SMIL und insbesondere SVG hinsichtlich *tatsächlich Web-gemäßer, multimedialer Präsentationen* [s.Kap.1] überzeugende Vorzüge gegenüber derzeit dominierenden, teils jedoch problematischen Präsentationslösungen wie *Flash* oder *PowerPoint* aufweisen und somit hervorragende Trägermedien

[1] vgl. hierzu [Wine88]

etwa für den im Rahmen dieser Diplomarbeit vorgestellten Prototypen *XML » SVG Presenter* darstellen, ist den unverdienterweise bislang noch weitgehend unbekannten Formaten für die Zukunft eine weitaus erfolgreichere Akzeptanz und Verbreitung zu wünschen, als dies in der „nicht gerade rosigen" Vergangenheit [vgl. Arci02][1] der Fall war.

7.2.1 SVG als wegweisendes Präsentationsformat

Insbesondere im Hinblick auf visuell orientierte Präsentationslösungen und grafische Web-Anwendungen hat speziell der SVG-Standard meiner Einschätzung nach durchaus das Potential, in diesem Bereich das bisherige „Universal"-Format HTML abzulösen

> *SVG will succeed because it satisfies the need for an easy-to-use, inexpensive, dynamic, nonproprietary way to build graphical interfaces, [which is] the key that will likely make it the HTML of this decade.*
>
> [Cagle02:12f]

Wie bereits in [s.5.4.7.4] diskutiert, ist es für einen diesbezüglich möglichen, zukünftigen Erfolg des Formates jedoch unabdingbar, so bald als möglich insbesondere drei problematische Teilaspekte konsequent anzugehen: Zum Einen erhoffe ich mir durch den für die unmittelbare Zukunft angekündigten SVG-Support maßgeblicher Softwareprodukte (Microsoft Office,[2] Adobe LiveMotion, ImageStyler etc.) einen erheblichen „Anschub" für den Vektorstandard hinsichtlich „notwendiger" [Dumb01] Authoring-Tools – im Hinblick auf momentan teils noch „lückenhafte" Funktionalität ruhen meine Hoffnungen hingegen, neben derzeit bereits praktikablen „Work-Arounds"[s.6.5.3], speziell auf den angekündigten Erweiterungen [vgl. Watt02:38,1032ff] der für Ende dieses Jahres [vgl. Jack03a] zu erwartenden Folgeversionen 1.2 und insbesondere der – leider erst erheblich später erscheinenden – zweiten Generation des viel versprechenden Vektorstandards.

7.2.1.1 Die Zukunft des Formats auf der Kippe

Insbesondere jedoch führt, um ein bereits prophezeites [vgl. Foss03], frühzeitiges Versinken des Formates in „völlige Bedeutungslosigkeit", an einer möglichst baldigen Integration SVGs in führende Webbrowser kein Weg vorbei: Ohne, dass etwa Microsofts Internet Explorer neben den derzeit bereits integrierten Vektorformaten VML und Flash auch SVG-Grafiken ohne Plug-In-Installation nativ darstellt, so meine Befürchtung, wird dem verheißungsvollen Standard wohl der Durchbruch verwehrt bleiben [s.5.4.7.3]. Angesichts der bereits jetzt vollständigen Flash-Dominanz [vgl. Macr98b] wird es SVG überdies nicht gelingen, sich gegen den durchaus nicht unproblematischen [s.4.5.4] Marktführer – auch langfristig – durchzusetzen, wenn eine derartige, stets vage angekündigte [Watt02:33] Integration nicht *bald* vonstatten geht: Bereits jetzt reagiert „Platzhirsch" [Curt01] Macromedia durch deutliche XML-Erweiterung der Flash MX-Generation sehr aggressiv [vgl. Fest02] auf die gefährliche „Alternative" [Bem02] SVG und droht dabei, den ungeliebten (da offenen, nicht-proprietären und somit im Gegensatz zu Flash kostenlosen)[3] Konkurrenz-Standard bereits vor dessen „voller Entfaltung" durch „erdrückende Marktbeherrschung" sowie durchschaubare Bemühungen, das eigene SWF-Format „auf webfähig zu prügeln" [vgl. NSW02], wieder in der Versenkung verschwinden zu lassen.

7.2.2 Einsichtige Microsoft-Entwickler?

Einen anderen Weg scheint indes der bisherige Präsentations-„Matador" Microsoft zu beschreiten: Insbesondere die angekündigte *Integration* des SVG-Formates etwa in die bislang problematische [s.2.3] PowerPoint-Software [vgl. Paol02] könnte nicht nur dem heftig umstrittenen „Präsentationsstandard" den Aus-

[1] "Less-than-stellar past." [Arci02]
[2] vgl. [Paol02]
[3] vgl. [Cagl02] pp.12ff

weg aus dessen derzeitigem, konzeptionellen[1] sowie formattechnischen[2] Dilemma weisen, sondern auch dem SVG-Format endlich zu wohlverdienter Blüte verhelfen: Nicht ausschließlich würde hierdurch der bislang „indiskutable" [Will02] Web-Export PowerPoints durch die überzeugende Funktionalität des SVG-Standards deutlich aufgewertet – durch die Weiterverarbeitbarkeit SVG-basierter Grafiken sowie des ausgesprochen „schicken" Renderings des SVG-Viewers könnte überdies auch exportierten PowerPoint-Präsentationen zu neuen (insbesondere auch) ästhetischen Möglichkeiten verholfen werden.

Die Entwicklung im Bereich multimedialer Web-Formate dürfte, angesichts der im Rahmen dieser Diplomarbeit untersuchten, bereits in der Vergangenheit äußerst dynamischen Entwicklung somit auch für die Zukunft interessant werden, insbesondere – wie eben zuvor ausgeführt – im Hinblick auf multimediale-Web-basierte Präsentationen.

In Gestalt des *XML » SVG Presenters* liegt mit dieser Diplomarbeit überdies ein funktioneller Prototyp sowie ein „konzeptionelles Framework" vor, welches, anders als die eingangs beleuchteten, „konventionellen" Präsentationsansätze eine Alternative hinsichtlich der Realisierung dramaturgisch bzw. hierarchisch strukturierter, Webfähiger Präsentationen aufzeigen soll. Obgleich als „proof-of-concept"-Studie durchaus bereit einsatzfähig, versprechen hingegen erst die bereits als „Follow-up" dieser Diplomarbeit geplanten, in [6.6] diskutierten GUI-Erweiterungen einen konsequenten Ausbau dieses momentan noch prototypischen Ansatzes zu einem einfachen, intuitiv zu bedienenden Präsentations-Erstellungs-Tool.

7.2.3 XML » SVG Presenter als alternativer Vorschlag

Die zukünftige Entwicklung auf dem Web-basierten Präsentations-Markt bleibt daher, nicht nur im Hinblick auf den vorgestellten *Presenter*, äußerst spannend, denn der Markt ist gerade in letzter Zeit deutlich in Bewegung geraten: War ebendieser Sektor noch in den letzten zehn Jahren primär durch starre, unbewegte Marktdominanz des sowohl theoretisch als auch technisch „unbefriedigenden" PowerPoint gekennzeichnet [s.2.3.2-3], so hat sich nicht zuletzt mit der Einführung von SVG in diesem Bereich doch erstaunliches getan: Nach einer schier „epidemischen" Veröffentlichung zahlloser OpenSource-Präsentationsansätze auf SVG-basis [HSL00, Fisc02, Herm02, Sue02], der OpenSource-orientierten SlideML-Initiative [Fisc02] und ebenso der eher auf Ästhetik bedachten PowerPoint-Konkurrenz *Apple Keynote* (mit ebenfalls XML-basierten Präsentationsformat APXL)[3] macht insbesondere die Öffnung Microsofts im Hinblick auf SVG deutlich, dass das ehemals unantastbare „PowerPoint-Prinzip" erheblich ins Wanken geraten ist – und lässt für die sicherlich spannende Zukunft noch einiges erhoffen.

Die vorliegende Diplomarbeit konnte –hoffentlich– ein kleinen, erhellenden Einblick in diese zweifellos aufregende Entwicklung geben und mit dem *XML » SVG Presenter* überdies einen bescheidenen Beitrag in Form eines als Konzeptstudie realisierten „Alternativ-Vorschlags" zu dieser Diskussion liefern. Wenn sich auch, realistisch betrachtet, das bisherige Präsentationsparadigma, zumindest in nächster Zeit wohl kaum „dramatisch" in diese Richtung verändern wird, so gilt es dennoch, die zukünftige Entwicklung dieses nun wieder erheblich dynamischeren Bereiches der Präsentationsformate und –Ansätze doch aufmerksam zu beobachten. Ich für meinen Teil bin jedenfalls auf die gerade in der nächsten Zeit zu erwartenden Veränderungen und Initiativen auf diesem Gebiet äußerst gespannt und freue mich bereits darauf, eventuell auch an den zukünftigen, aufregenden Entwicklungen teilhaben zu können.

[1] s. 2.3.1-2
[2] s. 2.3.3 (m.E. 2.3.4)
[3] vgl. [Ogbu03]

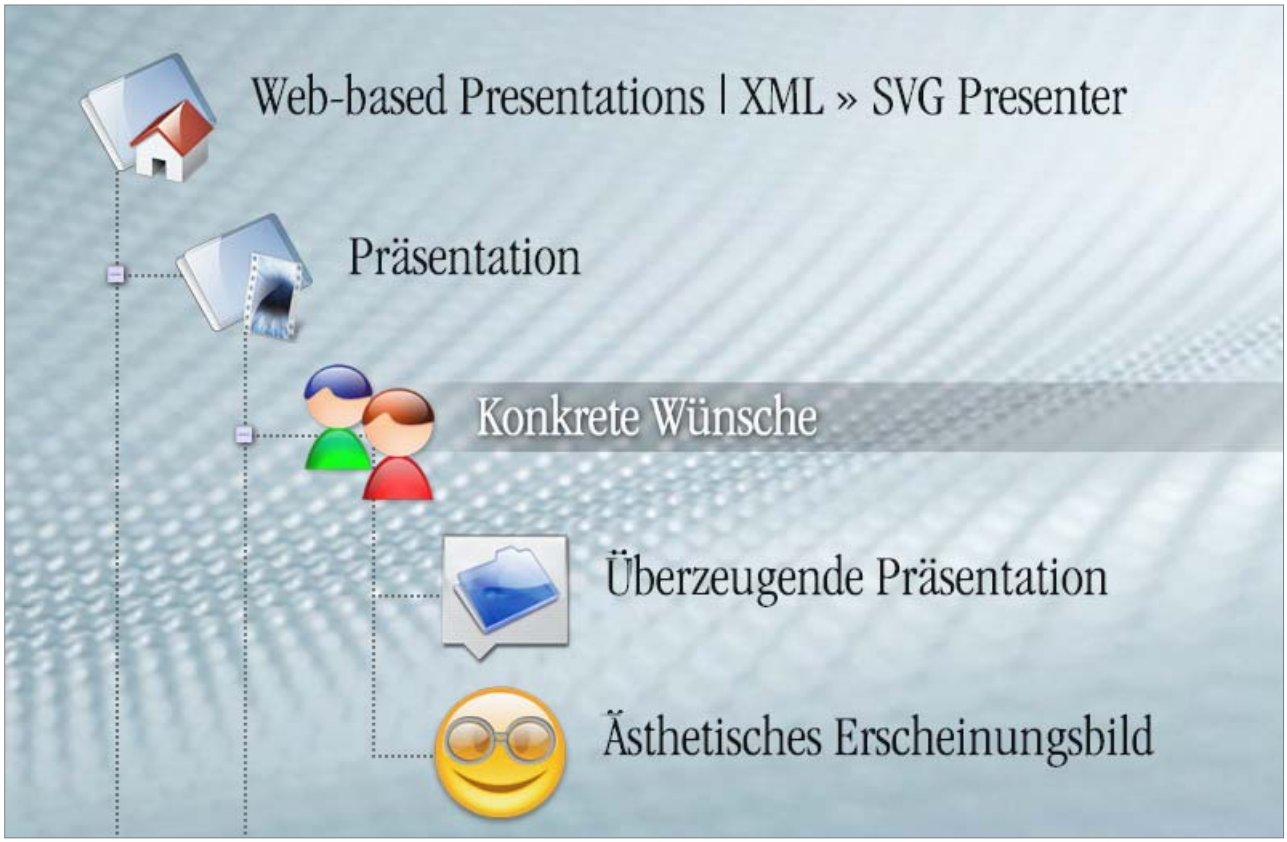
Anhang

Anhang A: Beispiel-Präsentationsdaten

Gemäß der in [6.2] erläuterten, eigenen XML-DTD:

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE PData SYSTEM "http://www.fh-furtwangen.de/~voswinck/pdata.dtd">
  <presentation id="pdata"
     xmlns:pd="http://www.fh-furtwangen.de/~voswinck"
     xmlns:svg="http://www.w3.org/2000/svg"
     xmlns:xlink="http://www.w3.org/1999/xlink">
  <head id="head">
      <title>Web-based Presentations | XML &#xbb; SVG Presenter</title>
      <author>Till Voswinckel</author>
      <desc>Bitte auf das Icon klicken, um anzufangen!</desc>
      <setting>Fachhochschule Furtwangen</setting>
      <date>10.1.2003</date>
    </head>
    <body id="body">
      <item>Web-based Presentations | XML &#xbb; SVG Presenter
        <item>Präsentation
          <item>Der Anlass
            <item>Das Bedürftnis
              <item>Diplomarbeit: Inhalte dokumentieren...</item>
              <item>... und überzeugend präsentieren!
                 <image xlink:href="images/powerpnt.png" />
              </item>
            </item>
            <item>Der Wunsch
              <item>Überzeugende Präsentation</item>
              <item>Ästhetisches Erscheinungsbild</item>
              <item>PowerPoint: Stichwort-Zwang
                  <text>Steve Jurvetson: &quot;I've seen people who get in a
                        mental rut and are unable to write anything other than
                        bullet-point lists&quot; (USA Today-Artikel)</text>
              </item>
              ...
            </item> <!-- Weitere Präsentationsdaten (hier weggelassen) -->
          </item>
        </item>
      </item>
    </body>
  </presentation>
```

Anhang B: Beispiel-Screenshots




Dave Winer


Der PowerPoint-Ansatz
Die "Tatwaffen"
"Böse" Zauberer
Die Konsequenz
Friends don't let Friends use PowerPoint! (Thomas Stewart, Fortune Magazine)

Bibliographie

Anmerkung: Alle angegebenen Internet-URLs wurden vor Drucklegung am 27. Feburar 2003 erneut auf ihre „Online-Funktionstüchtigkeit“ überprüft – ein separater Datumsvermerk entfällt.

Literaturhinweise und Quellenverzeichnis

[Abbe00] Heidi Abbey: “Beyond the Bells and Whistles: A Practical Look at Macromedia’s Flash and Its Usability for Libraries.” *ITIG TechCorner*, (Information Technology Interest Group, Univ. and Research Libraries) Chicago, Juli/August 2000

[Adam02] Alexander Adam: *SVG – Scalable Vector Graphics*. Franzis’ Verlag, München 2002

[Airb96] Max Airborne: “Adobe moves to standardize the Web with Java.” *JavaWorld*, Ausg. 6/96. Vancouver, 15. Mai 1996

[Alla02] Jeremy Allaire: “Macromedia Flash MX - A Next-Generation Rich Client.” Whitepaper, Macromedia Inc. San Francisco, März 2002. URL: http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf

[Ande00] David Anderson (Hrsg.) “Evangelize with Usability.” *User Interface Design*, Glasgow 2000 URL: http://www.uidesign.net/2000/papers/evangelize.html

[Andr93] Marc Andreessen: *NCSA Mosaic: Technical Summary*. National Center for Supercomputing Applications der Universität Illinois, Urbana-Champaign 1993

[Appn02] Timothy Appnel: “Comparing SWF (Flash) and SVG is missing the point.” *MPlode / TIMA thinking out loud*. Jersey City NJ, 11. Juni 2002. URL: http://www.mplode.com/tima/archives/000017.html

[Arah99] Tom Arah: “SVG: The Future Web Format.” *Designer-Info.com*. Edinburgh, August 1999 URL: http://www.designer-info.com/Writing/svg_format.htm

[Arah01] Tom Arah: “Xara X: Recommended.” *Designer-Info.com*. Edinburgh, Januar 2001 URL: http://www.designer-info.com/Writing/xara_x.htm

[Arci02] Fabio Arciniegas: “A Realist’s SMIL Manifesto.” *XML.com*, O’Reilly & Associates, Sebastopol CA, 29. Mai 2002. URL: www.xml.com/pub/a/2002/05/29/smil.html

[Arm01] Eric Armstrong: *Working with XML* (Version 1.1) Sun Microsystems, Palo Alto 2001 URL: http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial

[Arno92] Lutz Arnold: *Moderne Bildkommunikation*. Hüthig Verlag, Heidelberg 1992

[Arpa96] Scott Arpajian: “The Future of HTML,” in: Scott Arpajian und Robert Mullen: *How to Use HTML 3.2*. Ziff-Davis Press, New York 1996

[Arty02] Jacek Artymiak: “SWF Is Not Flash (and Other Vectored Thoughts)” *O’Reilly Network*, Sepastopol CA, 21. Juni 2002. URL: http://www.oreillynet.com/pub/a/javascript/2002/05/24/swf_not_flash.html

[AuBa02] Meike Aulbach und Alexandra Balschun: „Produktion multimedialer Inhalte mit Macromedia Director.“ Ausarbeitung i.R.d. Proseminars „Multimedia und Internet“, Universität Hamburg (Arbeitsbereich für Angewandte und Sozialorientierte Informatik) Hamburg, Wintersemester 2001/02

[Bach02] Stefan Bachert: „Mit Lecturnity auf dem Weg zum digitalen Hörsaal.“ *AboutIT*, Ausg. 42, p.4. Ehningen, 30. November 2002

[Badr01] Greg Badros, Jojada Tirtowidjojo, Kim Marriott, Bernd Meyer, Will Portnoy und Alan Borning: “A Constraint Extension to Scalable Vector Graphics.” *Proceedings of the 10th international conference on World Wide Web* (Hong Kong) pp.489-498. ACM Press, New York 2001

[Baja02] Geetesh Bajaj: “PowerPoint and Director.” *Indezine*, Hyderabad (Indien) 10. Juli 2002 URL: http://www.indezine.com/products/powerpoint/ppdirector.html

[Balo00] Peter Balogh: “Sympathy for the Plug-In.” *A List Apart*, New York 2000 URL: http://www.alistapart.com/stories/sympathy

[BaLy96] Michael Barnsley und Lyman Hurd: *Bildkompression mit Fraktalen*. Vieweg Verlag, Braumschweig / Wiesbaden 1996

[BaSl87] Michael Barnsley und Alan Sloan: “Chaotic Compression.” *Computer Graphics World*, Ausg.11/87, pp.107f. Nashua NH, November 1987

[Baum98] Michael Baumgardt: *Web Design Kreativ*. Springer Verlag, Berlin 1998

[Bayn98] Michael Bayne: “What’s our Vector Victor?” *Blue Sky*, The Mozilla Organization (o.O) 8. Juni 1998 URL: http://www.mozilla.org/blue-sky/extension/199806/vector-graphics.html

[Beck02] David Becker: “Flash: More than just eye candy.” *News.com* (c|net Networks) San Francisco, 3. März 2002

URL: http://news.com.com/2100-1040-849981.html

[BEng00] Simon Wistow BEng III: *Deconstructing Flash*. Projektdokumentation, Imperial College (Department of Computing) London 2000

[Bens98] Les Benson: "Corel Xara." *Big Blue & Cousins Newsletter*, 15. Jahrg. Ausg. 7. Victoria BC, September 1998

[BCM01] Tim Bienz, Richard Cohn und James Meehan: *Portable Document Format Reference Manual (Third Edition) Version 1.4*. Addison-Wesley, Boston 2001

[Behm02] Henning Behme: „Schön viel Grafik." *iX*, Ausg. 12/02, pp.52-59. Verlag Heinz Heise, Hannover 2002

[Behz02] Sassan Behzadi: "Using Macromedia Flash MX for Business Presentations." Macromedia Flash MX Application Development Center, San Francisco 2002
URL: http://www.macromedia.com/desdev/mx/flash/articles/flash_presentations.html

[Bell00] Cathleen Belleville: "PowerPoint – A Historical Review." A Bit Better Corporation, Los Altos, CA 2000
URL: http://www.bitbetter.com/downloads/belleville_ppthistory.ppt

[Bels97] David Belson: "Who's Using Java? – Survey Results." *Web Developer*, Darien CT 1997
URL: http://www.webdeveloper.com/java/java_who_is_using_it.html

[Berg01] Jarle Dahl Bergersen: "OpenLGX – The SWF revolution is here?" *FlashMagazine*, Oslo, Juni 2001
URL: http://www.flashmagazine.com/html/441.htm

[Bern99] Tim Berners-Lee (mit Mark Fischetti): *Weaving the Web*. Harper Collins Publishers, San Francisco 1999

[Bern01] Tim Berners-Lee, James Hendler und Ora Lassila: "The Semantic Web." *Scientific American*, 5. Jahrg. Ausg. 284, pp.34-43. New York, Mai 2001

[Bers97] Jesse Berst: "Java Authoring Tools Could Kill HTML." *AnchorDesk*, Ziff-Davis Publishing, New York, 2. Mai 1997. URL: http://www.zdnet.com/anchordesk/story/story_882.html

[Bert96] Manfred Bertuch: „Bits im Bilde." *iX*, Ausg. 9/96, pp.116ff. Verlag Heinz Heise, Hannover 1996

[Bert01] Manfred Bertuch: „Bilder schrumpfen." *iX*, Ausg. 8/01, pp.108ff. Verlag Heinz Heise, Hannover 2001

[BGTM02] Kristian Besley, Hoss Gifford, Todd Marks und Brian Monnone: *Macromedia Flash MX Video*. Friends of Ed / Wrox Press, Chicago 2002

[Birc99] Nick Birch (Hrsg.) *Digital Terrestrial Television MHEG-5 Specification*. ONDigital PLC, London 1999

[BJS94] Mike Bearrte, Sara Jones und John Sapsford-Francis: "Towards usability guidelines for multimedia systems." *Proceedings of the 2nd ACM international conference on Multimedia* (San Francisco) pp.105-110. ACM Press, New York 1994

[Blat97] David Blatner: "Dropping In on Drop Shadows." *MacWorld*, 14. Jahrg. Ausg. 1, pp.178f. San Francisco, Januar 1997

[Blei96] Andreas Bleicher: *Medientechnologien im Internet*. Diplomarbeit, Fachhochschule Furtwangen, Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, WS 1995/1996

[Blum98] Astrid Blumstengel: *Entwicklung hypermedialer Lernsysteme*. Dissertation, Wissensch. Verlag Berlin 1998

[BoEi02] Susanne Boll und Horst Eidenberger: „Medienmix: SMIL 2.0: Markup-Sprache für Multimedia-Präsentationen." *iX*, Ausg. 11/02, pp.113ff. Verlag Heinz Heise, Hannover 2002

[Boei03] Niels Boeing: „Das Web der Maschinen." *taz*, Ausg. 6964, p.14. Berlin, 9. Januar 2003

[Bole98] Dietrich Boles: *Multimedia-Systeme*. Begleitbuch und Skript zur gleichnamigen Vorlesung. Universität Oldenburg, Fachbereich Informatik (Abteilung Informationssysteme) Oldenburg 1998

[Boli00] Martin Boliek (Hrsg.) Charilaos Christopoulos und Eric Majani: *JPEG2000 Image Coding System (Part 1: Still Pictures)*. ISO/IEC 15444-1:2000 Final Committee Draft Version 1.0. Genf, März 2000

[BoVi02] Erich Bonnert und Karsten Violka: „Microsoft muss Java integrieren." *Heise Newsticker*, Verlag Heiz Heise, Hannover, 24.Dezember 2002. URL: http://www.heise.de/newsticker/data/kav-24.12.02-000

[Bord97] Monica Bordegoni, GiorgioFaconti, Steven Feiner, Mark Maybury, Thomas Rist, Salvatore Ruggieri, Panos Trahanias und Mary Wilson: "Standard Reference Model for Intelligent Multimedia Presentation Systems." *Computer Standards and Interfaces*, 18. Jahrg. Ausg. 6/7, pp.477-496. Elsevier Publishing, New York, Dezember 1997

[Born90] Günter Born: *Referenzhandbuch Dateiformate*. Addison-Wesley, München 1990

[Bosa97] Jon Bosak: "XML, Java, and the future of the Web." *World Wide Web Journal*, 2. Jahrg. Ausg. 4, pp.219-227. O'Reilly & Associates, Sebastopol CA 1997

[Bosa98] Jon Bosak: "Media-Independent Publishing: Four Myths about XML." *IEEE Computer*, 31. Jahrg. Ausg. 10, pp.120-122. Washington DC, Oktober 1998

[Bout96] Thomas Boutell (Hrsg.) *PNG (Portable Network Graphics) Specification*. W3C Recommendation, 1. Oktober 1996. URL: http://www.w3.org/Graphics/PNG

[Bozw00] David Bozward: "Technologies for the Third Generation (White Paper)" Imperial Communication Consultants, Birmingham 2000. URL: http://www.imcoco.com/umtspaper.htm

[BrDw94] Lorinda Brader und Carol Dwyer: *ToolBook*. Educational Technology Publications, Englewood Cliffs1994

[Broc95] Kraig Brockschmidt: *Inside OLE*. Microsoft Press, Redmond WA 1995

[Brod96] Neville Brody: "Preface", in: Willem Velthoven und Jorinde Seijdel (Hrsg.) *Multimedia Graphics*. Verlag Herrmann Schmidt, Mainz 1996

[Brow96] Mark Brown: *Special Edition Using Netscape 3*. QUE Publishing, Indianapolis 1996

[Brow99] Mark Brown: "Multimedia Applets", in: Molly Holzschlag (Hrsg.) *Special Edition Using HTML 4*. Macmillan Publishers, London 1999

[Brow02] Daniel Brown: "Understanding PowerPoint (Special Deliverable Nr. 5)." *Boxes and Arrows* Journal, Bethesda MD, 29. September 2002.
URL: http://boxesandarrows.com/archives/understanding_powerpoint_special_deliverable_5.php

[Brun01] Ronald Brunswig: "Intelligent Web Graphics." Paper i.R.d. *Spring 2001 Software Engineering Symposium*, DePaul University (School of Computer Science, Inf. Systems, and Telecommunications) Chicago 2001

[Bryn99] Jens Brynildsen (Hrsg.) "The SWISH-Maker." *FlashMagazine*, Oslo 1999
URL: http://www.flashmagazine.com/html/402.htm

[Bugg03] Keith Bugg: "SVG and Smart Maps." *Dr. Dobb's Journal*, 26. Jahrg. Ausg. 3 (Nr. 346) pp.38-41. San Mateo CA, März 2003

[Bult98] Dick Bulterman, Lynda Hardman, Jack Jansen, Sjoerd Mullender und Lloyd Rutledge: "GRiNS: a graphical interface for creating and playing SMIL documents." *Computer Networks and ISDN Systems*, 30. Jahrg. Ausg 1-7, pp.519-529. Elsevier Science Publishers, Amsterdam 1998

[Bult01] Dick Bulterman: "SMIL Perspective", in: Savitha Srinivasan und Dulce Ponceleon (Hrsg.) "Is Streaming Media becoming Mainstream?", pp.181f. *Proceedings of the 9th ACM international conference on Multimedia* (Ottawa) pp.181-186. ACM Press, New York 2001

[Burk96] Jonathan Burke: "Stretching Director." *Red Herring*, 4. Jahrg. Ausg. 35. San Francisco, September 1996

[Bush45] Vannevar Bush: "As we may think." *Atlantic Monthly*, Vol. 176, Ausg. 1, pp.101-108. Boston, Juli 1945

[Cagl02] Kurt Cagle: *SVG Programming: The Graphical Web*. APress, Berkeley 2002

[Cail97] Robert Cailliau: "Foreword", in: Hakon Wium Lie und Bert Bos: *Cascading Style Sheets: Designing for the Web*. Addison-Wesley, Boston 1997

[Calh96] Kevin Calhoun: "Hypercard 3.0: The Phoenix Rises." *Proceedings WWDC'96*, San Jose 1996

[Capi03] Tolga Capin (Hrsg.) *Mobile SVG Profiles: SVG Tiny and SVG Basic*. W3C Recommendation, 14. Januar 2003. URL: http://www.w3.org/TR/SVGMobile

[CGM92] International Organization for Standardization (o.V) *CGM Standard: Metafile for the storage and transfer of picture description information - Part 1: Functional specification*. ISO/IEC 8632, Genf 1992

[Cham02] Mike Chambers: "Favorite Feature in Macromedia Flash MX." *Logged In*, Macromedia Designer & Development Center, San Francisco 2002.
URL: http://www.macromedia.com/desdev/logged_in/mchambers_mx.html

[ChHu01] Michael Christel und Chang Huang: "SVG for Navigating Digital News Video." *Proceedings of the 9th ACM international conference on Multimedia* (Ottawa) pp.483- 485. ACM Press, New York 2001

[Chro02] Robert Chromow: „GIF-Animationen als ‚Online-Videos'." *Akademie.de*, Berlin 2002
URL: http://www.akademie.de/websiteaufbau/tipps_tricks/gestaltung/weitere_grafik-tools/gif-videos.html

[ChNg02] Benyam Chekol und Diem Nguyen: "Components of DHTML." Ausarbeitung i.R.d. Seminars "e-Commerce Technology", North Carolina State University (e-Commerce Learning Center) Raleigh 2002

[ChYu98] Wo Chang und Jin Yu: "S2M2 - A Java Applet-based SMIL Player." *Proceedings of the International Conference on Multimedia & Telecommunications Management*, pp 353-36. Hong Kong. Dezember, 1998

[Clon00] Curt Cloninger: "Usability Experts are from Mars, Graphic Designers are from Venus." *A List Apart*, New York 2000. URL: http://www.alistapart.com/stories/marsvenus

[CKR97] Dan Connolly, Rohit Khare und Adam Rifkin: "The Evolution of Web Documents." *XML.com*, O'Reilly & Associates, Sebastopol CA, 2. Oktober 1997. URL: http://www.xml.com/pub/a/w3j/s3.connolly.html

[Cove99] Andy Covell: "Introduction to SMIL." *Streaming Media Magazine*, New York 1999
http://www.streamingmedia.com/tutorials/smil.asp

[Croo01] Clayton Crooks: "An SVG Tool Kit for Java: Batik SVG Toolkit." *WebTechniques*, 6. Jahrg Ausg. 4, pp.40f. New York, April 2001

[CSS96] Håkon Wium Lie und Bert Bos: *Cascading Style Sheets, Level 1*. W3C Recommendation, 17. Dezember 1996. URL: http://www.w3.org/TR/REC-CSS1

[CSS98] Bert Bos, Håkon Wium Lie, Chris Lilley und Ian Jacobs (Hrsg.) *Cascading Style Sheets, Level 2 (CSS2) Specification*. W3C Recommendation, 12. Mai 1998. URL: http://www.w3.org/TR/REC-CSS2

[Curt01] Michael Curt: „Im Namen des guten Geschmacks: «Skip Flash».“ *Computerworld* (Schweiz) Special, Ausg. 2/01. Zürich, 2. Februar 2001

[Dabb01] Hussayn Dabbous (Hrsg.) *Saxess Wave 1.0 Manual*. Saxess Software Design, Köln 2001
URL: http://www.saxess.com/wave/swfml-ref

[D'Amo00] Stefan D'Amore: „SVG von Adobe soll der Konkurrenz Druck machen.“ *Macwelt*, Ausg 8/20, pp.100-104. München, August 1998

[Dana97] Laurel Danara: *Riven. The Sequel to Myst*. Sybex-Verlag, Düsseldorf 1997

[Dele97] Carmen Delessio: “Converting Windows Metafiles to Java.” *Dr. Dobb's Journal*, Ausg. 5/97. San Mateo CA, Mai 1997

[Depe02] Jens Depenau: *Die Zukunft von Navigationssystemen im World Wide Web*. Diplomarbeit, Fachhochschule Furtwangen. Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Wintersemester 2001/2002

[DHH02] David Duce, Ivan Herman und Bob Hopgood: “Web 2D Graphics: State-of-the-Art.” *Computer Graphics Forum*, 21. Jahrg. Ausg. 1, pp.43-65. Blackwell Publishing, Oxford 2002

[DMO01] Steve DeRose, Eve Maler und David Orchard: *XML Linking Language (XLink) Version 1.0*. W3C Recommendation, 27. Juni 2001. URL: http://www.w3.org/TR/xlink

[Doer00] Holger Dörnemann: „Softwareentwicklungsprozess und schnelllebiges Internet.“ *Proceedings Net.ObjectDays'2000* (Erste vereinigte GI Fachtagung „Objektorientierte Programmierung für die vernetzte Welt“) Erfurt, 10. Oktober 2000

[DOM02] Ben Chang, Elena Litani, Jeroen van Rotterdam, Johnny Stenback, Andy Heninger, Joe Kesselman und Rezaur Rahman (Hrsg.) *Document Object Model (DOM) Level 3 Abstract Schemas and Load and Save Specification*. W3C Working Draft, 25. Juli 2002. URL: http://www.w3.org/TR/DOM-Level-3-ASLS

[Doug98] Dale Dougherty: “XML'98: The Gathering.” *XML.com*, O'Reilly & Associates, Sebastopol CA, November 1998. URL: http://www.xml.com/lpt/a/98/11/xml98-1.html

[Doug01] Dale Dougherty: “Why Flash Is Significant.” *O'Reilly Network*, Sepastopol CA, 2. Februar 2001
URL: http://www.oreillynet.com/pub/a/network/2001/02/02/epstein.html

[Dowd99] John Dowdell: “What's the difference between Shockwave and Flash?” Macromedia General TechNotes, San Francisco, 2. August 1999
URL: http://www.macromedia.com/support/general/ts/documents/sw_flash_differences.htm

[Dowd00] John Dowdell: “How can I draw the intersection of two shapes?” *TechNote*, Macromedia Flash Support Center, San Francisco, 10. März 2000
URL: http://www.macromedia.com/support/flash/ts/documents/boolean_drawing.htm

[Draw98] Håkan Lothigius: *DrawML Specification*. W3C Note, 3. Dezember 1998.
URL: http://www.w3.org/TR/NOTE-drawml

[Duck01] Dirk Duckwitz: *Plattformübergreifendes Programmieren und Design*. Diplomarbeit, Fachbereich Digitale Medien (Studiengang Medieninformatik) Sommersemester 2001

[Duda99] Peter Dudar: „Deneba Canvas Professional Edition, Version 6.” *Graphic Exchange Magazine*, 4. Jahrg. Ausg. 2, pp.51-54. Toronto, April/Mai 1999

[Dudr98] Andrea Dudrow: “Deneba promises efficiency increase with the announcement of Colada 2.0” *MacWeek Japan*, Ausg. 1/98. ZDNet Press, Tokio, 13. Januar 1998

[Duff03] Stefan Duffner: *Transformation statischer HTML-Seiten in XML*. Diplomarbeit, Fachhochschule Furtwangen, Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Wintersemester 2002/03

[DuHo00] David Duce und Bob Hopgood: *Web Schematics*. Rutherford Appleton Laboratory (Council for the Central Laboratory of the Research Councils) Advanced Interactive Systems Group, Oxford 2000

[Dumb01] Edmund Dumbill: “Picture Perfect.” *XML.com*, O'Reilly & Associates, Sebastopol CA, 12. September 2001
URL: http://www.xml.com/pub/a/2001/09/12/svg.html

[Dyso96] Peter Dyson (Hrsg.) “Adobe tips Internet strategy.” *Seybold Report on Desktop Publishing*, 10. Jahrg. Ausg. 9. Seybold Publications, Media PA, Mai 1996

[ECMA97] Andrew Clinick, Clayton Lewis und Jan van den Beld (Hrsg.) *ECMAScript Language Specification*. European Computer Manufacturers Association (ECMA-262) Genf 1997

[Edri00] Don Edrington: “WordArt is a very useful tool among Microsoft products.” *North County Times*, Escondido CA, 17. Oktober 2000

[EfSt98] Wolfgang Effelsberg und Ralf Steinmetz: *Video Compression Techiques*. dpunkt Verlag, Heidelberg 1998

[EHHP99] Joachim Euchner, Dirk Höpfner, Olaf Hörnlein, Teodor Pirkmayer: *Integration von MHEG5 in das WWW*. Abschlußbericht, „IMW-Projekt", PRZ (Forschungszentrum für Netzwerktechnologien und Multimedia-Anwendungen) TU Berlin, Januar 1999

[Eise02] J. David Eisenberg: *SVG Essentials*. O'Reilly & Associates, Sebastopol CA 2002

[EnEh00] Tobias Engler und Stephan Ehrmann: „OS mit X-tras." *c't*, Ausg. 2/00, p.19f. Verlag Heinz Heise, Hannover 2000

[Endi01] Jim Endicott: "Don't force PowerPoint to do another program's job." *Presentations*, Ausg. 12/01. VNU Business Media, Minneapolis, Dezember 2001

[Endi02a] Jim Endicott: "It Always Pays to Have a Clean, Professional Package." *Presentations*, Ausg. 6/02. VNU Business Media. Minneapolis, Juni 2002

[Endi02b] Jim Endicott: "True creativity involves more than just pretty slides." *Presentations*, Ausg. 9/02. VNU Business Media, Minneapolis, September 2002

[Engs96] Adam Engst: „Weitere Gedanken über das Web." *TidBITS*, Ausg. 345 (dt. Edition) 16. September 1996

[EnSc03] Tobias Engler und Peter Schmitz: „Selber lesen!" *c't*, Ausg. 1/03, p.47. Heise-Verlag, Hannover 2003

[Essi96] Kristi Essick: "Adobe takes on the Internet." *JavaWorld*, Ausg. 6/96. San Francisco, 7. Mai 1996

[Farm98] James Farmer: "XML & Graphics – Vector Markup: Hyper Graphics Markup Language." *'net prophet*, Florida State University (School of Information Studies) Tallahassee FL, 28. Juli 1998

[FaSu97] Peter Faraday und Alistair Sutcliffe: "Designing Effective Multimedia Presentations." *Proceedings of the SIGCHI conference on Human factors in computing systems* (Atlanta) pp.92-98. ACM Press, New York 1997

[Fibi01] Iris Fibinger: *Scalable Vector Graphics - Untersuchung des XML-Standards für zweidimensionale Vektorgraphiken und Erstellung eines SVG-Tutorials*. Diplomarbeit, Fachhochschule Karlsruhe, Fachbereich Sozialwissenschaften (Studiengang Technische Redaktion) Karlsruhe, März 2001

[Fick99] Markus Fick: „Fraktale Bildkompression auf Basis von Iterativen Funktionssystemen und adaptiver HV-Partitionierung." Universität Siegen, 1999. URL: http://www.stud.uni-siegen.de/markus.fick/zfc/zfc.html

[Film99] Paul Filmore: *Master of Sciences Project Catalogue 1999/2000*. University of Plymouth, School of Electronic (Department of Communication and Electrical Engineering) Plymouth, 6. Dezember 1999, p.5

[Fisc95] Detlev Fischer: *A theory of Presentation and its Implications for the Design of Online Technical Documentation*. Dissertation, Universität Coventry 1995

[Fisc02] Roger Fischer (Hrsg.) Robin Berjon, Lon Boonen, Paul Everitt, Roger Fischer, Gregor Rothfuss, Christian Stocker, Eric Vitiello und Michael Wechner: *SlideML – XML for slides*. Bitflux GmbH, Zürich, 6. September 2002. URL: http://bitflux.ch/slideml/slideml.html

[Fish98] Yuval Fisher (Hrsg.) *Fractal Image Compression: Theory and Application*. Springer-Verlag, New York 1998

[Fish02] Wayne Fisher: "The Latest AutoCAD Extensions." *Proceedings of the Autodesk University 10th Annual User Conference*, Las Vegas, 5. Dezember 2002

[Ferr00] Jon Ferraiolo: "Getting Inside SVG: What Web Content Creators and Software Developers Should Know." *Proceedings XML Europe 2000*, Paris, Juni 2000

[Fest02] Paul Festa: "W3C sees graphics on cell phones." *News.com* (c|net Networks) San Francisco, 15. November 2002 (Ebenfalls unter dem Titel "W3C focuses on cell phone graphics" bei Ziff-Davis veröffentlicht) URL: http://news.com.com/2100-1001-966104.html

[FFJ03] Jon Ferraiolo, Jun Fujisawa und Dean Jackson (Hrsg.) *Scalable Vector Graphics (SVG) 1.1 Specification*. W3C Recommendation, 14. Januar 2003. URL: http://www.w3.org/TR/SVG11

[FlRi01] Robert Flisser und Wendy Richardson: *Just the Tips, Man: Microsoft PowerPoint 2000*. Nerdy Books, New York 2001

[Flüg96] Joachim Flügel: „Adobes ‚Bravo' soll Internet-Grafiken revolutionieren." *DTP-News*, Ausg. 6. Alling bei München, Juni 1996

[Fren02] William French: *How Flash Unseated Fireproof Digital*. Midterm Paper, University of California (School of Information Management & Systems) Berkeley 2002

[Frie03] Ian Fried: "Mac users find Keynote out of tune." *ZDNet News*, New York, 23. Januar 2003 URL: http://zdnet.com.com/2100-1104-981729.html

[Froo96] John Evan Frook: "Intranet Watch: Radical Changes at RadMedia." *CommunicationsWeek Interactive*, Manhasset NY, 2. Dezember 1996. URL: http://internetwk.com/cwi/netnews/120296/news1202-3.html

[Fors98] Michael Forster: *Analyse des klassischen GUI; Oberfläche im Internet/Intranet; Trendanalyse für einen Style-Guide*. Diplomarbeit, Fachhochschule Furtwangen, Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Sommersemester 1998

[Foss03] Kurt Foss: "Future PDF presentations will sing and dance with embedded SVG." *Planet PDF*, Melbourne, 31. Januar 2003. URL: http://www.planetpdf.com/mainpage.asp?webpageid=2571

[FRS94] Yuval Fisher, Daniel Rogovin und Tsae-Pyng Shen: "A Comparison of Fractal Methods with DCT and Wavelets," in: Su-Shing Chen (Hrsg.) "Neural and stochastic methods in image and signal processing" *Proceedings of the SPIE*, Vol. 2304, pp.132-134. San Diego 1994

[Fult01] Nancy Fulton: "Presedia Producer – Making PowerPoint-to-Flash translation easy." *e-Learning Magazine*, 1. Jahrg. Ausg. 10. Cleveland OH, 1. Oktober 2001

[Gall02] Peter Galli: "XML to Drive Office Update." *eWeek*, 19. Jahrg. Ausg. 39, pp.1,14. Ziff-Davis Publishing, New York, 30. September 2002

[Gay01] Jonathan Gay: "The Dawn of Web Animation", in: *The History of Flash*. Macromedia Showcase (i.R.d. flashforward'2001-Konferenz) San Francisco 2001
URL: http://www.macromedia.com/macromedia/events/john_gay

[Gibb01a] Mark Gibbs: "Cool graphics in XML." *Network World*, 18. Jahrg. Ausg. 23, p.56. Southborough, 4.6.2001

[Gibb01b] Mark Gibbs: "Ready for animation." *Network World*, 18. Jahrg. Ausg. 25. Southborough, 18.6.2001

[Gibs00] Brad Gibson: "Adobe ships SVG Viewer; designers skeptical." *Mac News Network*, Santa Clara, 6.6.2000
URL: http://www.macnn.com/feature.php?id=33

[Glue96] Erhard Glück: „Java – Eine lebende Programmiersprache." *Cyberzine*, Ausg. 1/96, Wien 1996

[Godi01] Seth Godin: *Really Bad PowerPoint*. Do You Zoom Books, New York 2001

[Gold92] Charles Goldfarb, Steven Newcomb, Eliot Kimber und Peter Newcomb: *Hypermedia/Time-based Structuring Language (HyTime)* ISO/IEC 10744, Genf 1992

[Gold03] Markus Goldstein: *Scalable Streaming of Medical Images using JPEG2000*. Diplomarbeit, Fachhochschule Furtwangen. Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Wintersemester 2002/03

[Good90] Danny Goodman (Hrsg.) *The Complete HyperCard 2.0 Handbook*. Bantam Books, New York 1990

[GoRi99] Michael Gould und Antonio Ribalaygua: "Implementation of Vector Markup Language for Cartographic Data Applications." *XML Europe'99 Conference Proceedings*, 24. Jahrg. Ausg. 9, p.48.Granada 1999

[Grae96] Gerald Graef: "Graphics Formats for Linux." *Linux Journal*, Ausg. 23es, pp.36-42. Seattle, März 1996

[Graf02] Klaus-Dieter Graf (Hrsg.) Margarita Esponda und Marco Rademacher: „Die Document-Object-Models (DOM)," aus: Unterl. Lehrveranstaltung „Partizipation im Internet", FU Berlin, Institut für Informatik (Fachbereich Mathematik und Informatik) AG Informatik in Bildung und Gesellschaft. Berlin 2002

[Gräf98] Hans Dieter Gräfen: „Anwendung von SGML und CGM am Beispiel Documentum." *Info Forum'98* (Tanner Documents) Lindau, 6. März 1998

[Grip03] Theodor Grip: *Konzeption und beispielhafte Realisierung von Skins für eine E-Democracy-Plattform*. Diplomarbeit, Fachhochschule Furtwangen. Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Wintersemester 2002/03

[GRM97] Michel Goossens, Sebastian Rahtz und Frank Mittelbach: *The LaTeX Graphics Companion*. Addison-Wesley, Boston 1997

[GrRa94] Kaj Grønbæk und Randall Trigg (Hrsg.) "Special Issue on Hypermedia." *Communications of the ACM*, 2. Jahrg. Ausg. 37. New York, Februar 1994.

[Grue01] Gabriele Gründer (Hrsg.) „Ausgezeichnete Artworks." *pocketPAGE*, Ausg. 1/01, pp.8f. MACup Verlag, Hamburg 2001

[Guhl01] Miguel Guhlin: "Streaming Video: Beyond Our Grasp?" *MindWrite*: Published Writing on Educational Technology. San Antonio TX, November 2001
URL: http://www.edsupport.cc/mguhlin/portfolio/writings/2001/streamingvideo.htm

[Hard99] Lynda Hardman, Jacco van Ossenbruggen, Sjoerd Mullender, Lloyd Rutledge und Dick Bulterman: "Do You Have the Time? Composition and Linking in Time-based Hypermedia." *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia* (Darmstadt) ACM Press, New York 1999

[Hard00] Vincent Hardy: "SVG and the Java2D API: Writing a custom Graphics2D implementation to generate SVG images." Sun Microsystems (Dept. Computer Systems) Palo Alto CA, August 2000
URL: http://wwws.sun.com/software/xml/developers/svg/java2d-api

[HaSc94] Frank Halasz und Mayer Schwartz: "The Dexter Hypertext Reference Model." *Communications of the ACM*, 37. Jahrg. Ausg.2, pp.30-39. New York, Februar 1994.

[HBR94] Lynda Hardman, Dick Bulterman und Guido van Rossum:. "The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model." *Communications of the ACM*, 37. Jahrg. Ausg. 2. New York, Februar 1994.

[HeGe94] Lofton Henderson und John Gebhardt, "CGM: SGML for Graphics? – A Structured Vendor Neutral Interchange Format for Graphics." *Gilbane Report,* 2. Jahrg. Ausg. 5. Cambridge MA 1994.

[Hent98] Johannes Hentrich: *Hypermedia Design*. Haessler Verlag, Schömberg 1998

[Herl99] Jan Herlitz: "DrawML and SVG." *XML Europe'99 Conference Proceedings*, 24. Jahrg. Ausg. 9. Granada 1999

[Hess97] Deborah Hess: "New Java Tools Cozy Up to Servers." *Byte* Magazine, 22. Jahrg. Ausg. 7, New York 1997

[Heid97] Jim Heid: "SuperCard 3.0: Multimedia Old-Timer Gets a Sporty Web Plug-In." *MacWorld*, 14. Jahrg. Ausg. 5, p.58. San Francisco, Mai 1997

[Heis98] Christian und Ansgar Heise (Hrsg.) „Multimedia im Web: SMIL, HGML." *iX*, Ausg. 8/98, p.22. Verlag Heinz Heise, Hannover 1998

[HGML98] Mike Evans, Steven Furnell, Andy Phippen, Paul Reynolds, Neil Lilly und John Hammac: *Hyper Graphics Markup Language (HGML)*. W3C Note, 19. Juni 1998
http://www.w3.org/TR/NOTE-HGML

[HoBo01] Jens Hoppe und Ligia Bolz: *Macromedia Director 8 Shockwave Studio*. Ausarbeitung i.R.d. Vorlesung „Mediensoftware II", FHTW Berlin, Studiengang Internationale Medieninformatik. Berlin 2001

[HoDu00] Scott Howlett und Jeff Dunmall: "VML Provides XML-based Graphics for the Web." *Microsoft Internet Developer Magazine*, Ausg. 1/00. Redmond WA, Januar 2000

[Hols99] Michael Holst: „Interaktives WWW mit ToolBook und Director." *Kompass*, Ausg. 73. Universität Köln, 18. Februar 1997

[Holz01] Molly Holzschlag: "Scalable Vector Graphics. (Integrated Design)" *WebTechniques*, 6.Jahrg. Ausg. 4, pp.30-34. New York, April 2001

[HORB99] Lynda Hardman, Jacco van Ossenbruggen, Lloyd Rutledge und Dick Bulterman: "Hypermedia: The Link with Time." *ACM Computing Surveys*, 31. Jahrg. Ausg. 4es. New York, Dezember 1999

[HTML97] Dave Raggett, Arnaud Le Hors und Ian Jacobs (Hrsg.) *HTML 4.0 Specification*. W3C Recommendation, 18. Dezember 1997
http://www.w3.org/TR/PR-html40-971107

[Hues97] Ralf Hüskes: „Mein Web gehört mir." *iX*, Ausg. 8/97, pp.140ff. Heise-Verlag Hannover, August 1997

[Hunt02] John Hunt: "From Java to SVG." *Application Development Advisor*, 6. Jahrg. Ausg. 2, pp.52-57. Farnham UK, März 2002

[IfDi02] Sebastian Iffländer und Stefan Dilger: *Virtual-Reality-gestützte Prozessvisualisierung auf Basis einer 3D-Game-Engine*. Diplomarbeit, Fachhochschule Furtwangen, Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Sommersemester 2002

[Jack03a] Dean Jackson (Hrsg.) "Scalable Vector Graphics Roadmap." World Wide Web Consortium, Canberra, 22. Januar 2003. URL: http://www.w3.org/Graphics/SVG/Roadmap.html

[Jaco02a] Anne Jacoby: „Präsentieren wie ein Profi." *Frankfurter Allgemeine* Hochschulanzeiger. Ausg. 62, pp.40-41. Frankfurt am Main, Oktober 2002

[Jaco02b] Anne Jacoby: „Die Multimedialen Showmaster." *Frankfurter Allgemeine* Hochschulanzeiger. Ausg. 62, pp.42-43. Frankfurt am Main, Oktober 2002

[Jaqu89] Arnaud Jaquin: *A fractal theory of iterated Markov operators with applications to digital image coding*. Dissertation (PhD Thesis) Georgia Institute of Technology, Atlanta 1989

[Jasn99] Uwe Jasnoch: *WWW - Weiterführende Konzepte*. Vorlesungsunterlagen „Internet - Technologie und Anwendungen", Fraunhofer IGD / Fachhochschule Darmstadt 1999

[Kais00] Jean Kaiser: "Good Flash, Bad Flash – Indulgent Design." *About.com*, New York, 19. September 2000

[Kali00] Sari Kalin: "Tim Berners-Lee – Inventing the Enterprise." *CIO Magazine*, New York, 15. Dezember 1999/1. Jan. 2000

[Kara02] Konstantinos Karagiannis: "Microsoft Produces – Free." *PC Magazine*, Ziff-Davis Publishing, New York, 26. Februar 2002. URL: http://www.pcmag.com/article2/0,4149,8011,00.asp

[Karv00] Kristiina Karvonen: "The Beauty of Simplicity." *Proceedings of the SIGCHI Conference on Universal Usability* (Arlington, VA) pp.85-90. ACM Press, New York 2000

[Kass95] Andreas Kassler: *FraComp: Fraktale Bildkompression unter Windows*. Diplomarbeit, Universität Augsburg (Institut für Mathematik) Augsburg 1995

[Katz00] Jon Katz: "In Defense of Flash," in: Cliff Wood (Hrsg.) "Vector Graphics On The Web?" *SlashDot*, 9. August 2000 (*mittlerw. offline*)

[Kaub00] Torsten Kaubisch: *Überblick über Sun's Java Media Framework 2.0*. Seminararbeit, Universität Mannheim (Lehrstuhl für Wirtschaftsinformatik) Mannheim 2000

[Kauf96] Jon Kaufthal: "ABC QuickSilver: Bring Your Graphics to Life." *PC Magazine InternetUser*, Ziff-Davis Publishing, New York, 31. Juli 1996. URL: http://kaufthal.com/portfolio/quicksil.htm

[Kay00] Michael Kay: *XSLT Programmer's Reference*. Wrox Press, Chicago 2000

[Keit00] Wolfgang von Keitz: *Das SMIL-Textbuch*. Hochschule für Bibliotheks- und Informationswesen, Stuttgart

2000. URL: http://v.hbi-stuttgart.de/~keitz/skripte/SMILStart.htm

[Kenn00] Tim Kennedy: "Repent from Flash Sins." *Streaming Media World*, 10. Mai 2000 URL: http://smw.internet.com/symm/voices/flashsins

[Khar98] Rohit Khare: "VML, PGML, and XML Marketing." *FoRK Archive*, University of California (Information and Computer Science) Irvine, 13. Juni 1998

[Kirb94] Tom Kirby: "117 Ideas for Better Business Presentations." *The Executive Speaker Company Clearinghouse & Digest*, Dayton OH 1994

[Kirs98] Christian Kirsch: „Echse in Freiheit." *iX*, Ausg. 5/98, p.48. Verlag Heinz Heise, Hannover, Mai 1998

[Kobl98] Ron Kobler: "Image Viewer Plug-Ins." *Smart Computing*, 6. Jahrg. Ausg. 4. Lincoln NE, April 1998

[Kres95] Johen Kreß: *Digitale Bewegtbild-Kompressionsstandards, Anwendungen und Zukunftsperspektiven*. Diplomarbeit, Fachhochschule Furtwangen. Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Wintersemester 1994/1995

[KrMa00] Jürgen Krüger und Christian Märtin: „Dehnbare Formate." *iX*, Ausg. 11/00, pp.148ff. Verlag Heinz Heise, Hannover 2000

[KrTe02] Thomas Kröckertskothen und Astrid Tessmer: „PDF – Portable Document Format." *RRZN-BI (Benutzerinformations-Journal des Regionalen Rechenzentrums Niedersachsen)* Ausg. 355. Uni Hannover, Januar 2002

[KSH01] Ivan Kopilovic, Dietmar Saupe und Raouf Hamzaoui: „Progressive Fractal Image Coding." *Proceedings of IEEE International Conference in Image Processing (ICIP'01)*, 3. Jahrg. Thessaloniki, Oktober 2001

[Kurz97] Michael Kurzidim: „Java einfacher." *c't*, Ausg. 11/97, p.366. Heiz Heise Verlag, Hannover 1997

[Kunz00] Ralf Kunze: „2D Vektorgrafik: Macromedia Flash / SVG." Vortrag i.R.d. Multimedia-Seminars, Universität Osnabrück (Fachbereich Mathematik/Informatik) Osnabrück, 1. Dezember 2000

[LaCa03] Alexander Lamb und Nicolas Cassoni: « La génération de documents PDF depuis un serveur applicatif.» *Flash Informatique*, 11. Jahrg. Ausg. 1, pp.1f. École Polytechnique Fédérale de Lausanne, 21. Januar 2003

[Laga98] Klaus Lagally: "Readability considerations", in: *7-bit Meta-Transliterations for 8-bit Romanizations*. Universität Stuttgart (Fakultät für Informatik) Stuttgart, 21. Mai 1998

[Lang99] Markus Lang, Wilhelm Berghorn, Tobias Boskamp, Holger Bettag und Heinz-Otto Peitgen: "Feasibility of wavelet image compression for digital mammograms." *Proceedings ECR'99*, Wien, 9. März 1999

[VaNi01] Steven Vaughan-Nichols: "Will Vector Graphics Finally Make It on the Web?" *IEEE Computer*, 24. Jahrg. Ausg. 12, pp.22-24. Washington DC, Dezember 2001

[VRKH02] Petri Vuorimaa, Teemu Ropponen, Niklas von Knorring und Mikko Honkala: "A Java based XML browser for consumer devices. " *Proceedings of the 17th ACM symposium on applied computing* (Madrid) pp.1094-1099. ACM Press, New York 2002

[Lero02] Philippe Leroy: "MVQ streaming may blow away MPEG-4." *ZDNet News France*, Levallois-Perret, 20. Februar 2002. URL: http://zdnet.com.com/2100-1104-840979.html

[Lu97] Ning Lu: *Fractal Imaging*. Academic Press, Oxford 1997

[LZ77] Abraham Lempel und Jacob Ziv: "A universal algorithm for sequential data compression." *IEEE Transactions on Information Theory*, 23. Jahrg. Ausg. 3, pp.337-343. New York, Mai 1977

[LiJa03] Chris Lilley und Dean Jackson (Hrsg.) "Scalable Vector Graphics (SVG) Overview." World Wide Web Consortium (Graphics Activity) Canberra / Sophia Antipolis, Januar 2003 URL: http://www.w3.org/Graphics/SVG/Overview.htm8

[Lill96] Chris Lilley (Hrsg.) "W3C Scalable Graphics Requirements." World Wide Web Consortium (Graphics Activity) Sophia Antipolis 1996. URL: http://www.w3.org/Graphics/ScalableReq

[Lill98] Chris Lilley (Hrsg.) "Comment on Web Schematics Submission." World Wide Web Consortium (Graphics Activity) Sophia Antipolis, 31. März 1998 URL: http://www.w3.org/Submission/1998/05/Comment.html

[Lill98b] Chris Lilley (Hrsg.) "Comment on the 'DrawML' submission." World Wide Web Consortium (Graphics Activity) Sophia Antipolis, 3. Dezember 1998. URL: http://www.w3.org/Submission/1998/20/Comment

[Lill99] Chris Lilley (Hrsg.) "SVG Implementations." World Wide Web Consortium, Sophia Antipolis 1999 http://www.w3.org/Graphics/SVG/SVG-Implementations

[Lill02] Chris Lilley: "Foreword", in: Andrew Watt et al [s.Watt02]: *SVG Unleashed*, pp.XXIVff. Sams Publishing, Indianapolis 2002

[Lind02] Kevin Lindsey: "Object Oriented JavaScript: ECMAScript Object Inheritance." *KevLinDev*, Carrollton TX, 11. November 2002. URL: http://www.kevlindev.com/tutorials/javascript/inheritance

[LiPl97] Chris Lilley (Hrsg.) und Roy Platon: "Use of CGM as a Scalable Graphics Format." W3C Note, 18. Juni 1997. URL: http://www.w3.org/TR/NOTE-cgm

[Lipm00] Julia Lipman: "So far, we're not impressed." *Digital Mass*, Boston, September 2000 (*offline*)

[LiWe01] Chris Lilley und Dieter Weidenbrück: "Web CGM and SVG: A Comparison." *XML Europe 2001 Conference Proceedings*, Berlin 2001

[Mach97] Ingo Macherius: „Revolution der Experten." *iX*, Ausg. 6/97, pp.106ff. Heise-Verlag Hannover, Juni 1997

[Maci00] Ross Macintosh: "Simply Amazing." *DesignStop.com*, Alcove Consulting, Dartmouth 2000
URL: http://www.designstop.com/free_stuff/products/corelxara/vectorgraphics.htm

[Mack86] Jock Mackinlay: "Automating the Design of Graphical Presentations of Relational Information." *ACM Transactions on Graphics*, 4. Jahrg. Ausg. P, pp.110-141. Pittsburgh, April 1986

[Mack88] Jock Mackinlay: "Applying a Theory of Graphical Presentation to the Graphic Design of User Interfaces." *Proceedings of the 1st annual ACM SIGGRAPH symposium on User Interface Software* (Alberta, KAN) pp.179-189. ACM Press, New York 1988

[Macr98] Macromedia Incorporated (o.V.) *Macromedia Flash File Format Specification*. Addison-Wesley, San Francisco 1998

[Macr02a] Macromedia Incorporated (o.V.) "Macromedia Flash Player Penetration." White Paper for Developers and Publishers. San Francisco, September 2002
URL: http://www.macromedia.com/software/flash/survey/whitepaper

[Macr02b] Macromedia Incorporated (o.V.) *Macromedia Flash File Format Specification, Version 6*. San Francisco, 18. Oktober 2002. URL: http://www.macromedia.com/go/flash_specification

[Macr02c] Macromedia Incorporated (o.V.) "Macromedia and Usability Guru Jakob Nielsen work together to improve Web Usability." Pressemitteilung, San Francisco, 3. Juni 2002
URL: http://www.macromedia.com/macromedia/proom/pr/2002/macromedia_nielsen.html

[MaFu01] Philip Mansfield und Darryl Fuller: "Graphical Stylesheets: Using XSLT to Generate SVG." *XML 2001 Conference Proceedings*, Orlando, Fla. 13. Dezember 2001

[Mage00] Johannes Magenheim (Hrsg.) „Der Gegensatz von Layout und Struktur." Dokumentation TIDE-Forschungsseminar, Universität Paderborn, Sommersemester 2000

[Magl98] Tom Magliery: "A Brief History of Mosaic." Universität Illinois, Urbana-Champaign IL 1998

[MaKo02] Filipe Pereira Martins, Anna Koblinska: *PDF-Workflow*. SmartBooks Publishing, Zürich 2002

[MaMu99] Philip Marden und Ethan Munson: "Why Current Style Sheet Standards Have Failed to Improve Document Engineering." *Proceedings of the Web Engineering Workshop, 8th International World Wide Web Conference*. Toronto, Mai 1999

[Mane99] Kevin Maney: "PowerPoint obsession takes off." *USA Today*, McLean VA, 12. Mai1999

[Mans01] Philip Mansfield: "'Catwalk', a RAD Tool for Dynamic SVG-Generating Web Applications." *XML 2001 Conference Proceedings*, Orlando, Fla. 13. Dezember 2001

[Mark01] Joe Marks (Hrsg.) "Graph Drawing." *Proceedings GD'2000*, Springer Verlag, Williamsburg VA 2001

[Mart95] Carlos Domingo Martinez: "Drawing Software: SmartSketch 1.0." *MacWorld*, 12. Jahrg. Ausg. 9. San Francisco, September 1995

[Marx99] Günter Marxen: „Microsoft kauft Visio." *Kompass*, Ausg. 83, p.34. Universität Köln, Dezember 1999

[Math98] Brian Mathews, Daniel Lee, Brian Dister, John Bowler, Howard Cooperstein, Ajay Jindal, Tuan Nguyen, Peter Wu und Troy Sandal: *A comparison of VML and the W3C Scalable Graphics Requirements.* World Wide Web Consortium, Boston 1998

[Math99] Patrick Ion und Robert Miner (Hrsg.) *Mathematical Markup Language (MathML™) 1.01 Specification*. W3C Recommendation, 7. Juli 1999. URL: http://www.w3.org/TR/REC-MathML

[McCa02] Steve McCannell: "SMIL: Multimedia for the Masses." *Webmonkey* (o.O.) 6. Oktober 2002
URL: http://hotwired.lycos.com/webmonkey/00/41/index4a.html

[McCl94] Scott McCloud: *Understanding Comics*. Kitchen Sink Press, Northhampton, MA 1994

[McGr00a] Chris MacGregor: "A Cancer on the Web called Flash." *FlaZoom* (Editorial) Seabrook, 1. Juni 2000
http://www.flazoom.com/news/cancer_06012000.shtml

[McGr00b] Chris MacGregor: "Hey Flasher, Stop Abusing your Visitors!" *FlaZoom* (Editorial) Seabrook, 20. Juni 2000. URL: http://www.flazoom.com/news/user_06202000.shtml

[McGr01] Chris MacGregor: "Flash: 99% Proof." *FlaZoom* (Editorial) Seabrook, 11. Januar 2001
URL: http://www.flazoom.com/news/99proof_01112001.shtml

[McGr02] Chris McGregor: "Developing User-Friendly Flash Content." Macromedia White Paper, San Francisco / Seabrook 2002. URL:
http://www.macromedia.com/software/flash/productinfo/usability/whitepapers/usability_flazoom.pdf

[McHu96] Michael McHugh: "Acrobat on a Wire: PDF on the Web." *NetWorker*, 6. Jahrg. Ausg.3, p.20. University

of Southern California, Los Angeles 1996

[McLa01] Brett McLaughlin: Java and XML. O'Reilly & Associates, Sebastopol CA 2001

[McLD01] Nigel McKelvey, Ruth Lennon und Thomas Dowling: "XML Applications and How They Interact in a Multimedia Networking Environment." *Proceedings IEI/IEE Symposium on Telecommunications Systems Research*, Dublin 2001

[McLe02] Drew McLellan: "Tableless Layouts with Dreamweaver MX." *Macromedia Dreamweaver MX Application Development Center*, San Francisco 2002
URL: http://www.macromedia.com/desdev/mx/dreamweaver/articles/tableless_layout.html

[MeEf95] Thomas Meyer-Boudnik und Wolfgang Effelsberg : "MHEG Explained." *IEEE Multimedia*, 6. Jahrg. Ausg. 1, pp.26-38. Washington DC, Januar 1995

[Meis97] René Meissner: „Elektronen statt Papier." *c't*, Ausg. 1/97, p.49. Heiz Heise Verlag, Hannover 1997

[Meit03] Wolf Meitz: *Transformation von XML-basierten Daten- und Programmstrukturen in objektorientierte Systeme*. Diplomarbeit, Fachhochschule Furtwangen, Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Wintersemester 2002/03

[Meyr91] Norman Meyrowitz: "Hypertext and pen computing." *Proceedings of the third annual ACM conference on Hypertext* (San Antonio, TX) p.379. ACM Press, New York 1991

[Mian99] John Miano: *Compressed Image File Formats*. ACM Press / Addison-Wesley, New York 1999

[Mich89] Steve Michel (Hrsg.) *Steve Michel's SuperCard Handbook*. Osborne McGraw-Hill, Berkeley 1989

[Micr97] Microsoft Corporation (o.V.) *Microsoft PowerPoint 97 File Format SDK*. Microsoft Graphic Production Unit, Redmond WA 1997

[Micr98a] Microsoft Corporation (o.V.) "Frequently Asked Questions About VML: Flash." *Microsoft Developer Network Library*. Redmond WA, 4. November 1998
URL: http://msdn.microsoft.com/workshop/author/vml/vmlfaq.asp#flash

[Micr98b] Microsoft Corporation (o.V.) "How to Use VML on Web Pages: VML Reference." *Microsoft Developer Network Library*. Redmond WA, 16. November 1998
URL: http://msdn.microsoft.com/workshop/author/vml/ref

[Micr99] Microsoft Corporation (o.V.) "How to Automate PowerPoint Using Visual J++." *Microsoft Knowledge Base*, Redmond WA 1999
http://support.microsoft.com/support/kb/articles/q215/4/84.asp

[Micr02] Microsoft Corporation (o.V.) *Windows Media Format 9 Series SDK Programming Guide*. Microsoft Developer Network Library, Redmond WA 2000
URL: http://msdn.microsoft.com/library/en-us/wmform/htm/programmingguide.asp

[Mint03] Stefan Mintert: „Ordnung muss sein." *iX*, Ausg. 3/03, pp.50ff. Verlag Heinz Heise, Hannover 2003

[MLL99] Jourdan Muriel, Tardif Laurent und Villard Lionel: "SMILY, a SMIL authoring environment." *Proceedings of the 7th ACM int'l conference on Multimedia, Part II* (Orlando, FL) p.198. ACM Press, New York 1999

[Moir97] Charles Moir (Hrsg.) Phil Martin, Mark Neves und Colin Barfoot: *Xara Flare 1.0 Specification*. Xara Limited, Hempstead UK 1997. URL: http://www.xara.com/support/docs/webformat/spec

[Mora00] Tom Moran: "Using Distance Education to Teach Introductory Multimedia Design and Production." *Proceedings of the 18th Annual ACM Conference on Systems Documentation* (Cambridge, MA) pp.93-98. IEEE, Piscataway NJ 2000

[Mori99] Koichi Mori, Koichi Wada und Kazuo Toraichi: "Function Approximated Shape Representation using Dynamic Programming with Multi-Resolution Analysis." Paper, PADC-Labor (Institut für Informationsverarbeitung und Elektrotechnik) Universität Tsukuba 1999

[Mori01] Jenny Morice: *Multimedia Authoring*. Vorlesungsunterlagen, Griffith University (School of Computing and Information Technology) Nathan, Queensland 20001

[Morn98] Allen Mornington-West: "MHEG-5 and Java." *EBU Technical Review*, Ausg. 275. Genf, März 1998

[Morr01] Michael Morrison: *HTML & XML for Beginners*. Irwin Professional Publishing, Scarborough 2001

[Mose99] Felix Moser: „VML und PGML – Endlich Vektor-Bilder fürs Internet?" *Publisher*, Ausg. 3-99, p.8. Zürich, Juni 1999

[Moss01] William Moss: "Customizing RealPresenter 8." Paper, Clemson University, College of Engineering and Science (Department of Mathematical Sciences) Clemson SC, 17. April 2001

[Moul02] Stuart Moulthrop (Hrsg.) "Hypermedia and Multimedia." *Proceedings of the 13th Conference on Hypertext and hypermedia*, p.196. ACM Press, New York 2002

[Muel00a] Florian Müller: *User Interface Design for Video on the Web*. Diplomarbeit, Fachhochschule Furtwangen, Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Winterse-

mester 1999/2000

[Muel00b] Heiko Müller: *Design and implementation of a browser interface for viewing XML documents*. Diplomarbeit, Fachhochschule Trier (Fachbereich Angewandte Informatik) Wintersemester 1999/2000

[Muel02a] Regina Müller: „Spielwiese." *iX*, Ausg. 5/02, pp.144ff. Verlag Heinz Heise, Hannover 2002

[Muel02b] Manfred Müller-Späth: „Composer Pro 2.0." *Internet Professionell*, Ausg. 8/02. VNU Business Publications, München 2002

[Muen98] Stefan Münz: „Formatierung für HTML-Elemente", in: *Self HTML 7.0*, München 1998

[Murd02] Arnold Murdock: *Multimedia Design Guidelines*. Seminar "Multimedia Production in Workforce Education", Southern Illinois University, College of Education and Human Services (Department of Workforce Education and Development) Carbondale IL 2002

[Musi02] Marko Musiat: „Webmigration unter ToolBook", in: *BWL-Lernsoftware Interaktiv*. Vortrag zum gleichnamigen Projektworkshop, Technische Universität Dresden, Media Design Center (Fakultät Wirtschaftswissenschaften) Dresden 2002

[Negr95] Nicholas Negroponte: *Total Digital*. Bertelsmann, München 1995

[Nels88] Ted Nelson: *Dream Machines*. Microsoft Press, Redmond WA 1988

[Neub96] Matt Neuburg: "Looking for the Future." *MacTech*, 12. Jahrg. Ausg.9. Westlake Village CA 1996

[Neum02] Andréas Neumann: "Comparing SWF (Shockwave Flash) and SVG (Scalable Vector Graphics) file format specifications." *Carto.net* / Kartographisches Institut der ETH Zürich (Dept. Bau, Umwelt und Geomatik) Zürich, 24. Januar 2002
http://www.carto.net/papers/svg/comparison_flash_svg.html

[Nevi96] François Neville (Hrsg.) "PowerMedia Pro." *PC Today*, 10. Jahrg. Ausg. 12. Lincoln NE, Dezember 1996

[NeWi00] Andreas Neumann und Andréas Winter: *Kartographie im Internet auf Vektorbasis, mit Hilfe von SVG nun möglich*. Universität Wien (Institut für Geographie und Regionalforschung), ETH Zürich (Institut für Kartographie) 4. Dezember 2000. URL: http://www.carto.net/papers/svg

[Newm00] Debbie Newman: "Spice Up Your Web Pages with HTML+TIME." *Microsoft Developer Network Library*, Redmond WA 2000. URL: http://msdn.microsoft.com/library/en-us/dntime/html/htmltime.asp

[Niel95] Jakob Nielsen: *Multimedia and Hypertext: The Internet and Beyond*. Morgan Kaufmann, S. Francisco 1995

[Niel96a] Jakob Nielsen: "Disabled Users and the Web." *Alertbox* / SunSoft, Palo Alto CA, Oktober 1996
URL: http://www.sun.com/columns/alertbox/9610.html

[Niel96b] Jakob Nielsen: "Why Frames Suck (Most of the Time)." *Alertbox*, Mountain View CA, Dezember 1996
URL: http://www.useit.com/alertbox/9612.html

[Niel97] Jakob Nielsen: "The Need for Speed." *Alertbox*, Mountain View CA, März 1997
URL: http://www.useit.com/alertbox/9703a.html

[Niel99] Jakob Nielsen: "User Interface Directions for the Web." *Communications of the ACM*, 42. Jahrg. Ausg. 1, pp.65-72. New York, Januar 1999

[Niel00a] Jakob Nielsen: "Flash: 99% Bad." *Alertbox*, Mountain View CA, Oktober 2000
URL: http://www.useit.com/alertbox/20001029.html

[Niel00b] Jakob Nielsen: *Erfolg des Einfachen*. Markt und Technik Verlag, München 2000

[NiLy90] Jakob Nielsen und Uffe Lyngbaek: "Two field studies of hypermedia usability," in: Ray McAleese und Catherine Green (Hrsg.) *Hypertext: State of the Art*, Ablex, pp.64-72, Intellect Press. Veröff. i.R.d. "Hypertext'2 Conference", York UK, Juni 1989

[NGWA02] National Ground Water Association (o.V) "No Bad Visuals!" *Preparational Documents*, June 2002 NGWA Conference on MTBE. Westerville OH 2002

[Norv99] Peter Norvig: "The Making of the Gettysburg PowerPoint Presentation." Mountain View CA, Januar 1999
URL: http://www.norvig.com/Gettysburg/making.html

[NSW02] Andreas Neumann, Peter Sykora und Andréas Winter: „Das Web auf neuen Pfaden." *c't*, Ausg. 20/02, pp.218-225. Verlag Heinz Heise, Hannover 2002

[Oaks01] Scott Oaks: *Java Security*. O'Reilly & Associates, Sebastopol CA 2001

[O'Don97] Bob O'Donnell: "Microsoft should open Office 97 file formats to the public domain." *InfoWorld*, San Mateo CA, 1. September 1997

[Ogbu00] Uche Ogbuji: "What comes after XML?" *Unix Insider (SunWorld Online)* Southboro MA, 23. Oktober 2000. URL: http://uche.ogbuji.net/tech/pubs/svgpres.xhtml

[Ogbu03] Uche Ogbuji: "Apple uses XML for new presentation format, but shuns SVG." *XMLHack*, York, 10. Januar 2003. URL: http://www.xmlhack.com/read.php?item=1865

[Oebb97] Alfons Oebbeke: „Illustrationen im Web: CAD.“ *AEC Web-Magazin*, Neustadt a.d. Weinstrasse 1997 URL: http://www.archmatic.com/m/2d3d-3.htm

[Olip96] Zan Oliphant: *Programming Netscape Plug-Ins*. Sams Publishing, Indianapolis 1996

[Orch97] Dave Orchard: Java 1997: “A detailed look at where Java is going this year and in the near future.” *JavaWorld*, Ausg. 6/97. Vancouver, Juni 1997

[OyGo01] Keizo Oyama und Hironobu Gotoda (Hrsg.) *Proceedings of the International Conference on Dublin Core and Metadata Applications*. National Institute of Informatics, Tokyo 2001

[PaAd98] Howard Park und Alexander Adams: “Microsoft PowerPoint Viewers,” in: *Conversion of Macintosh Formatted PowerPoint Files to PC Format*. Projektdokumentation EE628, University of Hawaii at Manoa (Department of Electrical Engineering) Honolulu 1998

[Paol02] Jean Paoli: “Bringing the XML Vision to the Desktop with ‘Office 11’.” *XML 2002 Conference Proceedings*, Baltimore, 11. Dezember 2002

[Para97] John Parazette-Tillar: “Java Software Reviews.” *Animation World*, 2. Jahrg. Ausg. 2, pp.47-49. Los Angeles, Calif. Mai 1997

[Park01] Ian Parker: “Absolute Powerpoint: Can a software package edit our thoughts?” *The New Yorker*, p.76. New York, 28. Mai 2001

[Pens00] George Penston: “The Creative Toolbox: Open-Standard Upstart SVG Takes on Macromedia’s SWF.” *CreativePro.com*, Livingston, 17. August 2000. URL: http://www.creativepro.com/story/feature/8057.html

[Perr97] Ralph Perrine: “Using Java in Your Internet Applications,” in: Randall Tamura (Hrsg.) *Lotus Notes and Domino Server 4.6*. Sams Publishing, Indianapolis 1997

[Pfei01a] Eckehard Pfeifer: „Verhaltensforschung.“ *VBA Magazin*, Ausg. 4/01. München, Juli 2001

[Pfei01b] Eckehard Pfeifer: „VML verstehen.“ *VBA Magazin*, Ausg. 6/01. München, November 2001

[Pieg95] Winfried Piegsda: *Integrierte Digitale Medienproduktion*. Diplomarbeit, Fachhochschule Furtwangen. Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Wintersemester 1994/1995

[Piem98] Claudia Piemont: „Ein Lächeln fürs Web.“ *c’t*, Ausg. 20/98, pp.80ff. Verlag Heinz Heise, Hannover 1998

[Piff02] Pamela Piffner: *Inside the Publishing Revolution: The Adobe Story*. PeachPit Press, Berkeley 2002

[Pirn01] Paul Pirner: “Brighten your Presentation Visuals with Fresh Images.” *InfoComm News & Information Network* (International Communications Industries Association) Fairfax VA, 18. Mai 2001

[PGML98] Nabeel Al-Shamma, Robert Ayers, Richard Cohn, Jon Ferraiolo, Martin Newell, Roger de Bry, Kevin McCluskey und Jerry Evans: *Precision Graphics Markup Language (PGML)*. W3C Note, 10. April 1998 URL: http://www.w3.org/TR/1998/NOTE-PGML

[Phil98] Barry Phillips: “Backing the Latest W3C Standards: Microsoft Frowns on SMIL”, in: “Browsers See Things Differently.” *IEEE Computer*, 31. Jahrg. Ausg. 10, p. 30. Washington DC, Oktober 1998

[PMEB01] Steve Probets, Julius Mong, David Evans und David Brailsford: “Vector Graphics: From PostScript and Flash to SVG.” *Proceedings of the 2001 ACM Symposium on Document Engineering* (Atlanta) pp.135-143. ACM Press, New York 2001

[Pomp99] Stevyn Pompelio: “Animated GIFs”, in: *How to Make an Annoying WebPage* (o.O.) 1999 URL: http://www.users.nac.net/falken/annoying/ani_gif.html

[Powe00] Thomas Powell: “SVG – Something Very Good, or a flash in the pan?” *ITworld.com*, Southboro MA, 22. Juni 2000. URL: http://www.pint.com/resources/articles/art_it_svg.htm

[PoWi00] Oliver Pott und Gunter Wielage: *XML. Praxis und Referenz*. Markt und Technik Verlag, München 2000

[Prob00a] Steve Probets: “Semantics of Macromedia's Flash (SWF) Format and its Relationship to SVG.” Electronic Publishing Research Group, University of Nottingham (School of Computer Science and Information Technology) Nottingham 2000. URL: http://www.ep.cs.nott.ac.uk/projects/SVG/flash2svg/swfformat.html

[Prob00b] Steve Probets: “How Flash Animation is managed by an SWF to SVG Translation Package.” Electronic Publishing Research Group, University of Nottingham (School of Computer Science and Information Technology) Nott’gham 2000. URL; http://www.ep.cs.nott.ac.uk/projects/SVG/flash2svg/swfsvganim.html

[Pusc00] Frank Puscher: *Das Flash-Kochbuch 4.0*. dpunkt Verlag, Heidelberg 2000

[Quin02] Antoine Quint: “SVG Tips and Tricks: Adobe’s SVG Viewer.” *XML.com*, O’Reilly & Associates, Sebastopol CA 3. Juli 2002. URL: http://www.xml.com/pub/a/2002/07/03/adobesvg.html

[QuVa94] Vincent Quint und Irène Vatton: “Making Structured Documents Active.” *Electronic Publishing – Origination, Dissemination and Design*, 7. Jahrg. Ausg. 2, pp. 55-74. John Wiley & Sons, Chichester 1994

[Ragu99] Dack Ragus: “Flash is Evil.” Minneapolis, September 1999. URL: http://dack.com/web/flash_evil.html

[Ragu00] Dack Ragus: “Flash vs. HTML: A Usability Test.” Minneapolis, Januar 2000

http://www.dack.com/web/flashVhtml

[Rand97] Glenn Randers-Pehrson: "MNG – A Multiple-Image Format in the PNG Family." *World Wide Web Journal*, 2. Jahrg. Ausg.1. O'Reilly & Associates, Sebastopol CA1997

[Rask03] Avery Raskin: "Adobe Photoshop Album – Feature Highlights." Adobe Systems Incorporated, San Jose 2003. URL: http://www.adobe.com/products/photoshopalbum/pdfs/photoshopalbum_nfhs.pdf

[Raux01] Rob Raux: "History of Flash," in: Rob Raux und Sean Haneberg: *ACM Flash Tutorials*. ACM at UB (University at Buffalo) The State University of New York, 2001: http://acm.cse.buffalo.edu/~rjraux/flash

[Reib99] Helmut Reibold: „Animierte Banner." *Internet Magazin*, Ausg. 8/99. WEKA-Verlag, Poing/München 1999

[Reid99] Rodney Reid: "Conversion table between SVG and VML structures." *ToxiCorp*, San Francisco 1999 URL: http://www.toxicorp.com/svg_to_vml.html

[Reid00] Rodney Reid: "VML Tutorial." *ToxiCorp*, S.Francisco 2000. URL:http://www.toxicorp.com/vml_start.html

[Reid01] Rodney Reid: "SMIL 2.0: Codeless Animation in HTML." *Webmonkey* (o.O.) 14. Juni 2001 URL: http://hotwired.lycos.com/webmonkey/01/24/index3a.html

[Rein98a] L Lisa Rein: "XML Resource Guide: HGML." *Seybold Publications*, O'Reilly & Associates, Sebastopol CA, 19. Juni 1998. URL: http://www.3cts.com/books/enxml/hgml.htm

[Rein98b] Lisa Rein: "VML and PGML: A Comparison." *XML.com*. O'Reilly & Associates, Sebastopol CA, 22. Juni 1998. URL: http://www.xml.com/pub/a/98/06/vector/vmlpgml.html

[Reus00] Fritz Reust: „«WAP» – heute enttäuschend, aber mit Zukunft." *Neue Zürcher Zeitung* (Special „Telekommunikation") Ausg.130, p.94. Zürich, 6. Juni 2000

[RFEP99] Paul Reynolds, Steven Furnell, Michael Evans und Andrew Phippen: "Hyper Graphics Markup Language for optimising WWW access in wireless networks." *Proceedings of the 4th Annual Scientific Conference on Web Technology* (Euromedia'99) München, 26. April 1999

[RHO99] Lloyd Rutledge, Lynda Hardman und Jacco van Ossenbruggen: "Evaluating SMIL: three user case studies." *Proceedings of the 7th ACM international conference on Multimedia, Part II* (Orlando, FL) pp.171-174. ACM Press, New York 1999

[RHOB99] Lloyd Rutledge, Lynda Hardman, Jacco van Ossenbruggen und Dick Bulterman: "Mix'n'Match: Exchangeable Modules of Hypermedia Style." *Proceedings of the 10th ACM Conference on Hypertext and hypermedia* (Darmstadt) pp.179-188. ACM Press, New York 1999

[Ricc98] Mike Ricciuti: "Adobe pitches graphics standard." *News.com* (c|net Networks) San Francisco, 13. April 1998. URL: http://news.com.com/2100-1001-210111.html

[Rieh01] Andreas Riehl: *Usability-Unterschiede zwischen Hypertextmedien und klassischen GUI-Oberflächen*. Diplomarbeit, Fachhochschule Furtwangen. Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Sommersemester 2001

[Robe95] Jason Roberts: *Director Demystified*. Addison-Wesley, Boston 1995

[ROHB98] Lloyd Rutledge, Jacco van Ossenhruggen, Lynda Hardman und Dick Bulterman: "Practical Application of Existing Hypermedia Standards and Tools." *Proceedings of the 3rd ACM Conference on Digital Libraries* (Pittsburgh) pp.191-199. ACM Press, New York1998

[Rose00] Scott Rosenberg: "DEN, Boo: R.I.P." *Salon.com*, San Francisco, 19. Mai 2000 URL: http://salon.com/tech/col/rose/2000/05/19/den_boo

[Roth98] Silvia Rothen: „Bringt uns Java mehr als winkende Pinguine?" *M+K Computermarkt*, Ausg. 96-8, pp.21-23. Luzern, August 1998

[Roto02a] Jennifer Rotondo: "Use design to communicate that you know your listeners." *Presentations*, Ausg 1/02. VNU Business Media, Minneapolis, Januar 2002

[Roto02b] Jennifer Rotondo: "Pushing PowerPoint charting options to the limit." *Presentations*, Ausg 4/02. VNU Business Media, Minneapolis, April 2002

[Rott01] Christian Rotterdam: *Grafikformate im WWW*. Referat i.R.d. „WissensNetzwerkes für Unternehmer" (ERFA-Gruppe) Billerbeck, 26. Juli 2001 http://www.lernen-im-internet.de/schwarzes_brett/referate_pdf/bildformate_referat_rotterdam.pdf

[Ruhl97] Matthias Ruhl: *Fraktale Bildkompression: Adaptive Partitionierungen und Komplexität*. Master-Thesis, Albert-Ludwigs-Universität Freiburg (Mathematische Fakultät) Freiburg im Breisgau, April 1997

[Rutl99] Lloyd Rutledge: „Tonangebend." *iX*, Ausg. 10/99, pp.58ff. Verlag Heinz Heise, Hannover 1999

[Saul02] Sanjay Sauldie: „Flash oder kein Flash?" *IUMM* (Internet Marketing u. Multimedia-Forum) Stuttgart 2002

[Scar97] Lori Scarlatos: "Designing interactive multimedia." *Proceedings of the 5th ACM international conference on Multimedia* (Seattle) pp.215-218. ACM Press, New York 1997

[Scar98] Kane Scarlett: "Advis designs business extensions for Apple's WebObjects." *JavaWorld*, Ausg. 6/98. Van-

couver, Juni 1998

[ScCo01] Patrick Schmitz und Aaron Cohen: *SMIL Animation*. W3C Recommendation, 4. September 2001 URL: http://www.w3.org/TR/smil-animation

[Scha90] Thomas Schaller: *Business Grafik*. Sybex Verlag, Düsseldorf 1990

[Schm95] John Schmitz: "Java, Super Browsers, and the Virtual Classroom." SCALE Seminar Series, University of Illinois (College of Agricultural, Consumer and Environmental Sciences) Urbana-Champaign, IL 1995

[Schm99] Mark Schmid: *Audio- und Video-Datentransfer im Internet*. Diplomarbeit, Fachhochschule Furtwangen, Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Wintersemester 1998/99

[Schm00] Patrick Schmitz (Hrsg.) "HTML+SMIL Language Profile", in: Ayars et al. [wie SMIL01]: *Synchronized Multimedia Integration Language (SMIL) Boston Specification*. W3C Working Draft, 22. Juni 2000 URL: http://www.w3.org/TR/2000/WD-smil-boston-20000622/html-smil-profile.html

[Schm02] Patrick Schmitz: "Multimedia meets computer graphics in SMIL2.0: a time model for the web." *Proceedings of the 11th international conference on World Wide Web* (Honolulu) pp.45-53. ACM Press, New York 2002

[Scho02] Peter Schonefeld: "SVG is Real Flash." *DigitalCraft*, Melbourne 2002 URL: http://www.digitalcraft.com.au/svg/blurbs/blurb001.asp

[Schr00] Bendetta Schroth: „Browser-Plug-ins zum Nulltarif." *tecChannel*, München, 10. Januar 2000 URL: http://www.tecchannel.de/internet/195

[Schu02] Stefan Schumacher: „Bildlich gesprochen." *iX*, Ausg. 12/02, pp.60-67. Verlag Heinz Heise, Hannover 2002

[Sear98] David Searls: "It's the Story, Stupid!" Emerald Hills, 16.August 1998. URL: http://searls.com/present.html

[SeLo97] Robert Seetzen, Jörn Löviscach: „Bilder mit Durchblick." *c't*, Ausg. 10/97. Heise-Verlag Hannover, 1997

[SGML86] Charles Goldfarb (Hrsg.) *Standard Generalized Markup Language (SGML)*. ISO/IEC 8879, Genf 1986

[Shah96] Rawn Shah: "Authoring Tools." *JavaWorld*, Ausg. 7/96. Vancouver, Juli 1996

[Shne02] Ben Shneiderman: "Museum of User Interfaces: Harvard Graphics," aus: *Human Factors in Computer and Information Systems*. University of Maryland (Department of Computer Science) College Park, MA 2002

[Shoy01] Vitaly Shoykhet: "Java Implementation in Multimedia Rich Embedded Devices." Paper i.R.d. *Spring 2001 Software Engineering Symposium*, DePaul University (School of Computer Science, Information Systems, and Telecommunications) Chicago 2001

[Simi02] Simone Simiona (Hrsg.): "Tips and Tricks for Using PowerPoint." *FMHS Learning & Teaching News*, Ausg 5. University of Auckland (Faculty of Medical and Health Services) Auckland NZ, Mai 2002

[Sipp02] Michael Sippey: "Round Two – Sippey on Harpold", in: *Click to Add Title*, Oakland 2002 URL: http://www.clicktoaddtitle.com/#000019

[SMIL98] Philipp Hoschka (Hrsg.) *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification* (Synchronized Multimedia Working Group) W3C Recommendation, 15. Juni 1998 URL: http://www.w3.org/TR/REC-smil

[SMIL01] Jeff Ayars, Dick Bulterman, Aaron Cohen, Ken Day, Erik Hodge, Philipp Hoschka, Eric Hyche, Muriel Jourdan, Michelle Kim, Kenichi Kubota, Rob Lanphier, Nabil Layaïda, Thierry Michel, Debbie Newman, Jacco van Ossenbruggen, Lloyd Rutledge, Bridie Saccocio, Patrick Schmitz und Warner ten Kate (Hrsg.) *Synchronized Multimedia Integration Language (SMIL 2.0)* W3C Recommendation, 7. August 2001 URL: http://www.w3.org/TR/smil20

[Smit95] Philip Smith: "Block-Based Formatting with Encapsulated PDF." Electronic Publishing Research Group, University of Nottingham (School of Computer Science and Information Technology) Nottingham 1995 URL: http://www.eprg.org/pdfcorner/complete.pdf

[Smit00] Adrian Smith: "VML - Vector Graphics on the Internet" *Vector*, 16. Jahrg. Ausg. 4. Cambridge, April 2000

[Smit01] Adrian Smith: "VML and SVG Compared." Causeway Graphical Systems, Malton UK 2001 URL: http://www.grapl.com/vmlnotes/introduction/vml_and_svg_compared.htm

[Smit02] Adrian Smith (Hrsg.) "VML, SVG, PDF" *Dyalog APL Release Notes*, Causeway Graphical Systems, Malton UK 2002. URL: http://www.causeway.co.uk/support/rainvml/dyalog

[SmFa96] Brenda Smith Faison: "Graphic Designers in Transistion – From Print Communications to Interactive Media Design." *Interactions*, 3. Jahrg. Ausg. 1, pp.39-58. ACM Press, New York, Januar 1996

[SSSA99] Jared Spool, Taran Scanlon, Will Schroeder, Carolyn Snyde und Terri DeAngelo: *Web Site Usability: A Designer's Guide*. Morgan Kaufmann Publishers, San Francisco 1999

[Stan94] Anton Stankowski: „Visualisierung", in: Anton Stankowski und Karl Duschek (Hrsg.) *Visuelle Kommunikation*, pp.20-52. Dietrich Reimer Verlag, Berlin 1994

[Star01] Benjamin Stark: *Flash Weather - Vektorisierung von Raum- und Zeitbezogenen Daten zur Visualisierung mit Macromedia Flash*. Diplomarbeit, Universität Osnabrück (Fachbereich Mathematik/Informatik). Osnabrück, Februar 2001

[Stei99] Ralf Steinmetz: *Multimedia-Technologie*. Springer Verlag, Berlin 1999

[Stei99] Dominik Stein: „Internetbasierte Informationssysteme II“ *Lernzettel*, Universität Essen (FB5) SS 1999

[Stew01] Thomas Stewart: “Ban It Now! Friends Don’t Let Friends Use PowerPoint.” *Fortune*, 134. Jahrg. Ausg. 3, pp.210f. New York, Februar 2001

[StLa02] Simon St. Laurent: “Macromedia reinvents the Web.” *O’Reilly Network*, Sepastopol CA, 11. März 2002 URL: http://www.oreillynet.com/lpt/wlg/1197

[Stub96] Erik Stubkjær: “WWW, Clickable Maps and Education in LIS/GIS,” in: *Proceedings FIG Commission 2 Joint Workshop on Computer Assisted Learning and Achieving Quality in the Education of Surveyors*, pp. 263-273. Helsinki, September 1996

[SuCi02] Stephanie Sullivan und Ginger Ciuperca: “Simple Styling with CSS.” *Macromedia Dreamweaver MX Application Development Center*, San Francisco 2002
URL: http://www.macromedia.com/desdev/mx/dreamweaver/articles/css_styling.html

[SVF95] Scott Sherman (Hrsg.) *Specification for the Simple Vector Format (SVF)*. Bellingham WA, 1995/2000 http://svf.org/spec.html [v.2.0] bzw. http://www.softsource.com/svf/spec.html [v.1.1]

[SVG99] Jon Ferraiolo (Hrsg.) *Scalable Vector Graphics (SVG) Specification*. W3C Working Draft, 11. Februar 1999 http://www.w3.org/TR/1999/WD-SVG-19990211

[SVG01] Jon Ferraiolo (Hrsg.) *Scalable Vector Graphics (SVG) 1.0 Specification*. W3C Recommendation, 4. September 2001. URL: http://www.w3.org/TR/SVG

[SYS98] Patrick Schmitz, Jin Yu und Peter Santangeli: *Timed Interactive Multimedia Extensions for HTML: HTML+TIME - Extending SMIL into the Web Browser*. W3C Note, 18. September 1998 http://www.w3.org/TR/NOTE-HTMLplusTIME

[Tard01] Laurent Tardif: “SVG Verbose.” Paper i.R.d. *XML Human-Readability Discussion*, Monash University (School of Computer Science and Software Engineering) Clayton, Victoria 2001

[Terra02] Alejandro Terrazas: *Java Media APIs: Cross-Platform Imaging, Media and Visualization*. Sams Publishing, Indianapolis 2002

[Tetz95] Jon Stephenson von Tetzchner: „MultiTorg Opera.“ *Proceedings of the 3rd International World-Wide Web Conference* (WWW’95) Darmstadt, April 1995

[This00] Frank Thissen: *Screen-Design-Handbuch*. Springer-Verlag, Berlin Heidelberg 2000

[Tren02] Andrea Trentini: “A Java-based framework to support computer-assisted creation of structured XML documents.” *ACM SIGAPP Applied Computing Review*, 10. Jahrg. Ausg. 1, pp.48-53. ACM Press, NY 2002

[Treu95] Jack Treuhaft: *Multimedia Design Considerations*. Vorlesungsunterlagen, Seminar “Using Technology in Education”. Algonquin College of Applied Arts and Technology, Ottawa 1995

[Trig88] Randall Trigg: “Guided Tours and Tabletops: Tools for Communicating in a Hypertext Environment.” *ACM Transactions on Office Information Systems*, 6. Jahrg. Ausg. 4, pp.398-414. New York, Oktober 1988

[Trin02] Andrea Trinkwalder: „Vektorgrafik fürs Internet: WebDraw 1.0.“ *c’t*, Ausg. 8/02, p.72. Verlag Heinz Heise, Hannover 2002

[Trip02] Bill Trippe: “SVG – The Future of Web Rendering?” *The Gilbane Report*, 10. Jahrg. Ausg. 6, pp.1-11. Cambridge MA, Juli/August 2002

[True01] Michael Truese: “Flash: 99% Good.” *WebmasterBase / SitePoint*, Melbourne, 3. April 2001 http://www.webmasterbase.com/article/374

[Viol01] Karsten Violka: „Streit um Java geht weiter.“ *Heise Newsticker*, Verlag Heiz Heise, Hannover, 17. August 2001. URL: http://www.heise.de/newsticker/data/kav-17.08.01-002

[Voel98] Hubert Voelzgen: „PDF-Format und Acrobat.“ *RHRZ-Aktuell*, Ausg. 16. Universität Bonn, Oktober 1998

[Voge96] Thomas Vogel: *HTML & WWW-Browser*. Ausarbeitung eines Vortrages i.R.d. Proseminars „World-Wide-Web“ an der TU Darmstadt, Fachbereich Informatik (Fachgebiet Verteilte Systeme) Darmstadt, Wintersemester 1995/96

[Voge02a] Martin Vogel: „Das FIF-Format“, in: *Tipps für Web-Autoren, oder: die Kunst, zu informieren ohne zu erschrecken*. Dortmund, 16. September 2002
URL: http://www.martinvogel.de/pc/webbing.html#fif

[Voge02b] Uwe Vogel: „Videoclips fürs Handy.“ *c’t*, Ausg. 13/02. Verlag Heinz Heise, Hannover 2002

[Vosh99] Shermin Voshmgir: “XML Tutorial – Glossar.” *JavaCommerce* Application Network, Wien / Farmington Hills, MI 1999. URL: http://www.javacommerce.com/tutorial/xmlj/glossary.htm

[VML98] Brian Mathews, Daniel Lee, Brian Dister, John Bowler, Howard Cooperstein, Ajay Jindal, Tuan Nguyen, Peter Wu und Troy Sandal: *Vector Markup Language (VML)*. W3C Note, 13. Mai 1998
URL: http://www.w3.org/TR/NOTE-VML

[Wagn98] Richard Wagner: "File | New: A Few Awards and a Good-Bye" *Delphi Informant*, Ausg. 1/98, Januar 1998

[Wald00] Rick Waldron: "The Flash History." *FlashMagazine*, Oslo 2000
URL: http://www.flashmagazine.com/html/413.htm

[Wald02] Harry Waldman: "Three good reasons to stop using PowerPoint." *Presentations*, Ausg. 9/02. VNU Business Media, Minneapolis, September 2002

[Wals98] Jeffrey Walsh: "Proposed language creates Web vector graphics on the fly" *InfoWorld*, S. Mateo, 28.5.1998

[Walt98] Mark Walter: "Adobe, IBM Brew a Java PGML Viewer." *Seybold Report on Internet Publishing Special, XML.com*. O'Reilly & Associates, Sebastopol CA, 17. November 1998
URL: http://www.xml.com/pub/a/seyboldreport/ipx981105.html

[Walt00] Wilhelm Walter: *Informatik 3*. Vorlesungs-Skript zur gleichnamigen Vorlesung. Fachhochschule Furtwangen, Fachbereich Digitale Medien (Studiengang Medieninformatik) Furtwangen im Schwarzwald, Sommersemester 2000

[Watt02] Andrew Watt, Chris Lilley, Daniel Ayers, Randy George, Christian Wenz, Tobias Hauser, Kevin Lindsey und Niklas Gustavsson: *SVG Unleashed*. Sams Publishing, Indianapolis 2002

[WAP98] WAPForum (o.V.) *WAP Binary XML Content Format Specification 1.0.* Wireless Application Forum / Open Mobile Alliance. Mountain View CA, 30. April 1998

[WCGM99] David Cruikshank, John Gebhardt, Lofton Henderson, Roy Platon und Dieter Weidenbrück: *WebCGM Profile*. W3C Recommendation, 21. Januar 1999. http://www.w3.org/TR/1999/REC-WebCGM-19990121

[Wein97] Harald Weinreich: *Ergonomie von Hypertext-Systemen und das World Wide Web*. Diplomarbeit, Fachbereich Informatik der Universität Hamburg, Februar 1997

[WeWi94] Louis Weitzman und Kent Wittenburg: "Automatic Presentation of Multimedia Using Relational Grammars Documents." *Proceedings of the second ACM international conference on Multimedia* (San Francisco) pp. 443-451. ACM Press, New York 1994

[Woda02] Krzysztof Woda: *Dateiformate*. Vorlesungsunterlagen Grundstudium Wirtschaftsinformatik, European University Viadrina (Lehrstuhl ABWL) Frankfurt/Oder, Wintersemester 2002/03

[WoHi00] Sascha Wolter und Gerald Himmelein: „Flashkiller? Webdesign mit Adobe LiveMotion 1.0." *c't*, Ausg. 13/00, p.78. Verlag Heinz Heise, Hannover 2000

[WSG00] Sandra Woolley, Mike Spann und Dave Gibson: "Inter-Frame Coding," in: *Data Compression of Coronary Angiography Video Sequences*. The University of Birmingham (School of Electronic and Electrical Engineering) Birmingham 2000

[WSML98] David Duce und Bob Hopgood: *Web Schematics on the World Wide Web*. W3C Note, 31. März 1998
URL: http://www.w3.org/TR/1998/NOTE-WebSchematics

[Wild99] Erik Wilde: *Wilde's WWW – Technical Foundations of the World Wide Web*. Springer Verlag, Berlin 1999

[Will01] Daniel Will-Harris: "The battle of the Vector Graphics." Point Reyes, CA 2001
URL: http://www.will-harris.com/wire/html/vector_graphics.html

[Will02] Dawn Williams: "Template Anyone?" *The Observer*, 3. Jahrg. Ausg. 6. Corpus Christi TX, 9. August 2002

[Wine88] Dave Winer: "Outliners and Programming." *Dave's Manila Playtown*, Burlingame, CA 1988
http://dave.editthispage.com/outlinersProgramming

[Wine98a] Dave Winer: "Flash and PGML." *DaveNet*, Acton MA, 14. April 1998
http://davenet.userland.com/1998/04/14/flashandpgml

[Wine98b] Dave Winer (Hrsg.) "Flash and PGML - News and commentary from the cross-platform scripting community." *Scripting News*, Burlingame CA, 15. April 1998
URL: http://www.scripting.com/mail/mail980415.html

[Wine98c] Dave Winer: "XMLizations." *DaveNet*, Acton MA, 20. April 1998
URL: http://davenet.userland.com/1998/04/20/xmlIzations

[Wu99] Peter Wu: "An Introduction to VML." *XML Europe'99 Conference Proceedings*, Granada, April 1999

[XML97] Tim Bray, Jean Paoli und Michael Sperberg-McQueen: *Extensible Markup Language (XML)* W3C Working Draft, 7. August 1997. URL: http://www.w3.org/TR/WD-xml-970807

[XML98] Tim Bray, Jean Paoli und Michael Sperberg-McQueen: *Extensible Markup Language (XML) 1.0*. W3C Recommendation, 10. Februar 1998. URL: http://www.w3.org/TR/1998/REC-xml-19980210

[YLQ98] Andrew Yang, James Linn und David Quadrato: "Developing integrated Web and database applications using Java applets and JDBC drivers." *Proceedings of the 29th SIGCSE technical symposium on Computer science education* (Atlanta) pp.302-206. ACM Press, New York 1998

[Zeld99] Jeffrey Zeldman: "Netscape Bites Bullet." *A List Apart*, New York 1999
URL: http://www.alistapart.com/stories/bullet

[Zeld01] Jeffrey Zeldman (mit Jim Heid) "SMIL When You Play That." *A List Apart*, Ausg. 101. New York,16 März 2001. URL: http://www.alistapart.com/stories/smil

[Zell89] Polly Zellweger: "Scripted Documents: A Hypertext Path Mechanism." *Proceedings of the second annual ACM conference on Hypertext* (Pittsburgh) pp.1-14. ACM Press, New York 1989

[Zeln98] Nate Zelnick: "Nifty Technology and Nonconformance: The Web in Crisis." *IEEE Computer*, 31. Jahrg. Ausg. 10, pp. 115-116,119. Washington DC, Oktober 1998

[Zieg99] Joakim Ziegler: "Structured data and the death of WYSIWYG." *Advogato*, 4. Dezember 1999
URL: http://www.advogato.org/article/19.html

[Ziel02] Dave Zielinski: "Clockwork." *Presentations*, Ausg. 2/02. VNU Business Media, Minneapolis, Februar 2002

[Zmoe00] Christine Zmoelnig: *Skip Intro - An Aesthetic and Economic Analysis of Macromedia's Flash Technology*. Dissertation, Hypermedia Research Centre (MA Mypermedia Studies) Westminster, Wintersemester 1999/2000

Software und Source Code-Referenzen

[Adob99] Adobe Systems GmbH (o.V.) „Skalierbare Vektor-Grafiken: Allgemeine Fragen" (SVG-FAQ) *Web Center Features*, Adobe Deutschland, Unterschleißheim 1999: http://www.adobe.de/web/features/svg/faq.html

[Adob02] Alexey Cherepakhin et al: *Adobe SVG Viewer*. Adobe Systems Incorporated, San Jose 2002
URL: http://www.adobe.com/svg/overview/whatsnew.html

[Appl03] Apple Computer Incorporated (o.V.) *Presenting Keynote: Great ideas start here*. Cupertino CA 2003
URL: http://www.apple.com/keynote

[Agui00] Eduardo Aguiar: *LibSWF*. Staatl. Universität Rio de Raneiro, 2000: http://sourceforge.net/projects/libswf

[Argo00] Alejandro Ramirez, Philippe Vanpeperstraete, Andreas Rueckert, Kunle Odutola und Jeremy Bennett: *ArgoUML User Manual*. Tigris Software Engineering, San Francisco 2000
URL:http://argouml.tigris.org/documentation/defaulthtml/manual

[Core95] Corel Corporation (o.V.) *Corel CMX Viewer*. Ottawa 1995. URL: http://www.corel.com/corelcmx

[Core98] Corel Corporation (o.V.) *Corel Barista - Java Based Publishing to the Web*. Ottawa 1998
URL: http://www.corel.com/internet/newbarista.htm 1998

[Dele99] Carmen Delessio: *WMF2SVG*. BlackDirt Software, New York 1999 http://www.blackdirt.com/graphics/svg

[DeLo02] Ken DeLong: *Fractal Image Compression Software*. Femtosoft Technologies, Oakland CA 2002
URL: http://www.femtosoft.biz/fractals/fractal.html

[Duge99] Jean-Luc Dugelay, Beat Fasel, Vincent Paoletti und Nicolas Vallet: *Fractal Image Compression*. Institut Eurécom, Sophia Antipolis 1999. URL: http://www.eurecom.fr/~image/DEMOS/FRACTAL

[FLSP99] International Business Machines Corporation (o.V.) *Lotus Web Screen Show Player*. Cambridge MA 1999
URL: ftp://ftp.lotus.com/pub/lotusweb/product/freelance/flgwssp.exe

[Frit02] Alexander Fritze: Mozilla SVG Project. The Mozilla Organization, Leicester 2002
URLs:http://www.mozilla.org/projects/svg bzw. http://www.croczilla.com/svg

[Gard02] Joolz Garden: *SVG-PL*. Electronic Publishing Research Group, University of Nottingham (School of Computer Science and Information Technology) Nottingham 2002: http://www.cs.nott.ac.uk/~jxm/SVG

[GBBW02] Glen Gersten, Thomas Barnes, Michael Bierman und Andre Watanabe: *Adobe SVG Draw 0.9*. Adobe Systems Incorporated, Palo Alto 2002. URL: http://www.adobe.com/svg/demos/svgDraw/svgDraw

[Hans02] Mattias Hansson: *Oplayo Media Designer 3*. Oplayo Oy, Helsinki 2002
URL: http://www.oplayo.com/products/Media Designer 3

[HaUl02] Tyson Haverkort, Joost van Ulden (Hrsg.) *VMLSource*. GroundControl GeoTechnologies, Vancouver 2000. URL: http://www.vmlsource.com

[Hayd01] Dave Hayden (Hrsg.) *Ming – an SWF output library and PHP module*. Opaque Industries, Portland 2001
URL: http://ming.sourceforge.net

[HeEz01] Dimitri van Heesch und Jesse Ezell: *SWFSource SDK* (o.O.) 2001: http://sourceforge.net/projects/swfsource

[Herm01] Ivan Herman: *SVGGraphics: SVG generator for Java*. Centrum voor Wiskunde en Informatica (CWI) Information Visualization Project. Amsterdam, 11. November, 2001
URL: http://www.cwi.nl/InfoVisu/SVGGraphics

[Herm02] Ivan Herman: *A Combined Slidemaker*. World Wide Web Consortium Niederlande, Amsterdam 2002
URL: http://www.w3.org/Consortium/Offices/Presentations/xsltSlidemaker

[HSL00] Vincent Hardy, Paul Sandoz und Ana Lindstrom-Tamer: *SVG Slide Toolkit*. Sun Microsystems, San Francisco 2000. URL: http://wwws.sun.com/software/xml/developers/svg-slidetoolkit

[Ice01] IceWEB Communications Incorporated (o.V.) *IceSLIDE*. Herndon VA 2001
URL: http://www.iceweb.com/site/iceslide.cfm

[Iked00] Shinya Ikeda: *MNG-LC Player*. Tokio 2000. URL: http://www.venus.dti.ne.jp/~i-shinya/mngplay

[Impa02] Brian Greenleaf (Hrsg.) Impatica for PowerPoint User Manual. LearningWell / Impatica Incorporated, Ottawa 2002. URL: http://www.impatica.com/imp4ppt

[Intr98] Intrinsyc Software Incorporated (o.V.) "Accessing PowerPoint from Java." Vancouver 1998
URL: http://www.intrinsyc.com/support/j-integra/doc/other_examples/powerpoint_from_java.htm

[Jack03b] Dean Jackson: "SVG Open 2002" (PDF zur Demonstration der SVG-Embedding-Funktionalität im Rahmen Adobe Image Viewer Plug-Ins) Canberra, Januar 2003
URL: http://www.grorg.org/2003/01/SVGOpen_2002.pdf

[JAI99] Sun Microsystems: *The JavaTM Advanced Imaging application programming interface (API)* Palo Alto CA 1999. URL: http://java.sun.com/products/java-media/jai

[Jona02] Piet Jonas: *WMF Tools for Java*. Universität Greifswald (Institut für Physik) Greifswald 2002
URL: http://piet.jonas.com/#wmf

[Juyn00] Gerard Juyn: *LibMNG - The MNG reference library and related information*. Rotterdam 2000
URL: http://www.libmng.com

[Kay01] Michael Kay: *The Saxon XSLT Processor*. Reading UK 2001. URL: http://sourceforge.net/projects/saxon

[Kern84] Brian Kernighan: *PIC – A Graphics Language for Typesetting*. Bell Labs, Murray Hill NJ 1984

[Klei00] Albrecht Kleine: „WMFView und WMFDecoder." *SaxNet*, Dresden 2000. URL:http://www.sax.de/~adlibit

[KuCu02] Richard Kunze und Torsten Curdt: *Spark: A Flash to XML converter*. Tivano Software, Neu-Isenburg 2002
URL: http://www.tivano.de/software/spark

[Main00] D. Nick Main: *JavaSWF2 - A Java Toolkit for Macromedia Flash Parsing and Generation.* Another Big Idea, Fremont / Rancho Cucamonga CA 2000. URL: http://www.anotherbigidea.com/javaswf

[MTH97] Corey Marion, Talos Tsui und Gedeon Maheux: *The Iconfactory*. Greensboro NC 1997
URL: http://www.iconfactory.com

[MWT99] Koichi Mori, Koichi Wada und Kazuo Toraichi: *fds for SVG*. PADC-Labor (Institut für Informationsverarbeitung und Elektrotechnik) Universität Tsukuba 1999
URL: http://padc23.padc.mmpc.is.tsukuba.ac.jp/member/morik/fdssvg

[NoDi00] Jorge Manuel Nogueira Diogo (Hrsg.) *ASP Flash Turbine*. Blue Pacific Software, Lissabon 2000
URL: http://www.blue-pac.com

[Oger02] Ronan Oger: *ROASP*. RO IT Systems, Zürich 2002. URL: http://roasp.com

[Pall00] Luca Palli. *Small Java Library for SVG* (SJLSVG) Lausanne 2000. URL: http://sourceforge.net/projects/sjlsvg

[Pres02] PresentationPro Incorporated (o.V.) *PowerCONVERTER*. Atlanta GA 2002
URL: http://www.presentationpro.com/powerpresenter/powerconverter.asp

[Prob00c] Steve Probets: *Convert Flash into SVG*. Electronic Publishing Research Group, University of Nottingham (School of Computer Science and Information Technology) Nottingham 2000
URL: http://www.ep.cs.nott.ac.uk/~sgp/swf2svg.html

[Rich00] Gareth Richards: *3DInteractive VML Animation.* GER Solutions, London 2000
URL: http://www.gersolutions.com/vml

[RoJa00] Bella Robinson and Dean Jackson. *CSIRO SVG Toolkit*. CSIRO (Commonwealth Scientific & Industrial Research Organisation) Mathematical and Information Sciences, Canberra 2000
URL: http://sis.cmis.csiro.au/svg

[Roya96] Royal Software Incorporated (o.V.) *LiveCard*. Largo, Fla. 1996
URL: http://www.royalsoftware.com/descriptions/livecard.html

[Safe01] Safe Software Incorporated (o.V.) "Vector Markup Language (VML) Format Writer," in: *FME Manual*. Surrey, CA 2001

[Schu02] Manfred Schubert: *PDF Browser Plugin*. Schubert Informationstechnologie, Freiburg im Breisgau 2002
URL: http://www.schubert-it.com/plugin

[Schw00] David Schweinsberg: *SVGFont*. Steady State Software Limited, Aylesbury UK 2000
URL: http://www.steadystate.com/svg

[Sher98] Scott Sherman (Hrsg.) *Simple Vector Format Plug-In*. SoftSource LLC, Bellingham WA 1998
URL: http://www.softsource.com/plugins/svf-plugin.html

[ShSi99] Shlomit Shachor-Ifergan und Sigalit Ur: *Xeena XML Editing Environment*. IBM Alphaworks Haifa Research

Lab, Haifa 1999. URL: http://www.alphaworks.ibm.com/tech/xeena

[Skav00] Dmitry Skavish: *JGenerator*. JZox Incorporated, Framingham MA 2000. URL: http://www.flashgap.com

[SPCO97] Software Publishing Corporation (o.V.) Harvard WebShow. Fairfield NJ 1997
URL: http://www.harvardgraphics.com/updates.asp

[SpTh00] Michael Sperberg-McQueen und Henry Thompson. *XML Schema*. World Wide Web Consortium, April 2000. URL: http://www.w3.org/XML/Schema

[Sue02] Hoylen Sue: *JackSVG*. "Titanium" Project, University of Queensland (Distributed Systems Technology Centre) St. Lucia 2002. URL: http://titanium.dstc.edu.au/xml/jacksvg

[Summ03] Jason Summers: *MNGPLG and MNG4IE: A simple browser plug-in for the MNG image/animation file format*. Fort Wayne IN 2003. URL: http://entropymine.com/jason/mngplg

[ToKe00] John Townley und Jeroen Kessels: "Introducing Form2Flash." *Streaming Media World*, Darien CT, 22. Juni 2000. URL: http://www.streamingmediaworld.com/symm/tutor/form2flash

[Vant02] Christophe Vantighem, David Deckeur, Nicolas Laurent und Vivien Plantinet: *CR2V – Celinea Raster to Vector converter*. Universität Nizza, Hochschule für Informatik (ESSI) Sophia Antipolis 2002
URL: http://www.celinea.com

[Wana02] Wanadu Incorporated (o.V.) *iCreate - Rich Media for the Rest of Us*. Los Altos CA 2002
URL: http://www.wanadu.com/new/site/icreate.html

[Wilk02] Alexis Wilke: *ScriptSWF*. Made to Order Software, London 2002
URL: http://sswf.sourceforge.net

[Wine99] Dave Winer: *Outliners.com - Archives from the golden age of outliners*. UserLand Software, Burlingame CA 1999. URL: http://www.outliners.com

[Xara02] Xara Limited (o.V.) *Xara Graphic Plug-In*. Hempstead UK 2002
URL:http://www.xara.com/downloads/plugin

Abbildungsverzeichnis