Günter Pomaska

Grundkurs Web-Programmierung

- Interaktion, Grafik und Dynamik
- Mit XHTML und CSS, XML, JavaScript, Applets, SVG, PHP

Mit Online-Service zum Buch

vieweg

Günter Pomaska

Grundkurs Web-Programmierung

Aus dem Bereich IT erfolgreich lernen

Pascal
von Doug Cooper und Michael Clancy
Grundkurs Programmieren mit Delphi
von Wolf-Gert Matthäus
Grundkurs Visual Basic
von Sabine Kämper
Visual Basic für technische Anwendungen
von Jürgen Radel
Grundkurs Smalltalk – Objektorientierung von Anfang an
von Johannes Brauer
Grundkurs Software-Entwicklung mit C++
von Dietrich May
Grundkurs JAVA
von Dietmar Abts
Aufbaukurs JAVA
von Dietmar Abts
Grundkurs Java-Technologien
von Erwin Merker
Grundkurs Algorithmen und Datenstrukturen in JAVA
von Andreas Solymosi und Ulrich Grude
Grundlegende Algorithmen
von Volker Heun
Objektorientierte Programmierung in JAVA
von Otto Rauh
Grundkurs Informatik
von Hartmut Ernst
Das PC Wissen für IT-Berufe
von Rainer Egewardt
Rechnerarchitektur
von Paul Herrmann
Grundkurs Relationale Datenbanken
von René Steiner
Grundkurs Datenbankentwurf
von Helmut Jarosch
Datenbank-Engineering
von Alfred Moos
Netze – Protokolle – Spezifikationen
von Alfred Olbrich
Grundkurs Verteilte Systeme
von Günther Bengel
Grundkurs Mobile Kommunikationssysteme
von Martin Sauter
Grundlagen der Rechnerkommunikation
von Bernd Schürmann

Masterkurs Computergrafik und Bildverarbeitung
von Alfred Nischwitz und Peter Haberäcker
Grundkurs MySQL und PHP
von Martin Pollakowski
Grundkurs Mediengestaltung
von David Starmann
Web-Programmierung
von Oral Avcı, Ralph Trittmann und Werner Mellis
Grundkurs UNIX/Linux
von Wilhelm Schaffrath
Das Linux-Tutorial – Ihr Weg zum LPI-Zertifikat
von Helmut Pils
Grundkurs Wirtschaftsinformatik
von Dietmar Abts und Wilhelm Mülder
Grundkurs Theoretische Informatik
von Gottfried Vossen und Kurt-Ulrich Witt
Anwendungsorientierte Wirtschaftsinformatik
von Paul Alpar, Heinz Lothar Grob, Peter Weimann und Robert Winter
Business Intelligence-Grundlagen und praktische Anwendungen
von Hans-Georg Kemper, Walid Mehanna und Garsten Unger
Grundkurs Geschäftsprozess-Management
von Andreas Gadatsch
Grundkurs SAP R/3®
von André Maassen und Markus Schoenen
Controlling mit SAP R/3®
von Gunther Friedl, Christian Hilz und Burkhard Pedell
Kostenträgerrechnung mit SAP R/3®
von Franz Klenger und Ellen Falk-Kalms
Kostenstellenrechnung mit SAP R/3®
von Franz Klenger und Ellen Falk-Kalms
Prozessmodellierung mit ARIS®
von Heinrich Seidlmeier
ITIL kompakt und verständlich
von Alfred Olbrich
Grundkurs Betriebswirtschaftslehre
von Notger Carl, Rudolf Fiedler, William Jórasz und Manfred Kiesel
Masterkurs IT-Controlling
von Andreas Gadatsch und Elmar Mayer
Grundkurs Web-Programmierung
von Günter Pomaska

www.vieweg-it.de

Günter Pomaska

Grundkurs Web-Programmierung

Interaktion, Grafik und Dynamik – Mit XHTML und CSS, XML, JavaScript, Applets, SVG, PHP

vieweg

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Höchste inhaltliche und technische Qualität unserer Produkte ist unser Ziel. Bei der Produktion und Auslieferung unserer Bücher wollen wir die Umwelt schonen: Dieses Buch ist auf säurefreiem und chlorfrei gebleichtem Papier gedruckt. Die Einschweißfolie besteht aus Polyäthylen und damit aus organischen Grundstoffen, die weder bei der Herstellung noch bei der Verbrennung Schadstoffe freisetzen.

1. Auflage Januar 2005

Lektorat: Dr. Reinald Klockenbusch / Andrea Broßler

Der Vieweg Verlag ist ein Unternehmen von Springer Science+Business Media.
www.vieweg-it.de

Konzeption und Layout des Umschlags: Ulrike Weigel, www.CorporateDesignGroup.de
Druck- und buchbinderische Verarbeitung: MercedesDruck, Berlin
Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier.

ISBN-13: 978-3-528-05895-1 e-ISBN-13: 978-3-322-80260-6
DOI: 10.1007/978-3-322-80260-6

Vorwort

Web, Statik und Dynamik

Hypertext Markup Language (HTML) ist die Beschreibungssprache für Internetseiten. Ein Dokument, formuliert in HTML, wird von einem Webserver auf Anfrage an einen Webbrowser, den Client, übermittelt. Der Webbrowser hat die Aufgabe der Umsetzung des Dokumenteninhalts in grafisch ansprechende Form. Durch eingebettete Hyperlinks wird auf beliebig verteilte Dokumente im Web verwiesen. Das Verhalten der Seite ist statisch. Der Bildschirminhalt verändert sich nur nach Anforderung einer neuen Webseite.

Der Anteil statischer Webseiten an den gesamten Internetanwendungen ist nur noch gering und verliert an Bedeutung. Das Internet ist dynamisch. Durch Benutzereingriffe verändert sich der Inhalt angezeigter Webseiten. Hierbei ist zu unterscheiden zwischen clientseitiger Dynamik und serverseitiger Dynamik. Im ersteren Fall erfolgen die Veränderungen durch Prozesse auf dem Clientrechner. Serverseitige Dynamik besteht in der Ausführung von Scripten oder Programmen auf dem Fremdrechner.

Sprachen und Werkzeuge

Sprachen und Werkzeuge zur Erstellung dynamischer Webseiten sind u. a. JavaScript, Java, XML, SVG, PHP, SQL, um nur einige mit ihren Abkürzungen zu benennen. Diese Technologien ermöglichen Internetanwendungen, die für ein breiteres Spektrum einsetzbar sind als nur zur Publikation von HTML-Dokumenten. Technisch-wissenschaftliche Anwendungen können webfähig programmiert werden. Es gibt für alle Anwender nur noch eine einzige, aktuelle Softwareversion auf einem Webserver. Der Zugang wird dem autorisierten Personenkreis von jedem Ort aus und zu jeder Zeit gewährleistet. Mit der Anwendung von Datenbanken stehen dem Benutzer der Webseite umfangreiche Datenmengen bei schnellem Zugriff zur Verfügung.

Mit welchem Wissen muss sich nun ein Anwender ausstatten, um die mit dem Web verfügbaren Ressourcen zu nutzen? Sprachspezifikationen, Referenzen und fertige Programme bietet das Web in hinreichender Fülle. Die Existenz, Verfügbarkeit und das Zu-

sammenspiel der Werkzeuge muss transparent werden. Kenntnisse in Auszeichnungssprachen, Scriptsprachen und objektorientierten Konzepten sind Voraussetzung zum Verständnis bestehender und sich neu entwickelnder Technologien. Das notwendige Basiswissen wird im vorliegenden Buch Studierenden und Praktikern vermittelt.

Programmieren lernt man nur durch Programmieren

Das Buch ist in erster Linie als Arbeitsbuch konzipiert und kann unterrichtsbegleitend oder zum Selbststudium verwendet werden. Durch die Vielzahl der Beispiele werden Programmiersituationen dargestellt, die in Anwendungen immer wieder auftreten. Insofern ist es auch ein Nachschlagwerk für den Praktiker. Es ersetzt aber in keinem Kapitel eine Sprachreferenz. Diese finden Sie im Web aktuell, umfangreich und komplett an autorisierter Stelle. In diesem Programmierpraktikum wird der Lernprozess als zielorientiertes *Learning by Coding* mit Reduzierung auf das Notwendige abgebildet.

Der Leser soll Modifikationen der Aufgabenstellungen selbst vornehmen können, die sich im Internet bietenden Ressourcen für eigene Anwendungen nutzen und mit Fachleuten der IT-Branche dialogfähig werden. Sie erhält mit dieser praxisorientierten Sammlung von Programmbeispielen kompaktes Lehrmaterial, mit dem in die populären Sprachen des Internets eingeführt wird.

Die Arbeit mit dem Buch

Den meisten Nutzen wird Ihnen das Buch bringen, wenn Sie es aktiv durcharbeiten und die Programmbeispiele gleichzeitig an Ihrem Computer ausführen und modifizieren. Es ist nicht notwendig, alle Kapitel des Buches der Reihenfolge nach durchzugehen.

XHTML und CSS ist ein Muss

Unbedingt erforderlich für Arbeiten an Internetprojekten ist ein gewisser aktiver Sprachschatz in *Extensible Hypertext Markup Language* (XHTML) und *Cascading Style Sheets* (CSS). Das Kapitel 2 ist demnach Pflichtlektüre.

Bei *Extensible Markup Language* (XML) ist das schon anders. Aber es ist anzumerken, dass die Grafiksprachen für das Web, SVG und X3D auf dem XML-Standard basieren. Zukünftig führt

wohl bei ernsthaften Web-Anwendungen kein Weg an XML vorbei. Wenn Sie wollen, überspringen Sie Kapitel 3 zunächst.

Objektorientierte Programmierung

Im Jargon der EDV-Leute ausgedrückt, ist das Kapitel über Java *hard stuff*. Grafische Anwendungen in Form von Applets sind ein attraktives Übungsfeld. Sie müssen sich schon intensiv mit Java auseinandersetzen. Geben Sie nicht so schnell auf. Wenn Sie ungeduldig sind, machen Sie doch weiter mit JavaScript. Vielleicht kehren Sie später einmal zum Kapitel 4 zurück und finden Gefallen an objektorientierter Programmierung.

JavaScript bringt Schwung auf die Seite

Mit wenig Aufwand gelangen Sie durch JavaScript zu Erfolgserlebnissen. Fertige Scripts aus dem Web bringen Schwung auch auf Ihre Webseiten. Vorgestellt werden die Applikationsobjekte, das Dom und die Sprachelemente von JavaScript. Eine etwas umfangreichere JavaScript-Anwendung macht ausgiebig von den Methoden des *Math-Objekts* Gebrauch.

Grafik mit SVG

Sofern Sie es mit zweidimensionalen Grafiken zu tun haben, ist *Scalable Vector Graphics* (SVG) das richtige Werkzeug zur Formulierung von Grafikdaten. Durch Einsatz von Interaktion und Animation erhalten Ihre Illustrationen Dynamik.

PHP Anwendungen auf dem Server ausführen

Mit der Installation eines lokalen Webservers und dem Einsatz von PHP (Hypertext Pre-Processor) beschreiten Sie den Weg zur serverseitigen Dynamik. Die Basisstrukturen der Programmiersprachen unterscheiden sich nicht wesentlich voneinander. Wenn Sie bereits programmieren, werden Sie sich schnell für PHP begeistern. Aber PHP ist auch ohne Vorkenntnisse leicht zu erlernen und kann Ihre erste Programmiersprache werden.

Eine Entwicklungsumgebung ist einfach zu installieren. Durch die Vielfältigkeit der Funktionen ist PHP auch eine Alternative zu nur lokal einsetzbaren Programmiersprachen.

Fahren Sie nun Ihren Rechner hoch. Laden Sie Ihren Lieblingseditor, und schon kann es losgehen mit der praktischen Web-Programmierung.

Günter Pomaska, November 2004

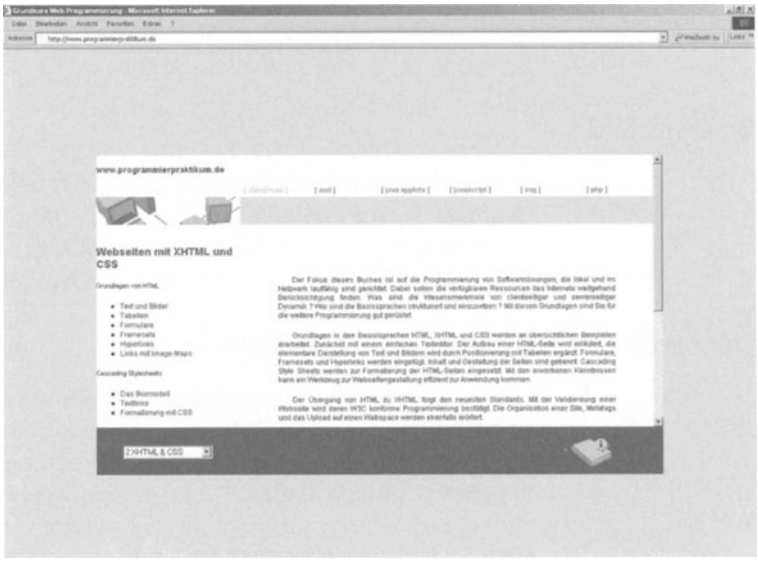
Online-Service zum Buch

Als Online-Service zu diesem Buch wurde die Internetpräsenz

www.programmierpraktikum.de

geschaltet. Alle Programme des Titels sind dort unter einer Benutzeroberfläche lauffähig installiert.

Als Leser des Buches können Sie sich registrieren und erhalten danach u. a. Zugang zum Download des vollständigen Quelltextes aller Programme und profitieren von Updates.



Inhaltsverzeichnis

1 Statische und dynamische Webseiten

Der Fokus dieses Buches ist auf die Programmierung von Softwarelösungen gerichtet, die sowohl lokal als auch im Netzwerk lauffähig sind. Dabei sollen die verfügbaren Ressourcen des Internets weitgehend Berücksichtigung finden. Das sind einerseits die Entwicklungswerkzeuge und andererseits die definierten Standards und gebrauchsfertigen Programmbausteine. Die Nutzung dieser Quellen erfordert Kenntnisse von der Infrastruktur des Internets. In diesem einführenden Kapitel werden die Komponenten des Webs kompakt dargestellt. Datenübermittlung und Adressierung der Rechner und das Zusammenspiel der Hardwarekomponenten sind im vorliegenden Zusammenhang weniger von Bedeutung. Primär ist der Unterschied zwischen statischen Webseiten und dynamischen Webseiten aufzuzeigen. Was sind die Wesensmerkmale von clientseitiger und serverseitiger Dynamik? Mit diesen Grundlagen sind Sie für die Programmierung gut gerüstet.

1.1 Das Internet – Infrastruktur und Software

Netzwerke

Ein Computernetzwerk besteht aus mindestens zwei miteinander verbundenen Computern. Die Nutzer des Netzwerkes können miteinander kommunizieren und gemeinsam die Ressourcen wie Festplatten oder Drucker nutzen. Die beteiligten Rechner können dabei unterschiedliche Plattformen (Hardware, Betriebssystem) verkörpern.

Das Internet ist ein weltweites Netz von Computernetzwerken. Die am Internet beteiligten Netzwerke sind autonom. Ein einzelner Benutzer wählt sich über eine Telefonleitung bei einem Internetprovider oder Online-Dienst ein und erhält hierdurch Zugang zum Netz. Der Einwahlknoten eines Providers wird mit *Point of Presence* (POP) bezeichnet, der wiederum über einen *Network Access Point* (NAP) ein Backbone-Netz anbindet, das die schnelle Verbindung der Netze garantiert.

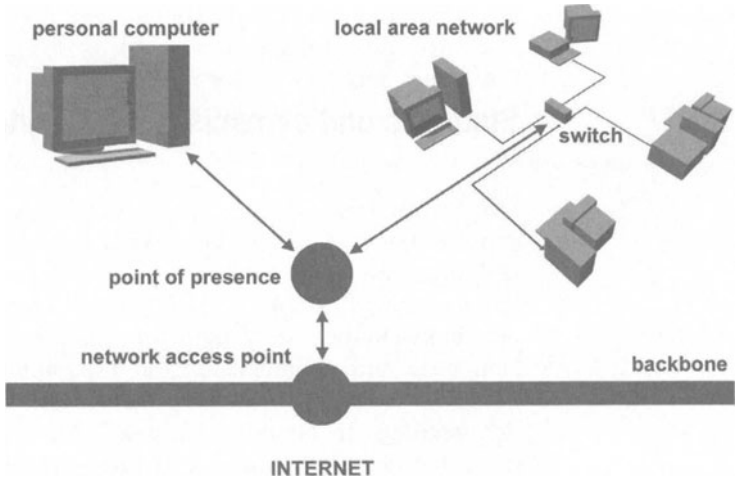


Abb. 1.1: Internet – Netzwerkstruktur

World Wide Web - www

Im Internet werden unterschiedliche Dienste wie Electronic Mail, NetNews, Telnet, File Transfer Protocol u.a. angeboten. Das *World Wide Web* (oder kurz Web) integriert Dienste verschiedener Art unter einer Oberfläche. Kennzeichnend für das Web ist die Fähigkeit, beliebig verteilte Multimediadokumente zu verknüpfen. Dadurch stellt das Web ein umfassendes Informationssystem dar.

Die Internetgemeinschaft ist auf nicht kommerzieller und nicht staatlicher Basis international organisiert. Die Internet Society *www.isoc.org* mit etwa 150 Mitgliedsorganisationen und 6000 Einzelmitgliedern in über 170 Ländern administriert die Infrastruktur.

Das World-Wide-Web-Konsortium *www.w3c.org* ist ein Zusammenschluss internationaler Unternehmen zur Entwicklung und Spezifikation offener Standards mit der Zielsetzung, das Web einheitlich zu gestalten.

Auskunft über den Inhaber einer Internetpräsenz bekommt man beim Network Information Center *www.nic.com* bzw. bei den nationalen Registrierungsstellen. Die ist *www.denic.de* für Deutschland, dort sind derzeit ca. 8 Millionen Domains registriert. Die Registrierung einer Domain erfolgt i. d. R. über den Internetprovider.

Client-Server Architektur

Ein Webserver ist eigentlich ein Programm. Die Bezeichnung *Server* ist aber auch für den Rechner zutreffend, auf dem die Software läuft. Ähnlich ist es mit dem Begriff *Client.* Häufig wird damit der mit dem Server verbundene Rechner bezeichnet, es kann aber aus der Softwaresicht z. B. auch ein Webbrowser sein.

Dokumente im Internet werden in der Auszeichnungssprache *Hypertext Markup Language* (HTML) bereitgestellt. Ein Browser fordert vom Server eine Datei an. Der Server übermittelt die Daten, der Browser stellt die Datei, entsprechend der HTML-Spezifikation, auf dem Client-Rechner dar. Die Kommunikation zwischen Server und Client wird über das *Hypertext Transfer Protocol* (HTTP) vereinbart. Die Komponenten des Web sind also der Internetbrowser, das Protokoll HTTP, die Beschreibungssprache HTML und der Webserver.

Datenübermittlung

Daten werden im Internet paketweise übermittelt. Informationen werden in Pakete aufgeteilt, getrennt versandt und am Zielort wieder zusammengesetzt. Diese Arbeit teilen sich die Protokolle TCP/IP. Während *Transmission Control Protocol* (TCP) die Daten aufteilt und beim Empfänger wieder zusammensetzt, regelt das Internetprotokoll (IP) die Zustellung. Andere Protokolle sind, außer dem oben bereits genannten HTTP, das *File Transfer Protocol* (FTP) oder die Email Protokolle *Post Office Protocol 3* (POP3) und *Simple Mail Transfer Protocol* (SMTP).

Adressen im Internet

Jedem Rechner wird im Internet eine eindeutige Adresse zugeordnet. Die 32-Bit IP-Adresse wird in vier 8-Bit Felder aufgeteilt. Die Felder werden durch Punkte getrennt und als Dezimalwert angegeben. Jede Adresse setzt sich zusammen aus einer Netzwerkkennung und der Rechnernummer im Netzwerk.

Kleinere private Netze sind über einen Router an das Internet angeschlossen. Nicht jeder Rechner hat hier permanent seine eindeutige IP-Adresse. Die IP-Adressen werden dynamisch aus einem reservierten Bereich zugeordnet. Der Router vermittelt dann zwischen der öffentlichen Adresse und der internen Rechneradresse, die nicht weitergeleitet wird.

Beispiele für IP-Adressen sind: 192.168.200.3 oder 127.0.0.1. Adressen, die mit 192.168 beginnen, sind für lokale Netzwerke reserviert. Die ersten drei Zahlen kennzeichnen das Netzwerk, die letzte Zahl steht für die Rechnernummer. Unter Windows können Sie sich mit dem Befehl *ipconfig* die IP-Adresse Ihres Rechners anzeigen lassen.

Geben Sie in die Adresszeile Ihres Browsers *http://217.12.3.11* ein, dann wählen Sie eine statische Adresse an. Zur Vereinfachung der Handhabung und auch aufgrund der dynamischen Adressvergabe geht man bei der Adressangabe den Umweg über den statischen Hostnamen. Dieser wird über das DNS Domain Name System festgelegt. Alle an einen Internet-Host angeschlossenen Benutzer gehören einer Domain an, die ihrerseits wieder an eine Domain angeschlossen ist. Diese hierarchische Verkettung führt von der untersten Ebene zur Top-Level-Domain. Die Domain *programmierpraktikum* gehört zur Top-Level-Domain *.de*. Das Präfix *www* ist nicht Bestandteil des Namens.

URI, URL, URN

Mit dem Adressierungsschema *Uniform Ressource Locator* (URL) werden bestehende Dienste im Internet integriert. Beispiele: *http://www.fh-bielefeld.de* ruft eine Homepage im Web auf, *ftp://ftp-bauwesen.fh-bielefeld.de* bedeutet Anmeldung auf einem FTP-Server.

Eine URL setzt sich zusammen aus dem Protokoll, der Serveradresse (Domain oder IP-Adresse) und der Pfad- und Dateiangabe. In den obigen Beispielen wurde die Dateiinformation nicht angegeben, es gilt die Voreinstellung des Servers (z. B. index.html).

In diesem Zusammenhang sind noch die Begriffe URN und URI zu erwähnen. Während eine URL einen Ort (location) angibt, wird mit einem Uniform Ressource Name (URN) eine Ressource oder ein Schema mit einem frei zu vergebenden Namen identifiziert. Der Oberbegriff für URL und URN ist Uniform Ressource Identifier (URI).

Sprachen - HTML, CSS, XML, XHTML, DHTML

Das Basisformat für Webdokumente ist die Auszeichnungssprache HTML. Anweisungen zur Strukturierung, Positionierung und Formatierung sind in sog. Tags, den Befehlsmarken, enthalten. Der Webbrowser als Client-Software interpretiert die Tags und führt die Befehle entsprechend aus. Dokumente können unmittelbar in HTML formuliert sein oder aber auch erst während der Anfrage auf dem Server generiert werden.

Die Trennung von Inhalt und Erscheinungsbild eines Web-Dokuments wird durch Einsatz von *Cascading Style Sheets* (CSS) erreicht. Bei unterschiedlichen Stilvorlagen kann das Erscheinungsbild einer Webseite wechseln oder dem Ausgabegerät (Monitor, Drucker, PDA) angepasst werden, ohne das ursprüngliche HTML-Dokument zu modifizieren. CSS-Anweisungen kön-

nen in einer externen Datei vorgehalten werden und sind auf alle HTML-Dokumente anwendbar.

Extensible Markup Language (XML) ist eine Metasprache zur Definition von Markup-Sprachen. Diese werden als XML-Anwendungen bezeichnet. Die Syntax, Struktur und Bedeutung der Tags wird für jede XML-Anwendung mit einer *Document Type Definition* (DTD) beschrieben. Das Erscheinungsbild der Tags kann mit CSS definiert werden. Weitere Möglichkeiten bestehen in der Anwendung der *Extensible Style Language* (XSL). Mit XML formulierte Datenbestände sind für die Darstellung mit Webbrowsern, für die Verarbeitung mit Textverarbeitung, Datenbankanwendungen und weitere Applikationen geeignet. Außerdem können XML-Anwendungen als Datenaustauschformate dienen. Mit XML ist die strenge Trennung von Inhalt und Erscheinungsbild realisiert.

Extensible Hypertext Markup Language (XHMTL) ist eine Umformulierung von HTML in eine XML-Anwendung. HTML liegt derzeit in der Version 4.01 vor und soll zukünftig durch XHTML, derzeit gültige Version 1.0, ersetzt werden. Der Umstieg von HTML zu XHTML wird empfohlen und ist problemlos zu vollziehen. Werkzeuge zur Konvertierung bestehender HTML-Dateien nach XHTML sind verfügbar. Es genügt, wenige Regeln bei HTML einzuhalten, die zu einem gültigen XHTML Dokument führen. Wenn im weiteren Text des Buches der Begriff HTML genannt wird, dann ist damit auch die Formulierung von Webdokumenten in XHTML und Anwendung von CSS zu verstehen.

Unter *Dynamic HTML* (DHTML) wurde eine Technologiererweiterung von HTML durch verschiedene Techniken wie JavaScript, Implementierung des *Document Object Model* (DOM), CSS u.a. spezifiziert. DHTML bezeichnet demnach keine eigene Sprache sondern den Einsatz von Techniken zur Erzielung clientseitiger dynamischer Effekte auf Web-Seiten.

1.2 Statische Webseiten

Die Anforderung einer Webseite erfolgt durch Eingabe einer URL in der Adressenzeile eines Webbrowsers. Die IP wird über einen Domain Name Server ermittelt. Der angewählte Web-Server sendet das angeforderte HTML-Dokument an den Client. Dort übernimmt der Browser die Darstellung der Seite. Der Seiteninhalt kann nicht ohne Neuanforderung vom Server verändert werden, daher wird diese Form der Darstellung als statisch bezeichnet.

Ggf. enthält die Seite Hyperlinks, die per Mausklick die Anforderung eines anderen Dokuments oder eine Verzweigung auf eine Stelle im gleichen Web-Dokument bewirken.

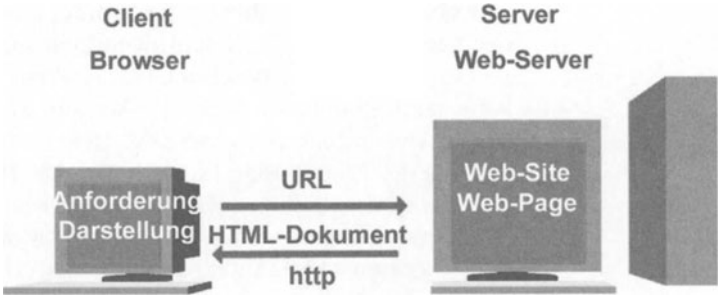


Abb. 1.2: Übermittlung statischer Web-Dokumente

1.3 Clientseitige Dynamik

Mit den Technologien JavaScript, Java und DOM, deren gemeinsamer Einsatz mit DHTML benannt ist, können Web-Seiten clientseitig ohne Neuanforderung verändert werden. Der Zugriff auf alle Elemente eines Dokuments wird über die Implementierung des DOM realisiert.

Scriptsprachen wie JavaScript werten Benutzereingaben in HTML-Formularfelder aus und können darauf basierende Modifikationen des Dokuments vornehmen. Abbildung 1.3 zeigt den Verarbeitungsprozess. Scripte können innerhalb des Dokuments eingebettet sein oder werden als externe Dateien durch Angabe einer URL eingebunden. Beispiele von einfachen Anwendungen mit JavaScript sind Währungsumrechner, Kalender oder Laufschriftanimationen.

Auch durch Java-Applets, das sind kleine Java Programme mit Sicherheitseinschränkungen, kann eine Website clientseitig dynamisch gestaltet werden. Während JavaScript bei der Interaktion weitgehend auf die Bedienungselemente des Browsers zurückgreift, sind Java-Applets nicht browserabhängig und benutzen die Implementierung der Programmiersprache. Der Browser muss lediglich die virtuelle Java-Maschine verfügbar haben. Java-Applets laufen in einem vom Browser reservierten Bereich und sind als compilierte Klassen in externen Dateien vom HTML-Dokument unabhängig gespeichert. Innerhalb einer Webseite

können auch Java-Applets, die sich auf einem anderen Webserver befinden, angefordert werden.

Dynamische Anwendungen von Java-Applets können wesentlich anspruchsvoller sein als die mit JavaScript realisierten Lösungen. Die Möglichkeiten von Java-Applets und JavaScript werden in nachfolgenden Kapiteln ausführlich betrachtet.

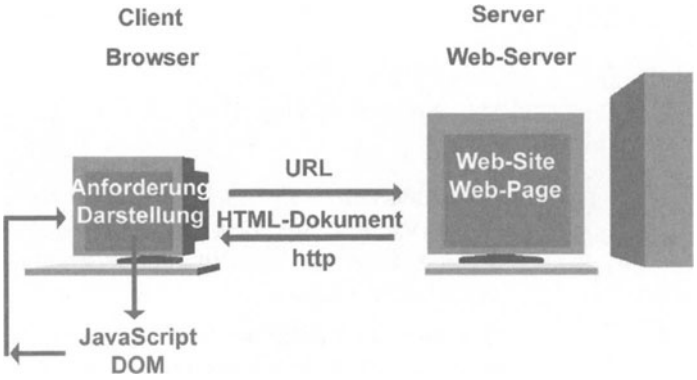


Abb. 1.3: Clientseitige Dynamik

1.4 Serverseitige Dynamik

Serverseitige dynamische Webseiten werden erst aufgrund der Anfrage des Clients auf dem Server erzeugt und dann als HTML-Datei dem Browser übermittelt. Hierbei ist zu unterscheiden zwischen Scriptsprachen, die als Servererweiterung anzusehen sind, und Programmen auf dem Server, die über das *Common Gateway Interface* (CGI) mit dem Server kommunizieren.

Serverseitige Dynamik mit CGI

Die Schnittstelle zwischen Client und Server stellt das HTTP-Protokoll dar. Zusammen mit der Übermittlung der URL erhält der Server Parameter, die an ein CGI-Programm weitergegeben werden. Diese Parameter können direkt mit der URL oder unter Verwendung von HTML-Formularen an den Server gesandt werden. Eine CGI-Programm kann in einer beliebigen Sprache geschrieben sein. Mit CGI werden die Kommunikationsregeln zwischen Server und Applikation beschrieben. Das CGI-Programm kann sehr komplex sein und sich beispielsweise der Dienste eines Datenbankservers bedienen. Ergebnis des CGI-Programms ist eine dynamisch generierte HTML-Seite, die der Webserver aufgrund der Anfrage an den Webbrowser zurück sendet.

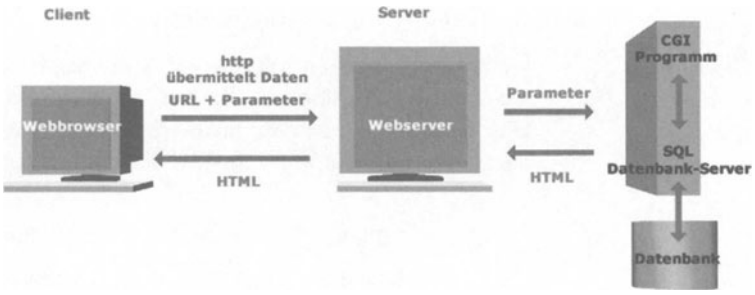


Abb. 1.4: Serverseitige Dynamik mit CGI-Programmen

Serverseitige Dynamik mit PHP

Eine Alternative zu CGI-Programmen stellen Scriptsprachen dar. Scriptsprachen sind Servererweiterungen. Die wohl populärste Scriptsprache ist derzeit PHP. PHP steht als Bezeichnung für *Hypertext Preprozessor.*

Der Browser übermittelt die Anfrage nach einem PHP-Script oder einer HTML-Datei zusammen mit den Parametern. Der Webserver lädt das Script bzw. die HTML-Datei mit eingebettetem PHP und stellt die übermittelten Parameter als Umgebungsvariable dem Script zur Verfügung. Das Script generiert die HTML-Ausgabe, die vom Webserver an den Client zurückgesandt wird. PHP ist eine mächtige Scriptsprache mit Funktionalitäten bzw. Klassen zum SQL-Datenbankzugriff und zur Verarbeitung von XML-Formaten.

Gegenüber clientseitigen Scriptsprachen ist der Zugriff auf das Filesystem des Servers nicht eingeschränkt. Es müssen lediglich die entsprechenden Zugriffsrechte eingerichtet werden. Eine Einführung in PHP erfolgt in Kapitel 7. Bei der Diskussion der Beispiele wird auch auf das Upload und die Einrichtung der Zugriffsrechte am Beispiel des FTP-Clients Filezilla eingegangen.

Zum Zeitpunkt der Erstellung dieses Buches erschien im Juli 2004 die Version PHP 5, die aber noch nicht unmittelbar von allen Providern unterstützt wurde. Im Rahmen der hier angeführten Programmbeispiele sind ausschließlich aufwärtskompatible Befehle der Version 4.3.8 eingesetzt. Die Unterstützung von PHP 4 ist bei den Providern weitgehend gewährleistet.

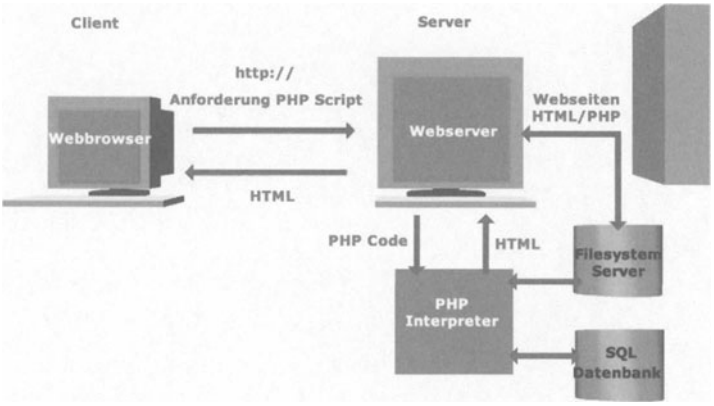


Abb. 1.5: Serverseitige Dynamik mit der Scriptsprache PHP

Zur Ausführung von PHP-Scripten ist eine Client-Server Architektur notwendig. Diese Architektur kann durch Einrichten eines Webservers auch auf einem lokalen Rechnersystem geschaffen werden. Eine Internetverbindung muss nicht bestehen. Daher kann PHP auch für lokale Anwendungen von Interesse sein.

Wenn Sie die PHP-Seiten auf einen Webspace hochladen, sollten Sie sich vergewissern, dass Ihr Provider auch diese PHP-Version unterstützt und die Programme mit entsprechenden Rechten ausgestattet sind. Das sind die Berechtigung zur Ausführung durch den Anwender und der Lese- und Schreibzugriff auf Dateien bzw. Verzeichnisse, die sich auf dem Server befinden.

2 Webseiten – Struktur und Publikation

Dieses Buch ist keine Anleitung zum Design von Webseiten hinsichtlich der grafischen Ausformung. Der Fokus ist auf die Architektur gerichtet. Darunter ist zu verstehen: die Trennung von Inhalt und Gestaltung, die Strukturierung der Inhalte, die Verknüpfung der Seiten und die Definition von Formatvorlagen. Darüber hinaus soll das Basiswissen zur Erstellung dynamischer Anwendungsprogramme für das Internet erarbeitet werden.

In diesem Kapitel liegt das Augenmerk auf der Strukturierung des Inhalts und den technischen Möglichkeiten zur Gestaltung von Internetseiten. Außerdem werden Aspekte der Publikation einer Website aufgegriffen.

Wenn Sie erstmalig Kontakt mit einer Programmiersprache haben, ist es zunächst empfehlenswert, die Grundlagen zu erarbeiten. Erst wenn man die Zusammenhänge kennt, sollte man sich komplexerer Entwicklungswerkzeuge bedienen. Internetprovider bieten häufig die Möglichkeit der Online-Generierung einer Hompage. Sie tippen Ihren Namen ein, etwas Text, klicken ein paar Bilder, suchen Farben und Zeichensätze für die Schrift aus und schon ist die Homepage fertig.

Derartige Seiten können sehr professionell aussehen. Auch viele HTML-Editoren bieten Gleichartiges. Der Anwender stellt nur noch seine Daten zusammen und wählt ein Layout. Noch weiter geht es bei der Nutzung von *Content Management Systemen* (CMS). Man setzt sich an einen Internetarbeitsplatz, loggt sich ein und publiziert einen Artikel. Alle Webseiten, die zusammen die Website bilden, haben immer das gleiche Layout. Doch auch diese hochkomplexen Systeme bieten die Möglichkeit der unmittelbaren Einbindung von HTML-Tags. Spätestens hier wird dann deutlich, wie sinnvoll Basiswissen sein kann. Wer auf soliden Grundkenntnissen aufbaut, ist in der Lage, komplexe Systeme schnell zu erlernen und anzuwenden und kann notwendige Modifikationen im Quelltext selbst vornehmen.

2.1 HTML Grundlagen

Werkzeuge

Alle nachfolgende Beispiele sind mit Notepad programmiert. Der Umfang der angesprochenen Tags ist auf die notwendige Untermenge reduziert. Nach Durcharbeitung dieses Kapitels werden Sie sich in der ausgezeichneten deutschsprachigen HTML-Referenz *selfhtml.org* bestens zurechtfinden. Wenn Sie dort nicht ausreichend informiert werden, hilft *www.w3c.org* sicher weiter. Sie merken schon, das vorliegende Buch stellt keine umfassenden Sprachreferenzen bereit, es bietet Anleitungen zur Anwendung der Web-Standards.

Gute Webeditoren gibt es viele. Visual Tools wie etwa Dreamweaver oder GoLive. Andere bekannte Werkzeuge sind Hotmetal oder der Namo Webeditor. An dieser Stelle können nur einige genannt werden. Probieren Sie vielleicht PhaseV aus, eine deutschsprachige Freeware, die unter der Portaladresse *www.qbaut.de* zugänglich ist. Sie benötigen noch einen FTP-Client zum Upload Ihrer Dateien auf einen Webserver. Auf dieses Werkzeug kommen wir im Verlauf des Kapitels zurück.

Aufbau einer HTML-Seite, Textdarstellung

Alle HTML-Auszeichnungen, die sog. Tags, befinden sich zwischen spitzen Klammern. Jeder Tag hat einen Start-Tag und einen Ende-Tag. Zwischen Groß- und Kleinschreibung wird bei HTML nicht unterschieden. Wir wählen hier die Kleinschreibung zwecks Kompatibilität zu XHTML. Ein HTML-Dokument besteht aus:

- HTML Versions-Information
- Abschnitt `head`
- Inhalt im Abschnitt `body`

Die Informationen im Abschnitt `head` werden nicht vom Browser dargestellt. Im `body` eines HTML-Dokuments befinden sich Tags zur Strukturierung des Inhalts, Angaben zur Positionierung, die Inhalte selbst bzw. Referenzen auf die Inhalte, Hyperlinks, Kommentare, eingebettete Scriptsprachen und Stilvorlagen.

Auf Darstellung der ursprünglichen HTML-Formatanweisungen wie z. B. `FONT` verzichten wir hier völlig, da die Gestaltung unserer Seiten ausschließlich mit Stilvorlagen erfolgen wird.

Alle HTML-Auszeichnungen müssen sich zwischen `<html>`

`</html>` befinden. Innerhalb von `<head></head>` ist ein Title-Tag vorgeschrieben, dessen Inhalt in der Titelzeile des Betriebssys-

temfensters erscheint, <title>*Text der Titelzeile*</title>. Einigen der sog. Metag-Tags wenden wir uns weiter unten zu.

Zwischen <body></body> befinden sich die Anweisungen für den Browser. Ein Tag kann Attribute enthalten, den Attributen wird über den Zuweisungsoperator = ein Wert zugewiesen. Mehrere Attribute sind durch Leerzeichen zu trennen. Ein Beispiel:

```
<body bgcolor = "#000000" text="#ffffff"></body>
```

würde den im Body-Tag eingeschlossenen Text mit weißen Buchstaben auf schwarzem Hintergrund darstellen. Aber wir wollten ja keine Gestaltungsanweisungen hier einbringen, es gibt im Moment leider noch kein besseres Beispiel.

Ein Kommentar in HTML befindet sich zwischen den Auszeichnungen

```
<!--
  hier ein Kommentar für den Benutzer des Quelltextes
 -->
```

und kann sich über mehrere Zeilen erstrecken.

Innerhalb des Body-Tags können die Elemente mit den Tags und gruppiert werden. Der Unterschied zwischen beiden Tags liegt in der Formatierung. HTML kennt Block-Level-Elemente und Inline-Elemente. Erstere werden immer in einer neuen Zeile dargestellt, letztere nicht. Hierin liegt auch der Unterschied zwischen span und div. Span definiert Inhalte mit der Eigenschaft inline, div behandelt Inhalte als Block-Level-Elemente. Bei der Anwendung von div gibt es demnach immer einige Leerzeilen mehr.

Überschriften, Absätze

Text kann durch Überschriften und Abschnitte gestaltet werden. Überschriften werden auch als Headings <h1></h1> bezeichnet. Für die Wertigkeit der Überschrift benutzt man die Zahlen von 1 bis 6. Der Text selber wird in Abschnitte (Paragraphen) eingeteilt. Es gibt weitere Tags zur Einteilung von Texten, die hier aber nicht näher behandelt werden.

Zur grafischen Unterteilung kann mit horizontalen Linien gearbeitet werden. Der Tag wird mit <hr> ausgezeichnet und hat in HTML keinen Ende-Tag. Ein Zeilenumbruch wird mit dem Tag <br> erzwungen.

Wir können jetzt die erste HTML-Datei erstellen. Dazu besorgen wir etwas Blindtext aus dem Internet oder von der eigenen Festplatte und fügen den Text wie unten dargestellt in den Tag <p>

ein. Die Datei wird gespeichert unter `test.html` und im Explorer aufgerufen.

```
<html>
  <! -- Strukturierung von Text -->
  <head>
     <title>HTML-Grundlagen</title>
  </head>
  <body>
    <div>
     <h1>2.1 HTML-Grundlage</h1><hr>
     <h2>Aufbau einer HTML-Seite</h2>
     <h3>Texdarstellung, Listen, Bilder</h3>
     <p>Ambigitur quotiens, uter utro sit
        prior, aufert Pacuvius doct
           ...
        famam senis Accius alti, dicitur
        iurgia finis.</p><hr>
     </div></body></html>
```

Listing 2.1: Strukturierung von Text mit HTML

Beachten Sie bitte, dass der Textumbruch durch den Browser realisiert wird. Je nach Bildschirmauflösung und Fenstergröße wird die Datei mit automatischem Umbruch dargestellt. Der Web-Autor kann ja nicht wissen, mit welcher Fenstergröße der Nutzer die Datei betrachten wird. Bei Bedarf werden vom Browser die Scrollbars eingeblendet.

Listen, Aufzählungen

HTML stellt für Aufzählungslisten `<ul>`, nummerierte Listen `<ol>` und Definitionslisten `<dl>` die benötigten Tags bereit. Bei Aufzählungslisten und nummerierten Listen werden die Listenelemente in den Tag `<li>` eingebunden. Die Tags können mit Gestaltungs- und Positionierungsattributen versehen werden. Wir teilen unser obiges Beispiel in zwei Textabschnitte und fügen eine Liste ein:

```
... quaedam nimis </p>
<ul>
   <li>antique</li>
   <li>peraque dure</li>
   <li>dicere credit eos</li>
 </ul>
 <p>ignave multa ...
```

Einfügen von Bildern

Webbrowser stellen Grafiken in den Formaten `GIF`, `JPG` und `PNG` dar. GIF-Formate zeigen maximal 256 Farben, können Animationen und Transparenz enthalten und werden bei Logos und grafischen Darstellungen benutzt. JPG-Formate stellen eine Palette bis zu 16,7 Millionen Farben dar und sind für Halbtonbilder (Photos) geeignet. Das Format ermöglicht Datenkomprimierung, so dass auch sehr große Bilder in kompakten Dateien gespeichert werden. Das Format PNG ist für das Web am geeignetsten, hat sich derzeit aber in der allgemeinen Anwendung noch nicht durchgesetzt.

Bilder werden mit dem Image-Tag `<img>` in einer Webseite platziert. Der Image-Tag hat in HTML kein Ende-Tag. Die Attribute haben folgende Bedeutung:

- src Bildquelle
- width Breite für die Darstellung des Bildes
- height Höhe für die Darstellung des Bildes
- alt Text für einen Tooltip
- border Bildrahmen
- hspace horizontaler Abstand zum Text
- vspace vertikaler Abstand zum Text
- align Positionierung zum umlaufenden Text

Dateireferenzen sollten grundsätzlich mit kleinen Buchstaben und ohne Sonderzeichen angegeben werden. Dabei ist immer eine relative Pfadangabe zu berücksichtigen. Der Pfad wird von der aktuellen Position des HTML-Dokuments aus verfolgt. Die Größe der Bilddatei ist für die Größe der Einfügung nicht maßgebend, das Bild wird mit den Werten der Attribute `width` und `height` eingefügt. Zur Vermeidung von Bildumrechnungen sollten bei Standardanwendungen die Angaben mit der Dateigröße selbst identisch sein.

Der Wert des Attributs `alt` enthält den Text für einen Tooltip, der bei Mausberührung angezeigt wird. Auf eine kurze Bilderläuterung sollte man nicht verzichten. Die Abstände des umlaufenden Textes zum Bild werden über die Werte der Attribute `hspace` und `vspace` festgelegt. `Align` gibt die Position des Bildes zum umlaufenden Text an. Die Werte können `left`, `right`, `top`, `middle`, `center` oder `bottom` sein.

Das einführende Beispiel wird nunmehr mit zwei Bilddateien ergänzt. Das Bild der Baumaschine ist ein JPG-Format, der Bauarbeiter liegt im Format animiertes GIF (gefunden im Web) vor. Die Bilder sind in einem eigenen Unterverzeichnis *gfk* abgelegt.

```
<img src="gfk/baustelle.jpg"
     alt="Baustelle"
     border="0" width=128" height="96"
     align="middle" >
<p>... esse reor, memini
<img src="gfk/baustelle.gif" alt="Arbeiter"
      border="0" hspace="6" vspace="6"
      align="left" width=96" height="66">
quae plagosum mihi ...</p>
```

Das erste Img-Tag befindet sich außerhalb des Textparagraphen, `align="middle"` ist hier ohne Wirkung, da das Bild nicht von Text umflossen wird. Die Positionierung des GIF-Bildes wird innerhalb von `<p></p>` im Text vorgenommen. Hiermit kann man die Formatierung durch den Browser dokumentieren. Versuchen Sie `align` mit den Werten `top`, `middle` und `bottom`. Letztlich ist alles nicht wie gewünscht. Bessere Lösungen erzielt man mit der Anordnung in Tabellen, die zur Strukturierung und Positionierung eingesetzt werden.

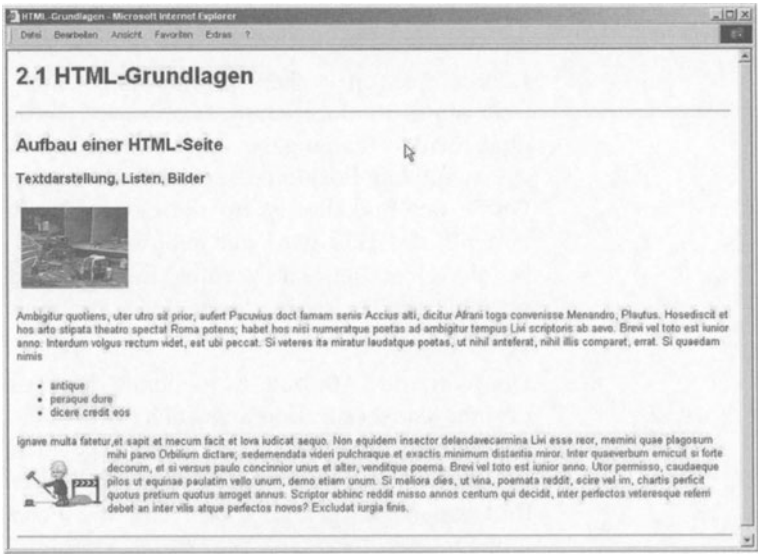


Abb. 2.1: Grundlagen von HTML, Screenshot von *test.html*

Pixelgenaue Positionierung mit Tabellen

Tabellen sind ein Beschreibungselement innerhalb von HTML mit denen die Daten in Zellen positioniert werden. Aufgrund der beliebigen Größe, die Tabellenzellen annehmen können, und der Möglichkeit, Ränder unsichtbar darzustellen, sind Tabellen nicht nur zur Aufnahme von Daten brauchbar, sondern stellen ein universelles Gestaltungsraster dar. Eine Tabelle wird durch den Tag `<table></table>` begrenzt. Bestandteile der Tabelle sind die Tabellenzeilen `<tr></tr>` und Tabellenzellen `<td></td>`.

Tabellen werden immer, von links oben beginnend, zeilenweise mit der Definition von Tabellenzellen aufgefüllt. Unterschieden wird zwischen regelmäßigen und unregelmäßigen Tabellen. In unregelmäßigen Tabellen können sich Datenzellen über mehrere Spalten, Attribut `colspan`, oder über mehrere Zeilen, Attribut `rowspan`, ausdehnen. Die Attributwerte definieren die Anzahl der belegten Spalten bzw. Zeilen.

Die wichtigsten Angaben zur Positionierung einer Tabelle sind die Breite der Tabelle, Attribut `width`, der Abstand zwischen den Zellen, Attribut `cellspacing`, die Polsterung, das ist der Abstand der Zelleninhalte zum Rand, Attribut `cellpadding`. Mit dem Attribut `border` wird die Anzeige der Ränder gesteuert.

Unser obiges Testbeispiel bauen wir jetzt zu einer mehrspaltigen Tabelle um. Die erste Zeile soll die Überschrift enthalten, ab der zweiten Zeile beginnen die Spalten. In der ersten Spalte befinden sich die Bilder, die zweite Spalte soll einen senkrechten Trennstrich zum Text aufnehmen, in der dritten Spalte befindet sich der Fliesstext. Die gesamte Tabelle wird in vier Zeilen aufgeteilt: Bei anfänglichen Versuchen empfiehlt es sich, den Wert für `border` auf 1 zu setzen, dadurch wird die Tabellenstruktur markiert. Wir wählen einen Abstand von 0 Pixel zwischen den Datenzellen, geben aber eine Polsterung von 2 Pixel für den Abstand zum Tabellenrand an.

```
<table width="640" cellpadding="2" cellspacing="0" border="1">
  <tr>
    <td colspan="3" align="center">
       <h1>2.1 HTML-Grundlagen</h1><hr>
       <h2>Aufbau einer HTML-Seite</h2>
       <h3>Textdarstellung, Listen,
           Bilder</h3></td></tr>
```

Die erste Tabellenzeile erstreckt sich über drei Spalten, daher ist nur eine Tabellenzelle `<td></td>` formuliert. Die Attribute sind

`colspan` und `align`, der Wert `center` positioniert die Inhalte, hier die Überschrift, mittig.

Die darauffolgende Zeile enthält in der ersten Spalte das Bild *baustelle.jpg*. Die Breite der Zelle ist nicht angegeben, diese wird automatisch vom Browser durch die Bildgröße bestimmt. Das Zeichen ` ` ist ein Leerzeichen, das nicht vom Browser entfernt werden kann. Die Zelle ist für drei Zeilen beschrieben. In den beiden Folgezeilen ist diese Spalte dann bereits belegt und muss nicht mehr definiert werden. Die noch verbleibenden Pixel, der insgesamt 640 Pixel breiten Tabelle, werden für den Text in Spalte 3 genutzt.

```
<tr>
  <td>
  <img src="gfk/baustelle.jpg"
       alt="Baustelle" border="0"
       width=128" height="96"></td>
 <td rowspan="3"> </td>
  <td>
    <p>Ambigitur ... Siquaedam nimi </p>
  </td></tr>
```

Nunmehr kann die dritte Zeile formuliert werden. Spalte 1 bekommt als Inhalt ein Leerzeichen, Spalte 2 ist bereits belegt, s.o. In Spalte 3 wird eine unsortierte Liste eingefügt.

```
<tr>
 <td> </td>
 <td>
   <ul>
      <li>antique  </li>
      <li>peraque dure</li>
      <li>dicere credit eos </li>
   </ul></td></tr>
```

Verbleibt noch die letzte Zeile. In Spalte 1 wird das GIF-Bild positioniert, Spalte 2 ist bereits belegt und Spalte 3 ist wieder für den Text vorgesehen.

```
<tr>
  <td>
    <img src="gfk/baustelle.gif"
         alt="Arbeiter" border="0"
         width=96" height="66"></td>
  <td>
    <p>ve ... iurgia finis.</p></td>
</tr></table>
```

Allgemein wurden keine Angaben für die Höhe der Tabellenzellen vorgenommen. Der Browser nimmt sich den Bereich, den er zur Darstellung des Inhalts benötigt. Ist dieser nicht im Fenster verfügbar, erscheinen die Scrollbars am rechten bzw. unteren Rand des Fensters.

Zur Ausgabe des senkrechten Strichs in Spalte 2 wenden wir einen kleinen Trick an. Mit einem Bildbearbeitungsprogramm wird ein Bild der Größe 1 x 1 Pixel mit der gewünschten Hintergrundfarbe erstellt. Das Bild wird vom Browser immer in der im Image-Tag angegeben Größe ausgegeben. Mit der Breite 1 können wir die Tabellenhöhe jetzt durch Angabe der Höhe des Bildes steuern.

```
<img src="gfk/ein_pixel_black.gif" alt="" width=1"height="360">
```

Verbleibt nur noch die Positionierung der Tabelle selbst. Diese ist in den Div-Tag eingebunden. Das Attribut `align` bekommt den Wert `center`, damit wird die Tabelle mittig in das Browserfenster gesetzt:

```
<div align="center"><table></table></div>
```

Nach Fertigstellung ist dann noch der Attributwert von `border` wieder auf den Wert null zurückzusetzen.

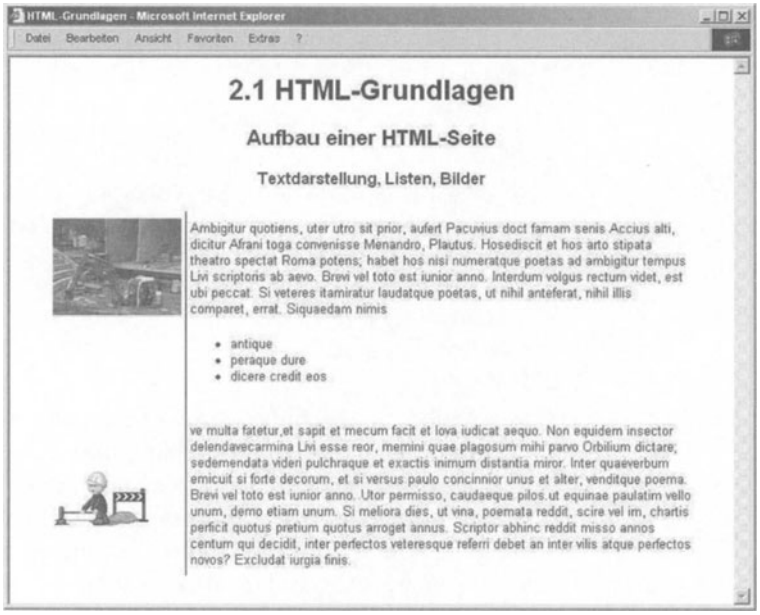


Abb. 2.2: Positionierung mit Tabellen

Formulare

Interaktive Benutzereingaben auf Webseiten werden über Formulare vorgenommen. Neben den Bedienelementen wie Checkboxen, Radiobuttons, Eingabefeldern u.a. stellen Formulare auch die Kommunikationsschnittstelle zum Webserver her.

Formulardaten können clientseitig mit JavaScript ausgewertet werden. Serverseitig werden die übermittelten Daten mittels dort laufender CGI–Programme ausgewertet oder als Umgebungsvariablen den Scripten verfügbar gemacht. Formulardaten können auch per Email weitergeleitet werden. Zur Auswertung der dann ziemlich kryptisch angekommenen Daten gibt es hilfreiche kleine Programme.

Ein Formular wird mit dem Tag `<form>` eingeleitet. Die Attribute des Tags legen die nach der Eingabebestätigung auszuführende Aktion fest. Innerhalb der Formulare gibt es die Befehle `<input>`, `<texteingabe>` und `<select>`. Die Positionierung der Formularelemente kann wieder geeignet mit Tabellen erfolgen.

Wir wollen ein kleines Kontaktformular aufbauen. Der Besucher einer Webseite erhält die Möglichkeit, eine Anfrage per Email an den Betreiber der Webseite zu stellen. Zunächst betrachten wir das sehr einfache Grundgerüst:

```
<html>
<head>
    <title>Formulareingaben</title></head>
<body>
  <form>
    <input name="absender" size="40" maxlength="60"
           value="eMail:">
    <textarea name="anfrage" rows="10" cols="50"
           wrap="virtual">Ihre Anfrage:</textarea>
    <input type=reset value="Reset">
    <input type=submit value="Senden">
  </form>
</body>
</html>
```

Innerhalb des Abschnitts `<body>` befindet sich das Form-Tag. Benutzt werden die Befehle `<input>` und `<textarea>`. Es wird ein Texteingabefeld mit dem Bezeichner `absender` für das Attribut `name` eingegeben. Die Größe des angezeigten Feldes beträgt 40 Zeichen, die maximale Anzahl der einzugebenden Zeichen ist auf 60 begrenzt. Das Eingabefeld ist mit dem Text eMail vorbesetzt. Durch die Auszeichnung `<textarea>` wird ein Texteingabebereich von 10 Zeilen und 50 Zeichen aufgebaut. Der automa-

tische Textumbruch wird mit `wrap="virtual"` ausgeführt. Das Textfeld ist mit dem Kommentar `Ihre Anfrage:` vorbesetzt.

Es folgen noch zwei Buttons. Mit Buttons vom Typ `reset` werden alle Eingaben auf die Anfangswerte zurück gesetzt, bei Klick auf den Button vom Typ `submit` werden die Formulardaten mit der im Form-Tag angegeben Aktion an den Server übermittelt. Das Formular ist zwar schon benutzbar, aber die Behandlung der Daten ist noch nicht angegeben. Zur besseren Übersicht müssen die Elemente auch noch positioniert werden. Nach der im vorhergehenden Abschnitt entwickelten Tabellenstruktur wird in das Formular eine zweispaltige Tabelle mit 640 Pixel Breite eingefügt. Innerhalb der Tabellenzellen finden die Dialogtexte und die Formularelemente ihren zugewiesenen Platz. Der Browser stellt das Listing mit bereits ausgefüllten Eingabedaten wie in der Abbildung 2.3 dar.

```
<html>
<head><title>Formulareingaben</title></head>
<body>
<div align="center">
<form>
<table width="640" cellpadding ="3" border="1">
<tr>
   <td>Absender eMail:</td>
   <td><input name="absender" size="40" maxlength="60">
   </td></tr>
<tr>
   <td>Ihre Anfrage:</td>
   <td><textarea name="anfrage" rows="10"
        cols="40" wrap="virtual">
       </textarea></td></tr>
<tr>
   <td><input type=reset value="Reset"></td>
   <td><input type=submit value="Senden"></td>
   </tr></table></form></div></body></html>
```

Listing 2.2: Formulare in HTML

Wie soll nun nach Absenden mit dem ausgefüllten Formular verfahren werden ? Die Auszeichnung `<form>` benutzt hierfür das Attribut `action`. Der Wert von `action` ist üblicherweise eine URL, die eine Webseite mit Anweisungen zur Auswertung der übermittelten Daten angibt.

Abb. 2.3: Formulareingaben in HTML

Die Art der Datenübermittlung wird im Attribut `method` durch `post` oder `get` festgelegt. Die Get-Methode hängt die Parameter sichtbar an die gesendete URL an und begrenzt die Datenmenge. Vorzuziehen ist die Post-Methode, die Daten unabhängig von der URL übermittelt und größere Informationsinhalte überträgt.

Zwei weitere Parameter von `form` dienen den Auswerteprogrammen zur Identifikation der Formularelemente, die Werte für `name` und `id` erhalten vom Programmierer frei zu vergebende Bezeichner. An dieser Stelle wird nur auf die Existenz der Attribute hingewiesen, in späteren Beispielen wird auch deren Auswertung behandelt.

In diesem Beispiel werden die Daten per Email übergeben. Es muss daher noch die Festlegung der Inhaltscodierung und eine Angabe über den verwendeten Zeichensatz folgen. Das Form-Tag bekommt hiermit abschließend folgende Notation:

```
<form
  action="mailto:info@programmierpraktikum.de" method="post"
  enctype="text/plain" name="anfrage" id ="anfrage">
```

Nach Absenden wird sich der auf Ihrem Rechner installierte E-mail-Client einschalten und die Datenübermittlung ausführen. Testen Sie das Programm mit Ihrer eigenen Email-Adresse. Sie erhalten die im Formular benutzten Bezeichner mit den eingegeben Werten zugesandt.

Weitere Bedienelemente und Möglichkeiten der Auswertung von Formularen werden wir bei den Anwendungen im Abschnitt JavaScript und PHP kennen lernen. An dieser Stelle wurde Ihnen

das Prinzip vorgestellt. Mit nur wenigen HTML-Befehlen konnten Sie ein brauchbares Kontaktformular erstellen. Anspruchsvollere Anwendungen sind an die weitere Vertiefung der Thematik gebunden.

Framesets

Frames unterteilen das Browserfenster in mehrere unabhängige rechteckige Bereiche. In jeden Frame wird eine separate Webseite geladen. Die Steuerdatei wird mit Frameset bezeichnet. Zwar steht Benutzung von Framesets in der Diskussion, es gibt aber sehr wohl geeignete Anwendungen bei der Seitengestaltung und Organisation, die für den Einsatz von Framesets sprechen. Beispielsweise kann die Navigation in einem eigenen Frame laufen und so vom eigentlichen Seiteninhalt getrennt werden.

Die Unterteilung innerhalb eines Framesets erfolgt immer zeilen- oder spaltenweise. Eine Kombination erfordert die Verschachtelung von Framesets. Jeder Frame bekommt einen Namen zur eindeutigen Identifikation. Hyperlinks werden mit dem Attribut `target` ausgezeichnet. Der Wert des Attributs gibt den Zielbereich für einen Hyperlink an. Beispiele zu Hyperlinks finden Sie im nächsten Abschnitt.

Einzelne Frames innerhalb eines Framesets können verschiedene Auszeichnungen zum Verhalten des Fensters und Positionierungsangaben zum Inhalt enthalten. Wir betrachten exemplarisch den Aufbau der Website: *www.programmierpraktikum.de.*

Die Eingangsseite wird mit *index.html* bezeichnet und definiert ausschließlich den Frameset.

Der Bildschirm soll zunächst horizontal in drei Bereiche aufgeteilt werden. Der mittlere Bereich bekommt eine feste Höhe von 540 Pixel. Das wird mit der Anweisung `rows (*,540,*)` erreicht. Wir beschränken uns hier auf die Einheiten Pixel und das Zeichen *, welches die relative Größe in Abhängigkeit von den weiteren Angaben beschreibt.

`Framespacing` und `frameborder` sind analog zu den Festlegungen einer Tabelle zu betrachten. Auf die Anweisung `frameset` folgen die einzelnen Frames. Der Abstand des Inhalts zum Frame wird mit den Attributen `margin` angegeben, `noresize` verhindert eine Veränderung der Fenstergröße, `scrolling` kann die Werte `yes`, `no`, oder `auto` annehmen und definiert das Vorhandensein der Scrollbars am Rande des Fensters. `Name` ist der Bezeichner für den Frame selbst. Das Attribut `src` steht für Source (die Quelle) und gibt die URL des einzufügenden Dokuments an.

```
<html> <head>
  <title>www.Programmierpraktikum.de </title></head>
<frameset rows="*,540,*" framespacing="0" frameborder="0" >
   <frame src="space_grau.html"
          name="space_top" marginwidth="0"
          marginheight="0" scrolling="no" noresize="noresize">
   <frame src="space_rot.html"
          name="space_center"
          marginwidth="0" marginheight="0" scrolling="no"
          noresize="noresize">
    <frame src="space_grau.html"
          name="space_bottom" marginwidth="0"
          marginheight="0" scrolling="no" noresize="noresize">
   </frameset>
 <noframes>
  <body>
     <p>Grundkurs Web-Programmierung, eine Vieweg-Publikation.
        Es werden keine Frames angezeigt</p>
  </body></noframes></html>
```

Wir benötigen noch die Dateien *space_grau.html* und *space_rot.html*. Diese Dateien definieren lediglich nur eine Hintergrundfarbe. Hier in hexadezimaler Darstellung für ein helles Grau #dddddd oder für Rot #e6001b.

```
<html>
  <head><title>space_grau</title></head>
  <body bgcolor = "#dddddd"></body>
</html>
```

Der Browser zeigt bei Aufruf der Datei *index.html* ein graues Fenster mit einem 540 Pixel breiten roten Mittelbereich. Die Frames tragen die Namen *space_top, space_center* und *space_bottom*. Probieren Sie es aus, bevor wir den mittleren Bereich in drei Spalten einteilen.

Wir ersetzen jetzt den mittleren Frame *space_center* durch einen dreispaltigen Frameset, der mittlere Frame bekommt eine Breite von 960 Pixel. Die Bezeichnungen für die neuen Frames werden `space_left`, `space_center` und `space_right`.

```
<frameset cols="*,960,*" framespacing="0" frameborder="0" >
  <frame src="space_grau.html"
        name="space_left" marginwidth="0"
        marginheight="0" scrolling="no" noresize="noresize">
  <frame src="space_rot.html"
        name="space_center" marginwidth="0"
```

```
 marginheight="0" scrolling="no" noresize="noresize">
  <frame src="space_grau.html"
         name="space_right" marginwidth="0"
         marginheight="0" scrolling="no" noresize="noresize">
</frameset>
```

Bei Seitenaufruf rendert der Browser nunmehr ein zentriertes rotes Rechteck der Größe 960 x 540 Pixel. Der rote Bereich wird nochmals aufgeteilt in einen unteren Navigationsbereich von 80 Pixel Höhe, der darüber liegende Bereich ist mit 460 Pixel Höhe ausgelegt. Beachten Sie bitte, dass nur zwei Werte im Attribut `rows` stehen. Die Namen der Frames lauten *menue* und *inhalt*. Das Attribut `scrollbars` des Frames *inhalt* erhält hier jetzt den Wert `yes`, später in der fertigen Anwendung dann `auto`. Es werden noch die HTML-Seiten *startseite.html* und *menue.html* angelegt. Die Seiten erhalten exemplarisch nur einen farbigen Hintergrund und einen kennzeichnenden Text.

```
<frameset rows="*,80" framespacing="0" frameborder="0" >
   <frame src="startseite.html" name="inhalt"
          marginwidth="0" marginheight="0" scrolling="yes"
          noresize="noresize">
   <frame src="menue.html" name="menue"
          marginwidth="0" marginheight="0" scrolling="no"
           noresize="noresize"></frameset>
```

Abbildung 2.4 zeigt das Ergebnis mit den grauen Frames `space_top`, `space_left`, `space_bottom` und `space_right`. Der dunkle (rote) Frame ist bezeichnet mit *menue*, der weiße Frame mit dem Namen *inhalt* wird i.d.R. der Target-Frame für die mittels Hyperlinks wechselnden Inhalte.

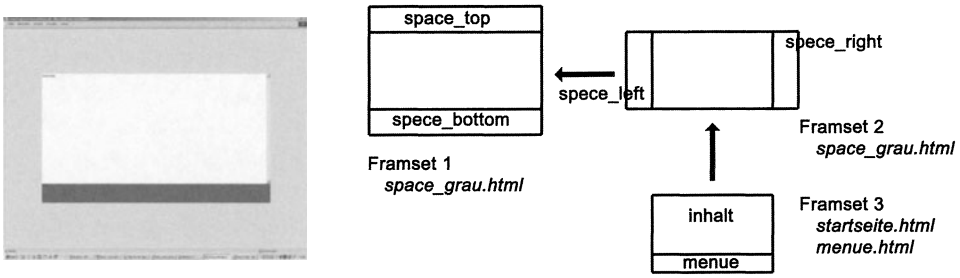


Abb. 2.4: Seiteneinteilung mit mehrfach geschachtelten Frames

Abschließend die HTML-Codierung noch einmal im Zusammenhang.

```
<html><head>
<title> Programmierpraktikum</title></head>
<!-- Frameset 1 -->
<frameset rows="*,540,*" framespacing="0" frameborder="0">
    <frame src="space_grau.html"
           name="space_top" marginwidth="0"
           marginheight="0" scrolling="no" noresize="noresize">
  <!-- Frameset 2 -->
  <frameset cols="*,960,*" framespacing="0" frameborder="0" >
    <frame src="space_grau.html"
           name="space_left" marginwidth="0"
           marginheight="0" scrolling="no" noresize="noresize">
  <!-- Frameset 3 -->
    <frameset rows="*,80" framespacing="0" frameborder="0" >
    <frame src="startseite.html"
           name="inhalt" marginwidth="0"
           marginheight="0" scrolling="no" noresize="noresize">
    <frame src="menue.html" name="menue"
           marginwidth="0"
           marginheight="0" scrolling="no" noresize="noresize">
    </frameset>
  <frame src="space_grau.html"
           name="space_right" marginwidth="0"
           marginheight="0" scrolling="no"
           noresize="noresize">
  </frameset>
  <frame src="space_grau.html"
           name="space_bottom" marginwidth="0"
           marginheight="0" scrolling="no" noresize="noresize">
</frameset>
 <noframes>
    <body>
       <p> ... Text ...</p></body></noframes>
</html>
```

Listing 2.3: Anlegen von Frames

Hyperlinks

Hyperlinks oder kurz Links sind Verweise in einem HTML-Dokument, mit denen bei Aktivierung durch Mausklick

- eine andere Position im Dokument
- eine neue Datei
- oder ein neuer Webserver

aufgerufen wird. Weiter gibt es neben diesen Verweisen auch die Möglichkeit eines Email-Links oder Dateiverweises. Dateiverweise bewirken den Download oder die Ausführung eines externen Programms.

Schaltflächen für Hyperlinks sind Texte, Grafiken, Bereiche in Grafiken (image maps) oder Buttons, die als Schaltflächen benutzt werden. Ein Hyperlink wird mit dem Tag `<a></a>` ausgezeichnet. Die Elemente innerhalb des Tags stellen die Schaltfläche dar. Standardmäßig wird ein Text, der als Link ausgezeichnet ist, in blauer Farbe und unterstrichen dargestellt. Die angeforderte URL wird bei Mausberührung in der Statuszeile des Browsers angezeigt.

Dem Attribut `href` wird die aufzurufende Referenz zugewiesen. Die Anzeige kann im eigenen Fenster, im eigenen oder einem anderen Frame oder in einem neuen Fenster erfolgen. Wertzuweisungen an `target` können sein:

- _blank neues Fenster
- _self eigenes Fenster
- _parent sprengt bei verschachtelten Framesets das aktuelle Frameset
- _top sprengt bei verschachtelten Framesets alle Framesets

oder ein beliebiger Name, der für einen Frame gilt, oder ein neues Fenster öffnet.

Als Beispiel legen wir eine Linkliste zu häufig benötigten Seiten und ein Email-Verzeichnis an:

```
<html>
<head><title>Linkliste</title></head>
<body >
  <h3>Web-Links</h3>
  <p>
    <a href="http://www.google.de" target="_blank">
            Google-Suchmaschine<br></a>
    <a href="http://w3c.org" target="_blank">W3C<br></a>
    <a href="http://de.selfhtml.org" target="_blank">
            SelfHTML-Referenz<br></a></p>
  <h3>eMails</h3>
  <p>
<a href="mailto:info@programmierpraktikum.de
   ?cc=gp@imagefact.de&subject=Anfrage
```

```
   Programmierpraktikum &body=Ihre Anfrage
   bitte:" target="_blank">an Webmaster<br>
</a></p></body></html>
```

Listing 2.4: Hyperlinks

Die Referenz in der Mailliste unterscheidet sich von den URLs durch Ersatz von `http` mit dem Begriff `mailto`. Der String zur Referenz kann weitere Angaben enthalten, die der benutzte Mailclient verarbeitet.

Soll innerhalb eines Links nicht eine externe URL referenziert werden, sondern eine Position im eigenen Dokument, dann muss diese zunächst durch einen Anker (anchor) gekennzeichnet sein. An der Position im HTML-Dokument wird ein Bookmark wie folgt notiert:

```
<a name="kap_1"><a>
```

Der Link auf diesen Bookmark würde dann wie folgt formuliert:

```
<a href ="#kap_1" target="_self">Kapitel 1</a>
```

Durch Angabe einer URL vor dem #-Zeichen kann auch auf ein Bookmark in einem Fremddokument verwiesen werden.

Wird statt des Textes ein Image-Tag zwischen `<a></a>` eingefügt, dann stellt die Grafik die Schaltfläche dar. Es muss aber nicht immer die gesamte Bildfläche als Link definiert sein. Eine Grafik kann mehrere Bereiche enthalten, die auf unterschiedliche Referenzen hinweisen. Man definiert für das Bild eine Map. Dies erfolgt entweder manuell mit Spezifikation der Bereiche (auch mit polygonaler oder kreisförmiger Begrenzung) oder besser durch Unterstützung eines HTML-Editors. Die Einheiten in der Bereichsdefinition sind Pixelkoordinaten. Nachfolgend ein kleines selbsterklärendes Beispiel. Das Browserfenster wird durch Angabe eines Framesets in die Bereiche links und rechts aufgeteilt. Die Image-Map ist statisch im linken Bereich angeordnet, gelinkt wird immer in den rechten Bereich. Die Datei *image_maps.html* enthält nur den Frameset:

```
<html>
  <head><title>Links in Image -Maps</title></head>
  <frameset cols="208,*" framespacing="0" frameborder="0">
     <frame src="personen.html" name="links"
        marginwidth="4"
        marginheight="0" scrolling="no" noresize="noresize">
     <frame src="mapping.html" name="rechts"
        marginwidth="2"
```

```
        marginheight="2"
        scrolling="no" noresize="noresize">
  </frameset>
  <noframes>
     <body>
         <p>... keine Framesets </p>
     </body>
  </noframes>
</html>
```

Nachfolgend sehen Sie den Quellcode der Datei *personen.html*. Der Image-Tag ist mit dem Attribut `usemap` versehen und kennzeichnet dieses Bild als Image-Map mit der Bezeichnung `map1`. Der Tag `<map>` wird mit `name = "map1"` dem Bild mit der URL *gfk/personen.jpg* zugeordnet. Es sind drei Rechtecke mit der jeweiligen Referenz auf die HTML-Dateien und den Namen des Zielframes definiert. Dieser Quellcode wird auch in der HTML-Datei *mapping.html* gezeigt. Besuchen Sie die Website *www.programmierpraktikum.de* und verifizieren Sie den Quellcode. Dort wird auch der Tag `<pre></pre>` benutzt, dieser Tag verhindert die Formatierung durch den Browser und dient der Anzeige von Quelltexten. Die Sonderzeichen `<` `>` innerhalb des Tags sind durch `&lt;` bzw `&gt;` ersetzt.

```
<html>
<head><title>Image Maps</title></head>
<body bgcolor="#dddddd">
<h3>Links in Image-Maps</h3>
<img src="gfk/personen.jpg" alt="Klicken Sie auf eine Person"
     width="200 height="150" border="0" usemap="#map1">
  <map name="map1">
    <area shape="rect" coords="130,10,176,128"
          href="person_rechts.html" target="rechts">
    <area shape="rect" coords="80,15,123,129"
          href="person_mitte.html" target="rechts">
    <area shape="rect" coords="28,15,76,129"
        href="person_links.html" target="rechts">
  </map>
</body></html>
```

Listing 2.5: Hyperlinks in Image-Maps

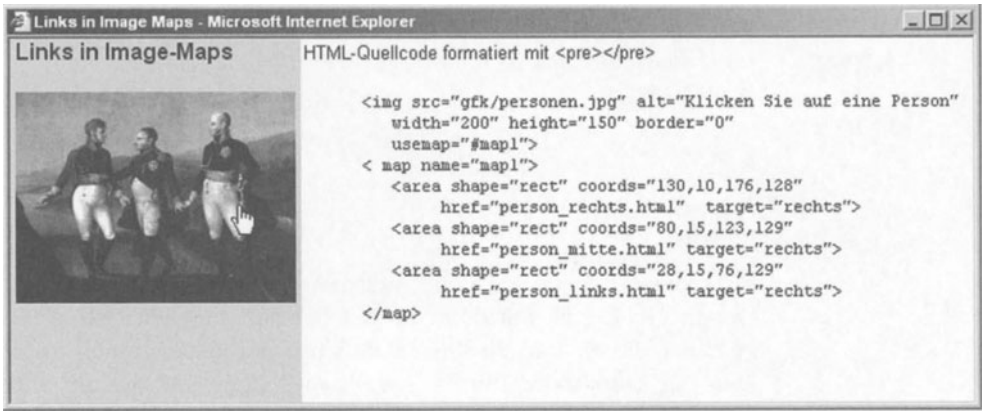


Abb. 2.5: Linkflächen in Image-Maps und Codedarstellung im Browser

2.2 Cascading Style Sheets CSS

Mit der Trennung von Struktur und Inhalt gegenüber dem Erscheinungsbild einer Webseite verbinden sich etliche Vorteile bei der Verwaltung und Programmierung der Web-Präsenz. Die Formatierung wird redundanzfrei an nur einer Stelle definiert. Ein einheitlicher Stil für alle Seiten ist gewährleistet. Änderungen können zentral innerhalb einer Datei vorgenommen werden. Verschiedene Ausgabegeräte oder Bildschirmauflösungen können durch Verknüpfung mit den entsprechenden Stilvorlagen bedient werden.

Die Syntax der Formatsprache für das Web ist durch die Sprache *Cascading Style Sheets* (CSS) definiert. CSS Level 1 wurde um CSS Level 2 mit Befehlen zur Positionierung und Benutzung unterschiedlicher Ausgabemedien erweitert. Derzeit ist bereits CSS Level 3 in der Entwurfsphase. Nicht immer konnten die Browser der Entwicklung folgen. Unterschiedliche Wiedergaben sind schon mal anzutreffen. In den Beispielen dieses Buches wird CSS Level 1 benutzt, das eine Untermenge von Level 2 darstellt. Die Positionierung der Elemente wird in diesem Buch mittels Tabellengestaltung ausgeführt.

Struktur und Syntax von CSS-Formatierungen

Stylesheet-Dateien können mit einem einfachen Texteditor verarbeitet werden. Heutige HTML-Werkzeuge bieten fast alle Unterstützung bei der Bearbeitung von Stilvorlagen. Spezielle Stylesheet-Werkzeuge liefern den entsprechenden Komfort. Im Anschluss an die Fertigstellung von Inhalt und Struktur einer Seite

erstellen Sie die Stildefinition mit Unterstützung eines derartigen Werkzeuges. Ein bekanntes und komfortables Tool ist *Top Style: http://www.bradsoft.com/topstyle.*

Hilfreich kann auch die Freeware Style Assistant sein, verfügbar unter *http://www.styleassistant.de.* Sie brauchen also in keinem Fall die einzelnen Attribute und Werte parat zu haben. Erwerben Sie sich hier den notwendigen Umfang an Kenntnissen über die Struktur und Regeln der Sprache. Zur Erstellung der Stilvorlagen setzen Sie das Werkzeug ihrer Wahl ein

Bevor wir in die Thematik einsteigen, zuvor noch eine kurze Anmerkung zur ansprechenden grafischen Gestaltung eines Web-Dokuments. Versuchen Sie sich nicht mit Gewalt in Mediendesign. Beschränken Sie sich auf einen Zeichensatz. Benutzen Sie zwei Farben, eine helle Pastellfarbe für den Hintergrund und eine dazu passende dunkle Farbe für den Text. Für Hervorhebungen können Sie noch eine Kontrastfarbe wählen. Die Farben sollen der Thematik der Site entsprechen. Hilfestellung zur Farbwahl finden Sie im Netz u.a. bei *www.webmart.de* oder *www.visibone.com.*

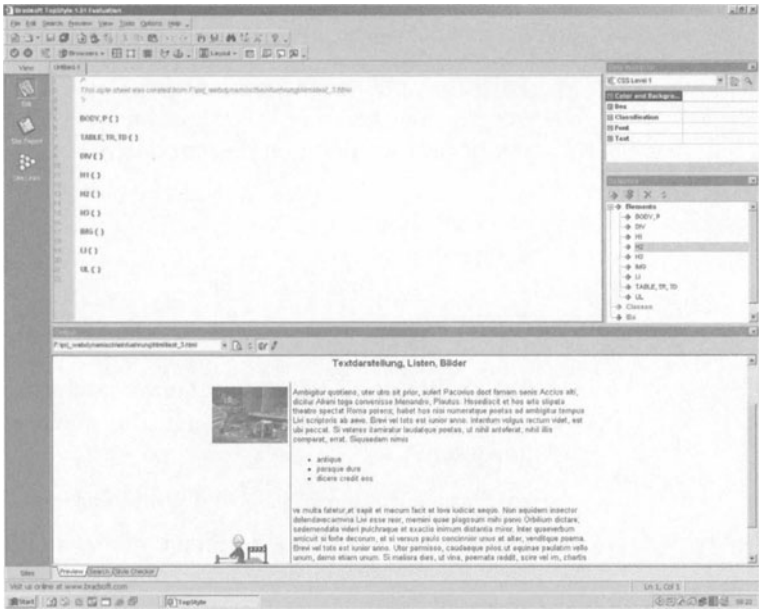


Abb. 2.6: CSS-Editor TopStyle in der Anwendung

Abbildung 2.6 zeigt den Editor Topstyle in der Anwendung. Die Stilvorlagen wurden aus der HTML-Testdatei generiert. Im unteren Fenster die Vorschau, oben links die Selektoren, noch ohne Attribute und Werte. In den Fenstern rechts können Sie das CSS Level und den Browser wählen. Darunter finden sich die Gruppierungen und die Elemente selbst, für die Sie die Stilvorlage erstellen wollen.

CSS Definitionen beginnen mit einem Selektor, dieser entspricht dem HTML-Element. In geschweiften Klammern folgen dann eine oder mehrere Deklarationen. Deklarationen sind die Attribute und deren Eigenschaften, voneinander getrennt durch Doppelpunkt. Mehrere Deklarationen werden durch Semikolon getrennt. Im unteren Fenster der Abbildung 2.7 ist der Code für den Selektor h1 erkennbar.

Zusammenfassung von Regeln

Regeln, die gleiche Deklarationen enthalten können durch Angabe mehrerer Selektoren zusammengefasst werden:

```
h3,h4,h5 {
          font-style :  italic;
          font-weight : normal;
}
```

Klassen als Selektor

Sollen sich Elementtypen in der Darstellung unterscheiden, so vergibt man Klassen. Die Bezeichner der Klassen folgen dem Selektor und werden von diesem durch einen Punkt getrennt.

```
p {
    font-size : 12px }
p.fussnote {
             font-size : 8px; }
p.anmerkung {
             font-style : italic; }
```

Innerhalb der HTML-Auszeichnung bekommt das Element durch das zugeordnete Attribut `class` die gewünschte Gestaltung zugewiesen.

```
<p class ="fussnote">.. eine Anmerkung der Klasse Fussnote</p>
```

Klassen, die sich auf kein Element beziehen und damit für beliebige Elementtypen gelten, bekommen lediglich den Punkt vorangestellt:

```
.allgemein { color: #000000 }
```

Dem HTML-Code steht dann für beliebige Elemente das Attribut `class` mit dem Wert *allgemein* zur Verfügung.

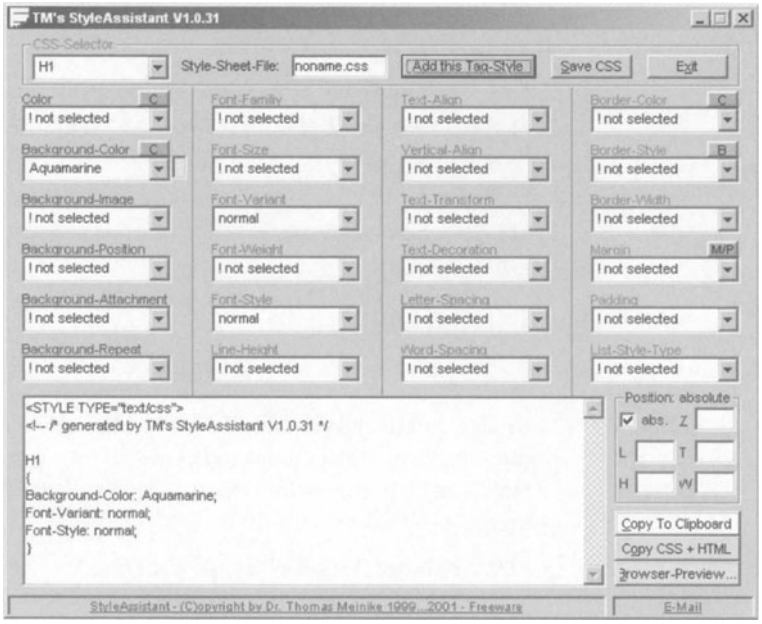


Abb. 2.7: Style Assistant, ausgewählt ist der Selector H1 mit drei Deklarationen

Vererbung der Stilvorlagen an Unterelemente

HTML-Elemente unterliegen einer Hierarchie. Elemente, die von anderen umschlossen werden, sind untergeordnete Elemente. Im unten aufgeführten Beispiel ist `<em>` Kind von `<p>` und `<p>` wiederum Kind von `<body>`. Die CSS-Eigenschaften würden von `<body>` an die Unterelemente vererbt werden. Diese Vererbung kann aber durch Überschreiben der Regeln aufgehoben werden. Werden die Regeln von `<p>` überschrieben, dann erbt `<em>` den Stil von `<p>`.

```
<body>Text im Body
 <p>Absatztext <em> Hervorhebung </em>
    weiter im Absatztext
 </p>
</body>
```

Einbindung von CSS in HTML

Stilvorlagen können als Attribut direkt im HTML-Code Elementen zugeordnet werden:

```
<h1  style="font-style:italic;
```

```
    text-align:center;">
    Überschrift kursiv und zentriert
</h1>
```

Das widerspricht aber der Philosophie von Stilvorlagen. Für Testzwecke kann man im Head-Bereich mit dem Style-Tag die Regeln dokumentenweit einbinden. Das Attribut type wird verlangt.

```
<style type="text/css">
  h1 {
    font-style:italic;
    text-align:center;
  }
</style>
```

In der Praxis wird man die Stilvorlagen in einer eigenen Datei mit der Erweiterungsbezeichnung .css vorhalten und mit dem Link-Tag, ebenfalls im Head-Bereich einer Seite, in die Dokumente einbinden.

```
<link rel="stylesheet" type="text/css"
     href="../css/stilvorlage.css" >
```

Die angegebene URL gilt exemplarisch für eine relative Adressierung. Vom aktuellen Verzeichnis geht es eine Hierarchieebene zurück, von dort wird das Verzeichnis css aufgerufen, in dem sich alle Stilvorlagen befinden, hier mit Referenzierung von *../css/stilvorlage.css*.

Warum Cascading ?

Mehrere Stilvorlagen können zu einem gemeinsamen Stylesheet zusammengefasst werden. Man könnte auf Bausteine zurückgreifen und hier individuelle Änderungen an ausgesuchten Regeln vornehmen. Kaskadierende Stylesheets werden durch den Import-Befehl erzeugt. Hierdurch werden Stilvorlagen nacheinander importiert. Bei gleichen Regeln überschreiben die zuletzt importierten Regeln die zuvor gültigen. Das gilt dokumentenübergreifend, aber auch bei Auszeichnungen innerhalb einer HTML-Datei dokumentenweit.

```
<style type="text/css" >
@import "http://www.programmierpraktikum.de/
            d/xhtml/html/css/stilvorlage.css";
@import "meinestilvorlage.css";
</style>
```

2.3 Seitengestaltung mit Stilvorlagen

CSS Level 1 gruppiert die Formatattribute in:

- Farbe und Hintergrund,
- Boxmodell,
- Klassifikation, Font und Text.

Wir folgen dieser Gruppierung, können im Rahmen dieser Publikation aber nur die notwendigsten Attribute und deren Eigenschaften ansprechen.

Farbe und Hintergrund / Color and Background

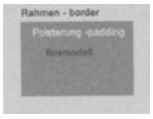
Farbe und Hintergrundfarbe von Elementen wird mit den Attributen `color` bzw. `background` festgelegt. Die Eigenschaften können durch die festgelegten Namen oder Werte für die RGB-Komponenten in dezimaler oder hexadezimaler Schreibweise angegeben werden. Dunkle Schrift auf leicht getöntem Hintergrund wird erreicht durch:

```
body, p {
          background : #eeeeee;
          color : rgb (64,64,64);
        }
```

Diese Ton-in-Ton-Darstellung kann als Empfehlung gelten, da einerseits der Hintergrund nicht zu hell ist und die Schrift nicht im Vollton dargestellt ist, wird ein zu starker Kontrast vermieden.

Das Boxmodell

HTML Auszeichnungen, die einen Block oder Absatz bilden, werden in eine Box eingebettet. Diese Box wird umgeben von der Polsterung (padding), dem Rahmen (border) und dem Rand (margin).



Das folgende Beispiel verwendet als Element `<h1>` und zeigt exemplarisch die Benutzung der Blockattribute, die Attribute können für die Begrenzungen `top`, `left`, `right` und `bottom` jeweils einzeln definiert werden. Unter Polsterung ist der Abstand des Elements zum umgebenden Rahmen definiert. Mit Rand wird der Abstand zum Nachbarelement festgelegt. Beachten Sie auch die Zusammenfügung von benachbarten Rändern. Bei vertikaler und horizontaler Anordnung zweier Elemente wird unterschiedlich vorgegangen.

```
<html>
<head>
<title>blockmodell</title>
```

```
<style type="text/css">
  h1 {  background : Gray;
        border     : 48px solid Aqua;
        padding    : 64px 64px 96px 64px;
        display    : block;
        margin     : 48;}
</style></head><body>
<h1>Blockmodell</h1></body></html>
```

Listing 2.6: HTML-Datei zum Boxmodell

Klassifizierungen

Elemente können unterschiedliche Charakteristika aufweisen. Das zugehörige Attribut wird mit `display` bezeichnet, die Eigenschaften können sein: `block`, `inline`, `list-item` oder `none`. Ein Block-Element erzeugt einen eigenen Absatz mit Zeilenumbruch vor dem Element und am Ende des Elements. Bei einem Inline-Element unterbleibt dieser Umbruch. Ein derartiges Element ist innerhalb vorhandener Absätze einsetzbar. Ein List-Element hat zusätzlich zum Zeilenumbruch die Eigenschaft eines vorangestellten Aufzählungszeichens, mit weiteren Attributen für `list-style` und `list-position` kann eine Aufzählungsliste grafisch gestaltet werden. Bei einem mit `none` klassifizierten Element unterbleibt die Darstellung, es wird nicht angezeigt. Letztlich kann das Attribut `white-space` noch das Verhalten für den Zeilenumbruch regulieren. Die Anwendungen dieses Buches benutzen die HTML-Voreinstellungen der Elemente.

Fonts – Schriftstil

Der Schriftstil wird über die Font-Attribute festgelegt. Bei der Schriftart können mehrere durch Kommata getrennte Angaben gemacht werden. Aus dieser Liste wird die erste der auf dem Rechner vorhandenen Schriftarten in der HTML-Darstellung verwendet. Mit `font-style` wird die Neigung der Buchstaben gewählt. Kapitälchen werden mit dem Attribut `font-variant` erreicht. Die Schriftgröße wird über `font-size` bestimmt. Hierbei können relative oder absolute Werte angegeben werden. Einheiten der absoluten Werte sind u.a. Pixel px oder Millimeter mm. Die Schriftstärke wird über `font-weight` gesetzt.

Textformatierung mit CSS

Das Textlayout regelt die Anordnung und Gestaltung einzelner Buchstaben. Es existieren Attribute für den Buchstabenabstand, den Wortabstand, die Textauszeichnung, die vertikale Positionierung, die Transformation von Buchstaben, die horizontale Textausrichtung, den Texteinzug und den Zeilenabstand. Nachfolgend ein komplettes Beispiel für die Textgestaltung, wie es in

den Beispielen des Buches Anwendung findet. Alternativen hierzu sind im vollständigen Testbeispiel auf der Website angegeben.

```
body,p {
 font-family : Arial, "Microsoft Sans Serif";
 font-size : 12px; font-style : normal;
 font-variant : normal; font-weight : normal;
 letter-spacing : normal; line-height:normal;
 text-align : justify; ext-decoration : none;
 text-indent : 24px;  text-transform : none;
 vertical-align : baseline;
 word-spacing : normal; color :#444444;
 background: #eeeeee;}
h1 {
 font-weight : bold;  font-variant : normal;
 font-style : normal; font-size : 18px;
 text-align : left;   display : block;}
h2, h3 {
 font-weight : bold; font-variant : small-caps;
 font-style : normal; font-size : 14px;
 text-align : left; display : list-item;
}
```

Listing 2.7: Style Sheet Datei *standard.css*, auszugsweise

Formatierung von Hyperlinks

In Verbindung mit dem Anchor-Tag können Pseudo-Elemente definiert werden. Selektoren für Pseudo-Elemente gestalten die Tags abhängig vom Zustand. Der normale Zustand ist `a:link` für noch nicht besuchte Seiten, `a:visited` kennzeichnet bereits besuchte Seiten, `a:hover` ist der Zustand eines Links bei Mausberührung, `a:active` ist der gerade besuchte Link und `a:focus` gilt für Links, die den Fokus erhalten.

Hier ein Beispiel wie man mit Text-Links und CSS eine den Karteikarten ähnliche Gestaltung erzielt. Die Attributfestlegungen `a:link` und `a:hover` bewirken hier die Farbumkehr.

```
<html><head><title>CSS-Links</title>
<style type="text/css">
  body {background:#dddddd;margin-top:48px;}
  td.link {background:white;
           text-align:center;}
  td{background:blue;}
  a:link{
    padding : 4px; color:blue; background:white;
    text-decoration : none;}
  a:visited{
```

```
    padding : 4px; color:blue; background:white;
    text-decoration : none;}
  a:hover{
    padding : 4px; color:white; background:blue;}
</style></head>
<body >
<div align="center">
<table width="480" cellspacing="0" cellpadding ="0">
<tr>
  <td class="link" witdh = "120" >
    <a href="dummy.html"> Link 1 </a></td>
  <td class="link" witdh = "120" >
    <a href="dummy.html"> Link 2 </a></td>
  <td class="link" witdh = "120" >
    <a href="dummy.html"> Link 3 </a></td>
  <td class="link" witdh = "120" >
    <a href="dummy.html"> Link 4</a></td></tr>
<tr>
   <td colspan="4" idth="480"height="24"></td>
</tr></table></div></body></html>
```

Listing 2.8: Textlinks mit CSS

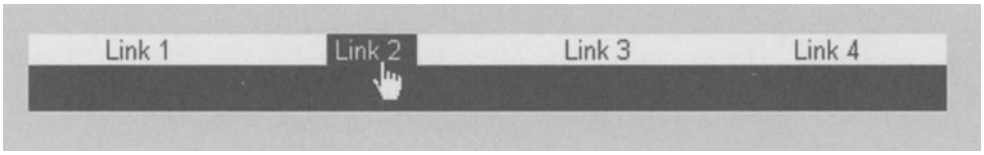


Abb. 2.8: Effiziente Menügestaltung mit einfachen Textlinks

Betrachten wir abschließend noch einen Auszug aus der Datei *tabak.css*. Bei Einbindung dieser Formatvorlage wird beim Status onMouseover eine Umkehr der Farben zwischen Hintergrund und Vordergrund erreicht. Beachten Sie bitte die Vererbung. Alle weiteren Eigenschaften erhalten die Links von den Elementen, in denen sie eingebettet sind. Die Reihenfolge der Zustände ist einzuhalten. Ein weiteres Pseudo-Element wurde mit `p:first-letter` eingeführt. Dieses Element gilt für die Formatierung des ersten Buchstabens eines Textabschnitts. Beim Test mit unterschiedlichen Browsern sind kleinere Unterschiede festzustellen. Beachten Sie auch die Hintergrundgestaltung. Abbildung 2.9 zeigt die Darstellung im MS-Internetexplorer. Alternative Stylesheets können Sie in der Kopfzeile auswählen.

```
a:link { color :  #3399ff;}
a:visited { color : #3399ff; text-decoration : line-through;}
```

```
a:hover { background : #3399ff; color : #cc9966;}
a:active{ background : #3399ff; color :  #cc9966;}
p:first-letter{    font : bold 18px;
       float : left;
       vertical-align : bottom;}
body {background: transparent
       url(../gfk/wasserzeichen.gif)
       no-repeat
       fixed center;
       margin-top: 24px;
       background-color : White;}
```

Abb. 2.9: Die HTML-Testseite mit CSS gestaltet

2.4 Von HTML zu XHTML

Die aktuelle HTML-Version 4.01 hat als Nachfolger *Extensible Hypertext Markup Language* (XHTML 1.0), eine XML-basierte Umformulierung der Auszeichnungssprache HTML. Alle HTML-Elemente sind jetzt in der Syntax von XML verfügbar, damit ist XHTML eine XML-Anwendung. XHTML als strukturorientierte Auszeichnungssprache weist nur wenige Unterschiede zu HTML auf. Bestehende HTML-fähige Browser können XHTML darstellen.

Mit XHTML 1.0 wurde die strenge Trennung von Struktur und Layout einer Seite vollzogen. Die Fähigkeiten einer Programmiersprache hat auch XHTML nicht. Die Erweiterung der Funktionalität erfolgt gleichermaßen wie bei HTML mit Scriptsprachen. Die Seitengestaltung wird mit CSS vorgenommen. Ein sinnvoller Einsatz von XHTML ist demnach nur in der Kombination mit CSS möglich.

Der Umstieg von HTML zu XHTML ist in unserem Fall nur noch ein kleiner Schritt. Wir haben in den hinter uns liegenden Abschnitten weitestgehend auf HTML-Formatanweisungen verzichtet und statt dessen CSS eingesetzt. Alle Tags wurden bereits in Kleinschreibung vorgenommen, der Unterschied zwischen Groß- und Kleinschreibung in XHTML ist signifikant. Ebenfalls haben wir die Werte der Attribute zwischen doppelte Hochkommata gestellt.

Für den Umbau älterer HTML-Dokumente wird vom W3C die Software *Tidy* angeboten. Tidy kann online benutzt oder lokal auf dem eigenen Rechner installiert werden. Ich habe Tidy erfolgreich online unter *infobound.net/tidy* getestet.

Wenden wir uns nun im einzelnen den Regeln zum Umbau von HTML nach XHTML zu.

Umlaute & Sonderzeichen, Codierung

Innerhalb eines XHTML-Dokuments dürfen keine Sonderzeichen auftreten. Für die Darstellung von Sonderzeichen und reservierten Zeichen müssen entweder der Zeichenname hinter dem Zeichen & oder der Dezimalcode hinter den Zeichen &# angegeben werden. Beide Angaben schließen mit einem Semikolon. Im Beispiel der Image-Maps haben wir die Zeichen < > bereits durch `<` und `>` ersetzt, mit Dezimalcode wäre `&#60;` und `&#62;` korrekt.

XML-Deklaration

Die erste Zeile eines XHTML-Dokuments muss eine XML-Deklaration aufnehmen. In dieser Zeile kann auch die benutzte Zeichencodierung deklariert sein.

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

für die westeuropäische Sprachen oder für das *Unicode Transformation Format* (UTF).

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

Dokumententyp – strict, transitional, frameset

Mit der Zeile:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

weist eine Dokumententyp-Deklaration auf eine der DTD-Dateien (wird im Kapitel XML erläutert) für XHTML hin. Der Typ kann `strict`, `transitional` oder `frameset` sein. Im Zusammenhang mit der Validierung, vgl. Abschnitt 2.5, unterliegt die Strict-Variante den strengsten Regeln. Nicht mehr unterstützte HTML-Elemente führen zu Fehlern bei der Validierung. In zukünftigen Versionen von XHTML wird es diese Unterscheidungen nicht mehr geben. Hier die alternativen Angaben für `strict` und `frameset`:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Namensraum

Innerhalb eines XHTML-Dokuments ist bei Angabe des Startelements von HTML ein Namensraum zu spezifizieren. Die Angabe des Sprachattributs `lang` sollte in den Varianten für XML und HTML angegeben werden

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
```

Struktur von XHTML

Die Struktur einer XHTML-Datei im Zusammenhang hat demnach folgendes Aussehen:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
lang="en">
  <head>
   <title>XHTML-Grundlagen</title>
   <link rel="stylesheet" type="text/css" href="referenz style
sheet" />
  </head>
  <body>
</body>
</html>
```

Listing 2.9: Gültige XHTML-Datei

Ende-Tag

Alle Elemente eines XHTML-Dokuments müssen korrekt verschachtelt sein. Jedes Element muss einen Schlusstag haben. Leere Elemente wie <br> müssen einen Schlusstag haben, oder das Anfangstag endet mit />, Leerzeichen vor /, zB.
. Vergleiche auch das zuvor notierte Element Link. Die gleiche Notation gilt auch für das Image-Tag.

Element-identifikation

Die Attributwerte `id` und `name` sind in XHTML zwingend vorgeschrieben für die Tags: `a`, `applet`, `form`, `frame`, `iframe`, `img`, `map`. Skript- und Stilelemente müssen das Attribut `type` aufweisen, vgl. wiederum obiges Link-Element. Alle Img- und Area-Elemente müssen das Attribut `alt` aufweisen.

Attribute

Attribute werden in doppelte Hochkommata eingeschlossen. Attribute dürfen nicht minimiert werden. Korrekte Beispiele sind etwa:

```
<td nowrap="nowrap"></td>
<hr noshade="noshade" />
```

PCDATA

Eingebettete Scripts oder Stilvorlagen werden in XHTML anders gehandhabt als bei HTML. Damit auch Sonderzeichen innerhalb von Scripts oder Stilvorlagen benutzt werden können, ist folgende Einbettung notwendig:

```
<script type="text/javascript">
  <!--
  <![CDATA[     /* Scriptformulierung */
  ]]>
  //--> </script>
```

Unter strenger Beachtung obiger Regeln sind wir jetzt in der Lage, gültige XHMTL 1.0-Dokumente zu publizieren. Wenn wir dann noch auf die zulässigen Elemente der Varianten Transitional und Frameset verzichten, nähern wir uns der nächsten Version: XHTML 1.1 modularisiert. Diese Version von XHTML kennt nur noch die Elemente der Variante `strict`. Die Modularisierung hat für die praktische Anwendung aber eher weniger Bedeutung.

Radikaler wird der Umstieg auf XHTML 2.0. Der Arbeitsentwurf für diese Version liegt seit August 2002 vor. Wenn jetzt noch XHTML-Basic genannt wird, dann dokumentiert das die Schnelllebigkeit der Entwicklungen im Web. Aber im Vergleich zum Softwareentwickler muss der Anwender nicht immer sofort die neueste Sprachversion einsetzen. Wir müssen schließlich noch auf die Implementierung der Spezifikationen in den Browsern

warten. Die Portalseite für alle hier noch unbeantworteten Fragen zu den Markup-Sprachen des Web ist: *www.w3.org/MarkUp* .

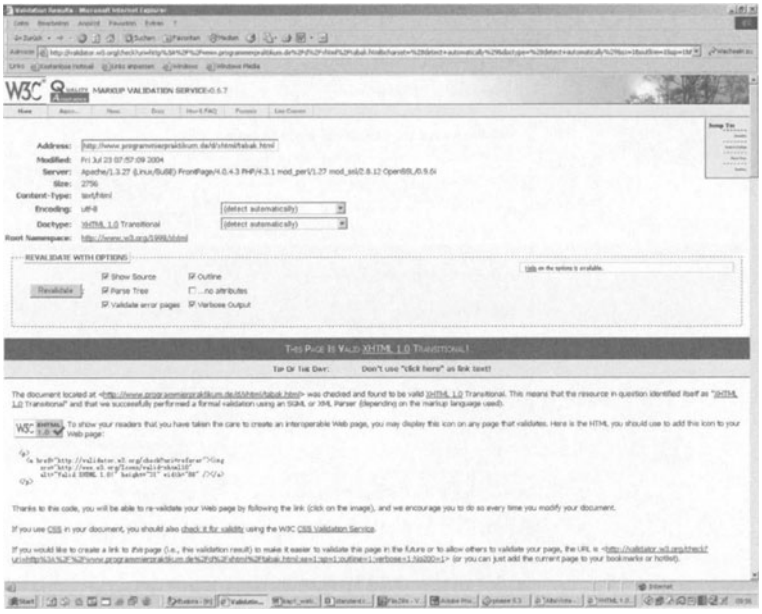


Abb. 2.10: Validierung einer Webseite beim W3C

2.5 Validierung von XHTML und CSS

Der Praktiker ist gut beraten, Webseiten in XHTML in Kombination mit CSS zu produzieren. Lieb gewordene Gewohnheiten braucht man nicht abzulegen, wenn die Variante `transitional` der DTD von XHTML 1.0 deklariert wird.

Zur Verbesserung der Zuverlässigkeit und Konformität der Kommunikation im Internet bietet das W3C einen Validierungsservice., den Sie unter *http://validator.w3.org.* nutzen können Abbildung 2.11 zeigt das Protokoll einer Validierungssitzung. Angegeben ist die URI des zu validierenden Webdokuments, Codierung und Variante lassen wir von der Software automatisch bestimmen. Nach Korrektur der Fehlermeldungen wird das Dokument als gültig erklärt. Man kann einen Hinweis auf die Validierung in das fertiggestellte Web-Dokument einbauen. Dieser enthält dann auch den entsprechenden Link zum Validerungsservice.

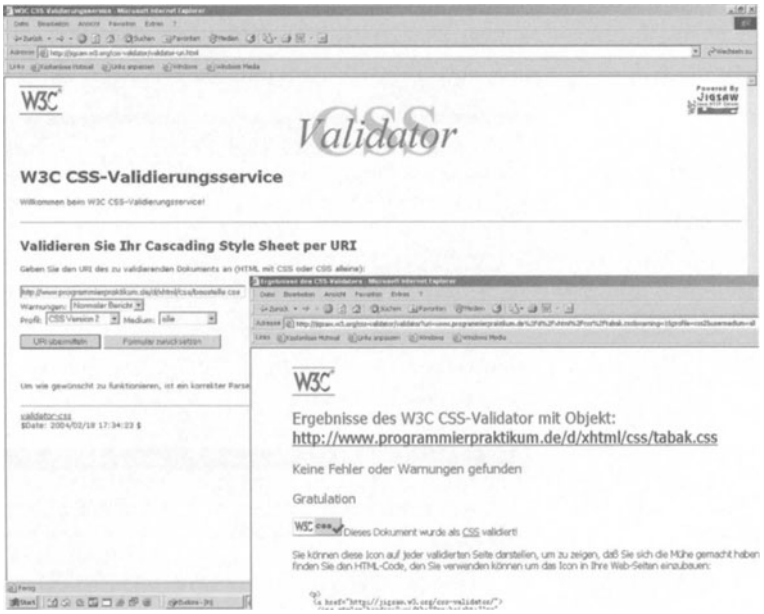


Abb. 2.11: Validierung von Stilvorlagen

Neben der XHTML-Codierung muss natürlich auch die Stilvorlage gültig sein. Die Validierung von CSS-Dateien kann online unter *jigsaw.w3.org/css-validator* vorgenommen werden. Nach Eingabe der Formularparameter gibt es entweder eine Seite mit Fehlermeldungen oder das Zertifikat.

Mit der Validierung schützen Sie sich nicht vor Browserinkompatibilitäten, Ihre Seite wird aber als fehlerfrei und W3C-konform ausgezeichnet. Die validierten XHTML-Seiten des Buches finden Sie unter *www.programmierpraktikum.de* im Kapitel XHTML. Mit einem Klick auf die Icons geleitet Sie Ihr Browser zu den W3C-Validierungsseiten.

2.6 Publikation von Webseiten

Siteorganisation und Entwurf

In der Praxis wird der Produktionsprozess einer Website oder einer Webanwendung nicht immer den wünschenswerten Verlauf von der Definitionsphase über den Entwurfsprozess zur Implementierung und Test mit gleichzeitig entstehender Dokumentation nehmen. Aus einer auf Zuruf gebastelten Homepage werden schnell etliche Webseiten mit vielen Dateien. Eine Website ist immer in der Aktualisierungsphase. Teamarbeit ist gefragt bei der Pflege der Inhalte, dem Kontakt zu den Besuchern, der Aktualisierung und den Erweiterungen. Auch bei mittleren Pro-

jekten stellt sich schon die Frage des Einsatzes eines *Content Management Systems* (CMS).

Neben den projektbezogenen und gestalterischen Fragen, ist für den Programmierer die Navigation und die Organisation der Dateien besonders zu beachten. Wichtig ist die Einhaltung einer übersichtlichen Verzeichnisstruktur und die Auswahl geeigneter Dateinamen. Ihre Site soll auf einem Server gehostet werden, dessen Betriebssystem Sie möglicherweise noch nicht kennen. Wählen Sie daher immer Kleinschreibung für Dateinamen, benutzen Sie keine Sonderzeichen in den Dateinamen und verwenden Sie auch nicht zu lange Bezeichner. Die alte *acht-Punkt-drei* Konvention hat sich bei einer übersichtlichen Verzeichnisstruktur immer bewährt. Geben Sie alle Referenzen in Ihren Dokumenten als relative Pfadangaben an. Sie gehen immer von dem Ort der Ursprungsdatei aus und verfolgen den Pfad bis zur Referenzdatei mit dem Schrägstrich als Trennung. Mit `../` gelangen Sie in der Verzeichnishierarchie eine Ebene höher. Der Verzeichnisbaum von *www.programmierpraktikum.de* entspricht der folgenden Abbildung 2.12:

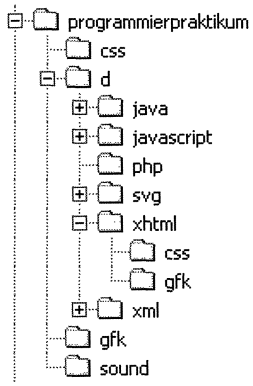
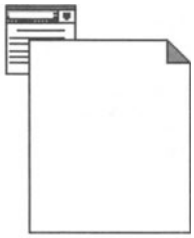


Abb. 2.12: Verzeichnisbaum www.programmierpraktikum.de

Auf der obersten Ebene bestehen die Verzeichnisse *css*, *d*, *gfk*, *sound* und natürlich die Datei *index.html* – die sog. Homepage. In *css* lagern die Stilvorlagen, in *gfk* allgemein benötigte Grafiken und in *sound* Audiodateien. Das Verzeichnis *d* enthält die Inhalte in deutscher Sprache. Benötigt man eine zweite Sprache, dann kann die Struktur z. B. für Englisch in ein Verzeichnis *gb* übernommen werden. Referenzen müssen bei Einhaltung dieser Vorgaben in übersetzten Seiten nicht geändert werden. Bei

grundsätzlich einsprachigen Seiten wäre der Verzeichnisname *html* auch eine gute Wahl für das Verzeichnis der Web-Dokumente. Jeder Menüpunkt bekommt sein eigenes Unterverzeichnis *java, svg, xhtml* usw. Jedes dieser Verzeichnisse sollte wieder die Datei *index.html* enthalten. *Index.html* wird vom Webserver bei Angabe des Verzeichnisses automatisch aufgerufen. Unterhalb der Navigationsstruktur, die sich hier abbildet, können wieder geeignete Einteilungen vorgenommen werden.

UML WAE



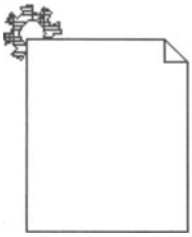
Unified Modeling Language (UML) ist ein Werkzeug zum Entwurf und zur Modellierung in der objektorientierten Softwareentwicklung. Verbreitet sind die Klassendiagramme als grafische Darstellungen der Softwarekomponenten. Die *Web Application Extension* (WAE) von UML sind Symbole und Piktogramme zur grafischen Darstellung des Entwurfs von von Webanwendungen. Im Internet wurde die Seite

www.cs.uwaterloo.ca/~dbbrown/cs338/cs338_uml_web_extensions.php

besucht. Dort verweist ein Link auf die Spezifikation

www.conallen.org/technologyCorner/webextension/WebExtension091.htm.

UML-WAEs stellen ein hilfreiches Werkzeug dar, spätestens beim Redesign eines ausgewucherten Projektes wird man sich eines derartigen Werkzeuges bedienen. Wir betrachten in Auszügen das Organigramm von *www.programmieren.de.*



Die Startseite *index.html* ist ein Frameset. Von *index.html* wird gelinkt auf die Navigationsstruktur *menue.html*, eine Titelseite im Hauptframe *inhalt* und die Leerseite mit der Hintergrundfarbe für den Rand. Über die Navigation gelangt man zu den Inhalten der oberen Ebene, die alle untereinander verlinkt sind. Die Linien tragen keinen Pfeil und gelten daher für beide Richtungen. Der Zielframe ist immer `target="inhalt"`. Alle Seiten sind Clientseiten. Unterhalb von *xhtml* existieren Links zu Clientseiten mit dem `target="_blank"`, es wird jeweils ein neues Fenster geöffnet. Ein weiterer Link ist ein Frameset mit den referenzierten Clientseiten.

Aus der Seite PHP heraus werden Formulare aufgerufen. Bei Eintreten des Ereignisses `submit` rufen diese Seiten ein serverseitiges Script auf, das dem Browser ggf. eine HTML-Seite liefert.

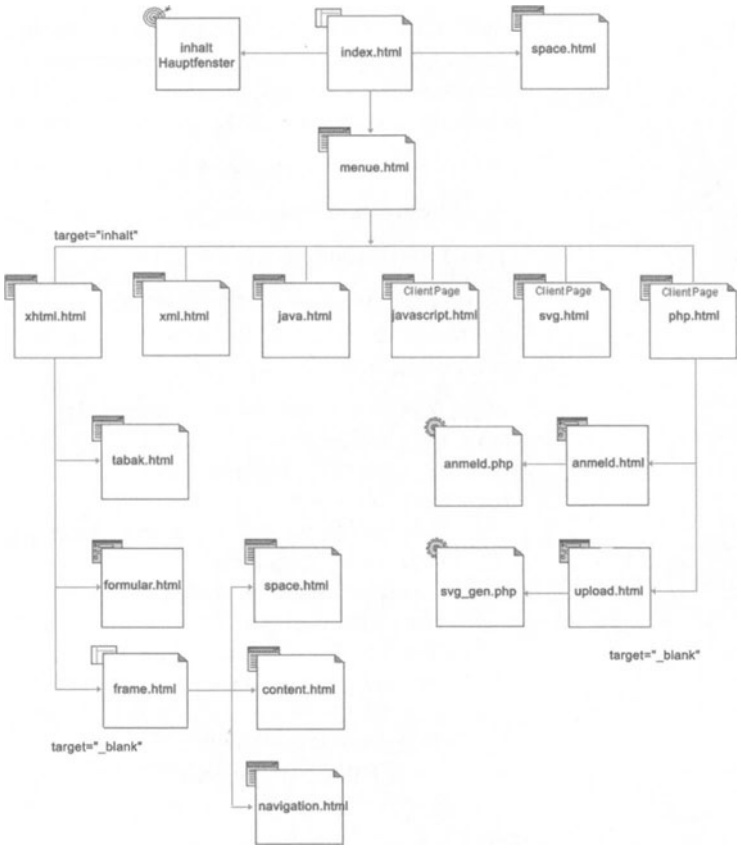


Abb. 2.13: Sitestruktur von www.programmierpraktikum.de dargestellt mit UML-WAE

Bereits bei kleineren Projekten kann man schon von dem Entwurf einer Webpräsenz mit UML-WAEs profitieren. Das Werkzeug ist noch nicht weit verbreitet. Im Rahmen dieses Buches konnte die UML-WAE nur kurz skizziert werden. Für vertiefte Betrachtungen sei auf die o.g. Web-Referenzen verwiesen. Im Leserbereich von *www.programmieren.de* finden Sie die UML-WAE-Symbole in verschiedenen Grafikformaten.

Meta-Tags

Das Internet bietet eine Fülle von Daten in nicht strukturierter Ordnung. Eine Methode, zu den gewünschten Daten zu gelangen, ist die Anfrage an eine Suchmaschine. Suchmaschinen halten sehr große Datenbanken vor, die Teile des Internets beinhalten. Diese Informationen bekommen die Suchmaschinen über sog. Web-Spider, Web-Crawler oder auch Robots genannt, die

das Internet nach Informationen durchsuchen. Neben den Textteilen einer Webseite werden auch die Informationen im Title-Befehl und in den Meta-Tags, beide im Kopfbereich der Seite, von den Crawlern ausgewertet und in die Datenbank der Suchmaschine übernommen. Bekannte Suchmaschinen sind u. a.;

www.google.com, www.webcrawler.com, www.altavista.com.

Die allgemeine Syntax des Meta-Befehls ist:

```
<meta name="Name" content ="Wert">
```

Suchmaschinen werten die Angaben zu den Attributen `keywords` und `description` aus. Wir bereiten die Metatags für die Hompage zu diesem Buch vor:

```
<title>Web-Programmierpraktikum</title>
<meta name ="author"
      content="the ImageFactory">
<meta name="keywords"
      content="Web-Programmierung, XHTML, CSS,
      XML, Java, JavaScript,
      Scalable Vector Graphics, SVG,PHP">
<meta name="description"
      content="Grundlagen der
       Webprogrammierung, statische und
       dynamische Webseiten, Grafik im Web,
       Programmierpraktikum">
<meta name="date" content ="2004-07-24">
<meta name="copyright" content="GWV Verlag">
```

Die folgende Angabe würde der Suchmaschine untersagen, Inhalte aus der Seite zu entnehmen:

```
<meta name="robots" content="noindex">
```

Eine Informationtion über die Aktualität der Seite mit dem Hinweis an die Suchmaschine, die Informationen nach 14 Tagen wieder aufzufrischen, wäre:

```
<meta name="revisit-after" content="14">
```

Sie brauchen die Meta-Tags nicht manuell codieren. Es gibt im Internet Meta-Tag Generatoren, die Ihnen aufgrund einer bestehenden Seite und nach Eintrag der zusätzlichen Werte die Meta-Tags automatisch generieren. Nachfolgend ein Ergebnis des Besuchs bei *www.metatag-generator.de.*

```
<meta http-equiv="content-type" content="text/html; char-
set=iso-8859-1">
<meta http-equiv="pragma" content="cache">
<meta name="robots" content="INDEX,FOLLOW">
<meta http-equiv="content-language" content="de">
<meta name="description" content="Programmierung statischer und
dynamischer Web-Anwendungen">
<meta name="keywords" content="Programmieren, Web, Java, SVG,
Grafik im Web">
<meta name="author" content="Guenter Pomaska">
<meta name="publisher" content="GWV-Verlag">
<meta name="audience" content="Alle">
<meta name="page-type" content="Anleitung">
<meta name="page-topic" content="Computer">
<meta http-equiv="reply-to" content="gp@imagefact.de">
<meta name="creation_date" content="2004-07-24">
<meta name="revisit-after" content="14 days">
<title>Grundkurs Web-Programmierung </title>
```

Zur Vermeidung von Problemen, die Suchmaschinen beim Antreffen von Framesets haben, sollte man die Inhaltsbeschreibung in diesem Fall als Text in den Noframes-Bereich einfügen.

Sie können Ihre Seiten manuell bei einzelnen Suchmaschinen eintragen oder einen kostenlosen Eintragsservice über das Internet nutzen. Um den Bekanntheitsgrad Ihrer Seite weiter zu erhöhen, können Sie sich in Webkataloge eintragen oder an Linklisten und Bannertausch teilnehmen.

Eine andere Anwendung von Meta-Tags besteht noch in dem sog. Client–Pull. Mit

```
<meta http-equiv="Refresh"
      content="5;
      URL=http://www.programmierpraktikum.de">
```

wird nach einer Zeit von 5 Sekunden die angegebene URL geladen. Mit dieser Anweisung können Sie Animationen oder automatisch ablaufende Präsentationen erstellen.

Upload

Als letzter Schritt ist jetzt noch das Hochladen der Dateien auf einen Webspace zu vollziehen. Für Testzwecke können Sie Provider finden, die Ihnen einen kostenlosen Zugang gewähren. Aber wir wollen später auch eigene PHP-Skripte ausführen, das muss der Provider unterstützen. Ich empfehle Ihnen eine eigene Webpräsenz, ohne Werbung, die Sie schon für geringe Monatsgebühren betreiben können - *www.programmierpraktikum.de* ist bei Neue Medien Münnich *www.allink.com* gehostet.

Wenn Sie Ihre Zugangsdaten bereit haben, benötigen Sie einen FTP-Client zum Hochladen der Dateien. Derartige Programme können Sie aus dem Internet beziehen. Hier wird Filezilla vorgestellt, download von *sourceforge.net/projects/filezilla.*

Filezilla hat eine recht übersichtliche Fensterstruktur. Adresse, Benutzer, Passwort und Port sind Ihre persönlichen Daten zum Einloggen auf dem Webspace bei Ihrem Provider. Das obere Fenster ist ein Statusfenster mit Nachrichten für Übertragungsbefehle. Links haben Sie den lokalen Bereich und rechts den serverseitigen Bereich. Dort können Sie auch die Benutzerberechtigungen für die Dateien eintragen. Im Kapitel PHP kommen wir auf diese Eintragungen zurück.

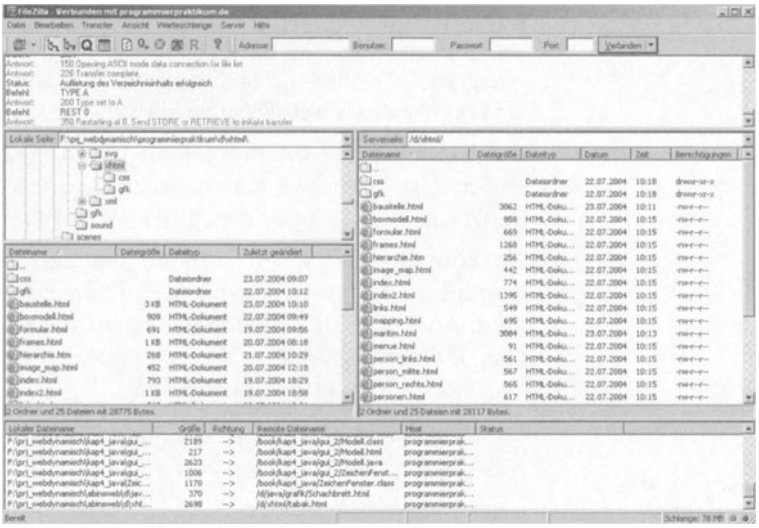


Abb. 2.14: Filezilla: FTP Client zum Dateitransfer mit dem Server

Sie markieren entweder serverseitig oder clientseitig die Dateien, bei Klick auf die rechte Maustaste erhalten Sie im Kontextmenü `Download` oder `Upload` angeboten. Mehrere ausgewählte Dateien erscheinen im unteren Fensterbereich als Warteschlange. Dort wird Ihnen auch der Status des Transfervorgangs angezeigt.

Jetzt ist es geschafft, Sie sind mit einer eigenen Internetpräsenz im Web. Rufen Sie Ihre eigene Seite im Browser zum abschließenden Test auf. Die Anforderung wird vom Webserver bedient, Ihr Browser rendert hoffentlich in der von Ihnen gewünschten Weise. Aber Vorsicht, löschen Sie Ihren Browser-Cache, sonst bekommen Sie ggf. Daten angezeigt, die sich noch im Zwischen-

speicher befinden und evtl. nicht mit der Präsenz auf dem Webserver übereinstimmen.

2.7 Zusammenfassung

Mit der Durcharbeitung dieses Kapitels haben Sie die Grundlagen zur Publikation statischer Webseiten gelegt. Sie können validierte XHTML-Seiten mit CSS-Stilvorlagen gestalten.

Eine XHTML-Seite ist in die Bereiche <head> und <body> eingeteilt. Es gibt für jeden Befehl einen Start-Tag und ein Ende-Tag. Die Befehle werden mit Attributen versehen, denen Werte zuzuweisen sind. Groß- und Kleinschreibung werden unterschieden. Spitze Klammern und doppelte Hochkommata stellen die reservierten Zeichen dar.

Den HTML-Elementen entsprechen die CSS-Selektoren. Der Block eines Selektors enthält in geschweiften Klammern die Attribute mit Doppelpunkt getrennt vom Wert, der mit einem Semikolon abgeschlossen wird. Gleiche Elemente werden unterschiedlich durch Einsatz von Klassen dargestellt. Die kaskadierende Stilvorlage kann sich irgendwo im Internet befinden. Individuellen Eintragungen überschreiben die globalen.

Zum Einstieg in die Syntax der Sprachen wird ein wenig Übung mit einem Texteditor empfohlen. Danach können Sie gängige Werkzeuge wie HTML-Editor, Stylesheet Generator, FTP-Client und Meta-Tag-Generator einsetzen. Ein Bildbearbeitungsprogramm zur Aufbereitung von Grafiken vervollständigt Ihren Werkzeugkasten.

Etliche nützliche Webreferenzen wurden vorgestellt, einige Tipps und Tricks wurden angewandt. Wissen Sie noch, was Client-Pull ist ? Hatten Sie schon einmal von UML-WAEs gehört ? Was waren doch noch Pseudo-Elemente, Text-Links mit CSS und Image-Maps ?

Lassen Sie Ihre Seiten auf Konformität zu den W3C-Standards überprüfen. Mit der gelungenen Validierung können die Seiten den Web-Crawlern bedenkenlos zur Verfügung gestellt werden.

3 Extensible Markup Language XML

Standard Generalised Markup Language (SGML) ist seit 1986 ein ISO-Standard. Die Auszeichnungssprache wurde zur Strukturierung und Darstellung von großen Informationsmengen entwickelt. Aufgrund der Komplexität von SGML war der Einsatz für Web-Anwendungen nicht geeignet. Aus SGML entwickelte sich HTML die *Hyper Text Markup Language*, eine Auszeichnungssprache mit der Definition von Methoden zur Darstellung von Dokumenten. In HTML ist aber noch nicht das Konzept der Trennung von Inhalt und Darstellung vorhanden. Mit Einführung der XML-Architektur sollen die Funktionalitäten von SGML in kompakter Form Web-Anwendungen zugänglich werden.

Extensible Markup Language (XML) selbst ist keine Sprache, es stellt eine strenge Formatspezifikation zur Strukturierung von Inhalten und Ableitung von XML-basierten Sprachen dar und ist die Basisdefinition einer Technologie-Gattung. Im vorhergehenden Kapitel wurde die *Extensible Hypertext Markup Language* (XHTML) vorgestellt. XHTML ist eine XML Anwendung zur Umformulierung von HTML und stellt die Brücke zwischen HTML und XML dar.

3.1 Verarbeitung von XML-Dokumenten

Die effiziente Verwendung von Informationen wird durch die Dreiteilung in Inhalt, Struktur und Layout gewährleistet. XML ist ein systemunabhängiges, medienneutrales, offenes Format. Ein Web-Browser stellt ein wohlgeformtes XML-Dokument als Baumansicht der Strukturelemente dar. Ein XML-Dokument gilt als gültig, wenn die Elemente in einer *Document Type Definition* (DTD) deklariert sind. Eine DTD enthält das Vokabular, welches in einem Dokument verwendet werden darf, und legt die Grammatik fest. Zur Grammatik gehören die Regeln der Verschachtelung und Angaben über die Elemente.

Ein XML-Parser ist ein Programm zur Verarbeitung eines wohlgeformten XML-Dokuments, der die XML-Datei einliest und den Knotenbaum weiterverarbeitenden XML-Anwendungen bereitstellt.

XML-Dokumente können mit CSS-Stilvorlagen clientseitig im Browser formatiert angezeigt werden. Zur Umsetzung von XML-Dokumenten wurde die *Extensible Style Sheet Language* (XSL) entwickelt. XSL ist unterteilt in Formatierung XSL-FO und Transformationen XSLT. Formatierer konvertieren ein XML-Dokument in Formate wie RTF oder PDF. XSLT-Prozessoren transformieren XML in HTML oder andere XML-Formate.

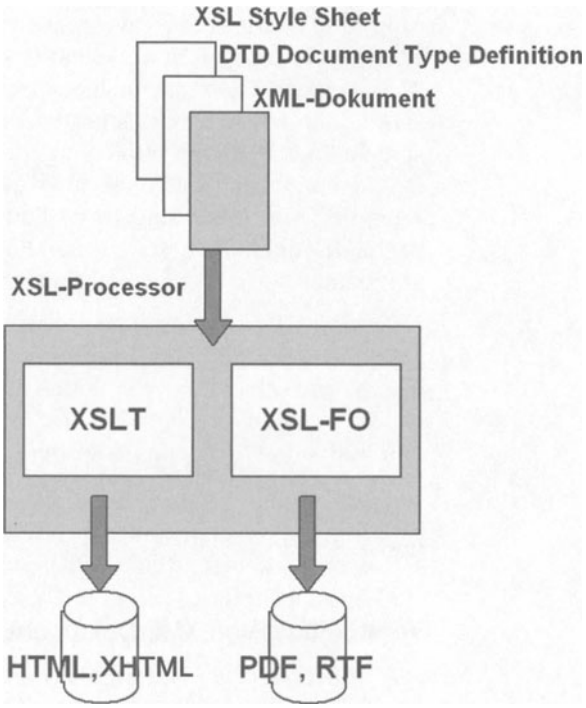


Abb. 3.1: Verarbeitung von XML-Dokumenten

Transformationen der XML-Dokumente werden üblicherweise serverseitig vorgenommen. Nach Anfrage wird das benötigte Format erzeugt und dem Browser zur Anzeige auf dem Endgerät (Monitor, PDA, Handy) übergeben. Java und PHP stellen umfangreiche Klassen zur Handhabung von XML-Dokumenten zur Verfügung.

In diesem Kapitel finden Sie auch eine kleine Java-Lösung, mit der Sie Ihre XML/XSL Dateien nach HTML clientseitig transfor-

mieren können. Auf dem Server stellt man dann die bereits konvertierte HTML-Version bereit.

Eine dritte Lösung ist die Übergabe von XML- und XSL-Dateien an den Browser und die clientseitige Transformation. Die Beispieldateien wurden mit MS Internetexplorer 6.0 und Firefox 0.9.2 getestet. Wenn kein Stylesheet angegeben ist, wird der Strukturbaum mit den Elementen des Inhalts vom Browser angezeigt. Andernfalls wird das Dokument entsprechend der Vorgaben gerendert.

Die obigen Ausführungen zur Anwendung der XML-Technologie mögen kompliziert erscheinen, Sie sollten aber in keinem Fall vor dem Einstieg in XML zurückschrecken. In den nachfolgenden Abschnitten wird die schrittweise Einführung in ein kleines Anwendungsbeispiel mit Einsatz der XML-Grafik SVG die Mächtigkeit von XML aufzeigen. Sie werden erfahren, dass die Anwendung von XML und XSL nicht so schwierig ist. Ihre Daten sind im XML-Format wohlgeformt gespeichert.

3.2 Struktur von XML-Dokumenten

XML-Dokumente sind Textdateien, die mit einem einfachen Texteditor zu bearbeiten sind. Die Syntax von XML entspricht der im Abschnitt 2.4 bereits für XHTML benutzten Notation. Alle Elemente müssen durch Anfangs- und Ende-Tag gekennzeichnet und korrekt verschachtelt sein. Attributwerte sind durch doppelte Hochkommata auszuzeichnen. Groß- und Kleinschreibung wird unterschieden. Reservierte Zeichen werden codiert. Abschnitte, die nicht vom Parser zu bearbeiten sind werden als CDATA gekennzeichnet. Kommentare sind in <!--*Kommentar* --> eingeschlossen.

Prolog

XML-Dokumente beginnen mit dem Prolog, das ist der Bereich vor dem Wurzelement, der hauptsächlich die Dokumentendeklaration und die DTD bzw. Referenz auf eine externe DTD enthält. Näheres zur DTD erfahren Sie im Abschnitt 3.3.

Die erste Zeile des Dokuments ist die XML-Deklaration, deren Attribute `version`, `encoding` und `standalone` sind optional. Unter `encoding` wird der im Dokument benutzte Zeichensatz angegeben.

`Standalone` kann die Werte `yes` oder `no` zugewiesen bekommen. Der Wert `yes` dieses Attributs signalisiert dem Parser, dass es für das Dokument eine externe DTD gibt, die mit der `!DOCTYPE` An-

weisung eingebunden ist. Der System-Identifizierer teilt dem Parser mit, an welcher Stelle die Datei mit den Dokumententyp-Spezifikationen zu finden ist. Der Name der Dokumentenklasse im Beispiel ist `literaturverzeichnis` und muss mit dem Bezeichner des Wurzelelements übereinstimmen.

```
<?xml version="1.0" encoding ="ISO-8859-1"
      standalone="yes"?>
<!DOCTYPE literaturverzeichnis SYSTEM "litera-
tur.dtd">
```

Processing Instructions

Processing Instructions (Verarbeitungsinformationen) enthalten Informationen und Anweisungen für die Anwendung und können an beliebiger Stelle im Dokument auftreten. Die allgemeine Syntax lautet:

```
<?PI   Anweisung ?>
```

Die XML-Deklaration ist eine Verarbeitungsinformation. Für PI ist das dem Parser bekannte Schlüsselwort `xml` einzusetzen. Das Schlüsselwort `xml` weist den Parser an, das Dokument nach den XML-Regeln zu bearbeiten und die Zeichen entsprechend dem angegebenen Zeichensatz zu interpretieren. Andere Verarbeitungsinformationen begegnen uns weiter unten.

Inhalt

Jedes XML-Dokument enthält genau ein Wurzelelement. Darunter befinden sich die Elemente, die als Tags ausgezeichnet werden, mit den Subelementen. Im angeführten Beispiel ist das Wurzelelement

```
<literaturverzeichnis>
  ... Inhalt des Dokuments ...
</literaturverzeichnis>
```

Elemente, Schachtelung

Mit dem Wurzelelement `literaturverzeichnis` wird eine Instanz der Dokumentenklasse `literaturverzeichnis`, die in der `!DOCTYPE`-Deklaration angegeben ist, gebildet. Aus diesem Grund ist die geforderte Übereinstimmung dieser Bezeichner notwendig.

Ein Element besteht aus den Markup-Tags und seinem Inhalt. Die Bezeichnung der Elemente ist frei wählbar. Im Literaturverzeichnis gibt es das Element <buch>. Ein Element mit Unterelementen stellt einen Knoten im Strukturbaum dar. Ein Buch wiederum besteht aus den Elementen Autor, Titel, Verlag und Erscheinungsjahr. Insofern ist Buch ein Knoten im Strukturbaum. Bei der Anzeige im Browser, vgl. Abbildung 3.2, wird ein Knoten

mit – oder + gekennzeichnet. Per Mausklick kann der Knoten zu- oder aufgeklappt werden. Das Element buch kann mehrfach im Inhalt des Dokumentenelements <literaturverzeichnis> vorkommen.

Elemente, die keinen Inhalt aufweisen, werden als Leerelemente bezeichnet. Bei Leerelementen folgt das Ende-Tag unmittelbar auf den Start-Tag, <name></name>. Die abgekürzte Variante ist <name />.

```
<literaturverzeichnis>
  <buch>
      <autoren>Guenter Pomaska</autoren>
      <titel>Grundkurs Web-Programmierung</titel>
      <verlag>GWV Verlag, Wiesbaden</verlag>
      <datum>2004</datum>
    </buch>
</literaturverzeichnis>
```

Fügen wir als erste Zeile der Datei den deklarierenden Tag

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

ein und speichern die Datei mit der Erweiterungsbezeichnung xml, dann zeigt der Internet-Browser das wohlgeformte Dokument als Strukturbaum, wie in der Abbildung 3.2 zu erkennen ist.

Attribute

Attribute sind Zusatzinformationen für die Verarbeitung von Elementen und beschreiben deren Eigenschaften. Fügen wir dem Element Buch das Attribut sprache an, dann wäre der Wert dieses Attributs beispielsweise deutsch. Die gesamte Datenbank könnte dann nach deutschen Titeln durchsucht werden. Attributwerte sind zwingend in Hochkommata einzuschließen Sie können ' oder " verwenden. Hier das Beispiel:

```
<buch sprache="deutsch"></buch>
```

Die Regeln für das Vorhandensein von Attributen in einem Tag werden ebenfalls in der DTD festgelegt.

Entitäten

Entitäten enthalten Daten, auf die im Inhalt eines Dokuments verwiesen wird. Der XML-Prozessor ersetzt diese Entity-Referenz durch ihren Inhalt. Entitäten werden in der DTD mit

```
<!ENTITY gwv "GWV-Fachverlage GmbH, Wiesbaden">
```

oder, wenn der Inhalt in einer externen Datei, steht mit

```
<!ENTITY gwv SYSTEM "datei.txt">
```

angegeben. In den Elementen der XML-Datei ist dann vor den Bezeichner das Zeichen & zu setzen, z.B.

```
<verlag>&gwv</verlag>
```

Der Parser ersetzt die Entität gwv durch die vollständige Bezeichnung *GWV-Fachverlage GmbH, Wiesbaden.*

Abb. 3.2: Strukturbaum der Elemente einer XML-Datei, angezeigt im MSIE

White-Space

Leerzeichen, Tabulatoren, Zeilenumbruch und Zeilenvorschub werden auch als White-Space-Character bezeichnet. Der XML-Parser ignoriert diese Zeichen bzw. fasst auftretende Leerzeichen immer zu einem Zeichen zusammen. In XHTML ist uns der Tag `<pre></pre>` begegnet, mit dem die standardmäßige Behandlung dieser Zeichen ausgeschaltet wurde. XML benutzt das Attribut `xml:space` mit den Werten `default` oder `preserve`. In der DTD z. B.:

```
<!ATTLIST autoren xml:space (default | preserve)  "preserve">
```

hebt für alle Tags <autoren> die Ignorierung der White-Spaces auf. Im Element selbst könnte die Überschreibung der Zuweisung mit

```
<autoren xml:space="default">namen</autoren>
```

vorgenommen werden.

CDATA

Eingebettete Daten, die nicht der XML-Deklaration unterliegen, z.B. Scriptprogramme, müssen vom Parser ignoriert werden. Diese Datenbereiche werden in die Anweisung

```
<![CDATA[   ... Textdaten ... ]]>
```

eingebunden. Der XML-Parser überliest die Zeichen, die sich innerhalb des inneren eckigen Klammernpaares befinden.

3.3 Document Type Definition DTD

Wie oben gezeigt, geht es auch ohne DTD, die Dokumente sind wohlgeformt, aber nicht gültig. Will man ein Dokument validieren, d.h. auf seine Konformität zu den Vorgaben des W3C überprüfen, dann benötigt man eine DTD. Aber die Document Type Definition hat noch mehr Bedeutung. Wenn Sie Änderungen in einem XML-Dokument vornehmen, die nicht der vorgegebenen Struktur entsprechen, kann der Parser das allein nicht feststellen, eine DTD muss her. Bei umfangreichen Datenbeständen, Bearbeitung von Dateien durch mehrere Personen und beim Datenaustausch ist eine DTD unverzichtbar. Sie vermeiden mit einer DTD fehlerhafte XML-Dokumente.

Die DTD enthält das Vokabular und die Regeln zur Verarbeitung der Informationen eines XML-Dokuments. Definiert werden alle Elemente, Attribute und Entitäten und Angaben zu Anzahl, Inhalt und Verschachtelung der Elemente. Die DTD wird mit der Anweisung !DOCTYPE im Prolog eingeleitet. Die Definition selbst kann dabei eingebettet oder extern in einer Datei gespeichert sein. Wir betrachten exemplarisch wieder das Literaturverzeichnis:

```
<!DOCTYPE literaturverzeichnis [
<!ELEMENT literaturverzeichnis (buch)+>
<!ELEMENT buch (autor, titel,verlag, datum)>
<!ATTLIST buch sprache
(deutsch|englisch|spanisch|franzoesisch) #REQUIRED>
  <!ELEMENT autor (#PCDATA)>
  <!ELEMENT titel (#PCDATA)>
  <!ELEMENT verlag #PCDATA)>
```

```
  <!ELEMENT datum (#PCDATA)>
<!ENTITY gp "Guenter Pomaska">
]>
```

Das Element *literaturverzeichnis* kann ein oder mehrere Kind-Elemente *Buch* enthalten. Der Indikator für das Mehrfachauftreten ist das Zeichen +. Das Element *buch* muss in der vorgegebenen Reihenfolge die Unterelemente (*autor, titel, verlag, datum*) enthalten. Das Attribut *sprache* des Elements *buch* wird verlangt und kann die in der Liste angegebenen Werte zugewiesen bekommen. Durch das Schlüsselwort #REQUIRED kann nicht auf das Attribut verzichtet werden. Die Datentypen der Elemente werden als Text gekennzeichnet, PCData steht für *parsed character data.* Ein so gekennzeichneter Bereich darf beliebige Zeichen enthalten. Mit der letzten Anweisung wird der Inhalt der Entität gp beschrieben.

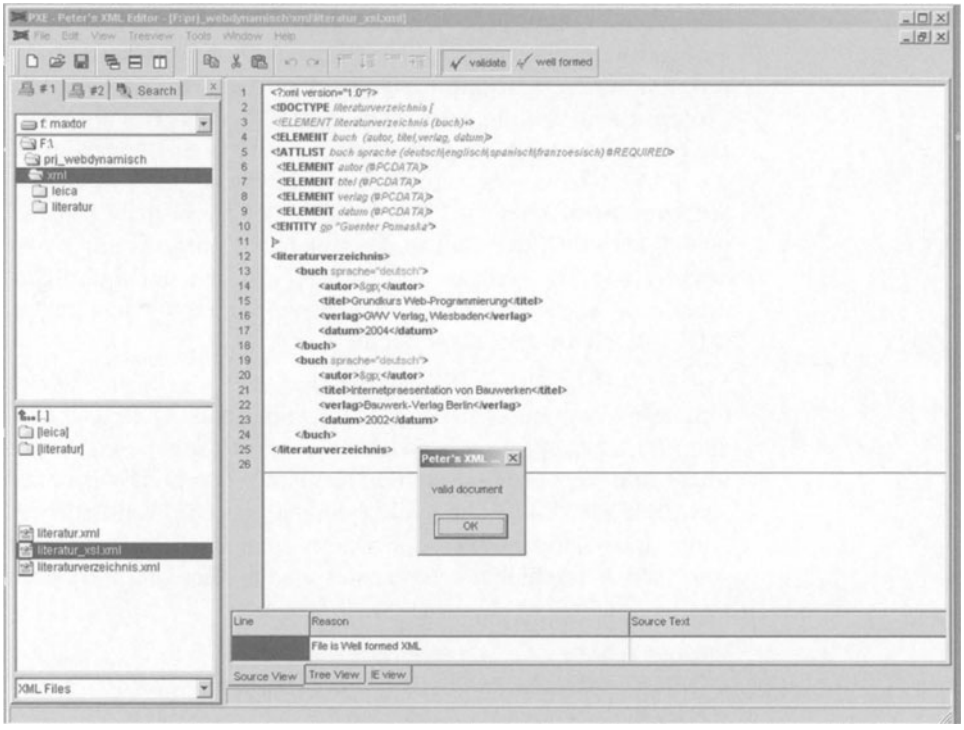


Abb. 3.3: Peter's XML Editor – Freeware zur Bearbeitung von XML-Dokumenten

Ist die DTD in der externen Datei *literaturverzeichnis.dtd* gespeichert, dann wird im Prolog wie folgt notiert:

```
<!DOCTYPE literaturverzeichnis SYSTEM "literaturverzeichnis.dtd">
```

Die Datei kann mit einem XML-Editor bearbeitet werden. Mit der Freeware Peter's HTML-Editor nutzen Sie im Rahmen dieser Einführung ein hinreichendes Werkzeug, das unter *www.iol.ie/~pxe* zu beziehen ist. Wie schon ausgeführt, kann ein XML-Dokument gültig oder wohlgeformt sein. Es ist wohlgeformt, wenn das Dokument durch Start- und Ende-Tag gekennzeichnete Elemente mit korrekter Verschachtelung enthält und der Syntax von XML entspricht. Gültig ist ein XML-Dokument, wenn eine DTD zugeordnet ist. Peter's Editor prüft das Dokument auf Korrektheit und gibt das positive Ergebnis mit der Meldung *valid document ok* an. Sie haben die Wahl zwischen Anzeige des Sourcecodes, Dokumentenbaums oder der Browser-Ansicht.

Wenn Sie nicht mit einem XML-Editor arbeiten, können Sie die Validierung auch online vornehmen. Legen Sie Ihre Datei auf den Webspace und validieren Sie z.B. mit

www.cogsci.ed.ac.uk/~richard/xml-check.html

Sie können danach bei der weiteren Bearbeitung von einem korrekten Dokument ausgehen.

Hier das gültige Dokument *literatur.xml* noch einmal im Zusammenhang:

```
<?xml version="1.0"?>
<!DOCTYPE literaturverzeichnis [
<!ELEMENT literaturverzeichnis (buch)+>
<!ELEMENT buch (autor, titel,verlag, datum)>
<!ATTLIST buch sprache (deutsch|englisch|spanisch|franzoesisch)
#REQUIRED>
  <!ELEMENT autor (#PCDATA)>
  <!ELEMENT titel (#PCDATA)>
  <!ELEMENT verlag #PCDATA)>
  <!ELEMENT datum (#PCDATA)>
<!ENTITY gp "Guenter Pomaska">]>
<literaturverzeichnis>
   <buch sprache="deutsch">
      <autor>&gp;</autor>
      <titel>Grundkurs Web ... </titel>
      <verlag>GWV Verlag, Wiesbaden</verlag>
      <datum>2004</datum>
```

```
</buch>
<buch sprache="deutsch">
   <autor>&gp;</autor>
   <titel>Internetpraesentation ... </titel>
   <verlag>Bauwerk-Verlag, Berlin</verlag>
   <datum>2002</datum>
</buch></literaturverzeichnis>
```

Listing 3.1: Ein gültiges XML-Dokument – *literatur.xml*

XMLStarlet Toolkit

XMLStarlet Toolkit ist ein Kommandozeilenwerkzeug zur Bearbeitung von XML-Dateien. Im Web finden Sie das Werkzeug unter *xmlstarlet.sourceforgce.net.* Hiermit können Sie Ihre XML-Dateien ebenfalls validieren. Das Tool kann u.a. analog zum Befehlszeilen Kommando `dir` bzw. `ls` die Angaben eines Festplattenverzeichnisses als XML-Datei ausgeben. Der Kommandozeilenaufruf lautet: `xml ls`. Das Ergebnis, hier im Ausschnitt , sieht noch ziemlich kryptisch aus.

```
<?xml version="1.0"?>
<directory>
  <d p="rwxrwxrwx" a="2004.07.27 13:49:59" n="."/>
  <d p="rwxrwxrwx" a="2004.07.27 13:48:09" n=".."/>
  <f p="rw-rw-rw-" a="2004.07.26 16:23:15" n="applets.html"/>
  <f p="rw-rw-rw-" a="2004.07.26 16:23:15" n="japplet.html"/>
  <d p="rwxrwxrwx" a="2004.07.27 11:10:12" n="java"/>
</directory>
</xml>
```

Es ist aber ein wohlgeformtes XML-Dokument. Das Wurzelelement ist `directory`. Die Elemente sind `d` und `f`, das sind leere Elemente die mit /> abschließen. Der eigentliche Dokumenteninhalt ist in den Attributen der Elemente abgelegt. Das Dokument wird gültig bei Angabe einer DTD, aber nicht lesbarer. Dieses Beispiel werden wir weiter unten entsprechend aufbereiten und in ein HTML-Format transformieren.

3.4 XML-Dokumente mit XHTML und CSS

Formatierung mit HTML und CSS

XHTML-Elemente können in XML-Dokumente eingebunden werden. Mit der Process Instruction

```
<?xml-stylesheet href="standard.css" type="text/css"?>
```

wird ein CSS Style Sheet referenziert. Unterhalb des Wurzelelements werden HTML-Befehle mit `html:` als Präfix für jeden Tag im Dokument eingeleitet.

Zur Einbindung der HTML-Befehle muss im Wurzelelement noch der Namensraum dem Attribut `xmlns:html` zugewiesen werden. Ein Namensraum stellt einen begrenzten Bereich dar, in dem ein vorgegebener Name aufgelöst wird.

```
<?xml version="1.0"?>
<?xml-stylesheet href="standard.css" type="text/css"?>
<!DOCTYPE literaturverzeichnis [
<!ELEMENT literaturverzeichnis (buch)+>
<!ELEMENT buch (autor, titel,verlag, datum)>
<!ATTLIST buch sprache
(deutsch|englisch|spanisch|franzoesisch) #REQUIRED>
  <!ELEMENT autor (#PCDATA)>
  <!ELEMENT titel (#PCDATA)>
  <!ELEMENT verlag (#PCDATA)>
  <!ELEMENT datum (#PCDATA)>
<!ENTITY gp "Guenter Pomaska">
]>
<literaturverzeichnis
   xmlns:html="http://www.w3.org/TR/REC-html40">
   <html:ul>
      <html:li>
         <buch sprache="deutsch">
            <autor>&gp;</autor>
            <html:br/>
            <titel> Grundkurs
                     Web-Programmierung</titel>
            <verlag>GWV Verlag,Wiesbaden</verlag>
            <datum>2004</datum><html:br/>
         </buch>
      </html:li>
      <html:li>
         <buch sprache="deutsch">
            <autor>&gp;</autor>
            <html:br/>
            <titel>Internetpraesentation
                von Bauwerken</titel>
            <verlag>Bauwerk-Verlag
                 Berlin</verlag>
         </buch>
      </html:li>
   </html:ul>
</literaturverzeichnis>
```

Listing 3.2: XML mit XHTML und CSS - *literatur_css.xml*

Dieses Beispiel ist aber noch wenig sinnvoll, müsste doch bei jedem Buchtitel das Listenelement und der Zeilenumbruch eingefügt werden. Verzichten wir andererseits auf die HTML-Befehle, dann erfolgt die Ausgabe aller Daten in einer Zeile. Ein Zeilenumbruch kann aber auch durch das Attributs `display` in der CSS-Datei erzwungen werden. Das XML-Dokument wurde modifiziert. Die DTD ist jetzt ausgelagert, die HTML-Angaben entsprechen der vollständigen Datei.

```
<?xml version="1.0"?>
<?xml-stylesheet href="standard_block.css"
                 type="text/css"?>
<!DOCTYPE literaturverzeichnis
          SYSTEM "literatur.dtd">
<literaturverzeichnis xmlns:html="http://www.w3.org/TR/REC-
html40">
<html:html>
<html:head><html:title>Literaturverzeichnis
           </html:title>
</html:head>
<html:body style="background-color:#ffffcc">
   <html:ul>
   <buch sprache="deutsch">
      <autor>&gp;</autor>
      <titel>Grundkurs Web-
            Programmierung</titel>
      <verlag>GWV Verlag, Wiesbaden</verlag>
      <datum>2004</datum>
   </buch>
      <buch sprache="deutsch">
      <autor>&gp;</autor>
      <titel>Internetpraesentation
            von Bauwerken</titel>
      <verlag>Bauwerk-Verlag Berlin</verlag>
   </buch>
   </html:ul></html:body></html:html>
</literaturverzeichnis>
```

Listing 3.3: XML mit XHTML und CSS: literatur_css_block.xml

Die CSS-Datei *standard_block.css* wird an dieser Stelle nur auszugsweise wiedergegeben:

```
AUTOR {
    display: list-item;
    font-family : Arial; font-size : 14px;
    font-style : normal;
```

```
    font-weight : normal; white-space : nowrap;
    color : Maroon;
}
DATUM,VERLAG { display : inline;}
TITEL { display : block; }
```

Mit dem obigen Beispiel wurde das Zusammenspiel zwischen XHTML, CSS und XML erläutert. Im nächsten Abschnitt wird zur Formatierung in die mächtigere Sprache *Extensible Style Sheet Language* (XSL) eingeführt

3.5 Extensible Stylesheet Language XSL

XSL steht für *Extensible Stylesheet Language* und ist den Bedürfnissen von XML angepasst. In Kombination mit XPATH, einer Sprache, mit der sich Ausdrücke zur Adressierung innerhalb von XML-Dokumenten formulieren lassen, stellt XSL ein mächtiges Werkzeug zur Verarbeitung von XML-Dokumenten dar. XSL umfasst zwei Teile: XSLT zur Transformation in andere Formate, etwa HTML, XHTML oder CSV, und XSL-FO zur Formatierung in PDF, RDF u.a.

Gegenüber dem Einsatz von CSS besteht mit XSL die Möglichkeit, Ablaufstrukturen wie bei einer Programmiersprache zu nutzen und auch Attributwerte auszugeben. Der Inhalt einer XSL-Datei kann zum großen Teil aus XHTML-Befehlen bestehen. Die Formatierung der Daten übernimmt in unserem Fall wieder CSS, eingebunden als *Character Data* (CDATA) oder externe Datei.

Eine XSL-Datei wird einem XML-Dokument zugeordnet mit der Processing Instruction:

```
<?xml-stylesheet href="meinxsl.xsl" type="text/xsl"?>
```

Die erste Zeile einer XSL-Datei deklariert den Namensraum. Die darauf folgende Zeile mit dem Element `template` gehört zum Namensraum `xsl` und definiert die Stilvorlage. Über das Attribut `match` lässt sich angeben, auf welche Elemente die Stilvorlage anzuwenden ist. Der Wert von match enthält ein Muster, das die Elemente beschreibt. Das Muster ist ein XPATH-Ausdruck und kann als Filter zur Behandlung ausgewählter Daten benutzt werden. Das Zeichen / zeigt im Beispiel auf das Root-Element, ein * kann als Wildcard genutzt werden.

```
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match ="/">
```

```
<! - es folgt die CSS-Definition in einem CDATA Abschnitt -->
< -- hier stehen XHTML-Befehle>
<xsl:for-each
     select="literaturverzeichnis/buch">
     <xsl:value-of select="autor"
 </xsl:for-each>
```

Die vorstehenden Anweisungen stellen eine Schleife mit einer Auswahl dar. `For-each` bedeutet mit jedem Element und `select` filtert das zu bearbeitende Element heraus. Mit `value-of select` werden die gewünschten Daten des aktuellen Datensatzes angesprochen. Betrachten wir die XSL-Datei für das Literaturverzeichnis im Zusammenhang:

```
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<style type="text/css">@import "standard.css";
</style>
<div align="center">
  <table cellspacing ="3" cellpadding ="3"
         bgcolor="#ffffff">
    <tr><td  colspan="3" class = "titel"
           style="color : #FFFFCE;
           background-color : Maroon;">
           Literaturverzeichnis</td></tr>
    <xsl:for-each
         select="literaturverzeichnis/buch">
    <tr><td class = "autor">
<xsl:value-of select="autor"/></td>
      <td></td><td></td></tr>
    <tr><td class="titel">
      <xsl:value-of select="titel"/></td>
      <td class="verlag">
     <xsl:value-of select="verlag"/></td>
      <td class="datum">
      <xsl:value-of select="datum"/></td></tr>
   </xsl:for-each>
 </table></div></xsl:template>
</xsl:stylesheet>
```

Listing 3.4: XSL-Datei template.xsl

Die Datei template.xsl ist in das XML-Dokument literatur_xsl.xml mittels:

```
<?xml-stylesheet href="template.xsl" type="text/xsl"?>
```

eingebunden. Die ersten beiden Zeilen deklarieren das Stylesheet und das Muster, auf das die Vorlage anzuwenden ist, hier das gesamte Dokument. Es folgen jetzt XHTML-Befehle. Die CSS-Stilvorlage wird im Style-Tag angegeben. Innerhalb eines Div-Tags wird eine Tabelle definiert. Die erste Tabellenzeile gilt über drei Spalten mit der Überschrift und einem eigenen Stil. Es werden jetzt für alle Unterelemente Buch zwei Zeilen der Tabelle benutzt. In der ersten Zeile, Spalte 1 wird der Autor mit `value-of-select` eingetragen. Spalten zwei und drei bleiben leer. Die Folgezeile wird spaltenweise mit den Werten `titel`, `verlag` und `autor` belegt. Alle Tags werden hiernach entsprechend der korrekten Verschachtelung geschlossen.

Im Browser aufgerufen, zeigt die Datei `literatur_xsl.xml` jetzt eine Tabellenform. Die XML-Datei wurde mittels einer XSL-Stilvorlage vom Browser in HTML transformiert.

Verarbeitung von Attributen

Es sollte noch gezeigt werden, wie der Zugriff auf die Attribute einer XML-Datei erfolgt. Weiter oben hatten wir mit dem Kommandozeilen-Werkzeug XMLStarlet durch Aufruf von

```
xml ls >directory.xml
```

das XML-Dokument *directory.xml* mit den Verzeichniseinträgen erzeugt. Der Dateianfang wird modifiziert

```
<?xml version="1.0"?>
<?xml-stylesheet version="1.0" href="template.xsl"
                 type="text/xsl"?>
<directory>
```

und das Wurzelelement geschlossen mit </directory>. Unser zugehöriges XLS-Dokument hat dann im relevanten Teil folgende Gestalt:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<xsl:for-each select="directory/f">
  <xsl:sort select="@n"case-order="lower-first"
            order="descending"/>
<tr>
 <td class = "name">
     <xsl:value-of select="@n"/></td>
    <td class="rechte">
      <xsl:value-of select="@p "/></td>
    <td class="groesse">
```

```
      <xsl:value-of select="@s"/></td>
    <td class="datum">
      <xsl:value-of select="@m"/></td></tr>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

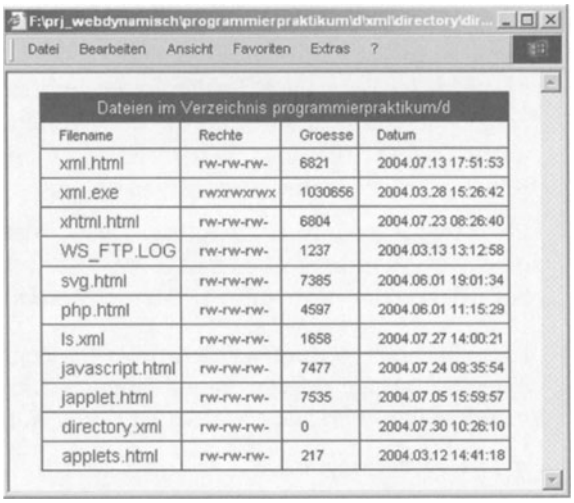


Abb. 3.4: Dateiverzeichnis aus XML generiert.

Im Attribut `match` wird das Wurzelverzeichnis des Dokuments angegeben, behandelt werden alle Elemente `directory/f`, also nur die Dateien, keine Verzeichnisse. Sortiert wird nach absteigenden Werten des Attributs `n`. Mit der Anweisung `select` wird auf Attribute zugegriffen, indem vor die Bezeichnung des Namens das Zeichen `@` gesetzt wird. In der Tabellenzeile werden die gewünschten Attribute ausgewertet.

3.6 XML nach HTML-Transformation

Online Transformation

Die Umformung eines XML-Dokuments in ein HTML-Dokument mittels XSL wird als XSLT-Transformation bezeichnet. Man benötigt eine wohlgeformte XML-Datei und eine gültige XSL Stylesheet-Datei sowie einen XSLT-Prozessor, der die Daten liest und entsprechend der Vorgabe in das neue Format umsetzt.

Sofern der Browser XSLT-Transformationen unterstützt, können Sie die XML-Datei direkt an den Browser übergeben, MS Internet Explorer hat hiermit keine Probleme.

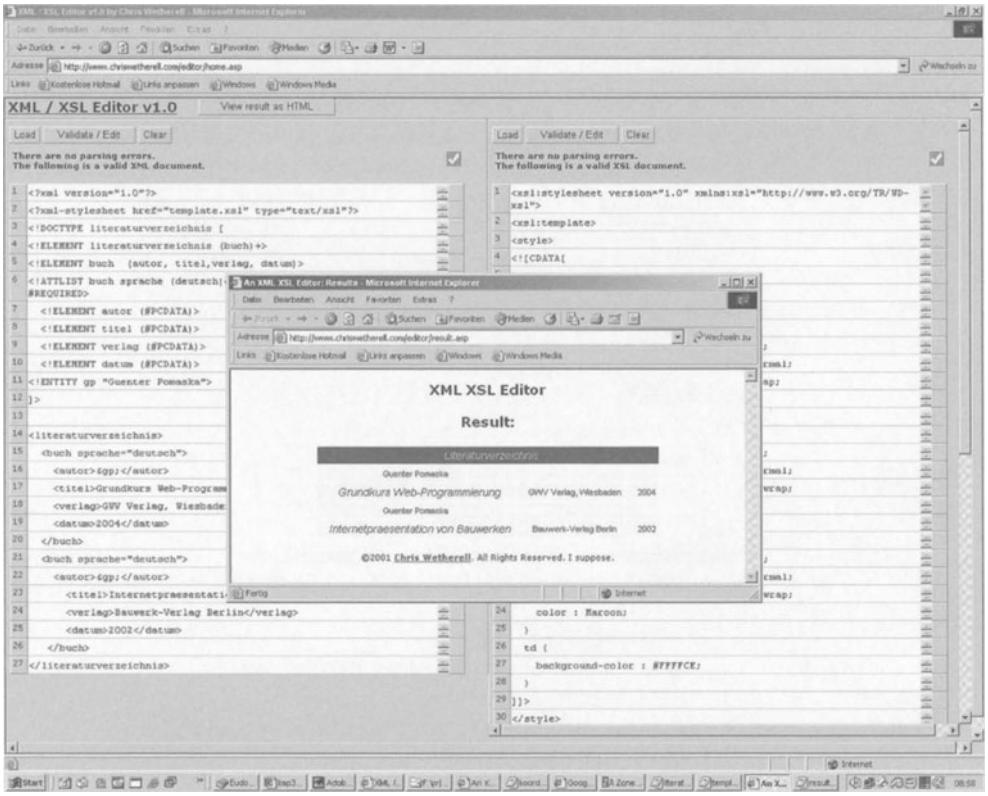


Abb. 3.5: Online XSL-Transformation

Wollen Sie jedoch auf Ihrer Webpräsenz unmittelbar die bereits in HTML transformierten Daten anbieten, dann können Sie die XSL-Transformation zunächst auch online veranlassen. Im Web finden Sie Kommandozeilenwerkzeuge und Seiten, mit deren Hilfe Sie die Transformation Ihrer Dateien durchführen können. Mit den Beispielen des Buches konnte ich die Transformationen auf den Seiten:

www.cherrypy.org/demo/xmlXslOnline

und

www.chriswetherell.com/editor/home.asp

erfolgreich bewerkstelligen. Letztere URL ist einfach und flexibel zu bedienen. Sie geben entweder die URL ihrer Dokumente an oder kopieren Ihren Sourcecode direkt in die Textfelder. Validierung und Transformation können getrennt erfolgen.

XSL-Transformation mit Java

Wenn Sie keine Java-Kenntnisse haben, wird Ihnen die nachfolgend vorzustellende Lösung hier noch nicht weiterhelfen. Nach Durcharbeitung des Kapitels Einführung in Java haben Sie aber sicher kein Problem, aus dem angegebenen Quellcode die Klasse zu kompilieren. Wie schon erwähnt, bietet Java umfangreiche Unterstützung zur Bearbeitung von XML-Dokumenten.

```
// jdk1.4.1
import javax.xml.transform.*;
import java.net.*;
import java.io.*;
public class HowToXSLT {
public static void main(String[] args) {
  try {
    TransformerFactory tFactory =
    TransformerFactory.newInstance();
    Transformer transformer =
      tFactory.newTransformer(new
      javax.xml.transform.stream.StreamSource
        ("template.xsl"));
    transformer.transform(new
      javax.xml.transform.stream.StreamSource
      ("literatur_xsl.xml"),
    new javax.xml.transform.stream.StreamResult
    ( new FileOutputStream
       ("literatur_java.html")));
    }
  catch (Exception e) {
    e.printStackTrace( );
    }
  }
}
```

Listing 3.5: XSLT-Transformation mit Java

Auf die Darstellung von Details müssen wir hier verzichten. Die Klasse HowToXSLT, Quelle:

www.rgagnon.com/javadetails/java-0407.html

importiert die benötigten Klassenbibliotheken, erzeugt die Instanzen und wendet darauf die Transformationsmethoden an. Im Beispiel sind die bekannten Dateien *template.xsl* und *literatur_xsl.xml* die Quelldateien. Zieldatei ist *literatur_java.html*. Wenn Sie Java ab jdk1.4.1 installiert haben, erzeugen Sie die Klasse und starten das Programm vorzugsweise aus der IDE. In-

stallieren Sie Programmcode und Daten im einfachsten Fall im selben Unterverzeichnis.

3.7 Eine XML-Applikation

Topographische Vermessungen werden heute mit elektronischen Tachymetern durchgeführt. Das Vermessungsinstrument wird auf einem festen Standpunkt positioniert. Auf Knopfdruck werden die Messdaten zu einem Zielpunkt (Reflektor) erfasst. Winkel und Strecken werden geräteintern umgerechnet und als Koordinatenwerte gespeichert. Zur Identifikation eines aufgemessenen Punktes dient die Punktnummer. Ein bedeutender Hersteller von Vermessungsgeräten ist die Firma Leica Geosystems. Die Firma benutzt unter der Bezeichnung GSI-8 ein internes Datenformat mit zeilenweiser Speicherung von Codeinformationen und Koordinaten.

Anhand der Codeinformationen ist die Bedeutung der folgenden Daten interpretierbar, das Format ist aber nur den Experten transparent:

```
110003+02100001 81..00+02008446 82..00+01050616
83..00+00055191 71....+00000070
```

Aus einem Datenfile der oben beschriebenen Struktur soll daher eine XML-Datei entstehen. Jeder aufgemessene Punkt wird mit Nummer, Code, y,x,z in einem Element gespeichert. Die einfache DTD hat folgend Struktur:

```
<!DOCTYPE koordinatenfile [
<!ELEMENT koordinatenfile (punkt)+>
<!ELEMENT punkt (pnr, code, y, x, z)>
<!ELEMENT pnr (#PCDATA)>
<!ELEMENT code (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT z (#PCDATA)>
]>
```

Die XML Datei hat dann auszugsweise folgende Gestalt:

```
<?xml version="1.0"?>
<!DOCTYPE koordinatenfile SYSTEM  "koordinaten.dtd">
<koordinatenfile>
  <punkt>
    <pnr>    1</pnr>
    <code> 21</code>
    <y>  2008.446</y>
```

```
    <x>  1050.616</x>
    <z>    55.191</z>
  </punkt>
</koordinatenfile>
```

Der fleißige Landmesser kann an einem Tag mit elektronischen Totalstationen sehr viele Punkte einmessen. Handarbeit ist bei der Umsetzung der Punkte nicht möglich. Die Umformung des GSI-8-codierten Formats in eine XML-Datei nehmen wir mit einem kleinen C-Programm vor. An dieser Stelle ein kurzer Abriss über den Ablauf des Programms, ohne zu sehr in die Details zu gehen. Die Referenz des Eingabefiles Leica-Koordinaten wird über die Tastatur eingegeben. Das File wird einmal zur Ermittlung der Punktanzahl und Extremwerte geparst.

Prolog und DTD werden in die XML-Datei geschrieben, im zweiten Durchgang werden die Elemente `<punkt>` mit den Unterelementen, nach Ermittlung der Inhalte aus der Quelldatei, in die Zieldatei geschrieben. Programmintern werden die Funktionen `parseGSIKor`, `iConvert`, `dConvert` und `writeContentXML` benutzt. Die ausführbare Datei können Sie von der Website *www.programmieren.de* aus aufrufen. Den Quellcode können Sie aus dem Leserbereich downloaden.

Nach Erzeugung des XML-Dokuments wird jetzt nur noch eine XSL-Stilvorlage benötigt, um die Koordinaten lesbar und sortiert vom Browser anzeigen zu lassen. Die CSS-Anweisungen wurden in der nachfolgend dargestellten XSL-Datei aus Platzgründen entfernt. Es wird, wie auch schon beim Literaturverzeichnis, eine Tabellenvorlage beschrieben. Mittels einer `for-each`-Struktur werden für jeden Punkt die Werte in die Tabellenzellen platziert.

Exemplarisch ist eine Sortierung nach aufsteigenden Y-Werten mit `xsl:sort` vorgenommen. Nun kann man im Browser durch das lesbare Koordinatenfile scrollen oder eine der oben beschriebenen XSL-Transformationen durchführen und unmittelbar ein HTML-File erzeugen.

```
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/
     Transform">
<xsl:template match="/">
<style>
<![CDATA[
 ... hier stehen die CSS-Anweisungen
  }
]]>
```

```
</style>
<div align="center">
<table cellspacing ="3" cellpadding ="3"
       bgcolor="#ffffff">
<tr>
  <td  colspan="5" class = "titel"
       style="color : #FFFFCE;
       background-color : Maroon;">
       Koordinatenverzeichnis</td></tr>
<tr><td class = "pnr"><p>PktNr</p></td>
   <td class = "code"><p>Code</p></td>
   <td class="y"><p>Y</p></td>
   <td class="x"><p>X</p></td>
   <td class="z"><p>Z</p></td></tr>
   <xsl:for-each
      select="koordinatenfile/punkt">
<!-- Pktfolge aufsteigend nach y sortieren -->
   <xsl:sort select="y"
   case-order="lower-first" order="ascending"/>
 <tr><td class = "pnr">
    <xsl:value-of select="pnr"/></td>
 <td class = "code">
    <xsl:value-of select="code"/></td>
 <td class="y">
    <xsl:value-of select="y"/></td>
 <td class="x">
   <xsl:value-of select="x"/></td>
 <td class="z">
   <xsl:value-of select="z"/></td></tr>
 </xsl:for-each>
</table>
</div>
</xsl:template></xsl:stylesheet>
```

Listing 3.6: XSL-Datei zur sortierten Listenausgabe der Koordinaten

Nachdem wir nun die Möglichkeiten der Datenstrukturierung mit XML kennengelernt und mit XSL die Daten in ein HTML-Format gebracht haben, soll die weitere Anwendung der Daten mit einer auf XML basierten Sprache erfolgen. Wir wollen die Quelldaten in ein *Scalable Vector Graphics* (SVG) Format konvertieren und eine grafisch maßstäbliche Darstellung des Punktauftrags im Web anbieten.

Abb. 3.6: Die Transformierte GSI-8-Datei in der XML-Struktur

Auf die Wiedergabe des C-Programms zur Konvertierung wird hier verzichtet und wiederum auf die Website zum Buch verwiesen. In Auszügen hat die Konvertierung in eine SVG-Datei folgendes Ergebnis:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type = "text/css"
      href="punktauftrag.css"?>
<svg width="600.0" height="600.0"
     viewBox="2015.0 -1008.0 10.0 11.0"
     preserveAspectRatio="xMidYMid meet">
  <title>Kartierung</title>
  <desc>Punktauftrag</desc>
  <defs>
  <g id="pkt"><desc>Punktsymbol</desc>
    <circle class="point"
            cx="0" cy="0" r=".05"/>
  </g></defs>
<! -- Punktsymbol und Punktnummer
<use x="2023.172" y="-1000.442"
     xlink:href="#pkt"/>
     <text class="pnr"
      x=" 2023.272" y=" -1000.542">1</text>
<! -- hierfolgen weitere Punkte -->
```

```
<! -- Gestaltung des Blattes, Koordinaten, Gitternetz, Be-
schriftung -->
<! - vollstaendiger Code im Web -->
</svg>
```

Listing 3.8: Konvertierung von GSI-8 in eine SVG-Datei

Die SVG-Datei wird hier noch nicht weiter diskutiert, Sie können aber bereits jetzt die XML-konforme Notation erkennen. Mit Kenntnis des Kapitels 6 wird die Angelegenheit transparent. SVG ist ein starkes Webwerkzeug für kartographische Anwendungen und gilt als Standardformat für interaktive 2D-Grafiken im Web.

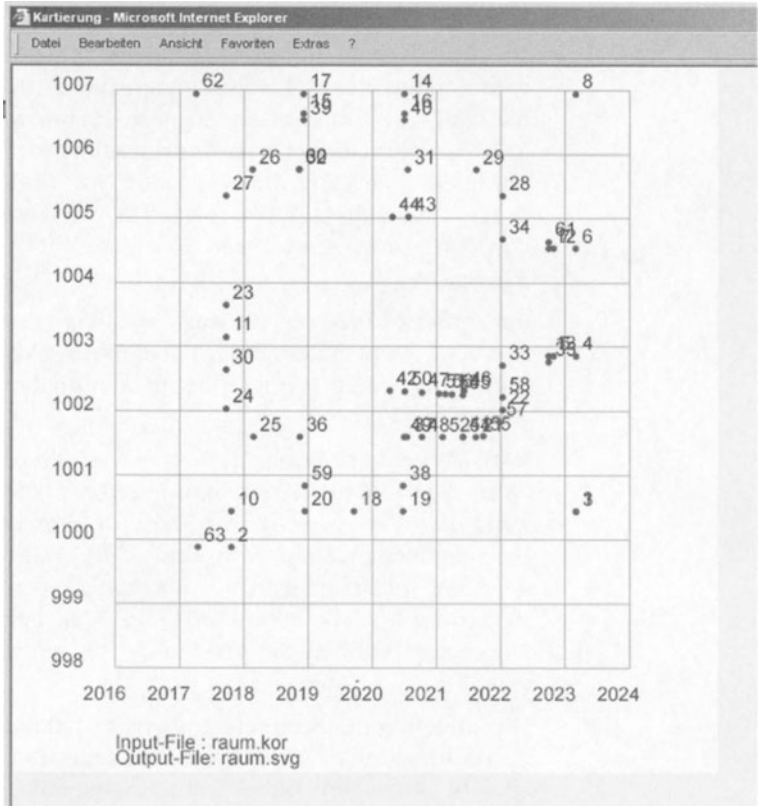


Abb. 3.7: Maßstäbliche Grafik (Punktauftrag) der GSI-8-Koordinaten als SVG-Datei

3.8 Zusammenfassung

Extensible Markup Language (XML) ist die Spezifikation einer Metasprache. Mit XML können Sie eigene Anwendungen ableiten und Daten geeignet strukturieren. Damit aus einem wohlgeformten XML-Dokument ein gültiges Dokument wird, benötigen Sie eine *Document Type Definition* (DTD), in der die Regeln für den Aufbau der XML-Datei festgelegt sind. Zur Bearbeitung von XML wird die *Extensible Stylesheet Language* (XSL) herangezogen. XSL beinhaltet ein Modell zum Zugriff auf Elemente des XML-Strukturbaums und hält Ablaufstrukturen ähnlich einer Programmiersprache bereit. In diesem Zusammenhang wurde auch XPATH benutzt und erwähnt.

Die Sprachelemente wurden anhand einer exemplarischen Anwendung erläutert. Das Zusammenspiel zwischen HTML und CSS mit XML zur formatierten Ausgabe im Browser wurde erläutert. XSL-Transformationen zur Konvertierung von XML/XSL in HTML erfolgten clientseitig, browserintern, mit online-Werkzeugen und durch Unterstützung von Java. Die serverseitige Konvertierung von XML-Daten und deren Übergabe an Webapplikationen ist gängige Praxis.

Ein Anwendungsbeispiel aus dem Vermessungswesen benutzt den XML-Webgrafik-Standard *Scalable Vector Graphics* (SVG) zur Kartierung einer topographischen Aufnahme mit interaktivem Zugriff im Web.

XML ist nicht zur Publikation von Webseiten konzipiert. Mit XML werden Datenbestände strukturiert verwaltet. Mächtige Unterstützung zur Verarbeitung von XML-Dokumenten bieten u.a. die Programmiersprachen Java und PHP. Das Potential von XML wird bei serverseitiger Dynamik voll ausgeschöpft. Mit der Formulierung von Datenbeständen in XML liegen Sie richtig, Ihre Homepage können Sie aber auch mit einfacheren Werkzeugen erstellen.

Die angeführten Beispiele sollten im Rahmen dieses einführenden Kapitels über XML etwas von den Möglichkeiten der Metasprache aufzeigen. Dabei war es notwendig, bereits auf noch zu behandelnde Themen wie Java und SVG einzugehen.

4 Programmieren von Java-Applets

Was ist Java? Die Programmiersprache Java wurde Mitte der neunziger Jahre bei SUN entwickelt. Die wichtigsten Merkmale der Sprache sind:

- objektorientiert
- plattformunabhängig
- internetfähig

Konventionelle Programmiersprachen orientierten sich an einem funktionalen Konzept, getrennt von der Datenstruktur, mit eindeutigem Kontrollfluss. Objektorientierte Programmiersprachen bilden Objekte durch Zustände und Verhalten ab, Daten und Methoden sind in Klassen abstrakt definiert. Aus Klassen werden durch Instanzbildung Objekte. Eigenschaften von Klassen können vererbt werden. Weitere Merkmale objektorientierter Programmiersprachen werden der Komplexität moderner Softwaresysteme gerecht. Darüber hinaus haben sich die Umgebungsbedingungen der Software durch heutige Betriebssysteme mit grafischen Benutzeroberflächen verändert. Programme werden durch Ereignisse, die von außen auf das Programm einwirken, wie etwa Benutzereingriffe durch Mausoperationen, gesteuert.

Die Plattformunabhängigkeit von Java wird durch Übersetzung des Quellcodes in einen Bytecode für eine virtuelle Maschine gewährleistet. Der Bytecode wird auf dem ausführenden System von einem Java-Interpreter geladen und ausgeführt. Somit ist ein einmal geschriebenes Java-Programm auf jeder Systemumgebung lauffähig.

Spezielle Bedeutung hat Java durch die Internetfähigkeit erlangt. Neben den Anwendungen können mit Java die sog. Applets erzeugt werden. Applets sind Java-Programme, die in einem Browser ablauffähig sind und Dynamik auf den statischen HTML-Seiten ermöglichen. Applets werden in Internetseiten mit HTML-Tags eingebunden. Aus Sicherheitsgründen unterliegen Applets Beschränkungen. Ein Zugriff auf das Dateisystem des Clientrechners ist nicht möglich.

Java gilt heute an den Hochschulen weltweit als die Standardprogrammiersprache. Die Mächtigkeit der Sprache liegt im Umfang der verfügbaren Klassenbibliotheken und der vielfältigen Applikationsunterstützung, seien es Datenbanken, Bildverarbeitung, 3D-Grafik, oder Netzwerke. Die Basissyntax ist mit schnellem sichtbaren Erfolg bei der Umsetzung eigener Programmierprobleme leicht erlernbar.

Programmieren lernen

Ein gutes Java-Lehrbuch für Einsteiger kommt mit einem Umfang von mindestens 500 Seiten daher. Wenn Sie beim Internetbuchversand Amazon nach Büchern zum Titel Java suchen, erhalten Sie etwa 750 Titel angezeigt. Die Thematik ist offenbar sehr umfangreich. Wo soll man anfangen? Wie soll man vorgehen?

In diesem Buch wird eine Einführung in die objektorientierte Programmierung auf wenige Seiten Text und die Analyse von ein paar Zeilen Sourcecode reduziert. Kann das erfolgreich sein? Zunächst ist festzustellen, dass beim Einstieg in eine Programmiersprache nicht sofort alle Details verständlich sein müssen. Es ist wichtig, einen Überblick zu bekommen. Dann ist auch bekannt, dass man Programmieren nur durch Programmieren lernt. Man braucht Willen, Beharrlichkeit und Geduld. Nur Lesen hilft nicht, vor Ihnen liegt ein Arbeitsbuch – oder besser neben Ihrer Tastatur.

Wir wollen sofort eigene Programme schreiben. Das sind zunächst kleinere Komponenten, die hilfreich bei der weiteren Programmierung sind. Unsere Anwendungen stellen Konzepte der Computergrafik vor, benutzen grafische Oberflächen und sind im Web verfügbar.

Die Strategie beim Erlernen einer Programmiersprache ist eigentlich immer gleich. Es wird die Entwicklungsumgebung installiert, die Programmstruktur und eine Befehlszeile in den Editor eingegeben, gespeichert, compiliert und die Anwendung gestartet. Hurra – es läuft. Sie wurden für die Mühe belohnt. Jetzt kann die Arbeit beginnen.

4.1 Installation einer Entwicklungsumgebung

Java ist im Internet frei verfügbar. Sie benötigen das Java Software Developer Kit und einen Texteditor. Aber warum sollten wir es nicht etwas komfortabler gestalten? Integrierte Entwicklungsumgebungen sind im Internet ebenfalls ohne Kosten erhältlich. Besuchen Sie *www.java.sun.com* und installieren Sie das Java SDK auf Ihrem Rechnersystem. Achtung, die Datei ist sehr

groß, ca. 50 Mbyte. Wenn Sie bei der Installation die Voreinstellungen akzeptieren, finden Sie unter Windows ein Verzeichnis `c:\j2sdk1.4.0_02`, im Unterverzeichnis `bin` befindet sich der Java-Compiler *`javac.exe`* und der Java-Interpreter *`java.exe`*.

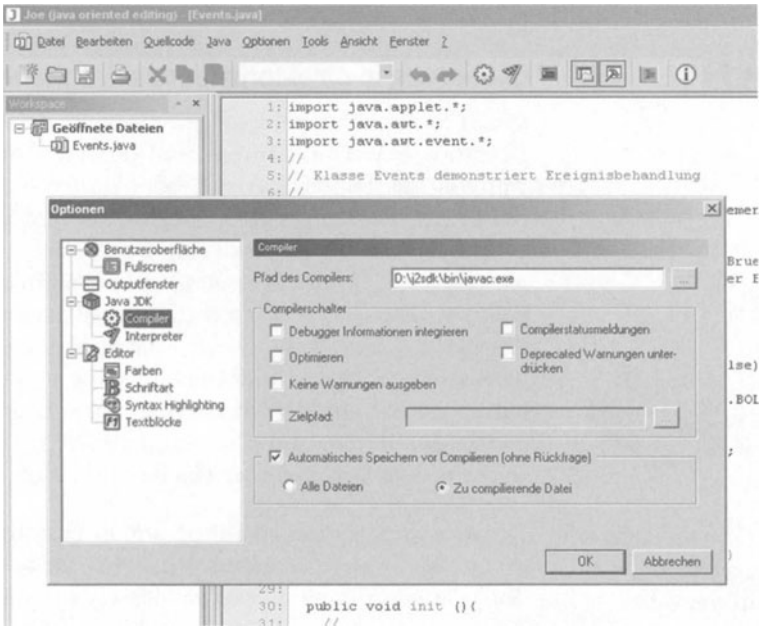


Abb. 4.1: Joe-Entwicklungsumgebung Optionen / Einstellungen Compiler bzw. Interpreter. Dort fügen Sie das Verzeichnis JSDK in den Suchpfad ein.

Der Entwicklungszyklus eines Programms beginnt immer mit Editieren und Kompilieren. War die Kompilation erfolgreich, ohne Fehlermeldungen, kann das Programm mit Java schon gestartet werden. Speichern Sie Ihr Programm, den Qellcode, in einer Datei mit dem Namen der Java-Klasse und der Erweiterungsbezeichnung `java`. Hiernach können Sie es kompilieren. Es ist aber notwendig, das Verzeichnis des Compilers im Systempfad einzutragen. Spätestens an dieser Stelle wird man nach einer Entwicklungsumgebung suchen, mit der die Arbeitsschritte automatisiert ablaufen. Joe, RealJ oder JCreator sind einige dieser nützlichen Helfer. Bei den Literaturangaben finden Sie die Webadressen zu den Werkzeugen.

Ein bewährtes und einfach zu handhabendes System ist Joe - *Java oriented Editing*. Dieser Editor mit deutschsprachigem Dialog steht unter *www.javaeditor.de* zum Download bereit.

Installieren Sie unter Windows zuerst Java, dann Joe (oder eine andere Entwicklungsumgebung) und schauen Sie bei den Optionen in der Entwicklungsumgebung nach, ob der Pfad zu Ihrer Java-Installation dort eingetragen ist, vgl. Abbildung. 4.1.Mit dem Befehl `Datei neu` kann es dann jetzt endlich losgehen.

4.2 Einführung in objektorientiertes Programmieren

Das Programmierproblem: Sie finden beim Aufräumen Ihres Desktops einen Goldbarren und möchten natürlich wissen, wie wertvoll das Fundstück ist. Leider haben Sie keine Waage aber ein Lineal. Sie kennen das spezifische Gewicht, für Gold unveränderlich – konstant, und den Goldpreis – der ist variabel. Sie messen den Goldbarren aus, erstellen ein Programm und berechnen über das Volumen die Masse und mit dem Tagespreis des Goldes den Wert. Die triviale funktional orientierte Lösung wird anschließend objektorientiert vorgestellt und als Java-Applet implementiert. In der Abbildung links erkennen Sie die Geometrie der Barren. Das Volumen des Pyramidenstumpfs ist aus dem Produkt von mittlerer Fläche und Höhe zu bestimmen.

Eine Java Klasse erzeugen und anwenden

Der Quelltext eines Java-Programms besteht aus Anweisungen, die in Methoden eingebunden sind. Methoden sind innerhalb von Klassen definiert. Das Schlüsselwort für Klassen ist `class`, der Klassenname beginnt üblicherweise mit einem Großbuchstaben. Die Klasse wird, wie auch die Methode, durch `{}` geschweifte Klammern begrenzt. Die Notation einer Methode akzeptieren wir zunächst ohne weitere Kommentierung. Eine Java-Anwendung wird immer mit der Methode `main()` gestartet. Hinter einem Methodennamen folgt ein Paar runder Klammern, diese schließen die Parameter der Methode ein. Achten Sie strikt auf die Einhaltung von Groß- und Kleinschreibung.

Nur zur besseren Lesbarkeit der gedruckten Quelltexte markieren wir Bezeichner für Klassen und Methoden fett. Kommentare innerhalb eines Programms ignoriert der Compiler, diese werden hier kursiv dargestellt. Einzeilige Kommentare werden mit einem doppelten Schrägstrich `//` eingeleitet. Längere Kommentare werden in `/* */` eingebettet.

Mit der Anweisung `import` werden existierende Methoden, die in Paketen zusammengefasst sind dem Programm zur Verfügung gestellt. Standardmäßig findet das Paket java.lang.* Anwendung.

```
import java.lang.* ;
class Goldbarren {
  public static void main(String[]args){
  }
}
```

Dieses Programm können Sie schon kompilieren und laufen lassen. Zuerst Speichern Sie die Datei unter *Goldbarren.java*. Bei der Entwicklungsumgebung JOE dann im Menüpunkt Java: Compilieren / Starten aufrufen. Es wird eine leere Konsole (Bildschirmfenster) angezeigt. Das ist korrekt. Bisher gab es ja noch keine Programmanweisungen.

Bei der weiteren Bearbeitung müssen zunächst die Daten bereitgestellt werden. Innerhalb von Programmen werden Daten in Variablen gehandhabt. Variablen sind Speicherplätze im Bereich des Hauptspeichers, in denen die Werte abgelegt sind. Der Typ einer Variablen legt die Größe des Speicherplatzes fest und bestimmt die Art der internen Interpretation. Jede Variable erhält vom Programmierer einen symbolischen Namen, den Bezeichner, unter dem die Variable innerhalb des Programms identifiziert wird. Nach der Deklaration der Variablen müssen die Werte initialisiert werden. Die Initialisierung erfolgt durch einen Zuweisungsoperator und die Auswertung des rechts vom Zuweisungsoperator stehenden Ausdrucks.

Datentypen, Deklaration von Variablen

Zwischen ganzzahligen Variablen `int` und Gleitkommazahlen `double` wird unterschieden. Diese Datentypen gehören zu den elementaren Variablen. Weitere Datentypen werden im Verlauf der Beispiele bei ihrem Auftreten erläutert.

```
class Goldbarren {
public static void main(String[]args){
// Deklaration der Variablen
int anzahl; // Anzahl der Goldbarren
// die Abmessungen:
double laengeUnten, breiteUnten,
       laengeOben, breiteOben,
       hoehe;
// das spez. Gewicht und der Preis:
 double dichteGold, preisGold;
// diese Werte muessen berechnet werden:
double masse, wert, mittlereFlaeche,volumen;
//...es folgt die Initialisierung der Variablen
  } // Ende Methode main
} // Ende Klasse
```

Wertzuweisung – Initialisierung von Variablen

Die Deklaration von Variablen erfolgt durch Angabe von Schlüsselworten `int` oder `double`, gefolgt von den durch Kommata getrennten Bezeichnern. Bei der Wahl des Bezeichners sollte man auf Sonderzeichen verzichten, mit einem kleinen Buchstaben anfangen und bei zusammengesetzten Worten mit einem Großbuchstaben fortfahren. Jede Anweisung wird durch ein Semikolon abgeschlossen. Kompilieren Sie diesen Programmteil. Hatten Sie Fehlermeldungen? Nein – dann geht es weiter mit der Wertzuweisung (Initialisierung) der Variablen.

```
laengeUnten =  9.0; // 9 cm ein wertvoller Fund
breiteUnten =  6.0; // die Abmessungen
laengeOben  =  7.0;
breiteOben  =  4.5;
hoehe       =  2.5;
dichteGold  = 19.3; // g / cm3
/*
   Feinunze Gold 374,90 US Dollar, der Tages-
   preis vom 24. Mai 2004
*/
preisGold   = 374.90;
/*
  1 Unze = 31.1035 g, Handelsgewicht fuer Gold
*/
unzeGramm    = 31.1035;
//
// ... es folgt der Berechnungsteil
```

Ausdrücke, Operanden, Operatoren – Zuweisungsoperator

Sie haben bemerkt, es gibt einen Dezimalpunkt. Das Komma wird ja bereits als Trennzeichen interpretiert. Die Einheiten Unzen und Gramm müssen umgerechnet werden. Nach Wertzuweisung können die Variablen in Ausdrücken verwendet werden. Ausdrücke bestehen aus Variablen, die durch Operatoren verknüpft sind. Die Wertigkeit (Reihenfolge der Abarbeitung) der arithmetischen Operatoren entspricht dabei den mathematischen Regeln. Sie können komplexe Ausdrücke aber auch durch Setzen runder Klammern lesbarer gestalten. Die Auswertung auf der rechten Seite einer Anweisung wird per Zuweisungsoperator der links stehenden Variablen zugeordnet.

```
mittlereFlaeche =(laengeUnten*BreiteUnten +
                  laengeOben*breiteOben)/2.0;
volumen         = mittlereFlaeche*hoehe;
masse           = volumen*dichteGold;
wert            = masse/unzeGramm * preisGold;
```

Nach der Berechnung sind die Variablen auf dem Bildschirm anzuzeigen. Wir benutzen zur Anzeige die Methode System.out.prntln(). Als Argument erwartet die Methode einen Ausgabestring (Zeichenkette). Mehrere Zeichenketten können durch den Operator + verknüpft werden. In der Verknüpfung befinden sich sog. String-Literale (Konstante) und die Bezeichner der Variablen, die von der Ausgabemethode in ihren aktuellen Wert umgewandelt werden.

Ausgabe der Variableninhalte

```
// Ausgaben auf die Konsole:
System.out.println("Berechnung des Wertes eines
                    Goldbarrens");
System.out.println("Abmessungen  [cm] Laenge
                     unten " +laengeUnten);
System.out.println("Breite unten "
                     +breiteUnten);
System.out.println("Laenge oben  "
                     +laengeUnten);
System.out.println("Breite oben "
                     +laengeUnten);
System.out.println("Hoehe " +laengeUnten);
System.out.println("Volumen [cm3] " +volumen);
System.out.println("Dichte rho  [g/cm3]"
                     +dichteGold);
System.out.println("Masse [cm3] " +masse); Sys-
tem.out.println("Wert[US Dollar" +wert);
```

Typumwandlung

Bei der Ausgabe ist festzustellen, dass die angezeigten Nachkommastellen der Berechnungsergebnisse nicht sinnvoll sind. Es ist daher vor der Ausgabe noch eine Rundung vorzunehmen. Mit dem Befehl (int) wird eine Gleitkommazahl in eine ganze Zahl umgewandelt. Die Nachkommastellen werden abgeschnitten. Daher wird zunächst mit der gewünschten Anzahl der Nachkommastellen multipliziert. Die darauf folgende Teilung des ganzzahligen Wertes durch die Gleitkommazahl liefert den Gleitkommawert mit den gewünschten Nachkommastellen.

```
//Rundung der Berechnungsergebnisse vor Ausgabe
masse = (int)(masse*10) /10.0;
wert  = (int) (wert *100)/100.0;
```

Nach erfolgreicher Kompilation *javac Goldbarren.java* legt der Compiler eine Datei *Goldbarren.class* an. Diese Javaklasse wird mit dem Interpreter *java Goldbarren* aufgerufen. Die Entwicklungssysteme stellen Menüaufrufe oder Icons für diese Arbeitsschritte zur Verfügung. Das Programm startet mit der ersten Anweisung in der Main-Methode. Die Anweisungen werden nach-

einander (sequentiell) abgearbeitet. Beim Aufruf der Methode `System.out.prntln` erfolgt eine Verzweigung in die aufgerufene Methode. Diese befindet sich nicht in unserem Quellcode, wird aber von Java bereitgestellt. Innerhalb der Methode werden die übergebenen Argumente abgearbeitet, und es erfolgt die Rückkehr in das aufrufende Programm. Hier geht es dann mit der nächsten Programmanweisung weiter, bis das Ende der Main-Methode erreicht ist.

Das gesamte Programm *Goldbarren* hier noch einmal im Zusammenhang, aus Platzgründen aber ohne Kommentare:

```
public class Goldbarren {
 public static void main(String[]args){
  int anzahl;
  double laengeUnten, breiteUnten,
       laengeOben, breiteOben,hoehe;
  double dichteGold, preisGold;
  double mittlereFlaeche, volumen,
         masse, wert;
  double unzeGramm;
  laengeUnten  =  9.0;  breiteUnten  =  6.0;
  laengeOben   =  7.0;  breiteOben   =  4.5;
  hoehe        =  2.5; dichteGold   = 19.3;
  preisGold    = 374.90;
  unzeGramm    = 31.1035;
  mittlereFlaeche = ((laengeUnten*breiteUnten)
             +(laengeOben*breiteOben))/2.0;
  volumen      = mittlereFlaeche*hoehe;
  masse        = volumen*dichteGold;
  wert         = masse/unzeGramm * preisGold;
  System.out.println("Berechnung des Wertes
                          eines Goldbarrens");
  System.out.println("Abmessungen  [cm] Laenge
                    unten " +laengeUnten);
  System.out.println("Breite unten "
                    +breiteUnten);
  System.out.println("Laenge oben  "
                    +laengeUnten);
  System.out.println("Breite oben  "
                    +laengeUnten);
System.out.println("Hoehe" +hoehe);
volumen = (int) (volumen *10)/10.0
System.out.println("Volumen[cm3]"+volumen);  Sys-
tem.out.println("Dichte rho  [g/cm3]"
                    +dichteGold);
```

```
masse = (int) (masse *10)/10.0 ;
System.out.println("Masse[gramm " +masse);
wert = (int) (wert *100)/100.0
System.out.println("Wert [US Dollar]" +wert);
 }
}
```

Listing 4.1: Das erste Java-Programm - *Goldbarren.java*

Alles ist gut gelaufen bisher. Sie haben die Kaffeepause verdient. Im Anschluss wird das gleiche Problem objektorientiert programmiert. Sollten Ihnen die nachfolgenden Erklärungen nicht hinreichend erscheinen, dann bedenken Sie bitte, dass in den späteren Beispielen Wiederholungen und Ergänzungen folgen. Es wurde schon darauf hingewiesen: Programmieren lernt man nur durch Programmieren !

Klassen, Objekte und Methoden

Der Goldbarren soll zu einem Objekt der Klasse *Edelmetall* werden. Diese Klasse wird zunächst eigenständig, ohne ein real existierendes Objekt, betrachtet und formuliert die Anforderungen an die Eigenschaften (Daten) und das Verhalten (Methoden) des Objekts vom Typ Edelmetall.

Die Bezeichnung der Klasse ist *Edelmetall.* Diese Klasse enthält eine Reihe von Methoden, die jeweils Teilaufgaben übernehmen. Eine Methode wird allgemein beschrieben durch

- den Typ des Rückgabewertes
- den Namen der Methode
- der Parameterliste in runden Klammern
- und dem Methodenrumpf in geschweiften Klammern

Methoden dienen der Abkapselung und Wiederverwendung von Programmcode. Dabei haben die Variablen, die in der Methode deklariert sind, nur lokal Gültigkeit. Es muss demnach Verfahren geben, Inhalte von Variablen mit Methoden auszutauschen. Das erfolgt über die Parameterliste der Methode und den Argumenten beim Aufruf sowie über den Rückgabewert. Eine Methode vom Typ `double` gibt genau eine Gleitkommazahl an das rufende Programm zurück. Eine Methode vom Typ `void` gibt keinen Wert zurück. Methoden werden aufgerufen durch Angabe des Namens und der aktuellen Argumente in runden Klammern. Elementare Variable, die in der Parameterliste beim Programmaufruf angegeben sind, müssen vom Typ her und in der Reihenfolge mit den lokalen Parametern der Methode übereinstimmen. Die aktuellen

Werte der Variablen werden in der Methode benutzt. Änderungen an den Variablen in der Methode haben keinen Einfluss auf das rufende Programm. Nach Abarbeitung der Anweisungen einer Methode wird in das rufende Programm zurückgekehrt.

Instanzvariablen

Ein Programm kann Variablen nur in dem Block benutzen, in dem diese auch deklariert wurden. Man spricht von der Sichtbarkeit von Variablen. Wir benötigen in unserer Klasse eine Anzahl von Variablen, auf die von verschiedenen Stellen aus zugegriffen werden soll. Diese Variablen werden innerhalb der Klasse, aber außerhalb der Methoden vereinbart. Man bezeichnet diese Werte auch als Instanzvariablen.

Die gewählten Bezeichner entsprechen dem bereits bekannten Programm. Vor dem Umrechnungsfaktor `unzeGramm` ist noch das Schlüsselwort `final` gesetzt. Hiermit haben wir den Umrechnungsfaktor als Konstante definiert, damit kann der Wert nicht mehr verändert werden.

Konstruktor

Nach Festlegung der Instanzvariablen folgt eine Methode, die den gleichen Namen trägt wie die Klasse. Das ist der Konstruktor. Der Konstruktor wird immer bei der Instanzbildung eines Objekts aufgerufen. Die Instanzierung wird weiter unten erläutert. Innerhalb des Konstruktors werden Vorbesetzungen und die Zuordnungen der eingehenden Parameter an die Instanzvariablen vorgenommen. Weiter wird auch gleich die Methode zur Berechnung des Volumens aufgerufen. Es folgen nun die Methoden der Klasse:

- berechneVolumen ()
- berechneWert ()
- setPreis( )
- setDichte( )
- ausgabeCon ()

Die Bezeichner und Funktionalität dieser Methoden sind selbsterklärend.

```
class Edelmetall {
  // Instanzvariablen
  private double laengeUnten, breiteUnten,
                 laengeOben, breiteOben, hoehe;
  private double preis, rho, volumen, masse, wert;
  final double unzeGramm = 31.1035;
```

```
  Edelmetall (double lu, double bu,
         double lo, double bo, double h){
  // eine Funktion mit dem gleichen Namen der
  // Klasse ist der Konstruktor
  // Vorbesetzung:
  preis = 1.0;  rho = 1.0; masse = 0.0;
  wert  =  0.0; volumen = 0.0;
  laengeUnten = lu; breiteUnten = bu;
  laengeOben  = lo; breiteOben  = bo; hoehe = h;
  volumen     = berechneVolumen(laengeUnten,
                breiteUnten, laengeOben,
                breiteOben, hoehe);
}
double berechneVolumen (double lu, double bu,
            double lo, double bo, double h){
/* Volumenberechnung wird vom Konstruktor
   aufgerufen */
  volumen = (((lu*bu)+(lo*bo))/2.0)*h;
  return volumen;
}
double berechneWert (){
  // Berechne Masse und Wert
  masse = volumen * rho;
  wert = masse/unzeGramm * preis ;
  return wert;
}
void setPreis( double p){
  // Festlegung des Preises
  preis = p;
}
void setDichte( double r){
  // Definition der Dichte
  rho = r;
}
void ausgabeCon (){
 // Ausgaben auf die Konsole:
System.out.println("Abmessungen  [cm] Laenge
                    unten " +laengeUnten);
System.out.println("Breite unten " +breiteUnten);
System.out.println("Laenge oben  " +laengeUnten);
System.out.println("Breite oben "  +laengeUnten);
System.out.println("Hoehe " +hoehe);
volumen = (int) (volumen *10)/10.0;      Sys-
tem.out.println("Volumen[cm3]" +volumen);
masse = (int) (masse *10)/10.0   Sys-
tem.out.println("Masse[gramm" +masse);
```

```
System.out.println("Dichte rho[g/cm3]"+rho);
System.out.println("Preis Feinunze [US Dollar]" +preis);
wert = (int) (wert *100)/100.0;
System.out.println("Wert[US Dollar]" +wert);
 } // Ende Methode ausgabeCon
} // Ende Klasse Edelmetall
```

Listing 4.2: Java objektorientiert – Klasse *Edelmetall.java*

Nach der Kompilation von *Edelmetall.java* steht nun alles bereit zur Einführung von Objekten der Klasse Edelmetall. Unser Programm zur Berechnung des Wertes von Edelmetallbarren ist bei Benutzung der Klasse Edelmetall jetzt sehr kompakt.

```
class Gold {
 public static void main(String[]args){
  Edelmetall goldBarren = new Edelmetall(9.0,6.0,7.0,4.5,2.5);
  goldBarren.setDichte(19.3);
  goldBarren.setPreis (374.90);
  goldBarren.wert = goldBarren.berechneWert();
  System.out.println("Wertberechnung Goldbarren");
  goldBarren.ausgabeCon();
 } // Ende Methode main
} // Ende Klasse Gold
```

Listing 4.3 : Java objektorientiert – die Klasse Gold

new

Es wird zunächst eine Instanz der Klasse Edelmetall mit der Bezeichnung *goldBarren* gebildet. Unter Benutzung des Schlüsselwortes new wird der Konstruktor aufgerufen, daher muss die Parameterliste, wie oben beschrieben, in Reihenfolge und Typ der Daten konsistent sein.

```
Edelmetall goldBarren = new Edelmetall(9.0,6.0,7.0,4.5,2.5);
```

Unter *goldBarren* ist dem Computer jetzt eine Speicheradresse bekannt, die auf das Objekt zeigt. Das Objekt beinhaltet Daten und Programmcode.

Wenn nachfolgend Methoden auf das Objekt anzuwenden sind oder dessen Daten manipuliert werden sollen, dann verknüpft der Punktoperator das Objekt und die Methode bzw. die Instanzvariable. Mit

```
goldBarren.setDichte(19.3);
```

wird für dieses Objekt das spez. Gewicht festgesetzt. Die Methode *berechneWert* ist vom Typ double. Das Berechnungsergebnis wird über den Rückgabewert geliefert und der Instanzva-

riablen *wert* zugeordnet, wieder mit dem Objekt `goldBarren` durch Punktoperator verknüpft.

```
goldBarren.wert = goldBarren.berechneWert();
```

Eingangs dieses Abschnitts haben wir über Methoden gesagt, dass diese Code zur Mehrfachverwendung bereitstellen und der Abkapselung dienen. In diesem Zusammenhang wurde auch etwas über den Zugriff bzw. die Sichtbarkeit von Variablen ausgesagt. Daten sind sichtbar in den Blöcken, in denen die Deklaration stattfand. Wir benutzten `final`, um einen Wert als unveränderbar zu definieren, und in der Klasse *Edelmetall* auch `private`. Die Main-Methode der ersten Anwendung innerhalb der Klasse Gold war `public static`.

Sowohl für Daten wie auch für Methoden existieren die sog. Modifizierer, das sind Schlüsselworte, die den Zugriff reglementieren. Man unterscheidet dabei inneren und äußeren Zugriff auf Daten oder Methoden einer Klasse. Der Zugriff auf Elemente der Klasse aus einer Methode heraus, die dieser Klasse angehört, ist der Zugriff von innen. Diese sind generell gestattet. Von außen greift man über eine Instanz auf Elemente einer Klasse zu. Hierbei werden die Zugriffsrechte wie folgt verwaltet:

- `public` von allen Klassen aus verfügbar
- `protected` sichtbar auch in abgeleiteten Klassen und bei Paketzugehörigkeit
- `private` nur in der Klasse selbst

Wird kein Modifizierer genannt, gilt Standardsichtbarkeit. Die Elemente müssen zum gleichen Paket gehören.

Weiter unten im Abschnitt Einführung in die 3D-Grafik wird eine Klasse *Transform* benutzt. Deren Methoden sind als `static` deklariert. Statische Methoden einer Klasse können auch ohne Instanzbildung aufgerufen werden. Dieser direkte Aufruf ist sonst nicht möglich. Statische Methoden wiederum können aber auch nur auf statische Elemente angewandt werden. Vertiefen können wir die Problematik hier in allen Einzelheiten nicht, sollte sich der Compiler beschweren, so haben Sie Hinweise auf die Ursache.

Zusammen-Fassung objektorientierte Programmierung

Die Programme bzw. Klassen *Goldbarren* und *Gold* zeigen die Einführung in die Programmierung mit Java und beschreiben das objektorientierte Konzept durch Gegenüberstellung mit der funktional orientierten Methode. Der Java-Compiler generiert eine Binärdatei für eine virtuelle Maschine aus dem Quelltext. Interpretiert wird die Binärdatei vom Java-Interpreter während der Ausführung auf der real existierenden Maschine. Ein Java-Programm besteht aus Klassen.

Klassen beschreiben ein abstraktes Objekt mit Daten und Methoden, die auf das Objekt angewandt werden. Die Instanz einer Klasse wird mit dem Schlüsselwort `new` und Aufruf des Konstruktors gebildet. Instanzen sind Referenzen auf das konkrete Objekt.

Elementare Variablen werden deklariert und initialisiert. Variablen eines Objekts werden als Instanzvariablen bezeichnet. Es wurde die Sichtbarkeit von Variablen und der Datenaustausch mit Methoden erörtert. Die Begriffe Operanden, Operatoren, Ausdrücke und Anweisungen wurden benutzt. Wir haben den Zuweisungsoperator und arithmetische Operatoren kennen gelernt.

Bezeichner sind vom Programmierer vorgegebene Namen für Klassen, Methoden und Variablen. Groß- und Kleinschreibung ist signifikant. Klassen sollten mit einem Großbuchstaben beginnen, Variablen und Methoden mit einem kleinen Buchstaben. Bei zusammengesetzten Begriffen kann das neue Wort durch einen Großbuchstaben gekennzeichnet sein. Auf Sonderzeichen, z. B. Umlaute, sollten Sie verzichten.

Die sequentielle Abarbeitung der Programmanweisungen wurde nur durch Methodenaufrufe unterbrochen. Kontrollstrukturen wie bedingte Anweisungen und Schleifen wurden noch nicht abgehandelt. Wir stellen fest: Die Programme *Goldbarren* und *Gold* sind eigenständige Java-Applikationen – Anwendungen, die nur für den lokalen Betrieb geeignet sind. Für das Internet benötigen wir Java Applets.

4. 3 Java-Applets

Besonderheiten von Java-Applets

Mit dem jetzigen Wissen über Java können wir uns den Applets zuwenden. Applets sind keine eigenständigen Programme. Applets werden in Internetseiten eingebettet und aus dem Browser heraus gestartet. Der Browser reserviert für das Applet einen eigenen Bereich, in dem grafische Ausgaben möglich sind. Aus

Sicherheitsgründen können Applets u.a. nicht auf das Dateisystem zugreifen. Das Schreiben und Lesen von Dateien ist nicht möglich. In der Entwicklungsphase können Applets auch mit einem Applet-Viewer gestartet werden.

Im Unterschied zu einer Java-Anwendung findet bei einem Applet der Einstieg nicht in der Main-Methode statt. Für Applets existieren die Methoden `init()` und `paint()`. Die Init-Methode wird einmalig bei Start des Applets ausgeführt, die Paint-Methode wird automatisch immer dann aufgerufen, wenn das Applet-Fenster neu zu zeichnen ist. Das ist die erstmalige Anzeige, ggf. bei Änderung der Fenstergröße oder wenn das Fenster wieder in den Vordergrund kommt.

Für ein Java-Applet werden die Packages *java.applet* und *java.awt* benötigt. Das AWT *Advanced Windowing Toolkit* stellt Klassen für die Oberflächengestaltung und grafische Darstellungen bereit. *Java.applet* ist die Basisklasse für Applets, von der unsere Anwendung erbt, d.h. die im Applet verfügbaren Klassen sind auch in der abgeleiteten Klasse enthalten. Die Klasse wird mit den Import-Anweisungen wie folgt formuliert:

```
import java.lang.*;
import java.applet.*;
import java.awt.*;
public class GoldApplet extends java.applet.Applet {
//... hier folgen die Anweisungen
}
```

Innerhalb der Klasse *GoldApplet* werden die Methoden `init()` und `paint()` zur Vorbereitung und Ausführung des Applets benutzt. Die Instanz `txtFont` der Klasse `Font` definiert einen Zeichensatz, der mit `setFont` als der aktuell zu benutzende definiert ist. `SetBackground` legt die Hintergrundfarbe für das Applet fest. Die Methode `init()` wird einmalig bei Start des Applets aufgerufen.

```
public void init(){
  Font txtFont = new Font ("Courier", Font.BOLD, 12);
  setFont(txtFont);
  setBackground(Color.lightGray);
}
public void paint(Graphics pen){
  pen.setColor(Color.blue);
  Edelmetall goldBarren =
        new Edelmetall(9.0,6.0,7.0,4.5,2.5);
goldBarren.setDichte(19.3);
```

```
goldBarren.setPreis (374.90);
goldBarren.wert = goldBarren.berechneWert();
pen.drawString ("Wertberechnung Goldbarren",10,10);
goldBarren.ausgabeCon(pen,30,30 );
 }// Ende paint
```

Automatisch erfolgt auch der Aufruf der Methode `paint()`. `Paint` enthält in der Parameterliste einen Parameter der Klasse `Graphics`. Dieses Objekt stellt den Grafik-Kontext her, ist sozusagen die Zeichenfläche. Auf dieses Objekt werden die elementaren Grafikmethoden angewandt. In unserem Beispiel `setColor()` und `drawString()`. An die Stelle von `System.out.println` tritt nun `drawString`. Der erste Parameter in der Liste ist die auszugebende Zeichenkette. Dahinter folgen die Angaben für die Position auf dem Bildschirm x, y. Angewandt auf das Graphics-Objekt, hier mit *pen* bezeichnet, ist wieder der Punktoperator von Bedeutung, der Objekt und Methode verbindet. Änderungen an der Klasse *Edelmetall* sind nicht vorzunehmen.

Im Entwicklungsstadium werden Java-Applets mit einem Appletviewer gestartet. Man benötigt dafür eine HTML-Datei, in der das Applet eingebettet wird. Die einfache Lösung ist der Applet-Tag, hier mit Angabe der notwendigen Attribute. Unter `code` wird die Klasse angegeben, die sich in diesem Fall im gleichen Verzeichnis mit der HTML-Datei befinden muss. Hiervon abweichende Verzeichnisse werden mit dem Attribut `codebase` angegeben. `width` und `height` legen die Größe des Appletbereichs fest. Zunächst ist diese minimale Einbindung hinreichend. Weiter unten werden wir eine anspruchsvollere Lösung behandeln.

```
<html>
  <head>
    <title>Goldbarren</title>
  </head>
  <body>
    <applet
       code="GoldApplet.class"
       width="400" height="400">
    </applet>
  </body>
</html>
```

Zusammenfassung Applets

Applets sind Java-Programme, die in Internetseiten eingebettet sind und aus dem Browser heraus gestartet werden. Aus Sicherheitsgründen haben Applets keinen Zugriff auf das Dateisystem des Clientrechners. Die notwendigen Klassen für ein Applet sind

in den Packages *java.applet* und *java.awt* enthalten. Die eigene Anwendung erbt von der Basisklasse und wird durch `extends java.applet.Applet` zum Applet. Die Methode `main()` einer Applikation wird ersetzt. Der Einstieg in ein Applet erfolgt durch den automatischen Aufruf der Methoden `init()` und `paint()`. Der Grafikkontext wird durch ein Objekt der Klasse Graphics bereitgestellt.

Hauptanwendungen für Java-Applets sind dynamische Ausgaben auf der Webseite wie Menüs, Animationen, MultiMedia-Anwendungen u.a. Sehr häufig werden auch physikalische und mathematische Zusammenhänge mit Java-Applets kommuniziert.

4.4 Computergrafik mit dem Java-AWT

Koordinatensysteme

Koordinatensysteme haben in der Computergrafik eine zentrale Bedeutung. Ein Benutzerkoordinatensystem stellt die Referenz für die Konstruktion der Geometrieelemente aus der Sicht des Benutzers dar. In diesem üblicherweise dreidimensionalen System formuliert der Anwender sein Modell. Das sind Bauteile oder Werkstücke, die durch Änderung der Abmessungen, Drehung und Positionierung zu einem Ganzen zusammengefügt werden. Beispielsweise entwirft ein Architekt den Grundriss eines Hauses, ein Maschinenbauer ein Drehteil oder ein Verpackungstechniker eine Faltschachtel. Diese Anwender denken in verschiedenen Maßstabsbereichen und richten ihre Systeme unterschiedlich aus. Ein umfangreiches Softwaresystem unterstützt die Modellierung und die grafische Ausgabe auf ein zweidimensionales Medium, Bildschirm oder Printer, in unterschiedlichen Ansichten und Darstellungsvarianten wie Strichzeichnung oder flächenhafte Wiedergabe. Ein ausgesprochen komplexer Vorgang, der im vorliegenden Kapitel einführend zu behandeln ist.

Bei der einfachen Grafikprogrammierung kommt man schon mit wenigen Programmanweisungen zu sichtbaren Erfolgen. Weiter unten wird ein Applet vorgestellt, mit dem ein Bauteil dreidimensional aus beliebiger Richtung zu betrachten ist – dort wird es dann schon etwas aufwändiger.

Grafiken bestehen aus elementaren Grundformen wie Punkte, Linien, Rechtecke und Ellipsen. Diese sind durch Koordinatenreferenzen und Abmessungen vorzugeben. Der virtuelle Zeichenstift verbindet die Grafiken entsprechend der Anweisungen des Grafikprogramms.

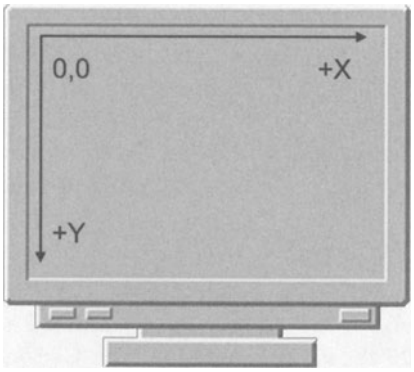


Abb. 4.2: Orientierung des Bildschirmkoordinatensystems

Für die weiteren Betrachtungen definieren wir die Referenzpunkte als x- und y-Werte im Koordinatensystem des Bildschirms. Das ist zwar wenig flexibel, reduziert die Angelegenheit aber zunächst auf die Ansteuerung des Grafikmediums.

Der Ursprung des Bildschirmkoordinatensytems liegt in der linken oberen Ecke. Die positive x-Achse zeigt nach rechts. Die Orientierung der y-Achse ist positiv nach unten, vgl. Abbildung 4.2.

Mit dem virtuellen Zeichenstift anzusteuernde Positionen sind ausschließlich ganzzahlig und demnach als Integer-Variablen anzugeben. Nach der Bereitstellung einer Zeichenfläche werden auf das Objekt Methoden zur Ausgabe von Grafiken angewandt. Grafikmethoden werden durch das *Advanced Windowing Toolkit* (AWT) bereitgestellt.

Anwendung elementarer Zeichenbefehle

Das Paket AWT enthält eine Klasse Graphics, diese beinhaltet Methoden zum Zeichnen, die auf ein Objekt der Klasse anzuwenden sind. Ein Objekt der Klasse Graphics kann als Zeichenfläche angesehen werden. Die Farbe ist eine Eigenschaft dieses Objekts, `drawLine()` eine Methode, die zwei Punkte durch eine gerade Linie verbindet.

Die Methode `paint()` eines Applets wird bei Bedarf automatisch aufgerufen. Das erfolgt, sofern ein Fenster in der Größe verändert wird oder wieder in den Vordergrund zu stellen ist und natürlich, wenn der Browser die Applet-Information erhält. Aus diesem Grund werden die Aufrufe der Grafikmethoden über diese Methode gesteuert. Als Parameter erhält die Methode ein Objekt der Klasse Graphics, im Beispiel mit *pen* bezeichnet. Die

minimale Anforderung zum Zugriff auf das Grafiksystem wäre demnach folgende:

```
import java.applet.*;
import java.awt.*;
public class Grafik_1
              extends java.applet.Applet {
  public void paint (Graphics pen){
    pen.drawLine(0,0,1024,768);
  }
}
```

Mit `pen.drawLine()` wird eine Diagonale von oben links nach unten rechts gezeichnet.

Wir werden aber sofort etwas anspruchsvoller. Linienzüge werden mit der Methode `drawPolygon()` gezeichnet. Der Zeichenstift kann auch an andere Methoden übergeben werden. In der Klasse *Grafik_2* stellt `paint()` die Zeichenfläche zur Verfügung und definiert die anzusteuernden Koordinaten. Durch Aufruf der Methode `zeichnePolygon()` werden das Graphics-Objekt, die Koordinaten und die Anzahl der zu zeichnenden Punkte übergeben. Lokal sind die Koordinaten unter den Bezeichnern `u, v` gültig. Der Grafikkontext ist dort *c*.

```
import java.applet.*;
import java.awt.*;
public  class Grafik_2 extends java.applet.Applet {
  public void paint (Graphics pen){
    int x [] = new int [4];
// Initialisierung der anzusteuernden Punkte
    int y [] = new int [4];
    int n    = 4;
    x[0] =  0; y[0]=  0;
    x[1] =  0; y[1]= 25;
    x[2] = 25; y[2]= 25;
    x[3] = 25; y[3]=  0;
    setBackground(Color.white);
    pen.setColor(Color.black);
    zeichnePolygon(pen,  x,  y, n);
  }
  void zeichnePolygon( Graphics c, int u[], int v[], int n){
    c.fillPolygon(u,v,n);
  }
}
```

Listing 4.4 : Grafik mit Java-Applets – Einführung von Arrays

Arrays

An dieser Stelle ist es nun notwendig, über Arrays zu sprechen. Arrays (oder Felder) bieten die Möglichkeit, unter einem Namen auf viele Elemente über den Index zuzugreifen. Arrays sind in Java Objekte und werden durch Angabe des Typs, den Feldnamen und eckige Klammern deklariert. Die Erzeugung der Array-Variablen selbst erfolgt durch Instanzenbildung mit dem Schlüsselwort `new`, dort wird dann auch die Größe des Arrays angegeben. Auf die Elemente eines Arrays wird über den ganzzahligen Index zugegriffen. Die Zählung beginnt immer bei null. Im Beispiel deklarieren wir ein Array für die x-Koordinaten und ein Array für die y-Koordinaten. Den Feldelemente werden Werte zugewiesen.

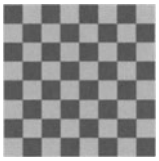
Das Array wird an Methoden weitergereicht, indem nur der Name als Argument übergeben wird. In der Parameterliste der Methode wird das Array wie bei der Deklaration notiert. Obwohl lokal die Bezeichner mit `u` und `v` von den Argumenten des Aufrufs abweichen, wird in der Methode auf die Originalspeicherplätze zugegriffen, da die Übergabe als Referenz erfolgt. Arrays stellen eine notwendige Datenstruktur dar, die bei den späteren Grafikanwendungen noch ausgiebig zu verwenden ist. Ein Arrayelement kann selbst wieder Arrays enthalten. Hierdurch können mehrdimensionale Arrays definiert werden, z. B. eine Matrix, ein rechteckiges Schema aus Zeilen und Spalten. Wir wenden Matrizenoperationen im späteren Abschnitt bei der Einführung in die 3D-Grafik an.

Das Beispiel *Grafik_2* zeichnet ein Quadrat von 10 x 10 Pixel in roter Farbe. Da könnte doch eigentlich noch ein Schachbrett daraus werden. Unser Crashkurs der Programmierung muss sich daher erst einmal mit der Ablaufsteuerung auseinandersetzen. Neben der sequentiellen Abarbeitung von Programmanweisungen und den Methodenaufrufen existieren in den Programmiersprachen Kontrollstrukturen zur Steuerung der Reihenfolge abzuarbeitender Programmanweisungen. Das sind u.a. die bedingten Anweisungen und Schleifen.

For-Loops

Im engen Zusammenhang mit Arrays und der Indizierung sind Schleifen zu nennen, von denen es unterschiedliche Typen gibt. Sofern der Programmierer weiß, wie oft genau Anweisungen zu wiederholen sind, ist eine Zählschleife das probate Mittel. Das Schlüsselwort für eine Zählschleife ist `for`. Im Kopf der Zählschleife wird der Schleifenindex, eine ganzzahlige Variable, initialisiert. Es folgt eine Laufbedingung und ein Kriterium zur Erhöhung des Schleifenzählers. Die Anweisungen folgen auf das

Schlüsselwort for in runden Klammern und sind durch Semikolon terminiert. Der Schleifenkörper wird durch die Anweisungen gebildet, die in den geschweiften Klammern folgen. Schleifen können geschachtelt werden. Die Reihenfolge der Abarbeitung findet von innen nach außen statt. Anhand der Grafik eines Schachbretts kann man die Funktion von Zählschleifen geeignet darstellen.



Ein Schachbrett besteht aus 64 Feldern, jeweils 8 Zeilen und 8 Spalten mit alternierenden Farben. Betrachten wir zunächst nur eine Zeile des Schachbretts. Zum Zeichnen eines Feldes benutzen wir die eigene Methode zeichneQuadrat(). Die x-Position des Quadrates würde sich von Feld zu Feld jeweils um die Feldbreite ändern. Wir beginnen unsere Schleife mit dem Schleifenindex j = 0; die Laufbedingung für die Schleife gilt, wenn j< 8 ist. Nach Schleifendurchlauf wird j um 1 erhöht (Benutzung des Inkrementoperators j++). Vor Neubeginn der Schleife erfolgt dann zunächst wieder die Abfrage der Laufbedingung. Im Schleifenkörper befinden sich der Methodenaufruf zeichneQuadrat() und die Veränderung der x-Position des Quadrats um die Seitenlaenge.

```
seitenLaenge = 25; x = 0;
pen.setColor(Color.black);
for (int j = 0; j<8; j++){
  zeichneQuadrat(pen, x, seitenLaenge);
  x+= seitenLaenge;
}
```

Bedingte Anweisungen

Es gibt in der Zeile des Schachbretts noch ein Problem. Alle Felder werden mit gleicher Farbe gezeichnet, sind also schwarz. Wir brauchen aber eine alternierende Darstellung mit wechselnden Farben. Hierzu führen wir eine Variable farbe ein. Mit dem Modulo-Operator kann man den ganzzahligen Rest einer Zahl nach Abzug der ganzen Vielfachen einer anderen bestimmen, farbe % 2 liefert 0 bei geraden Zahlen und 1 bei ungeraden Zahlen. Mit dem Schlüsselwort if führen wir eine weitere Anweisung zur Ablaufsteuerung ein – die bedingte Anweisung. Der Ausdruck innerhalb der runden Klammern von if wird ausgewertet. Liefert diese Bedingung den Wert wahr (true) dann erfolgt die Ausführung der folgenden Anweisung oder des Anweisungsblocks. Ist die Bedingung falsch (false), wird die nächste Anweisung bzw. der Anweisungsblock nicht ausgeführt.

```
seitenLaenge = 25; farbe = 0; x = 0; y=0;
```

```
pen.setColor (Color.black);
for (int j = 0; j<8; j++){      // Spaltenindex
  if ( farbe % 2 == 1){
   pen.setColor(Color.white);  // Farbe weiß
  }
  zeichneQuadrat(pen, x, seitenLaenge);
  x+= seitenLaenge; // ein Feld weiter
  pen.setColor (Color.black); // Farbe schwarz
}
```

Für eine Zeile ist das Verfahren jetzt brauchbar. Die weiteren Zeilen werden über eine zusätzliche äußere Schleife gesteuert, in dieser Schleife ist y variabel. Als Index wird k benutzt. Wir müssen noch beachten, dass nach Abarbeitung einer Zeile die x-Koordinaten wieder auf den Anfangswert zu setzen sind. Die Abfrage für die alternierende Zeichenfarbe wird über die Summe von i und j geführt.

```
seitenLaenge = 25; farbe = 0; x = 0; y= 0;
pen.setColor (Color.black);
for (int i = 0; i < 8; i++){ //Zeilenindex
  for (int j = 0; j<8; j++){ // Spaltenindex
    // ... innere Schleife s.oben
  }
  x=0; // Spalte beginnt wieder vorne
  y+= seitenLaenge; // eine Zeile weiter
  farbe++; // Farbwechsel nicht vergessen
}
```

Nachfolgend das vollständige Schachbrett-Programm im Zusammenhang und mit der eigenen Methode `zeichneQuadrat()`. Ein Tipp: Verändern Sie im Quellcode die Größen und die Anzahl der Felder, machen Sie sich mit der Funktion von For-Schleifen vertraut. Erinnern Sie sich noch an die Einführung zum Kapitel? Programmieren lernen - man braucht Ausdauer, Willen und Geduld.

```
import java.applet.*; import java.awt.*;
public  class Schachbrett extends
                          java.applet.Applet {
  public void paint (Graphics pen){
    int seitenLaenge, x,y,farbe;
    seitenLaenge = 25; farbe = 0; x = 0; y= 0;
    pen.setColor (Color.black);
    for (int i = 0; i < 8; i++){ //Zeilenindex
      for (int j = 0; j<8; j++){ // Spalteninx.
        if ( farbe % 2 == 1){
```

```
        pen.setColor(Color.lightGray);
        }
      zeichneQuadrat(pen, x,y, seitenLaenge);
      x+= seitenLaenge; // ein Feld weiter
      pen.setColor (Color.black);
      farbe++;
      }
    farbe++; //Farbwechsel nicht vergessen
    x=0; // Spalte beginnt wieder vorne
    y+= seitenLaenge; // eine Zeile weiter
    }
  } // ende paint
  void zeichneQuadrat( Graphics c, int pos_x,
                       int pos_y, int a){
    int u [] = new int [4];   // Eckpunkte
    int v [] = new int [4]; int n    = 4;
    u[0] = 0; v[0]= 0; u[1] = 0; v[1]=  a;
    u[2] = a; v[2]= a; u[3] = a; v[3]=  0;
    for (int i = 0; i<n; i++){// Transformation
     u[i] = pos_x +u[i];
     v[i] = pos_y +v[i];
    }
    c.fillPolygon(u,v,n);
  } // Ende zeichneQuadrat
}// Ende class Schachbrett
```

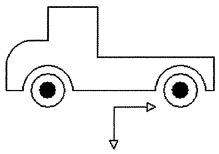
Listing 4.5: Bedingte Anweisungen und Schleifen, Grafikausgabe eines Schachbretts

Grafikelemente

Mit der folgenden Programmieraufgabe wird eine Teilaufgabe gelöst, die im weiteren Verlauf zu einem komplexeren Applet ausgebaut wird. Zunächst geht es nur darum, das Symbol eines Autos zu zeichnen. Dieses Auto soll beliebig auf der Zeichenfläche positioniert werden können. Wir benötigen einen Einfügepunkt, wie auch schon bei den Quadraten des Schachbretts gezeigt. Ferner ist vorzusehen, dass die Figur mit unterschiedlicher Größe zu zeichnen ist. Das Symbol ist also in einem lokalen System zu definieren. Die Abmessungen werden durch einen Skalierungsfaktor verändert, und die Position des Autos wird über die Angabe eines Einfügepunktes variabel gehalten. Wir erzeugen eine Klasse Auto, die ausschließlich aus der Methode `zeichne()` mit folgenden Parametern besteht:

```
Graphics c          das Graphics Objekt
Color cRaeder       die Zeichenfarbe Räder
Color cKarosse      die Zeichenfarbe für die
                    Karosserie
```

```
int scale          den Maßstabsfaktor
int x0, int y0     den Einfügepunkt
```



Innerhalb der Methode werden die Radien `r1, r2, r3` für Räder und Radkasten und der Achsabstand mit Konstanten zugewiesen und mit dem Maßstabsfaktor skaliert. Abgeleitet werden in Relation zu den Abmessungen und dem Einfügepunkt `x0,y0` die Mittelpunkte der Räder `xm1,ym1` und `xm2, ym2`. Alle Koordinaten für das Zeichnen der Linien werden immer durch Umspeichern den Variablen `x1,y1,x2,y2` zugewiesen. Hierdurch lautet generell der Aufruf zum Zeichnen einer Linie `c.drawLine (x1,y1,x2,y2);` In der Grafik links ist der Einfügepunkt `x0,y0` durch das Koordinatensymbol markiert.

```
import java.awt.* ;
public class Auto {
// Defaultkonstruktor
public void zeichne (Graphics c,
            Color cRaeder, Color cKarosse,
            int scale, int x0, int y0){
  int x1,y1,x2,y2;
  int xm1, ym1, xm2, ym2;
  int r1,r2,r3, achsabstand;
  achsabstand=8*scale;       // Sakalierung
  r1 = 2*scale; r2 = 4*scale; r3 = 6*scale;
  xm1 =  x0-achsabstand  ;    // linkes Rad
  ym1 =  y0-r2/2;
  c.setColor(cRaeder);
  c.drawArc(xm1-r2/2,ym1-r2/2,r2,r2,0,360);
  c.fillArc(xm1-r1/2,ym1-r1/2,r1,r1,0,360);
  xm2 =  x0+achsabstand ;    // rechtes Rad
  ym2 =  y0-r2/2; ;
  c.drawArc(xm2-r2/2,ym2-r2/2,r2,r2,0,360);
  c.fillArc(xm2-r1/2,ym2-r1/2,r1,r1,0,360);
  c.setColor(cKarosse);    // Fahrzeugkontur
  c.drawArc(xm1-r3/2,ym1-r3/2,r3,r3,0,180);
  c.drawArc(xm2-r3/2,ym2-r3/2,r3,r3,0,180);
  c.drawLine(xm1+r3/2,ym1,xm2-r3/2,ym1);
  x1 = xm2+r3/2; y1 = ym2;
  x2 = xm2+r2; y2 = ym2;
  c.drawLine(x1,y1,x2,y2); x1=x2; y2=y1-r2;
  c.drawLine(x1,y1,x2,y2); x1=x2; y1=y2;
    x2=x1-achsabstand*2+r2/2;
  c.drawLine(x1,y1,x2,y2); y1=y2-r3;x1=x2;
  c.drawLine(x1,y1,x2,y2); x2=x1-r3;y2=y1;
  c.drawLine(x1,y1,x2,y2); y2=y1+r2;x1=x2;
```

```
    c.drawLine(x1,y1,x2,y2);
      x1=x2; x2=x1-r2/2;y1=y2;
    c.drawLine(x1,y1,x2,y2); x1=x2-r3/2; y1=y2;
    c.drawArc(x1,y1,r3,r3,90,90);
      x2=x1; y1=y2+r3/2; y2= ym1;
    c.drawLine(x1,y1,x2,y2);
      x1=x2; y1=y2; x2=xm1-r3/2;
    c.drawLine(x1,y1,x2,y2);
  } // Ende Methode zeichne
  } // Ende Klasse Auto
```

Listing 4.6: Die Klasse Auto mit der Methode `zeichne()`

Die Methode `zeichne()` ist sicher leicht zu verifizieren. Lediglich die Methoden `drawArc()`, `drawOval()` bzw. `fillArc()`, `fillOval()` bedürfen der Erläuterung. Mit allen Funktionen wird ein Ellipsenbogen gezeichnet.

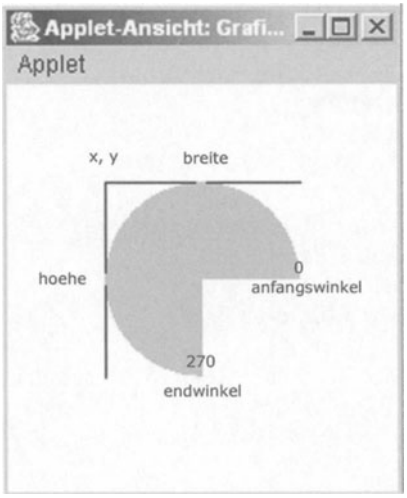


Abb. 4.3: Orientierung der Parameter bei der Zeichnung von Ellipsenbögen

Der Bezugspunkt x, y liegt in der oberen linken Ecke. Breite und Höhe bestimmen die Form. Sind die Werte gleich, entsteht ein Kreis bzw. Kreisbogen. Die Winkel sind mathematisch orientiert und werden in Grad gegen den Uhrzeigersinn angegeben. Der allgemeine Aufruf lautet:

```
c.fillArc (x,y,breite,hoehe,anfangswinkel,endwinkel);
```

vgl. auch Abbildung 4.3.

Es wird noch ein kleines Testprogramm mit dem Aufruf der Methode zeichne() benötigt. In der Methode init() werden die Vorbereitungen für das Applet getroffen: Es wird eine Instanz lkw der Klasse Auto angelegt, der Einfügepunkt wird initialisiert und der Zeichnungsmaßstab festgelegt. Innerhalb von paint() wird die Methode zeichneAuto() aufgerufen.

```
import java.applet.*;
import java.awt.*;
public class GrafikAuto
            extends java.applet.Applet {
//
//  Elementare Grafik - zeichnet ein Autosymbol
//  an Position x0, y0 im Maßstab scale
  Auto lkw;
  int x0,y0,scale;
  Font txt = new Font ("Courier",Font.BOLD,12);
  public void init(){
    lkw = new Auto();
    setBackground(Color.lightGray);
    setFont(txt);
    setSize(400,400);
    x0 = 175; y0=200; scale = 10;
  }
  public void paint(Graphics pen){
    Color cRot = Color.red;
    Color cBlau = Color.blue;
    pen.setColor(Color.black);
//
//  Methode zeichne der Klasse Auto auf das
//  Objekt lkw anwenden
    lkw.zeichne (pen,cRot,cBlau, scale,x0,y0);
  }
}
```

Listing 4.7: Klasse Grafik

Sie können die Dateien in der Reihenfolge *Auto*, *GrafikAuto*, getrennt compilieren. Der Compiler legt zwei Klassen an. Auto ist beliebig wiederverwendbar. Wovon wir bei den späteren Animationen Gebrauch machen.

Mit drawLine(), drawPolygon(), drawArc(), drawString() und setColor() wurden elementare Grafikfunktionen des AWT benutzt. Ersetzt man in den entsprechenden Aufrufen *draw* durch *fill*, dann wird die Form gefüllt dargestellt. Eine weitere

wichtige Funktion ist `drawImage()`, die ein Bitmap anzeigt. Auch diese Funktion werden wir bei den Animationen nutzen.

Zusammenfassung elementare Grafik mit dem AWT

Die grafische Ausgabe von Java-Applets erfolgt in einem vom Browser reservierten Bereich des Bildschirms. Die Gerätekoordinatenachsen sind von rechts nach links und von oben nach unten definiert. Mit dem automatischen Aufruf der Methode `paint()` wird der Grafikkontext hergestellt. Auf der Zeichenfläche werden mit elementaren Grafikmethoden Punkte, Linien, Rechtecke, Kreise und Text ausgegeben.

Das Schachbrett war ein geeignetes Objekt zur Darstellung der Kontrollstrukturen For-Schleife und If-Bedingung. Mit `paint()` wurde eine Zeichenfläche bereitgestellt, auf das Graphics-Objekt wurde die Methode `fillPolygon()` angewandt. Die Parameter für `fillPolygon()` sind u.a. Arrays mit dem Elementtyp `int`. Daher musste die Behandlung der Klasse Array hier eingefügt werden.

Mit der Zeichnung des Autos wurde die Möglichkeit der Positionierung und Maßstabsänderung aufgezeigt. Die Figur Auto hat einen Referenzpunkt, für den der Einfügepunkt in der Zeichnung anzugeben ist. Eine Instanz der Klasse `ZeichneAuto` wurde mit `lkw` bezeichnet. Dieses Objekt wird mit der Methode `lkw.zeichne()` grafisch dargestellt.

Sie sollten jetzt in der Lage sein, grafische Figuren in Applets darzustellen und dabei Konzepte der objektorientierten Programmierung anwenden können.

4.5 Animationen

Der Prozess zur Erzeugung bewegter Grafiken ist zunächst denkbar einfach. Man weist das Programm an, die Ausgabe einer Grafik vorzunehmen, und teilt anschließend Windows mit, die Grafik darzustellen. Nach einer kurzen Warteschleife wird der Vorgang mit veränderter Grafik wiederholt. Bevor diese jedoch angezeigt wird, ist der bisherige Bildschirminhalt zu löschen. Zur Erzielung einer flimmerfreien Animation gibt es aber auch einige Probleme zu lösen.

Alle Methoden, die zur Darstellung einer animierten Grafik benötigt werden, sind über das AWT verfügbar, die mit `import java.awt.*` eingebunden werden.

Das AWT stellt die Methoden paint() und repaint() sowie alle benötigten Grafikmethoden bereit. Wie oben schon erwähnt wird paint () automatisch bei Programmstart durch das Laufzeitsystem der Rechnerplattform oder bei Applet-Start aufgerufen und mit dem Graphics-Objekt versehen, auf das die Zeichenmethoden Anwendung finden.

Weiter wird paint() bei Änderungen auf der Windowsoberfläche bedarfsweise automatisch aufgerufen. Die Methode repaint() bewirkt eine Aktivierung von paint() durch das Anwendungsprogramm. Das Anwenderprogramm wird daher nicht unmittelbar paint() aufrufen, sondern sich des Repaint-Aufrufs bedienen.

Betrachten wir die Klasse *Animiation_1*. Dort wird das bereits bekannte Fahrzeug der Klasse zeichneAuto() erneut benutzt. Diese Klasse beinhaltet nur eine Methode, die mit zeichne() benannt ist. Aus der Methode paint() heraus wird die Methode animiereAuto() aufgerufen. Als Argument wird der Grafikkontext pen übergeben.

```
import java.applet.*;
import java.awt.*;
public class Animation_1 extends
                        java.applet.Applet {
//  Bewegte Grafiken in Java-Applets
//  Auto faehrt vorwaerts und rueckwaerts
 int scale,x0,y0,x1,xr;
 Color c1,c2,c3; Auto lkw;
 public void init(){
   lkw = new Auto();
   c1 = Color.blue; c2 = Color.red;
   scale = 10;                // Groesse
   x0 = 200; y0 = 200;    // Einfuegepunkt
   x1 = x0 - 50;              // linke Position
   xr = x0 +50;             // rechte Position
   this.setSize (400,300);
   setBackground(Color.lightGray);
 }

 public void paint(Graphics pen){
   animiereAuto( pen);
 }
```

In der Methode animiereAuto() erfolgt zunächst die Bestimmung der Hintergrundfarbe mit getBackground(). Weiter werden

die Farben und die Positionen sowie der Maßstabsfaktor definiert, und es wird ein Objekt `lkw` der Klasse `Auto` instanziert. Der Einfügepunkt des Autos ist `x0, y0`. Die Fahrbewegung beträgt nach links und nach rechts jeweils 50px.

```
public void animiereAuto(Graphics c ){
  int xl,xr,k,j, x0,y0, i;
  Color c1, c2;
  setBackground(Color.black);
  c1 = Color.yellow; c2 = Color.red;
  Color c3;
  // bestimme Hintergrundfarbe
  c3 = getBackground();
  scale = 10; x0 = 200; y0 = 200;
  xl = x0 - 50; xr = x0 + 50;
```

Innerhalb zweier For-Schleifen wird die x-Position schrittweise um ein Pixel geändert. Dies erfolgt für eine Fahrt vorwärts und für eine Fahrt rückwärts. In der Vorwärtsfahrt wird der Schleifenzähler dekrementiert `j--`, in der Rückwärtsfahrt inkrementiert `j++`. Beachten Sie die Vergleichsoperatoren der Schleifenbedingung. Zunächst wird das Auto in den gewünschten Farben `c1,c2` gezeichnet und anschließend in der Hintergrundfarbe, dadurch wird die bestehende Zeichnung gelöscht. Alternativ wäre auch die Löschung des gesamten Rechtecks mit `clearRect()` möglich. Zwischen den Zeichnungsausgaben erfolgt mittels Aufruf von `warteSchleife()` eine Zeitverzögerung. Die Geschwindigkeit beim Vorwärtsfahren ist doppelt so hoch wie bei der Rückwärtsfahrt. Nach einer Vorwärts- und einer Rückwärtsfahrt wird über `repaint()` die Paint–Methode erneut aktiviert Dadurch entsteht eine Endlosschleife bis zur Beendigung des Applets im Browser.

```
for (j = xr ; j>xl; j--) {
  // Fahrt vorwaerts
  lkw.zeichne (c, c1, c2, scale, j,y0);
  warteSchleife(25)          // warte 25 ms
  lkw.zeichne (c, c3, c3, scale, j,y0);
  // Hintergrundfarbe, Bild wurde geloescht
}
for (j = xl ; j<xr; j++) {
  // Fahrt rueckwaerts
  lkw.zeichne (c, c1, c2, scale, j,y0);
  warteSchleife(50);           // warte 50 ms
  lkw.zeichne (c, c3, c3, scale, j,y0);
  // Hintergrundfarbe, Bild wurde geloescht
}
```

```
repaint();
}   // Ende Methode animiere Auto
void warteSchleife(int ms){
  try {
    Thread.sleep(ms);
  }
   catch(InterruptedException e){
  }
}
// Ueberschreiben der Methode update
public void update (Graphics screen){
  paint(screen);
}
} // Ende Klasse Animation_1
```

Threads, Warteschleife, Thread.sleep

Ein Thread ist innerhalb eines Programms ein eigener Prozess, der parallel zu anderen Aufgaben ablaufen kann. Im Zusammenhang mit Animationen wird durch Unterbrechung des Threads die Bewegung gesteuert. Jedes Programm bekommt automatisch einen Hauptthread zugeordnet. Ein Thread kann mit `sleep()` unterbrochen werden und beansprucht während dieser Zeit auch nicht den Prozessor. Die Methode `sleep()` wird immer in einem Try-catch-Konstrukt aufgerufen, um Ausnahmen abzufangen.

Die Grafikausgabe bei Animationen wird ständig wiederholt. Damit das System nicht überlastet wird, wird der Prozess immer wieder kurz angehalten, auch um `paint()` genug Zeit zur vollständigen Ausführung des Updates zu geben.

Animationen sind mit Printmedien nicht darstellbar. Starten Sie deshalb das Applet von der Website *www. programmierpraktikum.de.* Je nach Rechnerleistung wird es noch ein wenig flimmern und ruckeln. Zusätzlich zur Unterbrechung des Threads ist durch Überschreibung der Update-Methode noch eine Verbesserung des Bewegungsablaufs zu erzielen.

Die Methode `paint()` wird automatisch aufgerufen. Man kann das Windows-System aber auch durch Aufruf von `repaint()` veranlassen, das Applikationsfenster neu zu zeichnen. Im Falle eines Aufrufs von `repaint()` wird `update()` aufgerufen. Letztere Methode löscht den Fensterinhalt, füllt diesen wieder mit der Hintergrundfarbe und ruft dann wiederum `paint()` auf. Das Löschen des Hintergrundes ist der kritische Vorgang, der den Flimmer-Effekt hervorrufen kann. Aus diesem Grund kann man die Methode wie folgt in die Klasse *Animation_1* einfügen:

```
public void update (Graphics screen){
  //
  // Ueberschreiben der Methode update
  paint(screen);
}
```

Ein etwas modifiziertes Beispiel mit eigenem Thread und einer Wartezeit von 3 Sekunden vor einer erneuten Fahrt ist auf der Website unter *Animation_2* abgelegt.

Doppelpufferung

Eine Technik, die mit Doppelpufferung (double buffering) bezeichnet wird, kann die Performance von Animationen erhöhen und stellt eine Alternative zur Benutzung von nur einer Zeichenfläche dar.

Zunächst zeichnet man alle Grafikelemente in einen Speicherbereich, der im Hintergrund gehalten wird. Die aktuelle Grafikanzeige wird erst dann beeinflusst, wenn der nicht sichtbare Bereich in den sichtbaren Bereich kopiert wird.

Wir gehen unmittelbar auf das Beispiel ein. Innerhalb der Methode `paint()` reserviert man Speicher für ein Bild mit der Größe des augenblicklich angezeigten Fensters. Es muss ein zweites Graphic-Objekt verfügbar sein, dem man diesen Speicherbereich zuweist.

```
import java.applet.*;
import java.awt.*;
//
// Animation mit Doppel-Pufferung
 public class Animation_3 extends
                          java.applet.Applet {
  public void paint (Graphics pen){
//
//  Bild fuer den Hintergrund bereitstellen
//  und einem Graphics-Objekt zuweisen
   Image bgImg =
    createImage(size().width, size().height);
    Graphics bgGraph;
    bgGraph = bgImg.getGraphics();
```

Das Objekt `bgImg` wird eine Instanz der Klasse `Image`. Die Größe wird durch Aufruf der Methode `createImage()` mit den Parametern für die Breite und Höhe des aktuellen Applikationsfensters festgelegt. Neben dem Graphics-Object `pen`, das durch die Methode `paint()` verfügbar wird, benötigen wir ein zweites Objekt dieser Klasse, hier bezeichnet mit `bgGraph` (Grafik im Hinter-

grund). Das Objekt bgImg wird mit der Methode getGraphics() dem Hintergrundobjekt zugewiesen und steht nun im Kontext zu bgGraph.

Es wird noch die Hintergrundfarbe gesetzt und die Animation wie gehabt aufgerufen. Das Bild selbst und beide Graphics-Objekte, bgGraph im Hintergrund, pen im Vordergrund, werden als Argumente übergeben.

```
    setBackground(Color.lightGray);
    animiereAuto( bgImg, bgGraph, pen);
} // Ende Klasse Animation_3
```

Die Methodenaufrufe zum Zeichnen des Objekts lkw innerhalb animiereAuto() werden wie folgt durchgeführt:

```
    lkw.zeichne (bg, c1, c2, scale, j,y0);
```

Mit den Vordergrundfarben wird das Objekt lkw in den nicht sichtbaren Hintergrundbereich gezeichnet. Da dieser im Kontext zu dem Image-Objekt steht, kann das Image mit der Methode drawImage() in den sichtbaren Vordergrundbereich, der mit Graphics pen durch paint() bei Initialisierung des Applets bereitsteht, übernommen werden.

```
  // Uebertrage Image ->Graphics
 c.drawImage(bgImg,0,0,this);
```

Die beiden vorgenannten Schritte werden jetzt mit der Hintergrundfarbe wiederholt. Die Grafik im Vordergrund bleibt dann kurzzeitig leer.

```
 // Zeichne mit Hintergrundfarbe
 lkw.zeichne(bg, c3, c3, scale, j,y0);
 // Graphics C -> leer
c.drawImage(bgImg,0,0,this);
```

Die vier notwendigen Schritte zur Doppelpufferung sind also:

- Hinzufügen von Instanz-Variablen für die Speicherung eines Bildes und eines Graphics-Kontextes Image bgImg, Graphics bgGraph;
- Erzeugen der Instanzen von Image und Graphics
  bgImg = createImage(size().width, size().height);
  graph= bgImg.getGraphics();
- Alle Grafikausgaben in den Hintergrundpuffer schreiben;

`lkw.zeichne(bgGraph, ...`

- Nach Ende der Graphikmethoden den Hintergrundpuffer in den sichtbaren Bereich schreiben. `pen.drawImage (bgImg,..`

Der SourceCode von *Animation_3* hier nunmehr im Zusammenhang :

```
import java.applet.*;
import java.awt.*;
// Animation mit Doppel-Pufferung
public class Animation_3 extends java.applet.Applet {
 int scale,x0,y0,xl,xr;
 Color c1,c2,c3;
 Graphics bgGraph;
 Image bgImg;
 ZeichneAuto lkw;
 public void init(){
   lkw = new ZeichneAuto();
   c1 = Color.blue; c2 = Color.red;
   scale = 10;              // Groesse
   x0 = 200; y0 = 200;      // Einfuegepunkt
   xl = x0 - 50;            // linke Position
   xr = x0 +50;             // rechte Position
   this.setSize (400,300);
   setBackground(Color.lightGray);
  }
public void paint (Graphics pen){
//  Bild fuer den Hintergrund bereitstellen
//  und einem Graphics-Objekt zuweisen
//  bestimme Hintergrundfarbe
    c3 = getBackground();
    bgImg = createImage(size().width, size().height);
    bgGraph = bgImg.getGraphics();
    animiereAuto( bgImg, bgGraph,pen); }
  public void animiereAuto(Image bgImg, Graphics bg,
                           Graphics c){
      //Auto faehrt vorwaerts
    for (int j = xr ; j>xl; j--) {
      // zeichne mit Vodergrundfarbe
      lkw.zeichne (bg, c1, c2, scale, j,y0);
      c.drawImage(bgImg,0,0,this);
      // Uebertrage Image ->Graphics
      c.drawString("doublebuffering",75,25);
      warteSchleife(25);
      // Zeichne mit Hintergrundfarbe
      lkw.zeichne(bg, c3, c3, scale, j,y0);
```

```
        c.drawImage(bgImg,0,0,this);
        // Graphics C -> leer
        }
// Auto faehrt rueckwaerts
  for (int j = xl ; j<xr; j++)
    lkw.zeichne(bg, c1, c2, scale, j,y0);
     c.drawImage(bgImg,0,0,this);
     c.drawString("double buffering", 175,25);
      warteSchleife(75);
      // Zeichne mit Hintergrundfarbe
      lkw.zeichne(bg, c3, c3, scale, j,y0);
      // Graphics C -> leer
      c.drawImage(bgImg,0,0,this);
   }
  // Aufruf update
     repaint( );
  }
  public void warteSchleife ( int n){
     try {
       Thread.sleep(n);
      }
      catch(InterruptedException e){
      }
  }
    public void update (Graphics screen){
      paint(screen);
   }
}
```

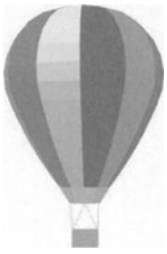
Listing 4.8: Animation mit Java-Applets

Animation mit Bitmaps

Das Verfahren der Doppelpufferung wird im folgenden Beispiel einer Animation mit Bitmaps erneut angewandt. Es existieren zwei Bilder im Format GIF unter der Bezeichnung *stadt.gif* und *ballon.gif*. Der Ballon soll kontinuierlich über der Stadt fahren. Wir betrachten die Klasse `Animation_img`. Dort werden global zwei Objekte vom Typ `Image` deklariert.

Innerhalb von `init()` werden die Bilder mit `getImage()` eingelesen. Unterstützt werden die Formate GIF und JPG. Die Parameter von `getImage()` sind `getCodeBase()` bzw. `getDocumentBase()` und der Namen der Bilddatei. `GetDocumentBase()` liefert die URL (uniform resource locator, Ort der Ressource im Web) der Website, in die das Applet eingebettet ist, `getCodeBase()` liefert die URL des Applets selbst zurück. Sofern HTML und Applet im gleichen Verzeichnis stehen sind die Werte identisch. Der Name des Bildes kann als Stringliterale (Konstante) oder besser als Va-

riable angegeben werden. Dann sind die Quellbilder auch über die HTML-Seite als Parameter referenzierbar. Die Methode `getImage()` kehrt direkt nach dem Aufruf zurück, das Bild ist ggf. noch nicht geladen. Das kann zu Problemen führen, die durch Verwendung eines MediaTracker-Objekts vermeidbar sind. Wir benutzen hier auch den This-Pointer, der immer das aktuelle Objekt bezeichnet.



```
import java.applet.*;
import java.awt.*;
// Ballonfahrt Animation mit Doppel-Pufferung
public class Animation_img extends
                              java.applet.Applet {
MediaTracker tracker;
Image stadt;      // Hintergrundbild
Image ballon;     // bewegtes Vordergrundbild
Font textFont;
int speed, count, iWidth, iHeight  ;
int iWidthB,iHeightB  ;
Image bgImg;
Graphics off ;
public void init(){
  tracker = new MediaTracker(this);
stadt = getImage(getCodeBase(),"stadt.gif");
  tracker.addImage(stadt,0);
  // Ladevorgang mit MediaTracker überprüfen
  try{
    tracker.waitForID(0);}
    catch(InterruptedException e){
    }
 ballon = getImage(getCodeBase(),"ballon.gif");
    tracker.addImage(ballon,0);
// Ladevorgang mit MediaTracker überprüfen
    try{
      tracker.waitForID(0);}
    catch(InterruptedException e){
    }
    iWidth  = stadt.getWidth (this);
    iHeight = stadt.getHeight(this);
    iWidthB = ballon.getWidth(this);
    iHeightB= ballon.getHeight(this);
    // Bildspeicher beschaffen
    bgImg = createImage(iWidth,iHeight);
    off= bgImg.getGraphics();
```

```
    textFont = new Font("Arial",Font.PLAIN,18);
    String theSpeed = getParameter("speed");
    if (theSpeed == null){
    // Vorbesetzung bei fehlendem Parameter
       speed = 150;
    }
    else {
      speed=Integer.parseInt(theSpeed);
    }
    count = 0;              // zaehlt die Fahrten
   }
```

In der Paint-Methode werden zunächst die Bildgrößen ermittelt, ein zweites Hintergrundbild wird generiert und dem zweiten Graphics-Objekt zugeordnet. Die Vorgehensweise entspricht der zuvor beschriebenen Animation mit Liniengrafik. Die Einfügeposition für das Hintergrundbild wird mit `xPos, yPos` festgelegt. Das Bild *`stadt.gif`* wird mit `drawImage()` in das sichtbare Objekt `pen` gezeichnet.

```
public void paint (Graphics pen){
int xPos = 0; int yPos=0;
pen.drawImage(stadt,xPos,yPos, iWidth,iHeight, this);
off.setFont(textFont);
off.setColor(Color.yellow);
```

Innerhalb einer For-Schleife erfolgt jetzt die Bestimmung der sich verändernden Ballonposition, die Ausgabe der Image-Objekte `stadt` an fester Position und `ballon` an variabler Position erfolgt aber zunächst im Hintergrund, auf dem `mit off` bezeichneten Graphics-Objekt. Erst das fertige Objekt wird in den sichtbaren Bereich `pen` eingetragen. Die Verzögerung erfolgt wieder durch die Unterbrechung des Hauptthreads. Mit `repaint()` wird die andauernde Animation erreicht.

```
for (int i = -iWidthB; i< iWidth+iWidthB; i++){
  off.drawImage(stadt,xPos,yPos, iWidth,iHeight, this);
  off.drawImage(ballon,i,30, iWidthB,iHeightB, this);
  pen.drawImage(bgImg,xPos,yPos,iWidth,iHeight,this);
  zeitSchleife(speed);
}
    repaint();
}
```

Das Applet bekommt noch zwei Ergänzungen. Es sollen die Ballonfahrten gezählt werden und zusammen mit der Zeitverzögerung und der Ballonposition in einer Textzeile ausgegeben wer-

den. Wir deklarieren global in der Klasse Animation_img folgende Objekte (Variablen):

```
Font textFont;
int speed;
int count;
```

Innerhalb der Init-Methode wird der Textfont definiert durch:

```
textFont = new Font("Arial",Font.PLAIN,18);
count = 0;
```

und der Zähler auf null gesetzt.

Bevor es in der Methode paint() in die For-Schleife geht, wird für das im Hintergrund befindliche Graphik-Objekt Zeichensatz und Farbe vereinbart.

```
off.setFont(textFont);
off.setColor(Color.yellow);
```

Innerhalb der Schleife wird dann noch ergänzend die Textausgabe mit drawString() vorgenommen.

```
off.drawString("Ballonfahrt Nr.: " +count +"
                Pos. x= " +i +", y = " +30
                 +" Delay: "+speed,125,185);
```

Der Zähler wird vor repaint() inkrementiert mit :

```
count++;
```

Das war aber noch nicht alles. Zum Abschluss soll noch der Parameter für die Zeitverzögerung aus dem Applet-Tag der HTML-Datei übernommen werden. Dieser Parameter kommt aus dem HTML-Code, sofern er dort angegeben ist, als Zeichenkette. Die Init-Methode wird wie folgt ergänzt:

```
String theSpeed = getParameter("delay");
    if (theSpeed == null){
    // Vorbesetzung bei fehlendem Parameter
      speed = 150;
    }
    else {
      speed=Integer.parseInt(theSpeed);
    }
```

Wir erwarten die Eingabe des String-Objekts theSpeed über den Parameter der im Applet-Tag mit speed zu bezeichnen ist. Für den Fall, dass die HTML-Datei den Wert nicht liefert, wird Sorge getragen und eine Vorbesetzung von 150 vereinbart.

Abb. 4.4: Animation mit Bitmaps. Vor dem Hintergrund der Stadtsilhouette fährt ein Ballon vorbei. Die Position wird laufend aktualisiert angezeigt.

Hier die vollständige HTML-Datei mit Applet-Tag und Parameterdefinition:

```
<html><head><title>Ballonfahrt</title>
</head>
<div align="center">
<body topmargin="0" leftmargin="0">
 <applet
   codebase="."
   code="Animation_img.class"
   width="600" height="200">
   <param name="speed" value="25">
 </applet>
</div>
</body>
</html>
```

Listing 4.8: Applet-Tag und Parameterübergabe

Die Darstellung von `Animation_img` abschließend im Zusammenhang:

```
import java.applet.*;
import java.awt.*;
// Ballonfahrt Animation mit Doppel-Pufferung
public class Animation_img extends
                            java.applet.Applet {
  MediaTracker tracker;
  Image stadt, ballon;
```

```
Font textFont;
int speed,count;
int iWidth, iHeight,iWidthB,iHeightB  ;
Image bgImg;
Graphics off ;
public void init(){
  tracker = new MediaTracker(this);
  stadt = getImage(getCodeBase(),"stadt.gif");
  tracker.addImage(stadt,0);
   try{
      tracker.waitForID(0);
      }
    catch(InterruptedException e){
    }
    ballon = getImage(getCodeBase(),"ballon.gif");
    tracker.addImage(ballon,0);
      try{
        tracker.waitForID(0);
      }
      catch(InterruptedException e){
      }
    iWidth  = stadt.getWidth  (this);
    iHeight = stadt.getHeight (this);
    iWidthB = ballon.getWidth (this);
    iHeightB= ballon.getHeight(this);
    bgImg = createImage(iWidth,  iHeight );
    off= bgImg.getGraphics();
    textFont = new Font("Arial",Font.PLAIN,18);
    String theSpeed = getParameter("speed");
    if (theSpeed == null){
       speed = 150;
    }
    else {
      speed=Integer.parseInt(theSpeed);
    }
    count = 0; // zaehlt die Fahrten
   }
   public void paint (Graphics pen){
    int xPos = 0; int yPos=0;
    pen.drawImage
    (stadt,xPos,yPos, iWidth,iHeight, this);
    off.setFont(textFont);
    off.setColor(Color.yellow);
    for (int i = -iWidthB; i< iWidth+iWidthB; i++){
      off.drawImage(stadt,xPos,yPos,iWidth,iHeight, this);
      off.drawImage(ballon,i,30,iWidthB,iHeightB, this);
```

```
        off.drawstring ("Ballonfahrt Nr.: " +count +"
           Pos. x= " +i +", y = " +30 +" Delay: "+speed,125,185);
        pen.drawImage (bgImg,xPos,yPos,iWidth,iHeight,this);
        zeitSchleife(speed);
      }
      count++;
      repaint();
    }
    private void zeitSchleife(int time){
       try {
         Thread.sleep(time);
        }
         catch (InterruptedException e){
        }
      }
     public void update (Graphics screen){
       paint (screen);
     }
}
```

Listing 4.9: Animation mit Bitmaps

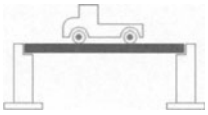
Zusammenfassung Animationen

Animationen gehören zu den beliebten Anwendungen von Java-Applets. Eine Animation entsteht durch wiederholtes Zeichnen und Löschen von Vektorgrafiken oder Bitmaps mit wechselnden Positionen. Die Zeitschleife wird durch Anhalten eines Threads erzeugt. Eine bewährte Methode für flimmerfreie Animationen ist das Verfahren der Doppelpufferung. Hierbei erfolgt die Zeichnungsausgabe generell im Hintergrund. Erst wenn die gesamte Zeichnung fertiggestellt ist, wird diese in den Vordergrund übertragen. Bitmaps werden mit `getImage()` im Internet übertragen. MediaTracker-Objekte überwachen den Ladevorgang. Mit der Methode `getParameter()` werden Werte aus HTML-Seiten an Java-Applets übergeben.

4.6 Grafische Benutzeroberflächen

In den bisherigen Java-Beispielen wurden alle Parameter innerhalb des Programmcodes den Variablen direkt zugewiesen oder durch die Auswertung von Ausdrücken berechnet. Bei Änderungswünschen muss der Editor her und eine neue Klasse compiliert werden. Das wird aber nur einem Programmierer Freude bereiten, der Anwender im Web kann damit nicht umgehen. Mit dem *Advanced Windowing Toolkit* werden auch die Werkzeuge zur Gestaltung einer grafischen Benutzeroberfläche bereitgestellt.

Wir werden jetzt ein GUI (*graphical user interface*) entwickeln und ausgewählte Bedienelemente kennen lernen. Weiter betrachten wir die Positionierung der GUI-Elemente auf der Zeichenfläche.



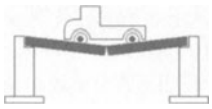
Traglast

Problemstellung ist die Vorbereitung zur Berechnung und Visualisierung der Traglast einer Brücke. Das Programm soll mittels einer grafischen Darstellung die unzulässige Belastung der Brücke anzeigen. Wir haben ja bereits die Klasse `Auto` vorgestellt. Nach dem gleichen Muster wird hier eine Klasse `Bruecke` hinzugefügt. Auf die Diskussion dieser Klasse wird verzichtet, da es keine wesentlichen Unterschiede zur Klasse *Auto* gibt. Unser Traglastmodell besteht also aus einer Brücke, deren Tragkraft erhöht werden kann, und einer Last, dem Auto, die ebenfalls variabel ist. Weitere Programmparameter sind der Bezugspunkt zum Einfügen der Brücke und des Autos in die Grafik und die Länge der Brücke. Für den Fall, dass die Brücke nicht mehr die notwendige Tragkraft aufweist, wird im Teststadium eine Variable `boolean crash` eingeführt. Das Programm kann in Abhängigkeit des Parameters über die Darstellungsart der Brücke entscheiden. Variable vom Typ `boolean` können nur die Werte `true` oder `false` annehmen.

Die Variablen sind in der Klasse `Modell` global definiert und werden im Konstruktor initialisiert. Bei Anlegen des Objekts Brücke wird die Länge und Materialstärke dem Konstruktor mitgeteilt.

Da wir uns hier mit der GUI-Programmierung beschäftigen, wollen wir auch eine Checkbox einbeziehen, die im geklickten Zustand signalisiert, dass die Tragfähigkeit der Brücke nicht mehr gegeben ist und somit die grafische Darstellung für die Bruchlast zu benutzen ist. Auf der Webseite finden Sie das kommentierte, erweiterte Programm. Dort wird über Tragfähigkeit aufgrund der Last und der Brückenabmessung durch Berechnung entschieden.

Zunächst betrachten wir nur die grafischen Darstellungen, die nach Studium der vorhergehenden Kapitel keine Probleme mehr bereiten sollten.



Bruchlast

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
// Klasse Modell zeichnet Bruecke mit Last
public class Modell extends
                              java.applet.Applet {
 Color cR = new Color(255,0,0);
```

```
 Color cS = new Color(0,0,0);
 boolean crash;
 int x0,y0, scale,l,h;
 Modell (){
setBackground(Color.lightGray);
x0=200; y0=150; scale=3; l = 200; h = 15;
crash=false;
 }
 public void paint(Graphics screen){
  // x0,y0 ... Einfuegepunkt Bruecke und LKW
  // scale ... Maszstabsfaktor LKW -> Last
  // l ..... Laenge der Bruecke
  // h ..... Traglast Bruecke (Querschnitt)
  // crash ... false -> Bruecke tragt Last
  Auto   lkw = new Auto();
  Bruecke b  = new Bruecke(l,h);
  b.zeichne   (screen,x0,y0,crash);
  lkw.zeichne (screen,cR,cB,scale,x0,y0);
 }
}
```

Komponenten

Komponenten einer Benutzerschnittstelle werden sog. Containern zugeordnet und dort platziert. Zunächst ist der übergeordnete Container unsere Applet-Klasse. Eine Komponente wird deklariert und mit der Methode `add()` einem Container hinzugefügt. Nehmen wir zunächst die Komponenten

- Label
- TextField
- Checkbox
- Button.

Ein Label kann nicht unmittelbar verändert werden, es dient üblicherweise dem Dialog des Programms mit dem Anwender. In Textfelder werden die Werte von Variablen als Zeichenketten eingegeben. Die Strings sind innerhalb des Programms in die entsprechenden Datentypen zu konvertieren. Eine Checkbox arbeitet wie ein Schalter. Im vorliegenden Beispiel kann `crash` auf `true` oder `false` gesetzt werden. Bei Klick auf einen Button ist die Auslösung von Programmanweisungen beabsichtigt, diese Ereignisbehandlung wird im nächsten Abschnitt behandelt. Hier richten wir die Betrachtungen zunächst nur auf die Anordnung und Funktionalität der Komponenten.

Die Komponenten werden global deklariert und mit Anfangswerten vorbesetzt. Die Labels beinhalten den Dialogtext. Textfelder

werden mit einer Länge von 8 Zeichen dimensioniert, der Button bekommt die Beschriftung Zeichne. Zusätzlich zu den Komponenten wird noch die Ausgestaltung des Textes festgelegt.

```
Label lbLaenge = new Label ("Laenge der Bruecke [200-300] ");
Label lbTragkraft = new Label("Tragkraft der Bruecke [1-40] ");
Label lbLast = new Label ("Last                    [1-10] ");
TextField tfLaenge    = new TextField(8);
TextField tfTragKraft = new TextField(8);
TextField tfLast      = new TextField(8);
Font txtFont = new Font("Arial",Font.BOLD,12);
Checkbox cbCrash = new Checkbox("Crash",true);
Button  btDraw   = new Button ("Zeichne");
```

Die Anweisungen zur Vorbereitung der GUI-Oberfläche werden geeignet in der Init-Methode positioniert. Wir verlegen einige Initialisierungen vom Konstruktor der Klassen Auto und Bruecke ebenfalls in die Methode init().

```
public void init (){
 setBackground(Color.lightGray);
 x0=200; y0=150; scale=3; l = 200; h = 15;
 crash=false;
 setFont(txtFont);
 add(lbLaenge); add (tfLaenge); add(lbTragkraft);
 add(tfTragKraft); add(lbLast);add (tfLast);
 add(cbCrash); add(btDraw);
}
```

Die Komponenten werden mit der Methode add() in das Applet eingefügt. Je nach Größe des Fensters erfolgt die Positionierung standardmäßig im sog. Flow-Layout in der angegebenen Reihenfolge. Keine besonders gelungene Anordnung. Wir wollen diese Anordnung verbessern und den Dialog von der Zeichenfläche trennen. Die Dialogelemente werden bei der vorgenommenen Anordnung in das Grafikfenster positioniert und überdecken ggf. Teile der Grafik.

Die Anordnung der Komponenten wird von einem Layout-Manager vorgenommen. Im Java AWT existieren die Klassen

- FlowLayout,
- GridLayout,
- BorderLayout
- CardLayout
- GridBagLayout.

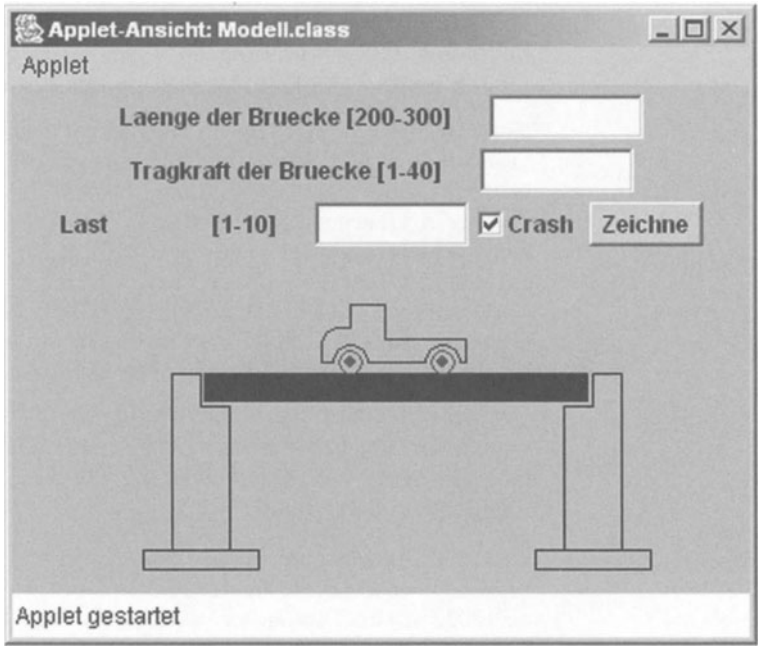


Abb. 4.5: Komponenten der Benutzeroberfläche im FlowLayout

Beim GridLayout werden die Komponenten in Zeilen und Spalten angeordnet. In unserem Beispiel können fünf Zeilen mit zwei Spalten definiert werden. Die weiteren Parameter beim Aufruf von GridLayout legen den Abstand zwischen den Komponenten fest.

Global wird

```
GridLayout  glRaster = new GridLayout (5,2,4,4);
```

vereinbart. Innerhalb von `init()` wird das Layout mit

```
setLayout(glRaster);
```

definiert.

Damit die Zeichenfläche ebenfalls in das Raster aufgenommen werden kann, muss der Grafikbereich in einer eigenen Unterklasse vom Typ `Canvas` eingebettet werden. Wir definieren eine Unterklasse von Canvas :

```
class ZeichenFenster extends java.awt.Canvas {
   public void paint(Graphics screen){
   }}
```

Nachdem ein Objekt dieser Klasse gebildet wurde, kann der Konstruktor aufgerufen werden und das neue Canvas-Objekt einem Container hinzugefügt werden.

```
ZeichenFenster viewport = new ZeichenFenster();
```

Jetzt besteht aber das Problem, dass GridLayout das gesamte Fenster ausfüllt und die Eingabefelder entsprechende proportionale Größen bekommen. Die Lösung besteht nun darin, dass mehrere Container in dem übergeordneten Container aufgenommen werden können und in diesen dann die Komponenten auch mit unterschiedlichen Layout-Managern zu positionieren sind. Eigene Container sind von der Klasse Panel.

Wir benutzen im übergeordneten Container das Borderlayout mit Anordnung der Komponenten um das Zentrum herum. Es sind noch die Layoutobjekte und das Containerobjekt zu bilden:

```
GridLayout glGrid   = new GridLayout(4,2,2,2);
BorderLayout blGrid = new BorderLayout();
Panel pDialog       = new Panel();
```

Wir trennen den Dialog von der Grafik, formulieren den Container pDialog und gestalten diesen mit Gridlayout in 4 Zeilen zu je zwei Spalten mit dem Abstand von 2 Pixel zwischen den Elementen. Das Applet wird im BorderLayout gestaltet. Die Objekte können um das Zentrum herum angeordnet werden. Variieren Sie im Sourcecode die Angabe North mit South, East, West. Unser Dialog wird entsprechend zum Zeichenfenster angeordnet. Zur besseren Unterscheidung kann man die Hintergrundfarben der Objekte verschieden definieren. Im Zusammenhang stellt sich der Quellcode wie folgt dar:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
// Klasse Modell zeichnet Bruecke mit Last
public class Modell extends
                  java.applet.Applet {
 // Dialogkomponenten definieren
 Label lbLaenge=new Label("Laenge der Bruecke [200-300] ");
 Label lbTragkraft=new Label ("Tragkraft der Bruecke [1-40] ");
 Label lbLast = new Label ("Last     [1-10] ");
 TextField tfLaenge      = new TextField(8);
 TextField tfTragKraft   = new TextField(8);
 TextField tfLast        = new TextField(8);
 Checkbox cbCrash = new Checkbox("Crash",true);
 Button  btDraw   = new Button ("Zeichne");
```

```
Font txtFont=new Font ("Arial", Font.BOLD,12);
// Layoutobjekte hinzufügen
GridLayout glGrid = new GridLayout(4,2,2,2);
BorderLayout blGrid = new BorderLayout();
// Eigenen Container fuer den Dialog
Panel  pDialog = new Panel();
// Canvas Objekt hinzufügen
// Konstruktor aufrufen
ZeichenFenster viewport = new ZeichenFenster();
public void init (){
 // Fenstergroessen und Layout vereinbaren
 this.setSize(400,500);
 this.setLayout(blGrid);
 pDialog.setSize(400,100);
 pDialog.setLayout(glGrid);
// Container Applet den eigenen Container pDialog hinzufuegen
   add("South", pDialog);
 // Objekte dem Container pDialog hinzufuegen
   pDialog.setBackground(Color.lightGray);
   pDialog.setFont(txtFont);
   pDialog.add(lbLaenge);
   pDialog.add (tfLaenge);
   pDialog.add(lbTragkraft);
   pDialog.add(tfTragKraft);
   pDialog.add(lbLast); pDialog.add (tfLast);
   pDialog.add(cbCrash);pDialog.add(btDraw);
  // Objekt Canvas im Zentrum des Applets
  // einfuegen
   add("Center",viewport);
 } // Ende Methode init
} // Ende Klasse Modell
class ZeichenFenster extends java.awt.Canvas{
Color cR = new Color(255,0,0);
Color cB = new Color(0,0,255);
Color cS = new Color(0,0,0);
boolean crash; int x0,y0, scale,l,h;
public void paint(Graphics screen){
 // x0,y0 ... Einfuegepunkt Bruecke und LKW
 // scale ... Maszstabsfaktor LKW -> Last
 // l ..... Laenge der Bruecke
 // h ..... Traglast Bruecke (Querschnitt)
 // crash ... false -> Bruecke tragt Last
 x0=200; y0=200; scale=5; l = 200; h = 15;
 crash=false; this.setSize(400,400);
 this.setBackground(Color.white);
 Auto   lkw = new Auto();
```

```
  Bruecke b  = new Bruecke(l,h);
  b.zeichne   (screen,x0,y0,crash);
  lkw.zeichne (screen,cR,cB,scale,x0,y0);
  } // Ende Methode paint
} // Ende Klasse Zeichenfenster
```

Listing 4.10: GUI Modell zur Traglastberechnung

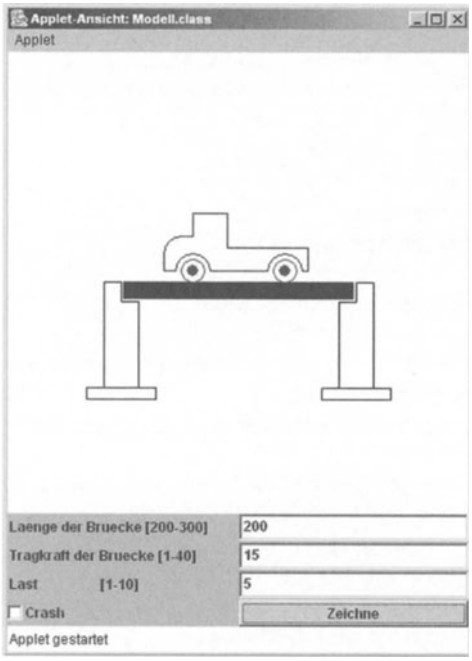


Abb. 4.6: GUI Oberfläche aufgeteilt in Container.

Der Screenshot, Abbildung 4.6, zeigt die fertige GUI-Oberfläche. Alle Dialogelemente arbeiten, man kann Eingaben vornehmen, die Checkbox bedienen und den Button `Zeichne` anklicken. Die Komponenten müssen aber noch auf die Benutzeraktion reagieren, hierzu ist jetzt die Auseinandersetzung mit dem Thema der Ereignisbehandlung notwendig.

Ereignisbehandlung

Wie können nun die Bedienungskomponenten der Benutzeroberfläche auf Ereignisse wie Mausklick, Tastatureingabe, Mausklick oder Mausbewegung reagieren? Hierzu existieren die sog. *Event Listener*. Das ist eine Klasse, die auf Ereignisse von außen reagiert und die weitere Verarbeitung nach Eintritt des Ereignisses veranlasst. Mit dem Schlüsselwort `implements` wird einer Klasse mitgeteilt, dass eines der Listener-Interfaces installiert

werden muss. Im vorliegenden Fall reagiert ActionListener auf das Anklicken einer Schaltfläche. Objekte der Klasse müssen dann noch als Listener registriert werden, hier mit btDraw.addActionListener(this). Der Listener überwacht nun die ihn betreffenden Ereignisse und benötigt eine Methode zur Auswertung. Ein Action-Listener benötigt die Methode actionPerformed(), der das auslösende Ereignis übergeben wird.

```
public class Events
             extends java.applet.Applet
             implements ActionListener {
 void init(){
  tfLaenge.setText("250");
  tfTragKraft.setText("15");
  tfLast.setText("3");
  btDraw.addActionListener(this);
 }
 public void actionPerformed (ActionEvent e){
  //
  // Konvertierung der Texteingabefelder von
  // String nach int
  // zulaessiger Wertebereich wird geprueft
  //
  int laenge, hoehe,last;  boolean crash;
  String s;
  s =  tfLaenge.getText();
  laenge = Integer.parseInt(s);
  if (laenge <100 || laenge >400) laenge = 250;
  s =  tfTragKraft.getText();
  hoehe = Integer.parseInt(s);
  if (hoehe <1 || hoehe >40) hoehe = 15;
  s = tfLast.getText();
  last = Integer.parseInt(s);
  if (last <1 || last >10) last = 3;
  //
  // Status der Checkbox ?
  crash = cbCrash.getState();
  // Dem Objekt viewport die Zeichnungs-
  // parameter mitteilen
  viewport.setDimension( laenge,
                  (int)hoehe,(int)last,crash);
  // Neuzeichnung
  viewport.repaint();
 }
}
```

Das ActionEvent selbst wird hier noch nicht weiter abgefragt, da wir ja wissen, dass es der Button gewesen sein muss. Ein späteres Beispiel bearbeitet die Selection der Ereignisse. In der Methode actionPerformed() werden die Textfelder ausgewertet. Es erfolgt der Umbau der Zeichenketten in die benötigten ganzzahligen Werte sowie die Abfrage auf den zulässigen Wertebereich. Der Status der Checkbox wird abgefragt.

Die Klasse aus dem obigen Beispiel wurde noch etwas modifiziert. Es wurde ein Konstruktor eingeführt und Methoden zum Zugriff auf die Klassenvariablen setDimension(), setPosition() ergänzend hinzugefügt. Es folgt der Programmcode im Zusammenhang.

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
// Klasse Events demonstriert Ereignis-
// behandlung
public class Events extends java.applet.Applet
                    implements ActionListener{
 // Dialogkomponenten definieren
 Label lbLaenge = new Label ("Laenge der Bruecke  [100-400] ");
 Label lbTragkraft = new Label ("Tragkraft der Bruecke [1-40]
");
 Label lbLast = new Label ("Last    [1-10] ");
 TextField tfLaenge      = new TextField(8);
 TextField tfTragKraft   = new TextField(8);
 TextField tfLast        = new TextField(8);
 Checkbox cbCrash = new Checbox("Crash",false);
 Button  btDraw = new Button ("Zeichne");
 Font txtFont = new Font("Arial",Font.BOLD,12);
 // Layoutobjekte hinzufügen
 GridLayout glGrid = new GridLayout(4,2,2,2);
 BorderLayout blGrid = new BorderLayout();
 // Eigenen Container fuer den Dialog
 Panel  pDialog          = new Panel();
 // Canvas Objekt hinzufügen
 ZeichenFenster viewport = new ZeichenFenster();
 public void init (){
  // Fenstergroessen und Layout vereinbaren
  this.setSize(400,500);
  this.setLayout(blGrid);
  pDialog.setSize(400,100);
  pDialog.setLayout(glGrid);
  // Dem Container Applet den neuen Container
```

```
 // pDialog hinzufuegen
 add("South", pDialog);
 // Objekte dem Container pDialog hinzufuegen
 pDialog.setBackground(Color.lightGray);
 pDialog.setFont(txtFont);
 pDialog.add(lbLaenge);pDialog.add (tfLaenge);
 pDialog.add(lbTragkraft);
 pDialog.add(tfTragKraft);
 pDialog.add(lbLast); pDialog.add (tfLast);
 pDialog.add(cbCrash);pDialog.add(btDraw);
 // Objekt Canvas im Zentrum einfuegen
 add("Center",viewport);
 tfLaenge.setText("250");
 tfTragKraft.setText("15");
 tfLast.setText("3");
 // Komponente beim Listener registrieren
 btDraw.addActionListener(this);
} // ende Methode init
public void actionPerformed (ActionEvent e){
 // Konvertierung der Texteingabefelder
 // von  String nach int
 // zulaessiger Wertebereich wird geprueft
 int laenge, hoehe,last;
 boolean crash;
 String s;
 // String aus Textfeld holen
 s =  tfLaenge.getText();
 // parsen von String nach int
 laenge = Integer.parseInt(s);
 // Voreinstell bei Bereichsueberschreitung
 if (laenge <100 || laenge >400) laenge = 250;
 s =  tfTragKraft.getText();
 hoehe = Integer.parseInt(s);
 if (hoehe <1 || hoehe >40) hoehe = 15;
 s = tfLast.getText();
 last = Integer.parseInt(s);
 if (last <1 || last >10) last = 3;
 //Status der Checkbox ?
 crash = cbCrash.getState();       //
 // Dem Objekt viewport die Zeichnungs-
 // parameter mitteilen
 viewport.setDimension( laenge, (int)hoehe,
                        (int)last,crash);
// Neuzeichnung durch Aufruf von repaint()
// auf das Canvas-Objekt
   viewport.repaint();
```

```
  } // Ende Methode actionPerformed
} // Ende Klasse   Events

class ZeichenFenster extends java.awt.Canvas{
 Color cR = new Color(255,0,0);
 Color cB = new Color(0,0,255);
 Color cS = new Color(0,0,0);
 boolean crash;
 int x0, y0, scale, l, h;
 public ZeichenFenster(){
  // Konstruktor definiert Voreinstellungen
  setPosition (200,200);
  setDimension(250,15,3,false);
 }
 public void setDimension( int laenge,
        int dicke, int s, boolean bruch){
  l = laenge; h = dicke;
  scale = s;crash = bruch;
 }
 public void setPosition(int x,int y){
  x0 = x; y0 = y;
 }
 public void paint(Graphics screen){
  // x0,y0 ... Einfuegepunkt Bruecke und LKW
  // scale ... Maszstabsfaktor LKW -> Last
  // l ....... Laenge der Bruecke
  // h ....... Traglast Bruecke (Querschnitt)
  // crash ... false -> Bruecke traegt Last
  Color bgFarbe = new Color(255,248,202);
  this.setSize(400,400);
  this.setBackground(bgFarbe);
  Auto   lkw = new Auto();
  Bruecke b  = new Bruecke(l,h);
  b.zeichne   (screen,x0,y0,crash);
  if ( crash)
     lkw.zeichne (screen,cR,cB,scale,x0,y0+h);
  if ( !crash)
     lkw.zeichne (screen,cR,cB,scale,x0,y0);
  }
```

Listing 4.11: GUI und Ereignissteuerung

Die Klasse Auto wurde schon weiter oben beschrieben, die Klasse Bruecke ist kaum unterschiedlich. Sie finden den kompletten Quelltext auf der Webseite im Leserbereich. Das wär's dann zunächst mit diesem Applet. Die Frage der Berechnung der zuläs-

sigen Last und Vergleich mit der tatsächlichen Last kann in einer eigenen Methode erfolgen. Auf die Beschreibung des algorithmischen Problems wird hier verzichtet. Der Aufruf aus `actionPerformed()` ist leicht nachzuvollziehen. Der Parameter `crash` wird entsprechend der Berechnungsergebnisse gesetzt. Sie finden die Berechnung mit hinreichend dokumentierten Sourcecode auf der Website.

Zusammenfassung GUI-Programmierung

Zur Programmierung einer grafischen Benutzeroberfläche stellt das AWT Klassen für alle gängigen Benutzerkomponenten bereit. Die Komponenten werden mit Layout-Managern auf einem äußeren Container, der Applet-Zeichenfläche ausgerichtet und können selbst auch wieder in eigenen Containern angeordnet sein. Die Zeichenfläche des Applets wird in mehrere Container eingeteilt. Layoutmanager können kombiniert eingesetzt werden. Bedienelemente werden deklariert, instanziert und in die Container eingefügt. Komponenten, die auf ein Ereignis reagieren, werden durch sog. Listener überwacht. Es existieren diverse Listenerklassen für Buttons, Scrollbars, Tastatur- oder Mauseingaben, die unterschiedlich zu handhaben sind. Implementiert wurde ein `actionListener` für einen Button mit der zugehörigen Methode `actionPerformed()`. Eine weitere Behandlung der Thematik wird in der folgenden Programmieraufgabe vorgenommen. Dort erfolgt die Eingabe von Variablen über Scrollbars, die durch `AdjustmentListener` überwacht werden. Beim Listener werden gleich mehrere Scrollbars registriert. Dabei ist dann auch noch das auslösende Ereignis selbst entsprechend zu behandeln.

4.7 Einführung in die 3D-Grafik

Dreidimensionale Grafiken sind in der Modellbildung und auch in der Abbildung sehr komplex. In diesem Abschnitt wird ein Programm diskutiert, mit dem die Möglichkeit besteht, sog. Drahtmodelle interaktiv im Browser aus beliebigen Richtungen bei Variation des Betrachtungsabstands und der Bildweite zu betrachten.

Modellierung und Transformationen

Wir wählen ein einfaches Bauteil aus, dessen Eckpunkte in einem köperfesten Koordinatensystem normiert vorgegeben werden, vgl. Abbildung 4.7. Durch Multiplikation mit Maßstabsfaktoren entlang der Koordinatenachsen können die Abmessungen des Werkstücks verändert werden. Das Bauteil wird durch Angabe eines Einfügepunktes in einem raumfesten Koordinatensystem positioniert. Rotationen können um die Körperachsen erfolgen.

Wir erstellen eine Klasse Transform, die Methoden zur Skalierung, Translation und Rotation um Koordinatenachsen beinhaltet.

Zunächst zur Klasse Werkstueck. Die Instanzvariablen zeigen die benutzte Datenstruktur zur Beschreibung des Bauteils. In einem zweidimensionalen Array werden die 18 Eckpunkte abgelegt. Weiter benötigen wir zunächst noch die umhüllende Box, ein Quader mit acht Eckpunkten, und sechs Punkte zur Beschreibung der körperfesten Koordinatenachsen. Zusätzlich werden die Extremwerte der Abmessungen in elementaren Variablen gespeichert. Die Variablen werden mit public static modifiziert, um ggf. auch von außerhalb Zugriff zu gewähren.

```
import java.awt.*;
public class Werkstueck {
 public static double [][] dPkt = new double [18][3];
 public static double [][] dBox = new double  [8][3];
 public static double [][] dAchsen = new double [4][3];
 public static double minx, miny, minz, maxx, maxy, maxz;
 public static int n;
 public static int n2;
 Werkstueck (){
  double [] scale = new double[3];
  double [] trans = new double [3];
  scale[0]=1.0; scale[0]=1.0; scale[0]=1.0;
  trans[0]=0.0; trans[1]=0.0; trans[2]=0.0;
  n = 18; n2 = 9;
  resetWerkstueck (scale,trans);
  boxWerkstueck();
 }// Ende Konstruktor
```

Im Konstruktor wird die Skalierung und die Translation voreingestellt. Mit diesen Argumenten wird eine Methode resetWerkstueck() aufgerufen, mit der die normierten Eckpunktkoordinaten erzeugt werden. Diese Ausgangswerte benötigen wir immer wieder bei Rotationen. Die Methode boxWerkStueck() liefert die umhüllende Box des Körpers.

```
public static void resetWerkstueck
                  (double s[], double t[]){
 // Eckpunkte untere Grundflaeche
 dPkt[0][0]=0.0;dPkt[0][1]=0.0;dPkt[0][2]=0.0;
 dPkt[1][0]=0.0;dPkt[1][1]=0.6;dPkt[1][2]=0.0;
 dPkt[2][0]=0.6;dPkt[2][1]=0.6;dPkt[2][2]=0.0;
 dPkt[3][0]=0.6;dPkt[3][1]=1.0;dPkt[3][2]=0.0;
 dPkt[4][0]=0.0;dPkt[4][1]=1.0;dPkt[4][2]=0.0;
 dPkt[5][0]=0.0;dPkt[5][1]=1.4;dPkt[5][2]=0.0;
```

```
 dPkt[6][0]=1.6;dPkt[6][1]=1.4;dPkt[6][2]=0.0;
 dPkt[7][0]=2.0;dPkt[7][1]=1.0;dPkt[7][2]=0.0;
 dPkt[8][0]=2.0;dPkt[8][1]=0.0;dPkt[8][2]=0.0;
 // Eckpunkte obere Flaeche, Hoehe 1.0
 for (int i = 0; I < n2; i++){
  dPkt[i+n2][0]=dPkt[i][0];
  dPkt[i+n2][1]=dPkt[i][1];dPkt[i+n2][2]=1.0;
 }
 // koerperfeste Achsen definieren
 dAchsen[0][0] = 1.0; dAchsen[0][1] = 0.7;
 dAchsen[0][2] = 0.5;
 dAchsen[1][0] = 2.0; dAchsen[1][1] = 0.7;
 dAchsen[1][2] = 0.5;
 dAchsen[2][0] = 1.0; dAchsen[2][1] = 1.4;
 dAchsen[2][2] = 0.5;
 dAchsen[3][0] = 1.0; dAchsen[3][1] = 0.7;
 dAchsen[3][2] = 1.0;
 // Skalierung, Dimension
 Transform.skalierung (s, dPkt, n);
 Transform.translation(t, dPkt, n);
 Transform.skalierung (s, dAchsen, 4);
 Transform.translation(t, dAchsen, 4);
} // Ende Methode resteWerkstueck
```

Die Punkte sind im Uhrzeigersinn festgelegt. In der linken unteren Ecke befindet sich der Bezugspunkt. Die Ausdehnungen sind in x-Richtung 2.0, in y-Richtung 1.5 und in z-Richtung 1.0. Die körperfesten Koordinatenachsen sind im Mittelpunkt des Körpers gelagert. Die Achslänge geht bis zur Begrenzung des umhüllenden Rechtecks. Aus der Klasse `Transform` werden die Methoden `skalierung()` und `translation()` aufgerufen. Das ist hier zunächst nicht notwendig, greift man aber von außerhalb auf die Methode `resetWerkstueck()` zurück, dann werden die aktuellen Argumente für Rotation und Skalierung mit übergeben und auf die ursprünglichen Daten angewandt.

Diese Datenstruktur ist für praktische Anwendungen nicht optimiert, aber hilfreich bei der Diskussion und Einführung von 3D-Verfahren. Es ist auch darauf hinzuweisen, dass in der Computergrafik üblicherweise mit homogenen Koordinaten gearbeitet wird. Die Vorteile homogener Koordinaten würden an dieser Stelle noch nicht deutlich, insofern kann auf die vierte Komponente hier verzichtet werden.

Wir bestimmen die Extremwerte des umhüllenden Rechtecks in der Methode `boxWerkStueck`. Die Methode ist selbsterklärend.

```
public static void boxWerkstueck (){
  // Berechnet umhuellende Box des Werkstuecks
  minx = dPkt[0][0]; maxx = dPkt[0][0];
  miny = dPkt[0][1]; maxy = dPkt[0][1];
  minz = dPkt[0][2]; maxz = dPkt[0][2];
  for (int i = 1; i < 18; i++){
   if (minx > dPkt[i][0]) minx = dPkt[i][0];
   if (maxx < dPkt[i][0]) maxx = dPkt[i][0];
   if (miny > dPkt[i][1]) miny = dPkt[i][1];
   if (maxy < dPkt[i][1]) maxy = dPkt[i][1];
   if (minz > dPkt[i][2]) minz = dPkt[i][2];
   if (maxz < dPkt[i][2]) maxz = dPkt[i][2];
  }
  dBox[0][0] = minx; dBox[0][1] = miny; dBox[0][2] = minz;
  dBox[1][0] = minx; dBox[1][1] = maxy; dBox[1][2] = minz;
  dBox[2][0] = maxx; dBox[2][1] = maxy; dBox[2][2] = minz;
  dBox[3][0] = maxx; dBox[3][1] = miny; dBox[3][2] = minz;
  dBox[4][0] = minx; dBox[4][1] = miny; dBox[4][2]=maxz;
  dBox[5][0] = minx; dBox[5][1] = maxy; dBox[5][2] = maxz;
  dBox[6][0] = maxx; dBox[6][1] = maxy; dBox[6][2] = maxz;
  dBox[7][0] = maxx; dBox[7][1] = miny; dBox[7][2] = maxz;
 }// Ende boxWerkstueck
```

Es werden noch Methoden zum Zeichnen benötigt. Zunächst wählen wir eine zum Grundriss orthogonale Abbildung. Auf der Zeichenfläche, hier mit c bezeichnet, werden nur die Komponenten x und y ausgegeben. Die Tiefe z wird ignoriert, wir erhalten damit eine Parallelprojektion.

```
public static void zeichneWerkstueckXY(Graphics c){
  c.setColor(Color.yellow);
  // Zeichne Draufsicht xy Grundriss
  int []x = new int [n2];  // untere Ebene
  int []y = new int [n2];
  for (int i= 0; I < n2;i++){
   x[i] = (int)dPkt[i][0]; // Typumwandlung
   y[i] = (int)dPkt[i][1];
  }
  c.fillPolygon(x,y,9);
  c.setColor(Color.black);
  for (int i= 0; I < n2;i++){// obere Ebene
   x[i] = (int)dPkt[i+n2][0];
   y[i] = (int)dPkt[i+n2][1];
  }
  c.drawPolygon(x,y,9);    // senkrechte Linien
  for (int i= 0; i < n2;i++){
   x[0] = (int)dPkt[i][0]; y[0] = (int)dPkt[i][1];
```

```
   x[1] = (int)dPkt[i+n2][0]; y[1] = (int)dPkt[i+n2][1];
   c.drawLine(x[0],y[0],x[1],y[1]);
  }
 }// Ende zeichneWerkstueckXY
```

Die untere Ebene des Werkstücks wird zur besseren Orientierung in der Abbildung mit `fillPolygon()` und unterschiedlicher Farbe dargestellt. Mit `drawPolygon()` wird die obere Ebene gezeichnet. Zur Darstellung der senkrechten Linien wird `drawLine()` benutzt. Die Koordinaten der Zeichnung werden in einem temporären Integer-Array gespeichert. Mit `(int)` werden die double-Werte „gecastet" (explizite Typumwandlung).

Damit wir später bei der Grafikausgabe flexibler sind, werden die Achsen und die umhüllende Box getrennt gezeichnet. Man beachte die unterschiedlichen Farben der Achsen (x: rot, y: grün, z: blau). In gleicher Farbe werden im Applet die Rotationen wx, wy und wz markiert. Das ist hilfreich bei der Betrachtung des gedrehten Bauteils. Die Typenumwandlung wird unmittelbar in der Argumenteliste vorgenommen.

```
 public static void zeichneAchsenXY (Graphics c){
  c.setColor(Color.red);
  c.drawLine((int) dAchsen[0][0],
   (int)dAchsen[0][1],(int) dAchsen[1][0],
(int)dAchsen[1][1]);
  c.setColor(Color.green);
  c.drawLine((int) dAchsen[0][0],(int)dAchsen[0][1],
    (int) dAchsen[2][0],(int)dAchsen[2][1]);
  c.setColor(Color.blue);
  c.drawLine((int) dAchsen[0][0],(int)dAchsen[0][1],
    (int) dAchsen[3][0],(int)dAchsen[3][1]);
 } // Ende zeichneAchsenXY
 public static void zeichneBoxXY(Graphics c){
  c.setColor(Color.gray);
  // Zeichne Draufsicht xy - Grundriss
  int []x = new int [4];   // untere Ebene
  int []y = new int [4];
  for (i= 0; i<4;i++){
   x[i] = (int)dBox[i][0]; y[i] = (int)dBox[i][1];
  }
  c.drawPolygon(x,y,4);
  for (i= 0; i<4;i++){     // obere Ebene
   x[i] = (int)dBox[i+4][0]; y[i] = (int)dBox[i+4][1];
  }
  c.drawPolygon(x,y,4);   // senkrechte Linien
  for (i= 0; i<4;i++){
```

```
    x[0] = (int)dBox[i][0]; y[0] = (int)dBox[i][1];
    x[1] = (int)dBox[i+4][0]; y[1] = (int)dBox[i+4][1];
    c.drawLine(x[0],y[0],x[1],y[1]); }
   // Beschriftung
   Font txt = new Font ("Courier",Font.BOLD,14);
   c.setFont(txt);
   c.drawString("Grundriss XY", (int) minx,(int) (maxy+20));
   } // Ende zeichneBoxXY
} // Ende Klasse Werkstueck
```

Bei entsprechender Modifikation der Parameterliste wäre für die Zeichnung auch nur eine Methode hinreichend. Die hier vorhandene Redundanz sollte man möglichst vermeiden. Man reduziert Fehlerquellen und verkürzt den Programmcode.

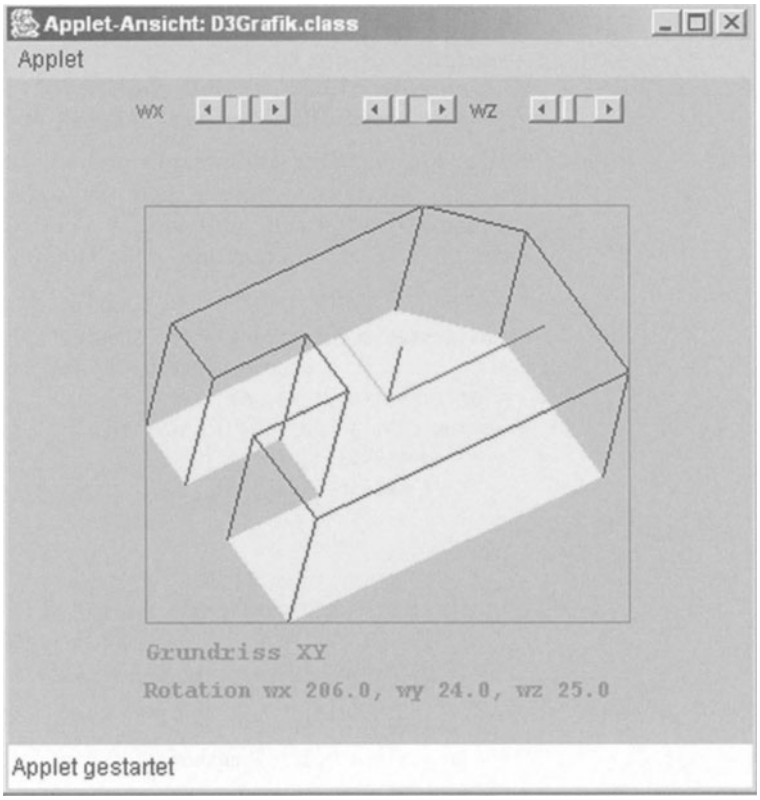


Abb. 4.7: Screenshot Werkstück in Parallelprojektion und Benutzerinterface

Wenden wir uns nun den Transformationen zu, die der Modellierung dienen: Skalierung, Translation und Rotation. Die Methoden der Klasse sind eingangs kommentiert.

```
public class Transform {
// Klasse Transform stellt Methoden zur Koordinatentransforma-
tion bereit
// vertex [n][3] * matrix[3][3]
// Translation:
//    translation (double t[3],double vertex[][],int n)
// Skalierung:
//    skalierung  (double s[3],double vertex[][],int n)
// Rotationen:
//  Definition der Rotationsmatrix:
//    rotation_x   (double rot,double vertex[][],int n)
//    rotation_y   (double rot,double vertex[][],int n)
//    rotation_z   (double rot,double vertex[][],int n)
//  Durchfuehrung derRotation:
//      private: vertexMatrixMult
// Ausgabe auf die Konsole:
//   printMatrix3x3   (double matrix[][])
//   printKoordinaten (int npts, double vertex[][])
```

Skalierung und Translation sind durch skalare Multiplikation gelöst. Für Rotationen werden getrennt nach Achsen die Rotationsmatrizen aufgestellt und anschließend unmittelbar die Matrixmultiplikation durchgeführt. Die Multiplikation ist daher als `private` deklariert.

```
public static void translation (double t[],
                 double vertex[][], int n){
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < 3; j++ ) {
      vertex[i][j] += t[j];
      // skalare Addition
    }
  }
}
public static void skalierung (double s[],
                               double vertex[][], int n) {
 for (int i = 0; i < n; i++) {
  for (int j = 0; j < 3; j++ ) {
    vertex[i][j] *= s[j];
    // skalare Multiplikation
  } // Ende for j
 } // Ende for i
} // Ende skalierung
```

Betrachten wir nun die Rotationen. `Math.sin(rot)` benötigt ein Argument im Bogenmaß. Bei Aufruf der Rotation ist die Umwandlung vorzunehmen. Die Aufstellung der Matrizen werden

wir hier nicht diskutieren. Lassen Sie sich von der Korrektheit durch das Grafikprogramm überzeugen.

```
public static void rotation_x ( double rot,
                                double vertex[][], int n) {
  double sin_rot = Math.sin (rot);
  double cos_rot = Math.cos (rot);
  double matrix [][] = new double[3][3];
  for (int i=0; i<3; i++) {
   for (int j= 0; j<3; j++) {
    matrix [i][j] = 0.0d;
    if (i == j) matrix [i][j]=1.0d;
   } }
  matrix[1][1]=  cos_rot;matrix[1][2]= -sin_rot;
  matrix[2][1]=  sin_rot;matrix[2][2]=  cos_rot;
  vertexMatrixMult (matrix,vertex,n); }
```

Eine rechteckige Matrix von n Zeilen und m Spalten kann mit einer quadratischen Matrix m x m multipliziert werden, indem die Summe der Produkte korrespondierender Elemente einer Zeile der Rechteckmatrix mit der Spalte der Quadratmatrix gebildet wird. Das Ergebnis ist wieder eine Matrix der Dimension n x m und enthält in diesem speziellen Fall die gedrehten Eckpunktkoordinaten unseres Werkstücks.

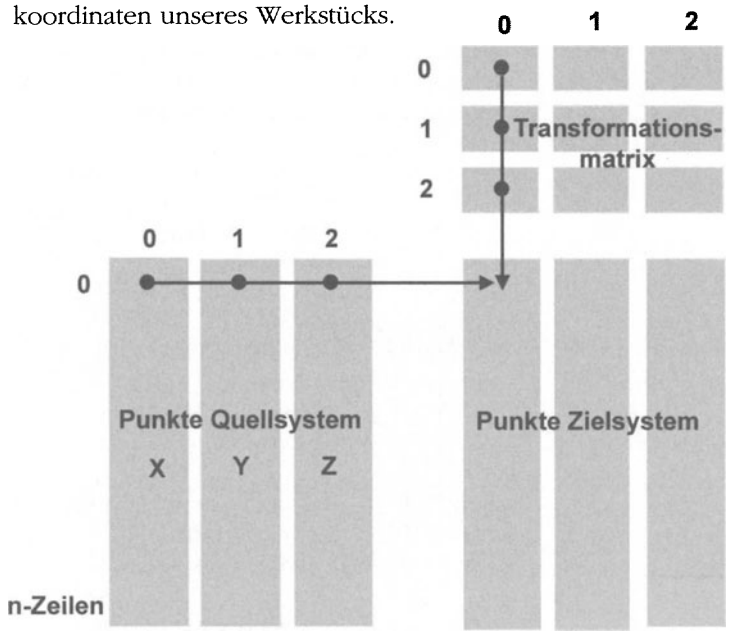


Abb. 4.8: Prinzip der Matrixmultiplikation

Wenn wiederholt kontrolliert rotiert werden soll, dann ist wieder von der Ausgangslage auszugehen, daher die Methode `resetWerkstueck()`. Ohne Arrays wären wir bei derart umfangreichen Berechnungen ziemlich hilflos. Die Grafik, Abbildung 4.8, versucht das Verfahren der Matrizenmultiplikation zu verdeutlichen. Für Programmieranfänger ist die Funktion nicht leicht durchschaubar. Erinnern Sie sich noch an die Einleitung? Geduld, Beharrlichkeit, Ausdauer ?

Zunächst zur Vervollständigung noch die Drehungen um die Achsen Y und Z. Und anschließend die Methode der Matrixmultiplikation. Damit wäre das Modellierwerkzeug bereitgestellt.

```
public static void rotation_y ( double rot ,
                                double vertex[][], int n) {
  double sin_rot = Math.sin (rot);
  double cos_rot = Math.cos (rot);
  double matrix [][] = new double[3][3];
   for (int i=0; i<3; i++) {
    for (int j= 0; j<3; j++) {
     matrix [i][j] = 0.0d;
     if (i == j) matrix [i][j]=1.0d;
    }
   }
   matrix[0][0]=  cos_rot; matrix[0][2]=  sin_rot;
   matrix[2][0]= -sin_rot; matrix[2][2]=  cos_rot;
   vertexMatrixMult (matrix,vertex,n);
  }
  public static void rotation_z ( double rot ,
                  double vertex[][], int n ){
  double sin_rot = Math.sin (rot);
  double cos_rot = Math.cos (rot);
  double matrix [][] = new double[3][3];
  for (int i=0; i<3; i++) {
   for (int j= 0; j<3; j++) {
    matrix [i][j] = 0.0d;
    if (i == j) matrix [i][j]=1.0d;
   }
  }
  matrix[0][0]=  cos_rot;  matrix[0][1]= -sin_rot;
  matrix[1][0]=  sin_rot;  matrix[1][1]=  cos_rot;
  vertexMatrixMult (matrix,vertex,n);
 }
 private static void vertexMatrixMult
    (double m[][],double vertex[][],int n ){
  // Multiplikation der Vektoren mit der
```

```
// Rotationsmatrix
double sum;
double help[] = new double[3];
for (int i = 0; i < n; i++) {
 for (int j = 0; j<3; j++) {
  help[j] = 0.0d;
  for (int k = 0; k<3; k++) {
   help[j] += vertex[i][k]*m [k][j];
  }
 }
 for (int l = 0; l<3; l++) {
  vertex [i][l] = help[l];
  // Umspeichern Hilfsvektor
 }
}// Ende for i
}// Ende Methode vertexMatrixMult
} // Ende Klasse Transform
```

Es existieren noch Methoden zur Ausgabe von Protokollen auf die Konsole. Auf deren Wiedergabe wird hier unter Verweis auf die Website verzichtet. Die letzte geschlossene Klammer zum Beenden der Klasse Transform bitte nicht vergessen.

Die Empfehlung, komplexere Lösungen immer schrittweise zu testen, gilt auch hier. Mit einer einfachen Appletklasse können wir zunächst die Funktionalität der bisherigen Programmteile testen.

```
import java.lang.*;
import java.applet.*;
import java.awt.*;
public class D3Grafik extends java.applet.Applet {
 Werkstueck a;
 double s[],t[],r[];
 final double rho = 180.0/Math.PI;
 Font txtFont;
 public void init(){
  s = new double [3];       // Skalierung
  t = new double [3];       // Translation
  r = new double [3];       // Rotation
  txtFont = new Font("Courier", Font.BOLD, 12);
  setBackground(Color.lightGray);
  a = new Werkstueck(); }
```

In der Init-Methode wird das Applet vorbereitet. Wir beschaffen uns Speicherplätze für Skalierung, Translation und Rotation.und definieren den Umrechnungsfaktor rho von Gradmaß in das Bo-

genmaß, bestimmen einen Textfont, definieren die Hintergrundfarbe und legen natürlich eine Instanz des Werkstücks an. Diese wird schlicht mit a bezeichnet.

In der Paint-Methode erfolgt die Definition von Skalierung und Translation. Hierbei müssen wir uns zunächst noch an dem Bildschirmkoordinatensystem orientieren. Das Werkstück bekommt Abmessungen von etwa 200 Einheiten und wird bei 100, 100 positioniert. Wir können Kontrollausdrucke vornehmen und die Rotationsparameter ebenfalls „fest verdrahten", durch Konstante vorgeben.

Die Anzeige des gedrehten Modell kapseln wir aber wieder in einer Methode `displayModell()` ab, denn hier gibt es noch einiges zu tun. `DisplayRotation()` zeigt uns nur die Drehwinkel an.

```
public void paint(Graphics pen){
  // Parameter fuer Skalierung und
  // Transformation  zuweisen
  s[0]=100.0; s[1]=100.0; s[2]=100.0;
  t[0]=100.0; t[1]=100.0; t[2]=0.0;
  a.resetWerkstueck(s,t);//vor Neuzeich. reset
  a.boxWerkstueck();// umhuellende Box ber.
   // Testausgaben auf die Konsole
  System.out.println("Box in X " +a.minx +" "
                     +a.maxx);
  System.out.println("Box in Y " +a.miny +" "
                     +a.maxy);
  System.out.println("Box in Z " +a.minz +" "
                     +a.maxz);
  // Rotation zuweisen
  r[0]=-45.0/rho;r[1]=45.0/rho;r[2]=45.0/rho;
  displayModell  (pen, r); // Modell zeigen
  displayRotation(pen, r); // Drehwinkel zeigen
 }
```

Damit die Rotation das Modell nicht aus dem sichtbaren Bereich dreht, wird zunächst eine Verschiebung in den Modellmittelpunkt vorgenommen. Wir kennen ja die umhüllende Box, bestimmen den Mittelpunkt, speichern diesen in den Translationsvektor und verschieben unser Modell einschließlich Achsen von hier in den Ursprung (negative Translation). Im Ursprung ist die Drehung kontrolliert durchzuführen. Nach der Drehung wird die Translation wieder zurückgenommen.

```
public void displayModell(Graphics pen, double r[]){
  // Mittelpunkt des Werkstuecks
  t[0]=(a.maxx+a.minx)/2.0 *(-1.0);
```

```
  t[1]=(a.maxy+a.miny)/2.0 *(-1.0);
  t[2]=(a.maxz+a.minz)/2.0 *(-1.0);
  // Mittelpunkt -> Ursprung
  Transform.translation(t,  a.dPkt, a.n);
  Transform.translation(t,  a.dAchsen,4);
  // Drehung durchfuehren
  Transform.rotation_x(r[0], a.dPkt,a.n);
  Transform.rotation_y(r[1], a.dPkt,a.n);
  Transform.rotation_z(r[2], a.dPkt,a.n);
  Transform.rotation_x(r[0], a.dAchsen,4);
  Transform.rotation_y(r[1], a.dAchsen,4);
  Transform.rotation_z(r[2], a.dAchsen,4);
  // Translation wieder aufheben
  t[0]= t[0] *(-1.0);
  t[1]= t[1] *(-1.0);
  t[2]= t[2] *(-1.0);
  // Ursprung -> Mittelpunkt
  Transform.translation(t, a.dPkt, a.n);
  Transform.translation(t, a.dAchsen, 4);
  a.boxWerkstueck();//wir brauchen auch die Box
  a.zeichneWerkstueckXY(pen); // Grafikausgabe
  a.zeichneBoxXY(pen);
  a.zeichneAchsenXY(pen);
 }
```

Und jetzt noch die Ausgabe der Drehwinkel mit drawString():

```
public void displayRotation(Graphics pen, double r[]){
  double wx,wy,wz;
  wx = (int)(r[0]*rho*100)/100.0;
  wy = (int)(r[1]*rho*100)/100.0;
  wz = (int)(r[2]*rho*100)/100.0;
  pen.setColor(Color.gray);
  pen.setFont(txtFont);
  pen.drawString("Rotation wx " +wx +", wy "
                 +wy + ", wz " +wz,
                 (int) a.minx,
                 (int)(a.maxy+40));

  }//Ende displayRotation
} // Ende Klasse D3Grafik
```

Beim Betrachten der Grafik stellen wir fest, dass die Orientierung des Werkstückes noch durch das Bildschirmkoordinatensystem verändert wird. Als nächsten Schritt werden wir eine perspektive Abbildung vornehmen und eine Abbildungsvorschrift formulie-

ren, die einen rechteckigen Bereich unseres Bildes auf einen rechteckigen Bereich der Zeichenfläche anpasst.

Abbildungstransformationen: Perspektive und Anpassung an das Ausgabegerät

Sofern bei einer zentralperspektiven Abbildung die Blickrichtung bereits entlang einer Achsrichtung des Modells orientiert ist, reduziert sich diese auf eine einfache Verhältnisrechnung. Der Betrachtungsabstand (Abstand zum Objekt), die unterschiedlichen Tiefen der Objektpunkte und der Bildabstand (Brennweite) beeinflussen die Perspektive. Wir nehmen ein einfaches Kameramodell an. Da wir die umhüllende Box kennen, sind unsere Bildkoordinaten parallel zur x-Richtung und z-Richtung des raumfesten Koordinatensystems. Unsere Blickrichtung ist horizontal und zeigt auf die Mitte des Objekts, vgl. Abbildung 4.9.

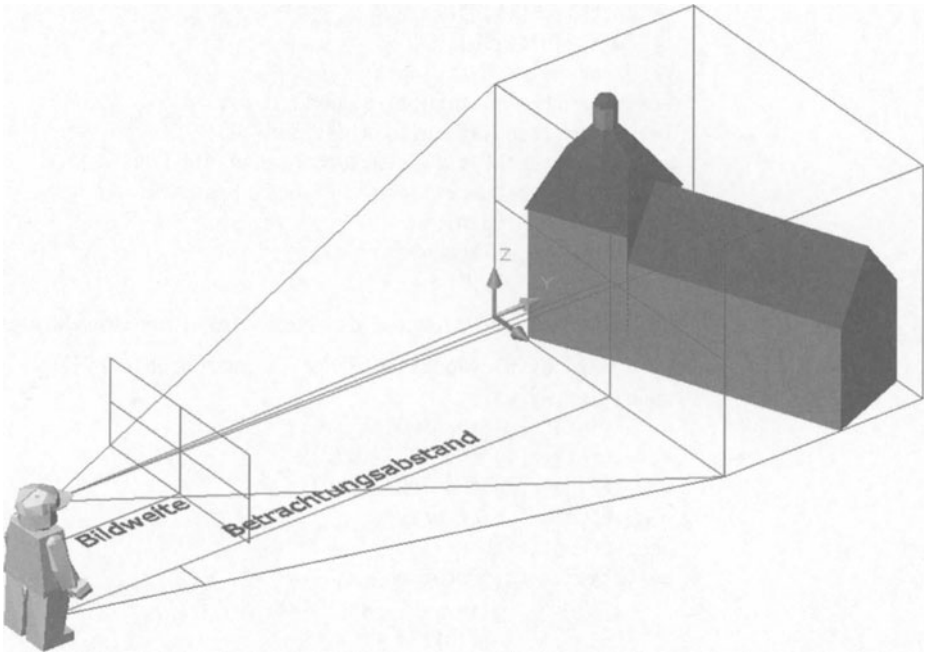


Abb. 4.9: Zentralprojektion, Kameramodell. Objektsystem parallel zum Betrachtungssystem. Der Ursprung dess Objektkoordinatensystems (Symbol) liegt in der Mitte der Box, die das Objekt umhüllt.

Was ist nun zu tun? Die bekannten Methoden `Werkstueck.zeichneWerkstueckXY()`,`Werkstueck.zeichneBoxXY()`,`Werkstueck.zeichneAchsenXY()` sind jetzt nicht mehr brauchbar und müssen alle modifiziert werden. Die neuen Bezeichner der Methodennamen entstehen, indem *`XY`* in den Methodennamen durch *`Zentral`* ersetzt wird. Wir können nun

nicht mehr die Komponenten der Modellkoordinaten abbilden, wir müssen uns ein temporäres zweidimensionales Array für Bildkoordinaten beschaffen. Die Bildkoordinaten berechnen und auf das Gerätedarstellungsfenster transformieren. Das machen für uns die Methoden `D3Grafik.transformRaumBild()` und `D3Grafik.transformWindowViewport()`. Die Parameterliste wurde erweitert um den Betrachtungsabstand und die Bildweite. Denn diese sollen ja auch später im Benutzerinterface veränderbar sein. Gezeichnet wird jetzt nach Typumwandlung mit den Bildkoordinaten. Die anderen Zeichenmethoden der Klasse `Werkstueck` sind entsprechend zu bearbeiten.

```
public static void zeichneWerkstueckZentral
 (Graphics c,double abstand, double fokus){
 // Perspektiv-Transformation
 double [][] biko = new double [18][3];
 D3Grafik.transformRaumBild  (dPkt,n,abstand,fokus,biko);
 D3Grafik.transformWindowViewport(biko,n);
 c.setColor(Color.yellow);
 // Zeichne Bildkoordinaten
 int []x = new int [n2];   // untere Ebene
 int []y = new int [n2];
 for (i= 0; i<n2;i++){
  x[i] = (int)(biko[i][0] ); y[i] = (int)(biko[i][1] );
 }
 c.fillPolygon(x,y,9);
 c.setColor(Color.black);
 for (i= 0; i<n2;i++){     // obere Ebene
  x[i] = (int)(biko[i+n2][0] );
  y[i] = (int)(biko[i+n2][1] );
 }
 c.drawPolygon(x,y,9),// senkrechte Linien
  for (i= 0; i<n2;i++){
   x[0] = (int)(biko[i][0] ); y[0] = (int)(biko[i][1] );
   x[1] = (int)(biko[i+n2][0] ); y[1] = (int)(biko[i+n2][1]
);
   c.drawLine(x[0],y[0],x[1],y[1]);
  }
 }// Ende zeichneWerkstueckZentral
```

Die Umrechnung in Bildkoordinaten ist recht einfach. Die parallel zur Bildebene verlaufenden Komponenten der Punkte werden zum Betrachtungsabstand und der Bildweite ins Verhältnis gesetzt. Hieraus ergeben sich die Bildkordinaten. Die Berechnung der Bildkoordinaten entsprechend der Methode `transformRaumBild()` wird anhand der Grafik, Abb. 4.9, anschaulich.

Die Bildweite ist in der Methode mit fokus bezeichnet. Zum Betrachtungsabstand wird die y-Koordinate x[i][1]addiert.

```
public static void transformRaumBild
                (double x[][],int nanz,
                 double abstand, double fokus,
                 double bk[][]){
  // Raum-Bild-Transformation / Perspektive
  for (int i = 0; i < nanz; i++){
    bk [i][0] = (x[i][0]/(x[i][1]+abstand))*fokus ;
    bk [i][1] = (x[i][2]/(x[i][1]+abstand))*fokus ;
  }
```

Es liegen jetzt zweidimensionale Bildkoordinaten vor. Diese müssen geeignet auf die Zeichenfläche abgebildet werden. Man definiert einen rechtwinkligen Bereich im Bild (*window*) und einen entsprechenden Bereich auf der Zeichenfläche (*viewport*). Die hieraus resultierenden Transformationsparameter werden einmalig in der Init-Methode durch Aufruf von setWindowViewportTrans() gesetzt. Vor der Zeichnungsausgabe der Bildkoordinaten erfolgt dann immer die zweidimensionale Abbildungstransformation. Durch entsprechende Angabe der Viewport-Koordinaten kann jetzt die Orientierung des Bildschirmkoordinatensystems umgekehrt werden, wir spiegeln unser Objekt an der Horizontalen, vgl. vPort[2] und vPort[3] mit window[2] und window[3] in der Abbildung 4.10. Es ist darauf zu achten, dass die Seitenverhältnisse der Rechtecke gleich sind. Im vorliegenden Fall ist das Darstellungsfenster mit 360px x 240px angegeben. Das Bildfenster hat eine Größe von 36 x 24 Einheiten mit dem Ursprung in der Mitte. Fotografen werden das Kleinbildformat in unserem Kameramodell erkennen. Die Festlegungen finden Sie in der Init-Methode des Applets.

```
public static void setWindowViewportTrans(){
  // 2D-Transformationsmatrix bestimmen
  // window, vPort global deklariert
  // und initialisiert
  double winx,winy,viewx,viewy;
  winx=window[1]-window[0];  winy=window[3]-window[2];
  viewx = vPort[1]-vPort[0]; viewy = vPort[3]-vPort[2];
  transX = viewx/winx;
  transY = viewy/winy; }
public static void transformWindowViewport
                (double BiKo[][],int nBiko ){
  // Bildkoordinaten auf Zeichenflaeche
  // abbilden
```

```
  for (int i=0; i<nBiko; i++){
   BiKo[i][0]=((BiKo[i][0]-window[0])*transX )
                + vPort[0];
   BiKo[i][1]= ((BiKo[i][1]-window[2])*transY )
                + vPort[2];
  }
} // Ende transformWindowViewport
```

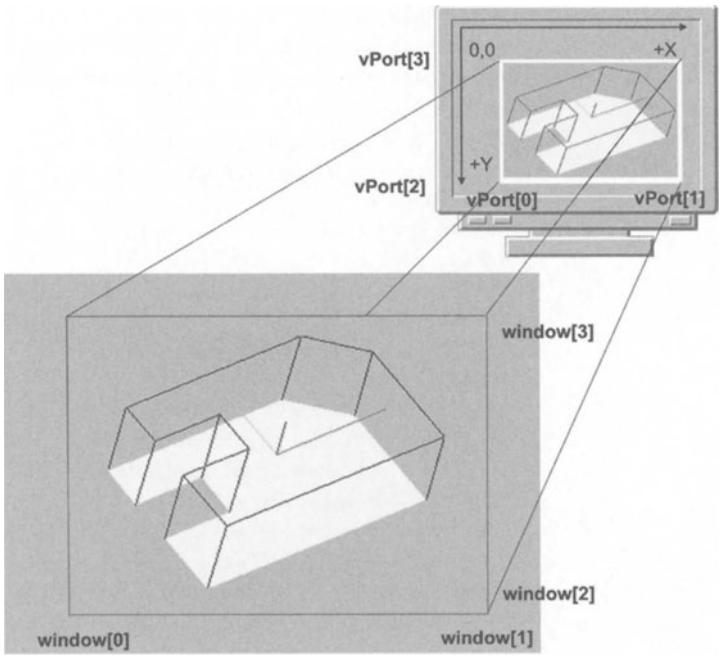


Abb. 4.10: Transformation eines Bildausschnitts

Das Benutzerinterface

Die Geometrie haben wir jetzt sicher im Griff. Alle Klassen und Methoden sind getestet. Wir müssen für unser 3D-Programm nur noch das Benutzerinterface konstruieren. Die 3D-Rotationen sollen durch Schieberegler steuerbar sein. Nach Einstellung einer Rotation soll unmittelbar die Neuzeichnung erfolgen. Den Betrachtungsabstand und die Bildweite geben wir in ein Textfeld ein. Für die Schieberegler müssen wir sog. AdjustmentListener installieren. Das erfolgt durch die Angabe `implements AdjustmentListener`. Wir deklarieren die benötigten Variablen und die Bedienelemente der Klassen `Scrollbar`, `Label` und `TextField`. Letzere kennen wir schon aus vorhergenden Abschnitten dieses Kapitels.

Die Scrollbars werden mit folgenden Parametern versehen: `Scrollbar.HORIZONTAL` oder `Scrollbar.VERTICAL` gibt die Orientierung an. Der folgende Parameter gibt den Anfangswert des Schiebereglers an. Danach folgt die Weite des Reglers sowie der Bereich, hier 0 bis 360 entspricht einem Vollkreis.

```
import java.lang.*;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
// Einfuehrung in die 3D-Grafik - perspektive  // Abbildung
public class D3Grafik extends
             java.applet.Applet
             implements AdjustmentListener {
 Werkstueck a;
 static double s[],t[],r[], abstand, fokus;
 static double rho = 180.0/Math.PI;
 static double [] window    = new double[4];
 static double [] vPort     = new double[4];
 static double transX,transY;
 Font txtFont;
 Scrollbar rotX=new Scrollbar(Scrollbar HORIZONTAL,0,1,0,360);
 Scrollbar rotY=new Scrollbar(Scrollbar.HORIZONTAL,0,1,0,360);
 Scrollbar rotZ=new Scrollbar(Scrollbar.HORIZONTAL,0,1,0,360);
 Label rX = new Label("wx");
 Label rY = new Label("wy");
 Label rZ = new Label("wz");
 Label lDist        = new Label("Abstand");
 Label lFokus       = new Label ("Fokus");
TextField txtDist  = new TextField();
TextField txtFokus = new TextField();
```

Wir können jetzt innerhalb der Init-Methode das Applet vorbereiten. Die Fenstergroesse wird auf 500 px x 400 px gesetzt. `Window[]`und `vPort[]` haben wir oben kennengelernt. Auch nehmen wir Voreinstellungen für `transX` und `transY` vor, diese werden aber durch `setWindowViewPortTrans` später überschrieben. Es wird ein einfaches `Flowlayout` deklariert. Die Arrays für Rotation und Translation werden reserviert, Eingabefelder vorbereitet und die Komponenten dem Applet hinzugefügt.

Jeder Schieberegler benötigt einen AdjustmentListener, einen Lauscher im Hintergrund, der die Regler beobachtet und auf das Ereignis Schieberegler reagiert. Wir benutzen einen Listener Regler. Welcher Regler bedient wurde, wird durch Entscheidung über das Objekt später abgefragt. Letztlich ist noch eine Instanz

des Werkstücks aufzurufen und die Abbildungstransformation unserer Klasse (`this`) für alle mit den Reglern eingestellten Transformationen unter Berücksichtigung des Betrachtungsabstandes und der Brennweite durchzuführen.

```
public void init(){
 this.setSize(500,400 );
 abstand = 400; fokus = 35;
 window[0]= -18.;  window[1] = 18.;
 window[2]= -12.;  window[3] = 12.;
 vPort[0] = 50.0;  vPort[1]  = 410.0;
 vPort[2] = 290.0; vPort[3]  = 50.0;
 transX   = 1.0;   transY    = 1.0;
 setLayout(new FlowLayout());
 s = new double [3];         // Skalierung
 t = new double [3];         // Translation
 r = new double [3];         // Rotation
 txtDist = new TextField("400",4);
 txtFokus = new TextField("35",4);
 txtFont=new Font ("Courier", Font.BOLD, 12);
 setBackground(Color.lightGray);
 setForeground(Color.red);
 add(rX); add(rotX);
 setForeground(Color.green);
 add(rY); add(rotY);
 setForeground(Color.blue);
 add(rZ); add(rotZ);
 setForeground(Color.black);
 add(lFokus);add(txtFokus);
 add(lDist);add(txtDist);
  rotX.addAdjustmentListener(this);
  rotY.addAdjustmentListener(this);
  rotZ.addAdjustmentListener(this);
  a = new Werkstueck();
  setWindowViewportTrans();
 } // Ende init
```

Es folgt die Methode paint mit der Bereitstellung des Grafik-Kontexts.

```
public void paint(Graphics pen){
 pen.setColor(Color.white);
 pen.fillRect((int)vPort[0],
              (int)vPort[3] ,360,240);
 // Anfangswerte fuer
 // Skalierung und Translation
 s[0]=100.0; s[1]=100.0; s[2]=100.0;
```

```
  t[0]=-100.0; t[1]=100.0; t[2]=-50.0;
  a.resetWerkstueck(s,t);//vor Neuzeich. reset
  a.boxWerkstueck(); // umhuellende Box ber.
  displayModell(pen, r);// Modell zeigen
  displayRotation(pen,r);
  } // Ende paint
```

Die Methode `displayModell()` wird mit neuen Aufrufen zum Zeichnen des Modells ausgestattet. Dahinter folgen noch `displayRotation()` und die oben beschriebenen Methoden zur Abbildungstransformation.

Wir müssen aber noch einen Blick auf `adjustmentValueChanged()` werfen. Diese Methode muss vorhanden sein, wenn wir einen AdjusmentListener implementieren. Das auslösende Ergeignis `e` wird mit übergeben. Wir werten die Textfelder `fokus` und `Abstand` immer bei der Gelegenheit aus und richten ein Objekt der Klasse `Object` ein, dem wir mit `e.getSource()` die Objektbezeichnung zuweisen. Anhand dieser Bezeichnung kann der entsprechend eingestellte Wert ermittelt werden und wird auch gleich in das Bogenmaß umgerechnet.

```
public void adjustmentValueChanged
            (AdjustmentEvent e){
  fokus =Double.valueOf(txtFokus.getText()).doubleValue();
  abstand=Double.valueOf(txtDist.getText()).doubleValue();
  System.out.println(" " +abstand +" " +fokus );
  Object obj = e.getSource();
  if(obj==rotX){
   r[0] =  e.getValue()/rho;}
  if(obj==rotY){
   r[1] =  e.getValue()/rho;}
  if(obj==rotZ){
   r[2] =  e.getValue()/rho;}
  repaint();
}
```

Hier noch die Modifikation von `displayModell()`. Im Zusammenhang finden Sie den kompletten Sourcecode auf der Website.

```
public void displayModell(Graphics pen, double r[]){
  // Schwerpunkt des Werkstücks
  t[0]=(a.maxx+a.minx)/2.0 *(-1.0);
  t[1]=(a.maxy+a.miny)/2.0 *(-1.0);
  t[2]=(a.maxz+a.minz)/2.0 *(-1.0);
  // Mittelpunkt -> Ursprung
```

```
 Transform.translation(t,  a.dPkt, a.n);
 Transform.translation(t,  a.dAchsen,4);
 // Rotation
 Transform.rotation_x(r[0], a.dPkt,a.n);
 Transform.rotation_y(r[1], a.dPkt,a.n);
 Transform.rotation_z(r[2], a.dPkt,a.n);
 Transform.rotation_x(r[0], a.dAchsen,4);
 Transform.rotation_y(r[1], a.dAchsen,4);
 Transform.rotation_z(r[2], a.dAchsen,4);
 t[0]= t[0] *(-1.0);  // Ruecktranslation
 t[1]= t[1] *(-1.0);
 t[2]= t[2] *(-1.0);
 // Ursprung -> Mittelpunkt
 Transform.translation(t, a.dPkt, a.n);
 Transform.translation(t, a.dAchsen, 4);
 a.zeichneWerkstueckZentral(pen, abstand,  fokus);
 a.boxWerkstueck();
 a.zeichneBoxZentral(pen, abstand, fokus);
 a.zeichneAchsenZentral(pen,abstand,fokus);
}
```

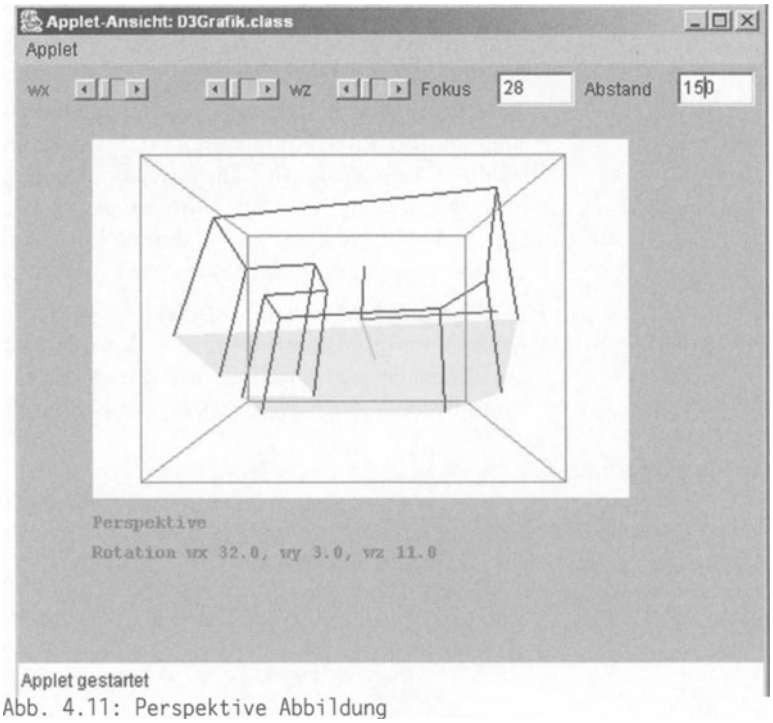


Abb. 4.11: Perspektive Abbildung

4.8 Zusammenfassung

Sie haben es geschafft. Objektorientiertes Programmieren mit Java ist Ihnen nicht mehr fremd. Mit Kenntnis der Programmstruktur und den Basisbefehlen kann man kleinere Problemlösungen erstellen. Auch die Programmierung von GUI-Oberflächen ist lösbar, wenn man das Problem modularisiert.

Am Beispiel eines 3D-Viewers haben wir die Prinzipien und Komplexität von 3D-Grafiken aufgezeigt. Ein Körper wird modelliert, in das Betrachtungssystem transformiert und auf die Bildebene reduziert. Ein rechteckiger Bereich der Bildebene wird in einem rechteckigen Bereich des Bildschirms abgebildet. Mit der Klasse `Transform` stehen Transformationsfunktionen bereit, die als `static` deklariert sind und daher auch ohne Instanzbildung aufgerufen werden können. Das Wissen zur 3D-Grafik bekommt man aber nicht geschenkt, man muss es sich erarbeiten. Sie haben Recht: Auch die Arrays, Methoden und For-Loops machten Anfangsprobleme.

Wir haben in den Anwendungen nur das einfache Java-AWT benutzt. Erweiterte 2D-Graphikmöglichkeiten existieren unter der Klassenbibliothek *Swing*.

Das hinter Ihnen liegende Kapitel sollte Anregungen zum Einstieg in die Programmiersprache Java geben. Einerseits ist der schnelle Erfolg mit der Basissyntax erkennbar, die Mächtigkeit von Java konnte aber im Rahmen dieser Einführung nicht deutlich gemacht werden. Es ist darauf hinzuweisen, dass Java eine Technologie ist und nicht nur eine Programmiersprache.

Sie können jetzt eigene Applets schreiben und frei verfügbare Applets aus dem Internet in Ihre Webseiten einbauen. Zur weiteren Einarbeitung empfehle ich Ihnen das Studium eines der im Literaturverzeichnis angegeben Java-Lehrbücher für Einsteiger.

5 JavaScript

Wenn Sie mit (X)HTML und CSS vertraut sind und sich intensiv mit dem vorausgegangenen Kapitel der Java-Applet-Programmierung auseinandergesetzt haben, dann wird Ihnen die Programmierung und Einbindung von JavaScript in HTML-Seiten keine Schwierigkeiten bereiten. Aber auch Neueinsteiger ohne Java-Kenntnisse können sich mit JavaScript in diesem Kapitel vertraut machen. Vorausgesetzt werden HTML-Grundkenntnisse, ohne die geht es nicht. Auch hier gilt wieder: Man muss nicht gleich jedes Detail verstehen. Erst verschafft man sich einen Überblick, danach vertieft man die Kenntnisse bei der praktischen Programmierung. Mit JavaScript kommen Sie schnell zu beeindruckenden Programmierergebnissen.

Im diesem Kapitel werden die Sprachelemente von JavaScript vorgestellt, die Entwicklung von Funktionen und die Handhabung von Objekten wird bearbeitet.

Wir setzten uns mit der Objekthierarchie, dem *Document Object Model* (DOM), auseinander und nutzen die vordefinierten Objekte mit deren Methoden zur Navigationsunterstützung und Gestaltungserweiterung von Web-Seiten.

Anhand eines Berechnungsbeispiels werden die mathematischen Fähigkeiten von JavaScript aufgezeigt und der Programmdialog mit Formularen realisiert.

Sie können nach Durcharbeitung dieses Kapitels eigene JavaScripts entwickeln und aus dem umfangreichen Web-Angebot frei verfügbarer Funktionen die geeigneten auswählen und in die eigenen Seiten implementieren.

5.1 Clientseitige Dynamik mit JavaScript

JavaScript ist konzipiert als Ergänzung der Funktionalität von Web-Browsern. Der Web-Browser kann den Inhalt einer Web-Seite nur statisch abbilden. Durch JavaScript kommt Dynamik auf die Seiten. Ohne eine Seite neu hoch zu laden, werden Elemente aufgrund von Benutzereingriffen dynamisch verändert. JavaScript stellt Dialogelemente bereit, bietet vordefinierte Objekte an, z. B.

mit dem Math-Objekt mathematische Funktionen, und gestattet über das DOM den Zugriff auf die Elemente einer Webseite und deren Eigenschaftsänderungen. Plattformunabhängigkeit besteht dadurch, dass JavaScript vom Web-Browser interpretiert wird. Die sehr erfolgreiche Entwicklung geht auf eine Kooperation der Firmen Sun und Netscape zurück. Mit JavaScript wurde eine einfache Programmiersprache entwickelt, die mit Java aber nur die Ähnlichkeit der Sprachelemente gemein hat. Neben den verschiedenen Versionen und Bezeichnungen für die Scriptsprache gilt als Standard die Bezeichnung ECMA-Script.

Hauptanwendungen von JavaScript sind u.a. die Unterstützung der Navigation durch Menüsteuerung, Kalenderfunktionen, Umgebungsabfragen, Animationen, Datumsfunktionen oder die Behandlung von Fenstern (pop-up-windows).

Durch die einfache Anordnung von Bedienelementen in den Formularen auf einer HTML-Seite entsteht mit wenig Aufwand eine komfortable Benutzeroberfläche. Die Auswertung von Formulardaten und deren Weiterverarbeitung bzw. Weiterleitung ist ein häufig vorkommendes Anwendungsgebiet von JavaScript.

Wenn Sie einen Währungsumrechner suchen oder Ihren Body-Maß-Index bestimmen wollen, dann werden Sie unter den JavaScript-Angeboten im Web sicher fündig. Die meist frei verfügbaren JavaScripts sind leicht in eigene Seiten zu integrieren.

JavaScript ist nicht Ersatz für HTML, sondern Ergänzung. Ergeben Sie sich nicht dem Multimediaoverkill – setzen Sie JavaScript gezielt ein. JavaScript ist keine Sicherheitslücke im Web. Abschreckend ist der Missbrauch, den Programmierer auf ihrer Seite treiben, indem sie dem Besucher Navigationswege aufzwingen, die rechte Maustaste sperren oder vergleichbare Restriktionen einbauen.

5.2 Einführung in JavaScript

Entwicklungsumgebung

Eine Entwicklungsumgebung für JavaScript braucht man eigentlich nicht. Sofern man den Quellcode einer HTML-Seite bearbeiten kann, ist man auch in der Lage JavaScript einzubinden. Es gibt Seiten die sind 100% mit Notepad erstellt. Die meisten HTML-Editoren bieten auch spezielle JavaScript-Unterstützung. Aber warum nicht einen JavaScript-Editor installieren? Antechinus von *c-point* kann in der Trial-Version von *www.c-point.com* heruntergeladen werden. Lizenzen liegen im angemessenen Sharewarebereich.

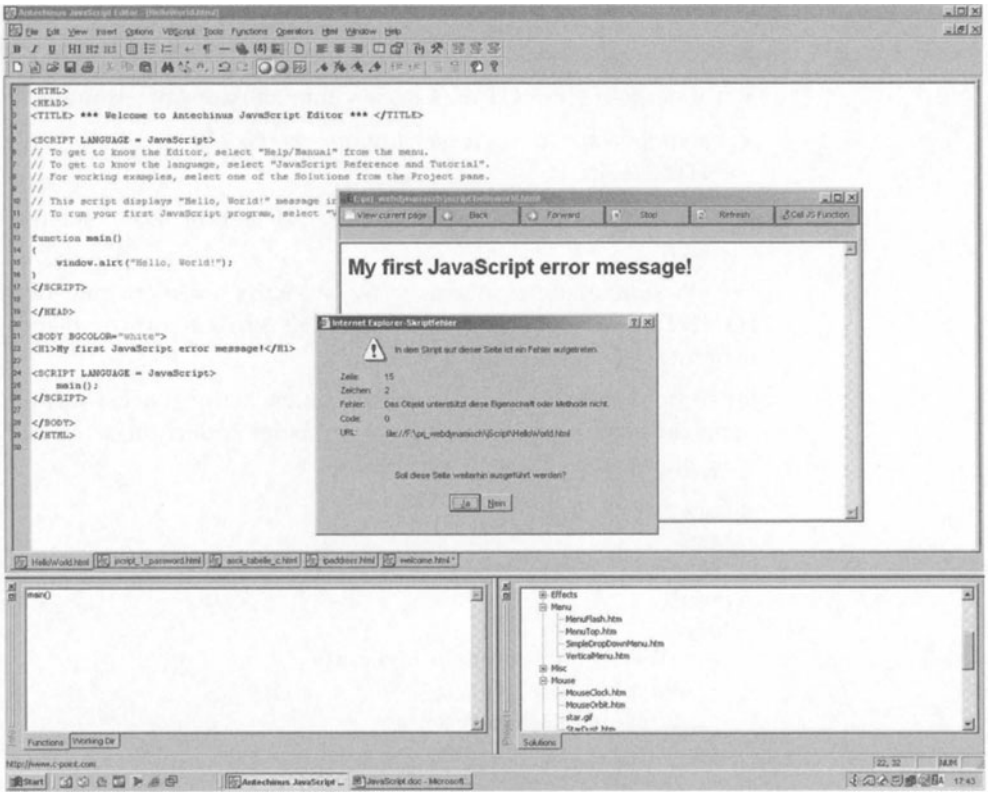


Abb. 5.1: Antechinus-JavaScript-Editor von *www.c-point.com*

Neben einer komfortablen Verwaltung der Projekte hat man Referenzen und Tutorials zur Hand. Das Editieren erfolgt wie gewohnt mit farblichen Markierungen. In einem eigenen Viewer werden die Funktionen getestet. Durch die Zeilennummerierung können Fehlermeldungen einfach lokalisiert werden. Bei neuen Projekten kann man aus Templates die notwendigen Voreinstellungen wählen. Code-Templates, Operatoren und Methodenaufrufe stehen in Auswahlmenüs zur Verfügung. HTML-Tags werden durch Klick auf ein Icon eingefügt. Abbildung 5.1 zeigt einen Screenshot nach einer Fehlermeldung im Script. Im Fenster unten rechts werden unter Solutions fertige Musterlösungen angeboten. Wenn Sie häufig JavaScript programmieren, lohnt sich der Einsatz dieses Editors.

Einbindung von JavaScript in HTML

Zur Einbindung von JavaScript in HTML-Programmcode bestehen die Möglichkeiten:

- innerhalb eines HTML-Tags als unmittelbare Anweisung
- mittels <script> Auszeichnung als Block innerhalb einer HTML-Seite
- im Kopf einer HTML-Seite oder als externe Datei referenziert

Die Programmanweisungen von JavaScript werden mit dem HTML-Tag <script> und dem Language-Attribut von der eigentlichen HTML-Auszeichnung getrennt.

JavaScript Anweisungen innerhalb eines Script-Blocks werden dann ausgeführt, wenn der Browser beim Laden diese Anweisungen erreicht.

```
<html>
 <head>
   <title>JavaScript in der HTML-Seite</title>
 </head>
 <body>
   <h1>JavaScript innerhalb HTML</h1>
   <p>Beispiel Anmeldung</p>
    <script language=javascript>
    <!--
      var nutzer = "Meier";
      var anmeldung = prompt("Geben Sie
                 Ihren Benutzernamen ein","");
      if (nutzer == anmeldung)
       alert("Willkommen " +nutzer);
      else
       alert("Sie sind nicht Nutzer Meier");
     -->
     </script>
     <p>Danke für Ihre Registrierung</p>
   </body>
</html>
```

Listing 5.1 JavaScript in HTML-Seiten, *js_block.html*

Mit dem Language-Attribut wird die Sprachversion spezifiziert. Die Anweisungen des Blocks sollten als Kommentar gekennzeichnet sein, damit der Browser ggf. diese Anweisungen überlesen kann. Im Script- Block wird eine Anmeldeprozedur simuliert, die einen Bildschirmdialog aktiviert.

Mit dem Schlüsselwort `var` wird die Variable `nutzer` deklariert und der Wert *Meier* implizit zugewiesen. Die folgende Variable `anmeldung` erhält durch Methodenaufruf `prompt()` ihren Wert. Innerhalb einer bedingten Anweisung wird der Benutzer Meier begrüßt oder alternativ mittels Aufruf der Methode `alert()` abgelehnt. Der JavaScript-Code ist in den Script-Tag eingebunden, der wiederum ist umgeben von HTML.

Ist der JavaScript-Code hingegen innerhalb einer Funktion definiert, dann muss diese explizit aufgerufen werden. Dieser Aufruf erfolgt ereignisgesteuert. Ereignisse können beispielsweise das Laden einer Seite `onload` oder ein Mausereignis `onMouseover`, `onclick` sein. Die Anweisungen der Funktion werden üblicherweise im Kopf der HTML-Seite eingetragen, sofern hier nicht Elemente referenziert werden, die noch nicht zur Verfügung stehen. Mit `function anmeldung(){}` haben wir jetzt eine Funktion mit der Bezeichnung *anmeldung* definiert. Der Anweisungsblock wird in geschweifte Klammern eingeschlossen. Der Aufruf erfolgt durch das mit dem Body-Tag ausgelöste Ereignis `onload`.

```
<html>
 <head>
   <title>JavaScript eingebettet </title>
   <script language=javascript>
   <!--
    function anmeldung(){
     var nutzer = "Meier";
     var anmeldung = prompt("Geben Sie Ihren
                        Benutzernamen ein","");
     if (nutzer == anmeldung)
      alert("Willkommen " +nutzer);
     else
      alert("Sie sind nicht Nutzer Meier");
    }
   -->
  </script></head>
 <body onload = 'anmeldung()'>
   <h1>JavaScript innerhalb HTML</h1>
   <p>Beispiel Anmeldung</p>
   <p>Danke für Ihre Registrierung</p>
 </body></html>
```

Listing 5.2 JavaScript embedded, *js_embedded.html*

Die Auslagerung von JavaScript in externe Dateien ist der Einbettung vorzuziehen. Der Programmcode kann mehrfach verwendet werden. So können Sie auch eine umfangreiche Scriptbibliothek unter der Erweiterungsbezeichnung `.js` bereitstellen. Mit dem Attribut `src` im Script-Tag referenzieren Sie die Datei, die jetzt nicht mehr den Script-Tag enthalten darf.

```
<html>
 <head>
    <title>JavaScript extern</title>
    <script language=javascript
       src="script_bibliothek.js">
    </script>
 </head>
 <body onload = 'anmeldung()'>
    <h1>JavaScript innerhalb HTML</h1>
    <p>Beispiel Anmeldung</p>
    <p>Danke für Ihre Registrierung</p>
   </body>
</html>
```

Listing 5.3 JavaScript extern, *js_extern.html*

Testen Sie die ersten einfachen Programme durch Aufruf von der Internetseite *www.programmierpraktikum.de.* Die HTML-Seiten werden in einem eigenen Browserfenster gestartet. Sie können den Quelltext im Editor modifizieren und in einer eigenen lokalen HTML-Datei speichern. Schon haben Sie Ihre ersten eigenen Java-Scripts. Die Datei *script_bibliothek.js* müssen Sie selbst erzeugen - cut & paste.

Objekthierarchie : Objekte, Methoden, Eigenschaften

Bei der Programmierung mit JavaScript hat man es auch mit Objekten zu tun. Objektorientierte Konzepte mit Klassen, Methoden, Eigenschaften und Instanzen wurden im Rahmen der Einführung in die Java-Programmierung diskutiert. Auch bei der Anwendung von JavaScript als Web-Programmiersprache sind Objekte anzutreffen. JavaScript kennt Objekte unterschiedlicher Typen:

- eingebaute Objekte wie Math, Date oder Array
- selbstdefinierte Objekte, die vom Programmierer in der Applikation deklariert und instanziert werden
- Applikationsobjekte, die vom Webbrowser bereitgestellt werden.

Die Applikationsobjekte unterliegen einer strengen hierarchischen Struktur. Mit JavaScript kann über die verfügbaren Metho-

den auf die Eigenschaften dieser Objekte zugegriffen werden. Dabei ist immer streng der Pfad zu den Objekten, beginnend bei der obersten Ebene, zu beachten. Die Objekte (Knoten der Baumstruktur) sind durch den Punktoperator voneinander zu trennen.

Die oberste Ebene der Objekthierarchie wird durch das Browserfenster repräsentiert und wird mit *window* bezeichnet. Unterhalb dieser Ebene befindet sich das Dokument. Objekte, die dem Objekt *document* untergeordnet sind, werden dem DOM (*document object model*) zugeordnet.

Hier zunächst ein sehr einfaches Beispiel: Mit der Methode `write()` des Objekts `document` können Inhalte im aktuellen Fenster manipuliert werden.

```
<html>
 <head>
   <title>Applikationsobjekte-
          HelloWorld.html</title>
     </head>
     <body  >
       <script LANGUAGE = JavaScript>
         window.document.write("hello world");
       </script>
     </body>
   </html>
```

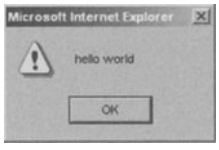
Listing 5.4 JavaScript Objekte, *js_helloworld.html*

Nun haben wir es also doch, das unvermeidliche Hello-World-Programm. Eigenschaften des Objekts sind u.a. `bgColor` und `fgColor`. Fügen wir vor die Methode `write()` die Zeilen

```
window.document.bgColor="#dddddd";
window.document.fgColor="#ff0000";
```

ein, dann erhält man eine rote Schrift auf grauem Hintergrund. Die Farbangabe ist hier in hexadezimaler Darstellung gewählt.

Auf die Angabe der obersten Hierarchie `window` kann verzichtet werden. Die schon benutzte Methode `alert()` ist eine Methode des Objekts `window`.



`window.alert ("Hello World");` bzw. `alert ("Hello World");`

erzeugt daher die Meldung in einem eigenen Fenster und erwartet eine Bestätigung. Abbildung 5.2 zeigt die hierarchische Struktur der Applikationsobjekte mit einem Pfad zu den Elementen In den folgenden Anwendungen wird der Zugriff auf ausgewählte

Applikationsobjekte erläutert. Im Abschnitt 5.5 wird ein Berechnungsbeispiel vorgestellt. Die Variablen werden dort über die Objekthierarchie referenziert.

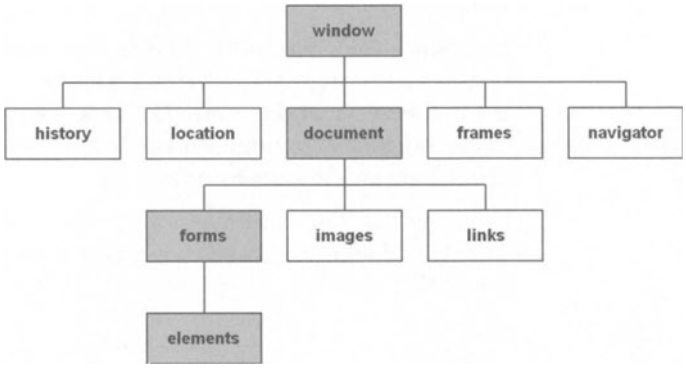


Abb. 5.2: Hierarchische Baumsstruktur der Applikationsobjekte

Umleitung einer Web-Site

Das erste Beispiel zu den Applikationsobjekten ist eine URL-Umleitung. Sie rufen eine Webadresse auf und werden von hier nach Bestätigung umgeleitet.

```
<html><head><title>Applikationsobjekte</title></head>
 <body  >
  <script LANGUAGE = JavaScript>
   document.write ("<b>URL = </b>" +location.href +"<br>" );
   document.write ("<b>PATH = </b>"+location.pathname+"<br>");
   document.write ("<b>PROTOKOLL = </b>" +location.protocol
                    +"<br>");
   document.write ("<b>Target    = </b>"+location.target
   +"<br>");
    if (window.confirm ("Sie werden jetzt weitergeleitet zu
                 www.divide-by-zero.com")){
     window.location.href = "http://www.divide-by-zero.com";
   }
    else{
     document.write("Sie haben die Weiter-
                    leitung nicht bestaetigt");
    }
   </script>
 </body>
 </html>
```

Listing 5.5: JavaScript Umleitung – *url_umleitung.html*

Die Methode `write()` des Objekts `document` erwartet als Argument einen String (Zeichenkette) in doppelte Hochkommata eingeschlossen. Dabei besteht die Möglichkeit, durch den Verknüpfungsoperator + einen zusammengesetzten String zu erzeugen. Sie können auch den arithmetischen Zuweisungsoperator += benutzen. Im ersten Aufruf von `document.write()` wird der String zusammengesetzt aus der String-Literalen URL = , der Eigenschaft `href` des Objekts `location` und der Literalen <br>. Die Bezeichnungen in spitzen Klammern <> sind HTML-Anweisungen zur Formatierung, Fettdruck und Zeilenvorschub. Weiter wird der Pfadname, das benutzte Protokoll und der Zielframe angezeigt.

Die Abfrage mit der Methode `confirm()` ist in eine Bedingung eingeschlossen. Die Methode liefert den Wert `true` zurück, falls der Button `ok` gedrückt wurde. Bei Entscheidung für Abbruch wird `false` zurückgegeben. Im letzteren Fall wird der Else-Zweig der Bedingung ausgeführt und der Besucher verbleibt auf der Seite. Erläuterungen zu Kontrollstrukturen und bedingten Anweisungen finden Sie weiter unten in diesem Kapitel.

Umgebungsabfrage

Den Typ des Browsers, dessen Einstellungen und die Bildschirmeinstellungen eines Benutzers kann man über die Eigenschaften der Objekte Navigator und Screen feststellen. Mit Kenntnis der Clientdaten können Webseiten, die in Abhängigkeit zu den Parametern stehen, geladen werden. Ggf. bekommt der Anwender Informationen über vorzunehmende Einstellungen wie Java aktivieren oder Cookies zulassen usw. Das nachfolgende Script gibt gleichzeitig eine Übersicht über die Eigenschaften der Objekte Navigator und Screen.

```
<html><head><title>Applikationsobjekte</title></head>
 <body onLoad="window.resizeTo(480,480)">
  <script LANGUAGE = JavaScript>
   document.write ("<b>Eigenschaften des Objekts
                    <i>navigator</i></b><br><br>");
   document.write
      ("<b>Codename: </b>" +navigator.appCodeName +"<br>");
   document.write("<b>Name: </b>" +navigator.appName +"<br>");
   document.write("<b>Version:
                 </b>"+navigator.appVersion+"<br>");
   document.write
      ("<b>Betriebssystem: </b>"+navigator.platform +"<br>");
   document.write("<b>userAgent: </b>"
                  +navigator.userAgent +"<br>");
```

```
document.write
  ("<b>Besuchte Seiten: </b>" +history.length +"<br>");
document.write
  ("<b>Java?: </b>" +navigator.javaEnabled() +"<br>");
document.write
  (<b>Cookies?: </b>" +navigator.cookieEnabled +"<br>")
document.write("<b>Sprache: </b>"
               +navigator.language +"<br><br>")
document.write("<b>Eigenschaften des
               Objekts <i>screen</i></b><br><br>");
document.write("<b>Aufloesung: </b>"
          +screen.width +" x " +screen.height +"<br>");
document.write("<b>Farbtiefe: </b>"
               +screen.colorDepth +"<br>");
document.write("<b>Farbaufloesung: </b>"
                  +screen.pixelDepth +"<br>");
document.write("<b>Verfuegbare Aufloesung: </b>"
  +screen.availWidth +" x " +screen.availHeight+"<br>");
</script> </body></html>
```

Listing 5.6: Objekte Navigator und Screen, *navigator.html*

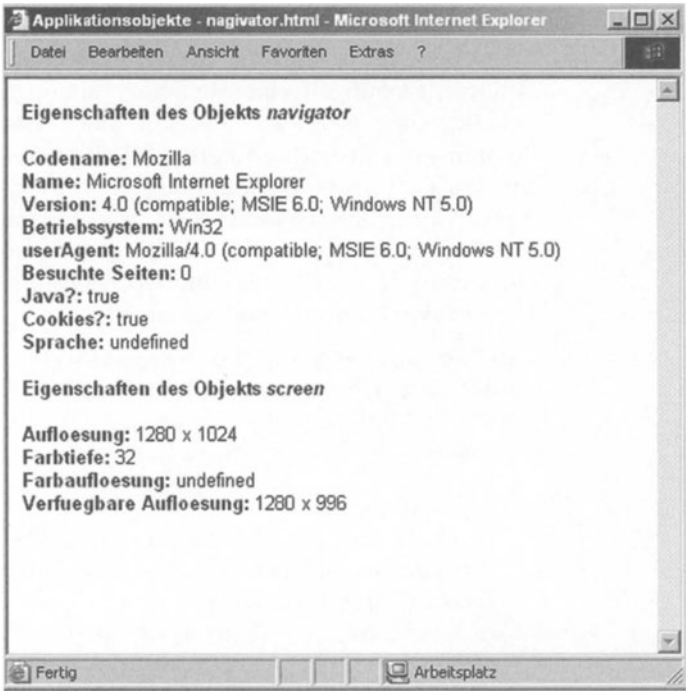


Abb. 5.3: Ausgabe Umgebungsabfrage des Client-Rechners

Pop-up-Windows

Eine häufige Anwendung von JavaScript-Funktionen ist die Fenstersteuerung und Platzierung der Fenster auf dem Bildschirm. Der eigentliche Sinn eines Pop-up-Fensters ist die Anzeige von Informationen innerhalb eines zusätzlichen Fensters. Das Fenster kann sich selbst wieder schließen und sollte möglichst wenig Auswirkung auf die weitere Navigation haben. Die Argumente der Funktion `window.open()` müssen in einer Zeile stehen. Der hier in der Druckversion angegebene Zeilenumbruch ist innerhalb von JavaScript unzulässig. Die Kommentare gehören in dieser Form nicht zum Quelltext.

Es werden zwei Versionen von Fenstern vorgestellt. Die Fenstergröße beträgt 400 x 300 Pixel-. Das Fenster wird zentriert dargestellt. Die erste Lösung zeigt alle Navigationselemente, während die zweite Lösung nur die Titelzeile zulässt.

```
<html><head><title>Popup-Fenster</title></head>
 <body bgcolor="#000000">
 <script language=JavaScript
         type=text/javascript>
   var winleft = (screen.width - 400) / 2;
   var wintop = (screen.height - 300) / 2;
   window.open(
    "dummy/dummy.html",  // URL
    "pop",                // Name des neuen Fensters
    "top="+wintop+",  // Position
      left="+winleft+",
      toolbar= 1,     // Navigationselementen
      location= 1,
      directories= 1,
      status= 1,
      menubar= 1,
      scrollbars= 1,
      resizable= 1,
      dependent= 1,
      fullscreen= 0,
      titlebar= 1,
      width=400,          // Fenstergröße
      height=300");
    </script>
  </body>
 </head>
</html>
```

Listing 5.7: Pop-up-Window mit Navigationselementen, *window-max.html*

```
<html><head><title>Popup-Fenster</title></head>
 <body bgcolor="#000000">
  <script language=JavaScript
   type=text/javascript>
    var winleft = (screen.width - 400) / 2;
    var wintop = (screen.height - 300) / 2;
    window.open(
          "dummy/dummy.html",
         "popup",
          "top="+wintop+",
          "left="+winleft+",
          width=400,
         height=300");
     </script></body></html>
```

Listing 5.8: Pop-up-Window ohne Navigationselemente *window-min.html*

Abb.5.4: Pop-up-Fenster mit und ohne Navigationselemente

Die Popup-Fenster erhalten eine Bezeichnung. Ein geöffnetes Fenster muss wieder geschlossen werden. Der Benutzer hat die Titelzeile zur Verfügung oder es wird eine kommentierte Schaltfläche angeboten, die bei Mausklick die Methode `close()` aufruft.

5.3 Sprachelemente von JavaScript

Variablen

Variablen sind Speicherplätze innerhalb des Hauptspeichers eines Rechnersystems, in denen die Daten während des Programmlaufs vorgehalten werden. Jede Variable erhält vom Programmierer eine Bezeichnung, den Variablennamen, unter dem die Variable innerhalb des Programms angesprochen wird.

Nach der Deklaration der Variablen müssen die Werte initialisiert werden. Das erfolgt durch einen Zuweisungsoperator mit Konstanten, Auswertung des rechts vom Zuweisungsoperator stehenden Ausdrucks, Funktionsaufruf oder Auswertung der Eingabefelder eines HTML-Formulars.

Variablennamen dürfen nicht mit einer Zahl beginnen und keine Sonderzeichen außer dem Unterstrich enthalten. Auszunehmen sind selbstverständlich auch die reservierten Schlüsselwörter der Sprache. JavaScript unterscheidet Groß- und Kleinschreibung, ist also *case sensitiv.*

JavaScript gilt als typenlose Sprache. Dennoch kennt Java Variablen vom Typ Zahl, Zeichenketten und Boolesche Werte. Der Datentyp einer Variablen kann sich innerhalb des Programmablaufs auch verändern. Mit dem Schlüsselwort `var` wird die Deklaration von Variablen eingeleitet. Die nachfolgende Liste enthält Variablennamen, die durch Kommata getrennt werden. Jede Anweisung muss mit einem Semikolon abgeschlossen werden.

Kommentare

Kommentare innerhalb von JavaScript unterscheiden sich von der Auszeichnung der HTML-Kommentare `<!--` *Kommentar in HTML* `-->`. Einzeilige Kommentare werden durch zwei Schrägstriche markiert. Mehrzeilige Kommentare werden zwischen die Zeichenkombination `/*` *mehrzeiliger Kommentar* `*/` eingeschlossen.

Zahlen

Mit der folgenden Anweisung werden die Variablen zur Berechnung einer Rechteckfläche deklariert. Die Initialisierung von `laenge` und `breite` erfolgt durch direkte Zuweisung von Konstanten. Die Ergebnisvariable `flaeche` sollte zunächst mit Null initialisiert werden und bekommt dann ihren Wert durch Auswertung des Ausdrucks zugewiesen.

```
var flaeche, laenge, breite;
    flaeche = 0.0;
    laenge  = 5.0;
    breite  = 3.0;
    flaeche = laenge * breite;
```

Die Verknüpfung der Variablen (Operanden) erfolgt durch Operatoren. Diese sind zu unterscheiden als Arithmetische Operatoren, Zuweisungsoperatoren, Vergleichsoperatoren, logische Operatoren und String-Operatoren. Wesentliche Unterschiede zu anderen Programmiersprachen bestehen nicht. Eine Zusammenstellung der Operatoren finden Sie in Tabelle 5.1

Arithmetische Operatoren:	+	Addition
	-	Subtraktion
	*	Muliplikation
	/	Division
	%	Modulo, Rest einer Ganzzahldivision
	++	Inkrement, erhöht Ganzzahl um 1
	--	Dekrement, verringert Ganzzahl um 1
	-	Vorzeichenwechsel
Zuweisungsoperatoren:	=	einfacher Zuweisungsoperator
	+=	arithmetische Zuweisungsoperatoren
	-=	Auswirkung auf den Speicherplatz der
	*=	Zielvariablen, Beispiel a *= 3
	/=	entspricht a = a * 3;
	%=	a %=2 entsprcht a = a%2;
Vergleichsoperatoren:	= =	Gleichheit zweier Ausdrücke
	!=	ungleich
	<=	kleiner gleich
	>=	größer gleich
	>	größer
	<	kleiner
Logische Operatoren:	&&	UND, true wenn alle Bedingungen erfüllt sind
	||	ODER, true wenn eine Bedingung erfüllt ist
	!	NICHT,
String-Operatoren:	+	Stringaddition, Verknüpfung
	parseInt	Umwandlung eines Strings in eine Ganzzahl
	parseFloat	Umwandlung eines Strings in eine Gleitkommzahl

Die Reihenfolge der Auswertung entspricht den mathematischen Regeln und kann durch runde Klammernpaare beeinflusst werden.

Tabelle 5.1: JavaScript-Operatoren

Strings
Boolesche Variablen

Stringvariable werden durch in Hochkommata eingeschlossene Literale initialisiert. Die folgenden Programmanweisungen verknüpfen die Variable txt mit einer Stringkonstanten.

```
var txt = "9" +"9";
alert("Stringverknuepfung " +txt);
alert(txt*3);
```

Die Variable `txt` beinhaltet die Zeichenfolge 99. Nach Multiplikation mit 3 wird der ganzzahlige Wert 297 angezeigt. Etwas ungewöhnlich für Programmierer, die Typenstrenge gewohnt sind. Fehlerquellen dieser Art kann man aber durch konsequente Programmierung vermeiden. Strings werden mit den Funktionen `parseInt()` und `parseFloat()` in Zahlen umgewandelt. Bei der Konvertierung von Strings in Zahlen treten Fehler auf, wenn im String für Zahlen unzulässige Zeichen vorkommen. Die erfolgreiche Umwandlung kann mit der Funktion isNaN geprüft werden. Probieren Sie folgende Anweisungen aus:

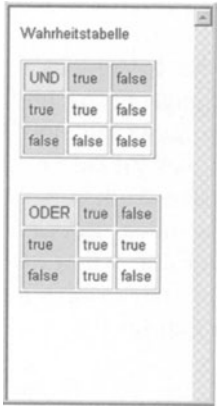
```
var test = false;
var zahl = "zwoelf";
test = isNaN (zahl);
alert (zahl +" isNaN test = "
       +test);
alert (parseInt(zahl));
//
zahl = 12;
test = isNaN (zahl);
alert (zahl +" ist eine Zahl test = "  +test);
alert (parseInt(zahl));
```

Boolesche Variablen signalisieren einen Zustand und arbeiten wie ein Schalter, die Inhalte der Variablen können nur die Werte `true` oder `false` annehmen. Der Booleschen Variablen `test` wird `false` zugewiesen, `zahl` erhält die Zeichenkette `zwoelf`. Anschließend wird `zahl` mit der Funktion `isNaN` geprüft. Da der Inhalt von `zahl` keine Zahl ist (*not a number*), liefert die Funktion den Wert `true` zurück. Die Variableninhalte werden mit `alert()` angezeigt, `parseInt()` liefert `NaN`.

Im zweiten Fall sind die Anweisungen korrekt, `test` wird `false` und `parseInt()` liefert 12.

Das folgende Script, eingefügt in den Body einer HTML-Datei, ist die Auswertung einer Wahrheitstabelle mit den logischen Operatoren && (*logisch und*) und || (*logisch oder*) verknüpft. Der Zeilenumbruch in den `document.write()`-Anweisungen ist im

Quellcode des Programms zu vermeiden. HTML-Anweisungen für eine Tabelle sind als Literale in Hochkommata angegeben. Die booleschen Variablen a und b und deren Verknüpfung werden mittels Stringaddition hinzugefügt.




```
<script language = JavaScript>
 var a = true; var b = false;
 document.write ("Wahrheitstabelle<br><br>");
 document.write("<table border = 1
    cellpadding = 4><tr><td bgcolor = yellow>
     UND</td><td bgcolor =#dddddd>" +a +"</td>
      <td bgcolor =#dddddd>" +b +"</td></tr>");
 document.write("<tr><td bgcolor =#dddddd>" +a
     +"</td><td>" +(a&&a)+"</td><td>" +(a&&b) +"</td></tr>");
 document.write("<tr><td bgcolor =#dddddd>" +b
     +"</td><td>" +(b&&a) +"</td><td>" +(b&&b) +"</td></tr>");
 document.write("</table>");
 document.write ("<br><br>");
 document.write("<table border = 1
     cellpadding = 4><tr>
      <td bgcolor = yellow>ODER</td><td bgcolor
      =#dddddd>" +a +"</td><td bgcolor
      =#ddddd>" +b +"</td></tr>");
 document.write("<tr><td bgcolor =#dddddd>" +a
      +"</td><td>" +(a||a) +"</td><td>" +(a||b) +"</td></tr>");
 document.write("<tr><td bgcolor =#dddddd>" +b
      +"</td><td>" +(b||a) +"</td><td>" +(b||b) +"</td></tr>");
 document.write("</table>");
</script>
```

Listing 5.9:Boolesche Variablen, *wahrheitstabelle.html*

Kontrollstrukturen

Programmanweisungen werden sequentiell nacheinander in der Reihenfolge ihres Auftretens abgearbeitet. Abweichungen hiervon werden durch Kontrollstrukturen vorgenommen. Kontrollstrukturen sind an Bedingungen geknüpft und stellen Schleifen und Verzweigungen dar.

Verzweigungen if, if ... else., switch

Verzweigungen werden mit den Anweisungen `if`, `if ... else` oder `switch` ausgeführt. Mit dem Schlüsselwort `if` wird eine Bedingung ausgewertet. Liefert diese Bedingung den Wert `true`, erfolgt die Abarbeitung der nachfolgenden Programmanweisung bzw. des in {} geschweiften Klammern folgenden Anweisungsblocks. Ist die Bedingung `false`, unterbleibt die Ausführung. Man beachte folgende Anweisungsfolge mit der Negation. Wird die Ausgabe mit `alert` aktiviert? Ja, denn der Wert 12 kann in

eine Zahl konvertiert werden, daher liefert `isNaN` `false` zurück. Die Negation wandelt die Auswertung der Bedingung in `true`, die Ausgabe erfolgt.

```
zahl = 12;
if( !isNaN (zahl)) alert("Wert ist zulaessig");
```

Mit der Verzweigung `if ... else` besteht die Möglichkeit, eine Alternative abzubilden. Entweder wird der auf `if` folgende Anweisungsblock ausgeführt, oder die dem `else` folgenden Anweisungen werden abgearbeitet.

```
zahl = "a2";
if( !isNaN (zahl)){
   alert("Wert ist zulaessig");
}
else{
   alert("Wert ist nicht zulaessig");
}
```

`if ... else`-Konstrukte können geschachtelt werden, wobei der `else` Zweig selbst wieder ein `if` nach sich ziehen kann. Die Lesbarkeit eines Programms wird durch eine zu tiefe Verschachtelung aber nicht erhöht. Mehrfachverzweigungen lassen sich besser durch die Fallunterscheidung mit `switch` ausführen. Mit `switch` wird der Wert einer Variablen abgefragt und mit `case` in die Verzweigungen geleitet. Die Switch-Anweisung arbeitet immer nur einen Zweig der Anweisungen ab. Nimmt eine Variable keinen der abgefragten Werte an, dann kann in den Zweig `default` mit den Anweisungen fortgefahren werden, `break` beendet eine Verzweigung. Im Beispiel wird eine bei trigonometrischen Berechnungen ggf. notwendige Falluntersuchung hinsichtlich des Quadranten vorgenommen. Die vier möglichen Quadranten eines Kreises könnten wie folgt behandelt werden:

```
var q = 3;
   switch(q) {
      case (1) :
         alert("  0 <= Q < 100");
         break;
      case (2) :
         alert("100 <= Q < 200");
         break;
      case (3) :
         alert("200 <= Q < 300");
         break;
       case (4) :
```

```
        alert("300 <= q < 0");
        break;
      default:
       alert("Nicht im abgefragten Bereich");
      } // ende switch Anweisung
```

Die Variable q kann beliebige Werte annehmen und wird in der switch-Anweisung abgefragt. Die Verzweigungen sind mit case angegeben. Trifft für q keiner der abgefragten Fälle zu, dann wird der Default-Zweig ausgeführt.

Schleifen for, while

Anweisungsblöcke, die wiederholt auszuführen sind, werden in Schleifen eingebettet. Dabei unterscheiden wir Zählschleifen, die genau n mal zu wiederholen sind und bei denen sich die Laufbedingungen während der Abarbeitung nicht ändert, und Wiederholungen, bei denen sich die Ausführungsbedingung während des Schleifendurchlaufs ändern kann.

Eine Zählschleife wird mit dem Schlüsselwort for eingeleitet. In runden Klammern folgen drei Ausdrücke getrennt durch Semikolon mit folgender Bedeutung:

- Initialisierung der Schleifenvariablen
- Formulierung der Abbruchbedingung
- Änderung der Schleifenvariablen

Der zu wiederholende Schleifenkörper wird in geschweifte Klammern eingeschlossen. Eine der Hauptanwendungen für For-Schleifen ist die Adressierung von Feldelementen mittels Schleifenvariablen. In den Anwendungsbeispielen werden wir den Gebrauch der Datenstruktur Array kennen lernen.

Als einführendes Beispiel für Schleifen wollen wir die Summe der Zahlen von 1 bis 10 berechnen.

```
<script language = JavaScript>
   var i,i1,i2,sum;
   sum = 0; i1 = 1; i2 =10;
   for( i=i1; i<= i2; i++){
      sum+=i;
   }
   alert("Summe der Zahlen von " +i1 +" bis " +i2 +" = "+sum);
</script>
```

Es werden die benötigten Variablen deklariert. Die Summe der Zahlen wird mit 0 besetzt. In der For-Schleife wird die Schleifenvariable mit i deklariert und der Anfangswert auf i1 gesetzt. Danach wird die Laufbedingung geprüft. Ist der Ausdruck i <=

i2 true, dann wird die Schleife abgearbeitet und nach Abarbeitung i inkrementiert. Danach geht es weiter mit Überprüfung der Laufbedingung. Die Schleifenvariable i soll innerhalb der Schleife nicht durch weitere Anweisungen verändert werden.

Eine andere Form von Wiederholungen wird durch die While-Schleife repräsentiert. Auf das Schlüsselwort while folgt die Formulierung einer Bedingung in runden Klammern. Liefert die Auswertung des Ausdrucks den Wert true, dann wird der Schleifenkörper so lange abgearbeitet, bis die Änderung der Bedingung die Beendigung der Schleife zur Folge hat.

```
var taste =" ";
while(taste != "x"){
    taste=prompt("Druecken Sie eine Taste","");
}
```

Es wurde eine Variable taste deklariert. Die Schleife soll so lange laufen, bis die Taste x eingegeben wurde. Die Zuweisung des Tastencodes erfolgt über die Funktion prompt.

Die Programmierung von Endlosschleifen sollte man tunlichst vermeiden. Für Testzwecke hier mal eine Version, bei der sich der Browser bald beschweren würde:

```
var i = 0;
while(true)
   document.write(i++ +" ");
}
```

Funktionen

Funktionen sind in sich abgeschlossene Programme, die Teilaufgaben lösen und innerhalb eines Programms Mehrfachverwendung finden. Durch Einsatz von Funktionen wird der Programmcode übersichtlicher und redundanzfrei, d.h. wiederkehrende Aufgaben sind nur an einer Stelle im Quelltext vorhanden. Eine Funktion wird eigenständig mit lokal gültigen Parametern definiert. Die Ausführung der Funktion erfolgt durch Funktionsaufruf, der Aufruf einer Funktion ist an ein Ereignis gebunden. Funktionen können danach aber auch von Funktionen aus aufgerufen werden. Üblicherweise werden die Funktionen im Kopfbereich einer HTML-Datei eingebettet oder sind in einer externen Datei gespeichert.

Das Schlüsselwort für eine Funktion ist function, hiernach folgt der Name der Funktion mit einer Parameterliste in runden Klammern. Die Anweisungen einer Funktion werden in geschweifte Klammern geschrieben. Die Parameterliste ist optional.

Funktionen tauschen Daten mit dem aufrufenden Programm über die Parameterliste oder einen Rückgabewert aus. Innerhalb eines Funktionsblocks sind mit `var` deklarierte Variable nur lokal in diesem Block gültig. Außerhalb von Funktionen deklarierte Variable gelten als global und sind auch innerhalb von Funktionen sichtbar.

Eine bereits bekannte Funktion ist `alert()`. Die Funktion wird im Beispiel durch das Ereignis `onLoad` beim Hochladen der Seite ohne Parameter aufgerufen.

```
<HTML>
   <HEAD>
    <TITLE>Funktionen: function_0.html</TITLE>
    <script language="JavaScript">
      function textBox(txt) {
      alert(txt);
      return;
     }
    </script>
   </HEAD>
   <BODY onload = alert();>
   </BODY>
</HTML>
```

Eine Stringkonstante kann als Argument beim Funktionsaufruf wie folgt übergeben werden:

```
<BODY onload = alert("hello world");>
```

Zur Demonstration des Parameteraustausches zwischen Funktionen und der Kapselung von Programmcode wird die Funktion `alert()` nicht unmittelbar aufgerufen. Wir erstellen eine Funktion `textBox()`.Diese Funktion hält als lokalen Parameter die Variable `txt` vor. Beim Aufruf der Funktion, ausgelöst durch das Ereignis `onload`, wird als Argument die Stringkonstante `"HelloWorld"` an die Variable `txt` übergeben und in der Funktion weiterverarbeitet. Strings werden in JavaScript als Objekte behandelt. Auf das Objekt `txt` wird eine Methode angewandt werden. Wir manipulieren den String hier mit `toUpperCase()`.

```
<html><head><title>Funktionen: function_3.html</title>
<script language="JavaScript">
  function textBox(txt) {
    txt=txt.toUpperCase();
    alert(txt);
    return;
```

```
  }
</script></head>
<body onload =textBox("hello world");>
</body></html>
```

Die Rückkehr aus einer Funktion erfolgt nach Abarbeitung aller Programmzeilen oder bei Erreichen der Anweisung `return`. Es geht dann mit der auf den Funktionsaufruf folgenden Anweisung weiter. Hier ist es der Funktionsaufruf von `alert()`, der den in Großbuchstaben gewandelten String `txt` anzeigt. Weitere Hinweise zur Anwendung von Funktionen finden Sie im Beispiel des Abschnitts 5.5.

5.4 Ereignisbehandlung

Ereignisse sind vom Benutzer ausgelöste Aktionen wie Tastendruck oder Mausbewegung. Auf diese Ereignisse reagieren die Eventhandler. Ein Eventhandler ist nicht in einen Script-Tag einzubinden. Dem Eventhandler wird ein String zugewiesen, der JavaScript-Code enthält. Das können auch mehrere aufeinanderfolgende Anweisungen sein. Üblicherweise ist es aber ein Funktionsaufruf. Die wichtigsten Ereignisse sind in der Tabelle 5.2 zusammengefasst. Nachfolgend werden drei Beispiele zur Ereignisbehandlung diskutiert.

Event-Handler:	wird ausgelöst durch:
onBlur	ein Element verliert den Fokus
onClick	ein Element wird angeklickt
onFocus	ein Eelement gewinnt den Fokus
onLoad	eine Web-Seite wird geladen
onMouseOut	Mauszeiger außerhalb des Element
onMouseOver	Mauszeiger über einem Element
on MouseMoved	Maus wurde bewegt
onMouseDown	Maustaste gedrückt
onMouseUp	Maustaste losgelassen
onUnLoad	Webseite wird verlassen

Tabelle 5.2: Gebräuchliche Eventhandler in JavaScript

Pop-up-Windows

Bei Betrachtung der Applikationsobjekte hatten wir die Funktionen zur Öffnung neuer Fenster bereits angesprochen. Die Inhalte der neuen Fenster benötigen eine eigene HTML-Datei. In der hier zu betrachtenden Anwendung handelt es sich um ein Schaubild, das den Softwareentwicklungszyklus mit der Programmiersprache C beschreibt.

Sobald die Maus einen der ausgezeichneten Bereiche berührt, erscheint ein Pop-up-Fenster, das bei Verlassen des Bereichs wieder geschlossen wird. Nun müsste man für jedes Fenster eine eigene HTML-Datei vorhalten. Dies kann man aber mit der Methode `document.write()` vermeiden, indem man den HTML-Text unmittelbar in das Dokument schreibt. Im Head-Bereich des HTML-Dokuments legen wir die JavaScript-Funktion `txtMessage()` ab.

```
<script language="JavaScript">
 function txtMessage(text,titel)
//
  ymouse =screen.height/2;
  xmouse = screen.width/2;
  var wintop  = ymouse - 120;
  var winleft = xmouse -90;
  popup=window.open("",
         "textFenster","top="
         +wintop +",left="
         +winleft   +",
          directories=no,
          menubar=no,
          width=240,
          height=180");
  popup.document.write("<html><head>
      <title>" +titel +"</title>
       <link href='textfenster.css'
             rel='stylesheet'
             type='text/css'>
        </head><body>");
    popup.document.write("<h2>" +titel
             +"</h2>")
    popup.document.write("<p>" +text
             +"</p></body></html>");
    }
   </script>
```

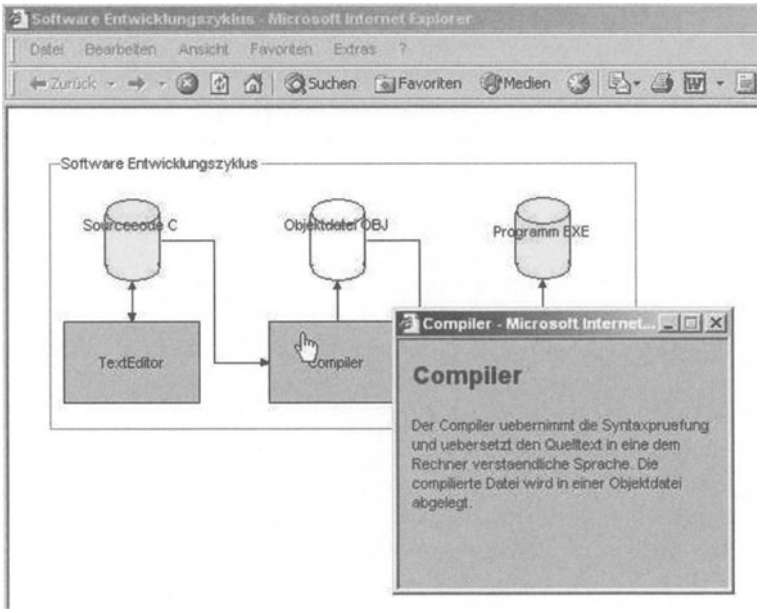


Abb.5.5: Anwendung von Pop-up-Windows, *software_entw.html*

Die Eingangsparameter, die von der Funktion erwartet werden, sind der auszugebende Text und der Titel der HTML-Seite. Mit `ypos, xpos` wird die Position des Fensters definiert. Das Objekt `window` wird mit `popup` bezeichnet. Somit können wir auf dieses Objekt als oberstes der Hierarchie zugreifen und den HTML-Inhalt unmittelbar als zusammengesetzten String einschließlich der übergebenen Parameter und Stylesheetdefinitionen mittels `popup.document.write()` in das Window schreiben.

Das HTML-Dokument ist mit einer Image-Map angelegt. Bereiche in der Grafik *zyklus.jpg* werden mit `area` deklariert. Dieser Tag wirkt zusammen wie der Anchor-Tag `<a></a>` bezogen auf den mit `shape` und `coords` definierten Bereich. Die Texte zur Darstellung im Textfenster sind hier der besseren Übersicht halber nur angedeutet.

```
<body>
 <map name="chart">
  <area href="exe"
   onmouseover="txtMessage ('Die ausfuehrbare ...','Exe-File')"
   onmouseout="popup.close()"
     shape="rect" coords="360 ,60 ,400 ,110">
    <area href="#"
    onmouseover="txtMessage('Der Linker...','Linker')"
```

```
    onmouseout="popup.close()"
    shape="rect" coords="330 ,140 ,430 ,200">
    <area href="#"
    onmouseover="txtMessage('Der Compiler ...,'Compiler')"
    onmouseout="popup.close()"
    shape="rect" coords="180 ,140 ,280 ,200">
    <area href="#"
    onmouseover="txtMessage('Mit einem...','TextEditor')"
    onmouseout="popup.close()
     "shape="rect": coords="30 ,140 ,130 ,200">
   </map>
   <img border="0" src="zyklus.jpg" alt="Entwicklungszyklus"
   usemap="#chart"
        width="450" height="220">
 </body>
```

Dem Eventhandler `onmouseover` wird der String mit dem Aufruf der Funktion `txtMessage()` zugewiesen. Bei Verlassen des Rechtecks wird über die Ereignisbehandlung von `onmouseout` die Methode `close()` auf das Objekt `popup` unmittelbar angewandt.

Ein Pulldown-Menü

Effektiv und übersichtlich kann die Seiten-Navigation durch ein Pulldown-Menü unterstützt werden. Wir betrachten hier das Pulldown-Menü der Website zum Buch. Der Variablen `index` wird der Index des ausgewählten Elements der Listbox zugewiesen. Index wird hier nicht weiter benutzt, kann aber ggf. sinnvoll Anwendung finden. Die Variable `neueSeite` bekommt die relative Adresse, die als Wert in der Auswahlliste definiert wurde, `pulldown_menue` ist der Name des Formulars, `listBoxAuswahl[]` enthält die Attribute des Objekts `Select`. NeueSeite wird dann an die Methode `open()` als erstes Argument der Liste übergeben.

Der Aufruf der Buchseite erfolgt durch einen Frameset, der im Kapitel 2 bereits erläutert wurde. In der Funktion ist der hierfür gültige Aufruf auskommentiert. Statt dessen sind die folgenden vier Anweisungen zum Aufruf eines sich selbst wieder schliessenden Popup-Windows eingefügt.

```
<html>
<head><title>Auswahlmenue</title>
   <script LANGUAGE="JavaScript">
   <!--
     function seiteLaden() {
      var index =
      document.pulldown_menue.
```

```
        listBoxAuswahl.selectedIndex;
      var neueSeite =
      document.pulldown_menue.
        listBoxAuswahl[index].value;
      var winleft = (screen.width - 400) / 2;
      var wintop = (screen.height - 300) / 2;
      popup=
    window.open
   (neueSeite,"neuesFenster","top="+wintop+",
   left="+winleft+",width=400,height=300");
   setTimeout("popup.close();", 5*1000);
       }
   -->
   </script>
</head>
<body bgcolor="#E6001B">
 <div align="center">
 <img src ="spix_trp.gif" width="1" height="24"
  border="0">
 <table width="360" cellpadding="0" cellspacing="0" border="0"
       bordercolor="#ffffff">
  <tr>
  <td width="170" valign="top">
  <form name ="pulldown_menue">
    <select name="listBoxAuswahl" size="1"
     onchange="javascript:seiteLaden();">
     <option value="dummy/dummy.html">Wählen Sie hier </opti-
on>
     <option value ="dummy/dummy.html">XHTML &amp;
CSS</option>
     <option value ="dummy/dummy.html">XML</OPTION>
     <option value ="dummy/dummy.html">Java-Applets</ option >
     <option value ="dummy/dummy.html">JavaScript</ option >
     <option value ="dummy/dummy.html">SVG</ option >
     <option value ="dummy/dummy.html">PHP</option >
    </select>
   </form></td></tr>
 </table></div>
</body></html>
```

Die Listbox wird innerhalb des Formulars mit dem Tag `select` formuliert. Die Auswahlalternativen sind mit dem Tag `option` beschrieben. Der Eventhandler ist onchange. Sobald eine Auswahl der Liste erfolgte, wird mit dem String `javascript:seiteLaden();` die oben beschriebene Funktion aufgerufen.

Mouseover-Effekte mit wechselnden Bildern – das Image-Objekt

Der Wechsel von Bildinhalten ist eine häufige Anwendung zur Navigationsunterstützung. Stellen Sie sich vor, Sie möchten zur Ansicht auf einer Webseite 400 Bilder darstellen. Für jedes Bild gibt es ein „Thumbnail" in schwarz-weiß und in Farbe. Sie ordnen die Bilder in einer Tabelle von 20 x 20 Bildern an. Gezeigt werden die schwarz-weiß Bilder. Bei Mausberührung wechselt das Bild zur Farbe, bei Klick wird das große Bild in einem neuen Fenster angezeigt. Unsere Darstellung beschränkt sich hier auf drei Bilder in einer Tabelle. Die Dateinamen unterliegen folgender Konvention: Wir nummerieren die Bilder durch. Die laufende Bildnummer wird dabei zwischen die Bezeichnungen `bild_` und `_color.jpg` bzw. `_gray.jpg` eingebunden. Das große Bild bezeichnen wir mit `bild_`, laufende Nummer und `_big.jpg`. Auf diese Weise lassen sich die Bilder bei der Verarbeitung leicht referenzieren.

Zur Einfügung der Bilder betten wir den Image-Tag in einen Anchor ein. Die Referenz zeigt dabei einen Leerstring oder den Character #. Jeder Img-Tag bekommt eine Identifikation. Über das Attribut `id` kann man so W3C-konform auf das Element zugreifen. Wir definieren die Objekthandler `onMouseOver`, `OnMouseOut` und `onClick` mit Funktionsaufrufen von `displayImage()` und `displayBigImage()`.

```
<table width = "335" border="0" cellspacing="1"
       cellpadding="3" bgcolor="#ffffff">
<TR>
 <TD bgcolor="#aaaaaa" width = 160 >
   <A HREF="#"
   onMouseOver = "displayImage ('img_0','bild_0_color.jpg')"
   onMouseOut  = "displayImage ('img_0','bild_0_gray.jpg' )"
   onClick="displayBigImage('bild_0_big.jpg')">
     <IMG SRC="bild_0_gray.jpg" border="0" id="img_0"></A>
   </td>
  <TD BGCOLOR="#aaaaaa" width = 160 >
    <A HREF="#"
    onMouseOver = "displayImage('img_1','bild_1_color.jpg' )"
    onMouseOut = "displayImage ('img_1','bild_1_gray.jpg' )"
    onClick="displayBigImage('bild_1_big.jpg')">
      <IMG SRC="bild_1_gray.jpg" border="0" id="img_1"></A>
   </td>
</TR>
</table>
```

Die Funktion displayImage() bekommt als Argumente den Inhalt des Attributs id und die URL des Bildes übergeben. In der Funktion benannt mit imageid und image_ref.

Über die Methode getElementById() wird auf das Element im Strukturbaum zugegriffen. Dem Attribut src wird die Bildreferenz zugewiesen.

```
function displayImage(imageid,image_ref){
    document.getElementById(imageid).src =
      image_ref;
    return true;
}
```

Die Funktion displayBigImage() arbeitet nach dem Muster der Popup-Windows (s. oben). Dabei wird der Image-Tag unmittelbar in das Dokument geschrieben, dadurch wird das Vorhalten von HTML-Dateien für jedes Bild entbehrlich.

```
function displayBigImage(image){
 var winleft = (screen.width - 480) / 2;
 var wintop =  (screen.height - 360) / 2;
 popup=window.open("", bildfenster",
   "top="+wintop+",left="+winleft+
   ",width=480,height=360");
 popup.document.write
   ("<html><head><title>" +image
     +"</title></head><body topmargin = 0
     leftmargin= 0 ><img src=" +image
     +"></body></html>");
}
```

Probieren Sie das komplette Script auf der Web-Site zum Buch aus. Wenn Sie einen VRML-Client installiert haben, dann können Sie mit der Maus das Objekt in der Tabelle rechts unten bewegen.

Bei Einführung des Objekts Image bieten sich weitere Alternativen an. Zunächst betrachten wir das sog. *Preloading*. Mit dem Ladevorgang der HTML-Seite können gleichzeitig auch alle Bilder geladen werden. Dazu müssen wir Instanzen der Bilder im Kopfbereich der HTML-Seite erzeugen.

Mit var bild = new Image(); wird die Objektvariable deklariert. bild.src ="bild_0_gray.jpg" erzeugt die Instanz. Für mehrere Bilder wenden wird das Verfahren in einer Schleife an und benutzen für die Objekte Arrays. Man ist auf diese Weise in der Lage, unter einem Namen viele Variablen anzusprechen. Die

Variablen werden durch den Feldindex unterschieden. Dieser wird üblicherweise in For-Schleifen zugeordnet. Der Programmcode hat folgende Gestalt:

```
var bilder = new Array();
var n = 3;
var j = 0;
for (i = 0; i < n; i ++){
   bilder[j]       = new Image();
   bilder[j].src   = "bild_" +i +"_gray.jpg";
   j++;
   bilder[j]     = new Image();
   bilder[j].src = "bild_" +i +"_color.jpg";
}
```

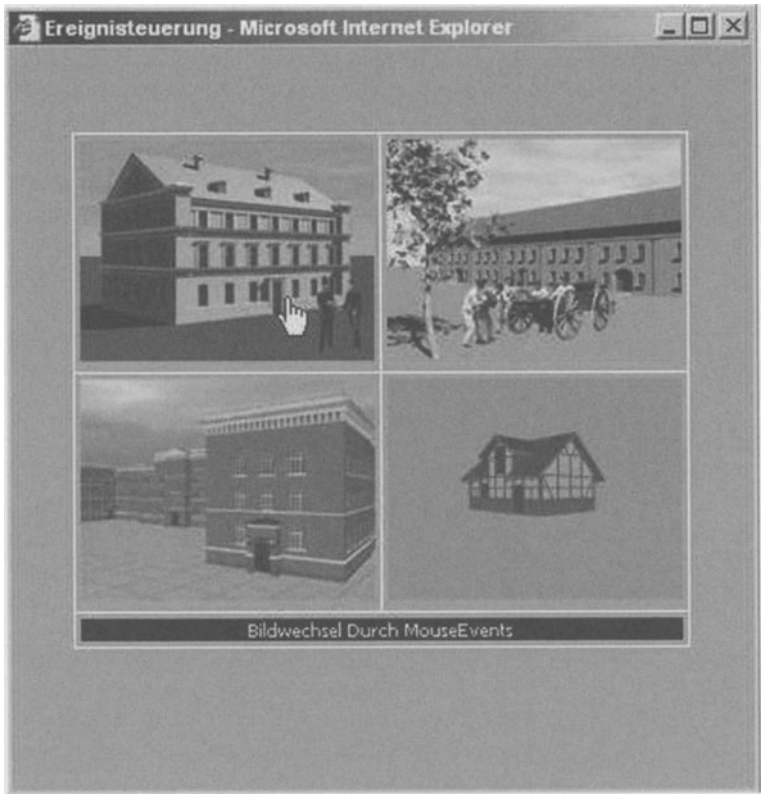


Abb. 5.6: Bildwechsel durch MouseOver-Ereignisse

Wir erhalten ein Feld von 6 Objekten: `bilder[0]`, `bilder[1]`, `bilder[2]`, `...[bilder[5]`.Auf den geraden Speicherplätzen 0, 2, 4 befindet sich immer das Graubild, auf den ungeraden Speicherplätzen haben wir die Farbbilder.

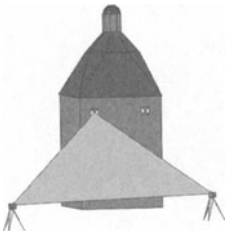
Die Ereignisbehandlung muss lediglich den Tausch zwischen Graubild und Farbbild vornehmen. Leider haben wir immer nur ein Attribut id für die Bilder. Daher führen wir ein zusätzliches Objekt swap ein. Dieses Objekt nimmt dann immer das zu überschreibende Graubild auf. Verlässt die Maus wieder das Bild, dann wird das zwischengespeicherte Graubild wieder dem Element zugewiesen. Der HTML-Code wird im Zusammenhang wiedergegeben. Wir verzichten bei der Wiedergabe aus Platzgründen jauf das onClick-Ereignis.

```
<html>
<head><title>Ereignisteuerung</title>
<script language="JavaScript" >
// Preload Images
 var swap = new Image();
 swap.src = "bild_0_gray.jpg";
 var bilder = new Array();
 var n = 4;
 var j = 0;
 for (i = 0; i < n; i ++){
   bilder[j]       = new Image();
   bilder[j].src   = "bild_" +i +"_gray.jpg";
   j++;
   bilder[j]     = new Image();
   bilder[j].src = "bild_" +i +"_color.jpg";
 }
</script>
</head>
<body   bgcolor = "#aaaaaa" >
 <script language = "JavaScript" >
   function displayImageColor(imageid,image_ref){
    // Graubild sichern
    swap.src = document.getElementById(imageid).src;
    document.getElementById(imageid).src = image_ref;
    return true;
   }
   function displayImageGray(imageid){
      document.getElementById(imageid).src =  swap.src;
   return true;
   }
   </script>
<div   align="center">
<table width = "335" border="0" cellspacing="1" cellpad-
ding="3"
       bgcolor="#ffffff">
 <TR>
```

```
 <TD BGCOLOR="#aaaaaa" width = 160 >
 <A HREF="#"
 onMouseOver = "displayImageColor('img_0','bild_0_color.jpg')"
 onMouseOut  = "displayImageGray ('img_0')">
   <IMG SRC="bild_0_gray.jpg" border="0" id="img_0"></A> </td>
 <TD BGCOLOR="#aaaaaa" width = 160 >
 <A HREF="#"
 onMouseOver="displayImagColor('img_1','bild_1_color.jpg')"
 onMouseOut  = "displayImageGray('img_1')">
   <IMG SRC="bild_1_gray.jpg" border="0"
   id="img_1"></A></td></TR>
<TR>
 <TD BGCOLOR="#aaaaaa" width = 160 >
 <A HREF="#"
  onMouseOver="displayImageColor('img_2','bild_2_color.jpg')"
  onMouseOut  = "displayImageGray('img_2')">
     <IMG SRC="bild_2_gray.jpg" border="0"id="img_2"></A>
  </td>
 <TD BGCOLOR="#aaaaaa" width = 160>
 <A HREF="#"
  onMouseOver="displayImageColor('img_3','bild_3_color.jpg')"
  onMouseOut  = "displayImageGray('img_3')">
  <IMG SRC="bild_3_gray.jpg" border="0" id="img_3"></A>
 </td>
</TR>
</TABLE></div></BODY></HTML>
```

Listing 5.10: Pre-Loading von Bildern, *bildanzeige.html*

5.5 Formularauswertung und Berechnungen



Unter der Bezeichnung Turmhöhenbestimmung ist in der Vermessungskunde ein Verfahren zur Berechnung der Koordinaten unzugänglicher Hochpunkte bekannt. Ohne zu sehr ins Detail zu gehen, wollen wir die Prinzipien der Vorgehensweise hier am Beispiel eines JavaScript-Programms erläutern. Im vereinfachten Fall werden auf den Endpunkten einer Basis die Horizontalwinkel zwischen dem gegenüberliegenden Basispunkt und dem Hochpunkt und der Zenitwinkel zum Hochpunkt gemessen. Die Winkel werden mit einem Theodolit oder Tachymeter gemessen. Die Höhe des Messgerätes über dem Standpunkt wird mit Instrumentenhöhe bezeichnet, die Tafelhöhe ist die Höhe der Zielmarke über dem Referenzpunkt.

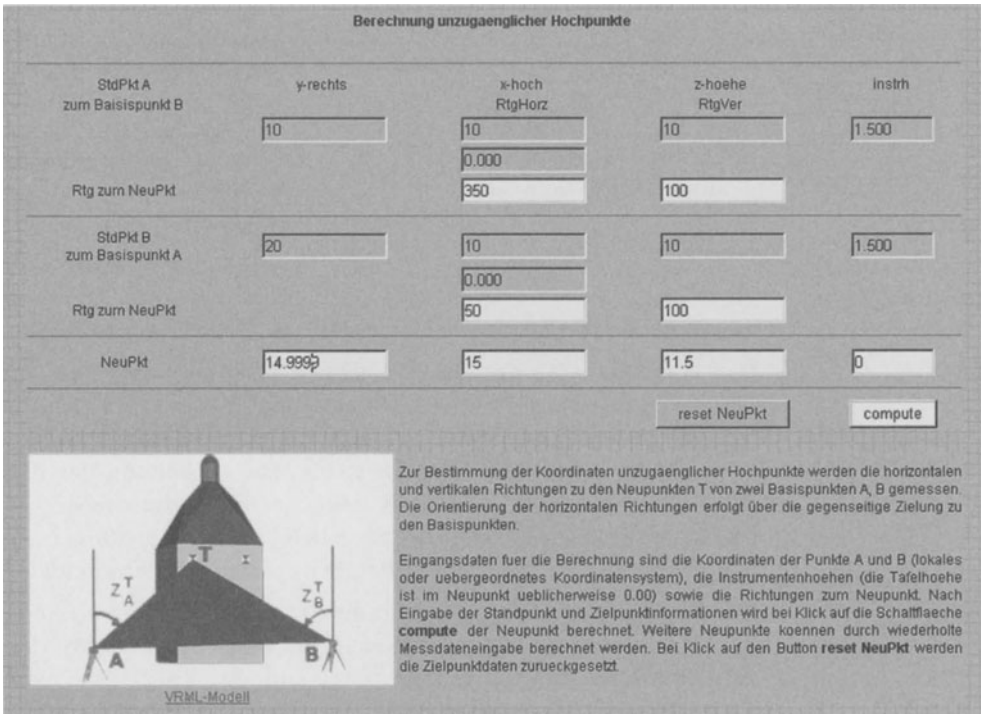


Abb. 5.7: Berechnung unzugänglicher Hochpunkte

Ausgangswerte für die Berechnung sind die Stationsdaten in den Basispunkten, das sind die Punktkoordinaten, die Instrumentenhöhe, die horizontale und vertikale Richtung zum Gegenüber und die Tafelhöhen. Hinzu kommen die Zielpunktdaten in jedem Standpunkt, das ist die horizontale und vertikale Richtung zum Hochpunkt, der i.d.R. direkt angezielt wird. Von einer Basis aus werden häufig mehrere Punkte bestimmt. Unser Programm sollte daher die wiederholte Eingabe neuer Zielpunkte bei unveränderter Basis vorsehen.

Über ein HTML-Formular kommt man mit wenig Aufwand zu einem komfortablen Benutzerinterface. Wir betrachten hier nur die wichtigsten HTML-Anweisungen: Im Head-Bereich befindet sich im Link-Tag die Referenz auf die Stilvorlage. Der Script-Tag enthält die Referenz auf das externe JavaScript-Programm *vws_dreidim.js*

```
<html>
<head> <title>Turmhoehenbestimmung</title>
  <link rel="stylesheet" type="text/css" href="css_form.css">
```

```
  <script language="JavaScript" src="vws_dreidim.js">
  </script>
</head>
```

Im Body-Bereich befindet sich das Formular mit den insgesamt 24 Input-Tags, dem Reset- und Compute-Button sowie einigen Erläuterungen. Die Positionierung ist mit einer Tabelle vorgenommen.

```
<form name="vws">
<input type="text" name="ya" size="12" width="12" value="0.00">
```

Das Formular wird mit vws bezeichnet. Die Input-Tags bekommen Namen entsprechend ihrer Bedeutung, hier z. B. ya für eine Koordinate des Basispunktes A und sofern möglich die Vorbesetzung eines Wertes im Attribut value. Die Gestaltung der Eingabefelder wird über Stylesheets gesteuert. Es wurde eine Klasse neuPkt definiert, die u.a. die Eingabefelder für Messungen zu den Hochpunkten farblich von den Stationsdaten unterscheidet.

Die Ereignisbehandlung der Buttons Reset und Compute besteht im Aufruf von JavaScript-Funktionen. Bei Klick auf Reset werden die Messwerte zu den Hochpunkten auf die Voreinstellungen zurückgesetzt danach können die Messungen zu einem anderen Neupunkt eingegeben werden

```
<input type="button" name="Button2"
       value="reset NeuPkt" onclick ="reset_neupkt()">
```

Bei Klick auf den Button Compute wird die Funktion zur Berechnung des Hochpunktes aufgerufen.

```
<input type ="button" name="Button1"
       style="background-color: #ffff99;"
       value="compute"
       onclick="vws_dreidim(
         document.vws.ya.value, document.vws.xa.value,
         document.vws.za.value,
                ...
         document.vws.vnp_b.value)">
```

Die Argumente beim Aufruf der Berechnungsfunktion werden in der Hierarchie der Applikationsobjekte angegeben, vgl. auch Abb.5.2 und die dortigen Erläuterungen. Auf die Angabe von window als oberstes Objekt kann man verzichten. Über document, den Namen des Formulars (hier vws) und den Namen des Elements (hier ya) kann man die Eigenschaft (hier value) erreichen. Die Notation bleibt bei der Wahl kurzer Namen übersichtlich.

Wie kommen nun die Argumente in der Funktion an und wie werden diese dort weiterverarbeitet? Das Programm *vws_dreidim.js* ist sehr übersichtlich und ausführlich dokumentiert. Wir betrachten exemplarisch die Parameterprüfung.

```
function vws_dreidim (
   y1, x1, z1, i1, y2, x2, z2, i2,
   h1, h2, hza, va, hzb, vb){
```

Die Parameter kommen als Zeichenketten und werden in Zahlen mit convert_to_num konvertiert. Der Aufruf lautet:

```
var ya = convert_to_number (y1);
```

In der Funktion wird der Parameter `number` mit `isNaN` geprüft. Es gibt entweder eine Fehlermeldung, oder das Ergebnis von `parseFloat()` wird zurückgegeben.

```
function convert_to_num (number){
  if(isNaN(number) == true){
    alert(number + " keine zulaessige Zahl");
  }
  return parseFloat(number);
}
```

Wenn alle Werte korrekt sind, kann man in die Berechnung einsteigen. Wir stellen ein geeignetes Datenmodell auf. Die Messdaten liegen im Gradmaß vor. Trigonometrische Funktionen benötigen die Argumente im Bogenmaß. Der Umrechnungsfaktor wird bestimmt mit

```
var RHO = (parseFloat(200.0)/Math.PI);
```

Sicherheitshalber wird die Konstante 200.0 mit `parseFloat()` konvertiert. PI ist eine Konstante des Objekts Math. Entsprechend gestaltet sich ein Funktionsaufruf

```
r1[0] = Math.sin(z[0]) * Math.sin(r[0]);
```

Der Ausdruck rechts ruft die Methode `sin()` des Objekts `Math` auf, Argument ist das erste Element des Arrays `z` bzw. `r`. Das Ergebnis des Ausdrucks wird dem ersten Element des Arrays `r1` zugewiesen. Wenn Sie mit Arrays nichts anfangen können, dann blättern Sie doch bitte zurück auf Seite 96. Auch im Kapitel 4.7 wurde die Benutzung von Arrays praktiziert.

Wenn nun alle Berechnungsergebnisse vorliegen, muss das auszugebende Ergebnis auf die gewünschten Nachkommastellen

gerundet werden und dem Attribut des Elements zugewiesen werden:

```
var df= 10000.0;
d = parseInt(d*df)
p0[0] = parseInt(p0[0]*df);

document.vws.ynpkt.value = p0[0];
```

Die Rundung wird durch Multiplikation und Umwandlung in eine Ganzzahl mit anschließender Division erzielt. Die Zuweisung an das Attribut des HTML-Dokuments erfolgt wieder über den Pfad durch die Applikationsobjekte.

Das JavaScript besteht aus ca. 180 Programmzeilen, die HTML-Datei etwa aus 120 Zeilen. Sie finden die Originalquelltexte gut dokumentiert auf der Webseite.

Zusammenfassung

Das JavaScript-Programm zur Berechnung der Koordinaten unzugänglicher Hochpunkte ist zur Lösung dieses komplexen Problems sicher hilfreich. Wir haben hier aber zwei Nachteile zu verzeichnen. Bei größeren Projekten kann man die Koordinaten der Basispunkte nicht aus einer Datei oder Datenbank entnehmen und die Berechnungsergebnisse stehen auch nicht zur unmittelbaren Weiterbearbeitung bereit. Vom Bildschirm abschreiben ist kein probates Mittel. Wir stellen daher fest, umfangreiche und häufig wiederkehrende Berechnungen sollten nicht mit JavaScript durchgeführt werden – es mangelt am Datenfluss.

Zum Einstieg in das Erlernen einer Programmiersprache und für kleinere Berechnungen ist JavaScript bestens geeignet. Hinzu kommen die Anwendungen bei Navigation, Interaktion und Überprüfung von Formulardaten, bevor diese an den Server übergeben werden.

Im übernächsten Kapitel werden wir Problemlösungen unter PHP auf dem Server entwickeln, die den Dateizugriff erlauben und damit echte DV-Lösungen in einer Client-Server-Umgebung ermöglichen.

6 Scalable Vector Graphics SVG

Scalable Vector Graphics (SVG) ist eine universell einsetzbare Grafiksprache für 2D-Darstellungen im Web. Einsatzbereiche von SVG sind u.a. technisch-wissenschaftliche Dokumentationen, interaktives Lehrmaterial und vor allem auch kartografische Darstellungen. In diesem Kapitel werden wiederverwendbare Bausteine für grafisch-interaktive Anwendungen entwickelt.

An Entwicklungswerkzeugen wird benötigt: ein Editor, ein SVG-Viewer und ein Webbrowser. Unter Windows ist der Editor Notepad hinreichend, von der Adobe-Website kann der SVG-Viewer heruntergeladen werden, als Browser ist der MS Internet Explorer geeignet. In dieser Umgebung wurden alle Anwendungen dieses Kapitels erfolgreich getestet.

Grafiken entstehen durch die Beschreibung und Platzierung von grafischen Elementen, unregelmäßigen Formen, Bildern und Text auf einer Zeichenfläche. Elementgruppen finden mittels Manipulationen Wiederverwendung. Die Maßstäblichkeit wird durch globale Abbildungstransformationen erhalten. Weitere Gestaltungseffekt bestehen u. a. durch die Anwendung von Schatten, Füllmustern und Farbverläufen.

Ergänzend zu diesen passiven Basistechniken bietet SVG Möglichkeiten der Interaktion und Animation. Letztere ist durch ein SVG-eigenes Animationsmodell oder durch Zugriff auf die Objektelemente über das DOM mit JavaScript realisiert.

In diesem Kapitel werden SVG-Komponenten vorgestellt, die einerseits die elementare Aufbereitung von Grafiken behandeln und andererseits die Möglichkeiten von Interaktion und Animation in einer webfähigen Umgebung aufzeigen. Der Import von massenhaften Datenmengen aus CAD-Umgebungen erfolgt über Konverter. Die geeignete Einsetzbarkeit unterschiedlicher Datenkonverter wird am Beispiel von Vektordaten und Bilddaten getestet.

Die beabsichtigten Applikationen richten sich auf die interaktive Darstellung von Grafiken im Internet zum Zwecke der Kommunikation und Information. Multimedia-Effekte wie Intros sind hier nicht Gegenstand der Betrachtungen.

6.1 Einführung in SVG

Das Akronym SVG trägt die Begriffe: skalierbar, Vektor und Grafik. Unter skalierbar versteht man im EDV-Sprachgebrauch eine Veränderung der Gesamtgröße, die Anpassung an sich verändernde Erfordernisse. In der grafischen Anwendung bedeutet skalierbar, dass keine Einschränkung auf eine feste Pixelgröße besteht. Beispielsweise nutzt eine SVG-Grafik die volle Druckerauflösung, ist aber mit dem gleichen Inhalt für den Bildschirm skaliert. Die im Web gültige Bedeutung für skalierbar gilt auch für SVG: Dateien, Benutzergruppen und Anwendungen müssen sich ohne Einschränkung veränderten Bedingungen anpassen können.

Vektorgrafiken bestehen aus geometrischen Formen, die mathematisch durch die Koordinaten der Stützpunkte und Art der Linienverbindung (Kreis, Gerade, Kurve) beschrieben sind. Rastergrafiken bestehen aus Pixelinformationen, die höheren Speicherplatzbedarf aufweisen und Qualitätsverlust beim Vergrößern zeigen. SVG gestattet die Einbettung von Rastergrafiken in die Vektorgrafik in Kombinationen auch mit unregelmäßigem Clipping.

Was ist SVG ?

SVG ist eine vom World-Wide-Web-Konsortium spezifizierte Beschreibung für zweidimensionale Grafiken und grafische Anwendungen in XML. Als XML-basierte Sprache profitiert SVG vom offenen Standard, der Plattformunabhängigkeit und dem mächtigen Funktionsumfang. Weiter ist SVG DOM-transparent und erlaubt dadurch den Zugriff auf die Elemente eines Web-Dokuments. Darüber hinaus ist der Einsatz von ECMAScript (standardisiertes JavaScript) möglich. Somit kann während der Laufzeit eine Veränderung der Attribute aller Elemente vorgenommen werden. SVG ist implementiert in Betrachtern (auch für mobile Geräte), Editoren, Exportern, Konversions-Programmen und serverseitigen SVG-Generatoren. Damit steht eine breite Palette von Dienstprogrammen für den rationellen Einsatz von SVG-Datenformaten zur Verfügung.

Die Betrachter, verfügbar als Stand-Alone-Version oder Plug-In, beinhalten einen XML-Parser, CSS-Parser, und eine SVG-Rendering-Engine zur Darstellung der Grafik. Neben der Darstellungsfähigkeit auf Bildschirmen und Mobilgeräten existiert auch eine Druckerfunktionalität. Typischerweise sind in interaktiven SVG-Betrachtern auch das Document Object Model, das CSS Object Model und ein ECMAScript Interpreter integriert.

Weit verbreitet ist der Adobe SVG-Viewer als Plug-In für Internetbrowser. Aus dem Batik SVG-Toolkit von Apache ist der Squiggle SVG-Viewer als Stand-Alone-Version zu nennen. Bekannte SVG-Editoren sind Jasc WebDraw oder Amaya. Beide sind WYSIWYG-Werkzeuge zur Erzeugung und Manipulation von SVG-Dateien.

Die populäre Grafik-Software Corel Draw kann SVG importieren und exportieren, ist also auch als Konverter einzusetzen. Konverter übersetzen CAD-Grafikformate und Bitmaps in SVG-Beschreibungen. Als zweidimensionales Format sind bei den Konvertierungen von CAD-Daten der dritten Dimension Einschränkungen zu akzeptieren. Die weiter unten angeführten Beispiele zur Konvertierung zeigen aber, dass auch kleinere Tools die dritte Dimension von der SVG-Konvertierung nicht ausschließen.

Mit Kenntnissen über die Struktur eines SVG-Dokuments und den wichtigsten Basisbefehlen kann man in die Bearbeitung von SVG-Dateien einsteigen. Detailinformationen und die Spezifikation befinden sich auf der Web-Site des W3C *http://www.w3.org/Graphics/SVG*. Zur Publikation im Internet wird die Datei mit der Erweiterungsbezeichnung `svg` oder `svgz` (für komprimierte Dateien) gespeichert und ist eigenständig oder in eine Internetseite eingebettet vom Server abzurufen.

Handhabung des SVG-Viewers von Adobe

Die Installation des SVG-Viewers von Adobe unter Windows-Systemen ist denkbar einfach. Rufen Sie die Web-Site *www.adobe.com/svg* auf und folgenden Sie den Anleitungen zum Download bzw. zur Installation des frei erhältlichen Viewers. Die Installation ist in nur wenigen Minuten realisiert. Auf der SVG-Seite von Adobe finden Sie auch beeindruckende Anwendungsbeispiele von Vektorgrafik im Internet. Mit SVG ist man nicht an Windows-Plattformen gebunden, Mac, Linux und Solaris werden ebenfalls unterstützt.

Nach Installation des Viewer-Plug-Ins können Sie eine SVG-Datei mit der Erweiterungsbezeichnung svg aufrufen. Auf der Seite *www.programmierpraktikum.de* finden Sie alle Beispiele dieses Buches. Rufen Sie eine Datei auf und klicken Sie mit der rechten Maustaste in den Bereich der SVG-Grafik. Das Kontextmenü mit den verfügbaren Befehlen zur Bedienung des Viewers, vgl. Abbildung 6.1, öffnet sich.

Sie können in der Grafik navigieren mittels Schwenken, Ein- und Auszoomen, oder zurück auf die Originalansicht schalten. Die

Qualität der Grafik kann zugunsten schnellerer Anzeige verringert werden.

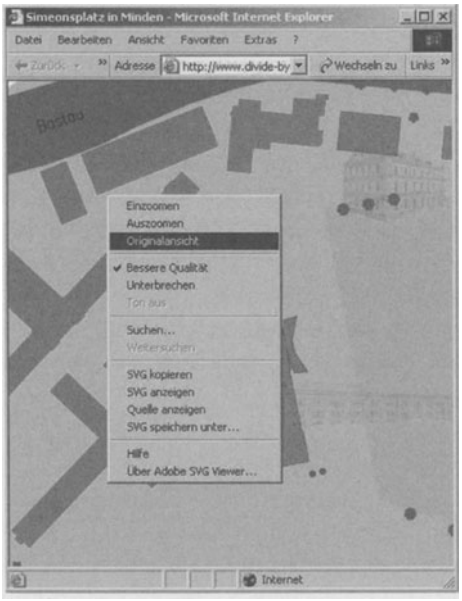


Abb. 6.1: Kontextmenü des Adobe SVG-Viewers

Verknüpfungen werden durch Änderungen des Mauszeigers signalisiert und können im aktuellen oder in einem neuen Fenster geöffnet werden.

Weiter bestehen Suchfunktionen zum Auffinden von Textstellen im Dokument. Unterschiedliche Kopierfunktionen erlauben die Speicherung der Anzeige als Bild bzw. die Übertragung des gesamten Dokuments, hier haben Sie auch die Option der komprimierten Speicherung. Einer kleinen Hilfedatei sind weitere Erläuterungen und die Beschreibung der Tastencodes zu entnehmen.

Struktur einer SVG–Datei

Die Dokumentenstruktur von SVG entspricht der einer XML-Datei und besteht im wesentlichen aus einem Header und einem Root-Element, darunter befinden sich verschiedene Kind-Elemente und deren Attribute. Zentrale Definitionen können in einem Definitionsabschnitt `<defs>` festgelegt werden und sind weiter unten in der Datei verwendbar. Weitere Strukturelemente sind Gruppen `<g>`, Beschreibungen `<desc>`, `<title>` und Gruppenreferenzen `<symbol>`. Elemente einer SVG-Datei werden in der Reihenfolge ihres Auftretens unter Berücksichtigung

der Opazität dargestellt, liegen also nach dem Prinzip „first in first out" übereinander. Jedes Element kann über einen Identifizierer eindeutig mit seinem Namen referenziert werden.

Die erste Zeile eines SVG-Dokuments definiert die verwendete XML-Version und ggf. den zu verwendenden Zeichensatz:

```
<?xml version="1.0" standalone="no" encoding="iso-8859-1"?>
```

Standalone ist ein Attribut zum DTD (*Document Type Definition*)-Bezug, das besagt, ob die DTD in der angegebenen Datei steht oder extern vorhanden ist. Zu allen XML-konformen Tags vgl. auch Kapitel 3.

Danach folgt die Angabe der benutzten DTD, diese bestimmt die verfügbaren Tags, Wertetypen für die Attribute und Optionen der Formatierung für das Style-Attribut.

```
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
    "http://www.w3.org/TR/2000/03/WD-SVG-
     20000303/DTD/svg-20000303-stylable.dtd">
```

Der eigentliche Code der SVG-Beschreibung wird in den Tag <svg></svg> eingebunden.

Attribute des SVG-Tags sind die Größe der Zeichenfläche `width, height` und die weiter unten zu erläuternden Parameter für die Abbildungstransformationen.

Bei Einführung einer Programmiersprache wird üblicherweise eine „Hello-World"-Botschaft programmiert. In der grafischen Anwendung muss die Flagge von Japan für ein erstes Programm herhalten. Dabei betrachten wir zunächst die Strukturierung der Datei. Im Root-Element ist eine Zeichenfläche von 500 x 500 Pixeln definiert. Der im Title-Tag eingeschlossene Text wird in der Titelzeile des Browserfensters angezeigt. Der Bereich desc wird nicht dargestellt.

Die darzustellende Grafik ist in drei Gruppen gegliedert: Hintergrundrechteck, Flagge mit Rechteck und Kreis und der Legende. Erläuterungen zu den Grafikelementen und deren Attribute folgen im weiteren Verlauf dieses Kapitels.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303
        Stylable//EN"
       "http://www.w3.org/TR/2000/03/WD-SVG-
       20000303/DTD/svg-20000303-stylable.dtd">
<!-- Kommentarzeile:
     Es folgt das-root-Element-->
```

```
<svg width="500" height="500">
  <title>Struktur einer SVG Datei</title>
  <desc>Beispiel Gruppe Kreis/Rechteck</desc>
  <g id="hintergrund">
    <!-- das Rechteck dient nur zur Darstellung
         der Hintergrundfarbe grau -->
    <rect id="hintergrund" x="-1024" y="-768"
      width="3072" height="2304"
      style="fill:rgb(128,128,128)"/>
  </g>
  <g id="flagge_japan">
    <rect id="rand" x="50"  y="100" width="400"
      height="300"style="fill:rgb(255,255,255);
      stroke:rgb(0,0,0); stroke-width:2"/>
    <circle id="punkt" cx="250" cy="250" r="75"
      style="fill:rgb(255,0,0)" />
  </g>
  <g id="Legende">
    <text x="50" y="450" style="font-family:
      'Verdana'; font-size:16px;
       fill:rgb(0,0,0)">
       Flagge von Japan
    </text>
  </g>
</svg>
```

Listing 6.1: Flagge von Japan, *svg_datei_1.svg*

Die Angaben zur Gestaltung sollten zweckmäßig mit Stilvorlagen erfolgen. Wie in XML-Dokumenten werden eingebettete Stylesheets in einem CDATA-Abschnitt eingefügt. Auf die Wiedergabe der Programmzeilen mit der XML-Version und der DTD-Definition wird in allen nachfolgenden Beispielen verzichtet.

```
<svg width="500" height="500">
  <style  type="text/css">
    <![CDATA[
      text {
        font-family:'Verdana'; font-size:16px;
        fill:rgb(0,0,0) }
      rect{
        fill:rgb(255,255,255);
        stroke:rgb(0,0,0);
        stroke-width:2}
      circle {
         fill:rgb(255,0,0)}
       ]]>
```

```
  </style>
  <title>Struktur einer SVG Datei</title>
  <desc>Beispiel Gruppe Kreis / Rechteck</desc>
  <g id="hintergrund">
    <!--Hintergrundfarbe grau,Style-
      sheet-Definition wird überschrieben -->
    <rect id="hintergrund" x="-1024" y="-768"
       width="3072" height="2304"
       style="fill:rgb(128,128,128)"/>
  </g>
  <g id="flagge_japan">
    <rect id="rand" x="50" y="100"
      width="400" height="300" />
    <circle id="punkt" cx="250" cy="250"
                        r="75"/>
  </g>
  <g id="Legende">
     <text x="50" y="450">
        Flagge von Japan</text>
  </g>
</svg>
```

Listing 6.2: Eingebettete Stylesheets, *svg_datei_2.svg*

Als externe Datei wird das Stylesheet mit folgender Anweisung vor dem Root-Element <svg> eingebunden:

```
<?xml-stylesheet href="stilvorlage.css"
      type="text/css"?>
```

Die Datei ist dann ohne Parser-Information unmittelbar mit den Definitionen aus der Stilvorlagen zu formulieren.

6.2 Grafikelemente: Geometrie und Attribute

Elementare Grafikformen wie Rechteck, Kreis, Ellipse, Linie, Polyline und Polygon sind Standardformen, die in SVG vordefiniert sind. Im einführenden Beispiel wurde aus den Grundformen Kreis und Rechteck die Flagge von Japan zusammengesetzt.

Grundlegende Formen

An dieser Stelle beschränken wir uns zunächst auf die Darstellung, Benennung, Positionierung und Gestaltung eines Rechtecks. Die Parameter für weitere Grundformen entsprechen weitgehend denen des Rechtecks oder sind aus der Geometrie heraus selbsterklärend.

Der Tag <rect /> wird mit folgenden Geometrieattributen aufgerufen:

`x = "koordinate"`	Koordinate der rechten unteren Ecke
`y = "koordinate"`	Koordinate der rechten unteren Ecke
`width = "länge"`	Breite des Rechtecks
`height = "länge"`	Höhe des Rechtecks
`rx = "länge"`	Radius der Abrundungsellipse für die Ecken
`ry = "länge"`	Radius der Abrundungsellipse für die Ecken

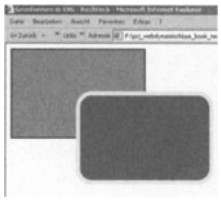
Alle Wertangaben für die Attribute beziehen sich auf das Benutzerkoordinatensystem, vgl. hierzu Abschnitt 6.3 Transformationen.

Hinzu kommen die Style-Attribute für die Gestaltung, die auch im Stylesheet zentral definiert sein können, s. oben.

`fill:`	bezeichnet die Füllfarbe oder das Füllmuster
`stroke:`	ist die Farbe der Umrandung
`stroke-width:`	Lienstärke der Umrandung

Farbwerte können als vordefinierte Namen, hexadezimale Werte oder RGB-Werte angegeben werden. Die Linienstärke kann in den von SVG akzeptierten Dimensionen angegeben werden. Die Anwendung von speziellen Füllmustern wird weiter unten abgehandelt.

Zur Benennung von Elementen (Tags) wird das Id-Attribut benutzt. Hierüber wird dem Element innerhalb der Grafik ein eindeutiger Name zugewiesen. Über die Identifikation kann eine Skriptsprache auf das Element zugreifen.



Die mit Listing 6. 3 erzeugte Grafik zeigt zwei Rechtecke, die absolut positioniert sind. Beide Rechtecke haben unterschiedliche Farbgestaltung und dokumentieren die Reihenfolge der Darstellung durch den SVG-Renderer. Das nachfolgende Element überlagert das vorhergehende.

```
<svg width="640" height="480">
  <title>Grunformen in SVG - Rechteck</title>
  <desc>Beschreibung des Tags rect</desc>
  <!--
     Rechteck 1:
       Hintergrundfarbe:
```

```
        Grau, definiert in RGB
      Randfarbe:
        Schwarz, definiert hexadezimal
      Strichstärke: 3 Pixel -->
  <rect id="rechteck_1"
    x ="10" y="10"
    width="240" height="160"
    fill="rgb(128,128,128)"  stroke="#000000"
          stroke-width="3px"/>
  <!--
    Rechteck_2: Gerundetet Ecken
      Hintergrundfarbe: Rot
      Randfarbe:Gelb,definiert über Namen
      Strichstärke    : 2 mm -->
  <rect id="rechteck_2" x="130" y="90"
    width="240" height="160"
    rx="25" ry="25"
    fill="red" stroke="yellow"
    stroke-width="2mm"/>
</svg>
```

Listing 6.3: Elementare Grafikformen, *svg_datei_3.svg*

Unregelmäßige Formen: Pfade

Pfade sind in SVG das grafische Element zur Beschreibung beliebiger unregelmäßiger Linien und Begrenzungen. Darüber hinaus werden Pfade bei Animationen zur Festlegung von Bewegungsrichtungen und zur beliebigen Ausrichtung von Text benutzt.

Die Definition von Pfaden ist vergleichbar mit herkömmlichen Grafiksprachen, `M` steht für *move* und bedeutet die Bewegung der „Schreibmarke" mit gehobenem Zeichenstift zur Zielposition, `L` steht für *line_to* und bedeutet die Bewegung mit gesenktem Zeichenstift zur Zielposition. Große Buchstaben stehen für absolute Koordinatenangabe, kleine Buchstaben geben die relative Position zur vorhergehenden an. Weiter existieren noch Befehle für unterschiedliche Arten von Linienverbindungen wie Kreisbogen, Bezierkurve u.a. Eine Zusammenstellung der Attribute für Pfade finden Sie auf der Website zum Buch in der Beispieldatei *svg_pfade.svg*. Die Formulierung eines Path-Tags beginnt mit der Kennzeichnung `d`. Zum Schließen eines Linienzuges wird die Bezeichnung `z` in den Wert des Attributes geschrieben.

Zur Gestaltung eines Pfades stehen, wie für alle anderen Grafikelemente auch, die Stil-Attribute `fill`, `stroke`, `stroke-width`

zur Verfügung. Erweiterte Elemente beziehen sich auf die Ausgestaltung der Linien in den Stützpunkten.

Wenden wir uns nun einem einfachen Beispiel eines Pfades zu. Für spätere interaktive Aufgaben benötigen wir eine „Sprechblase". Eine Figur wird diese bei Bedarf anzeigen. In die Sprechblase werden erläuternde Texte mit Hyperlinks eingeblendet. Hier bemühen wir uns zunächst um die Definition der Umrandung. Die Koordinatenangaben werden im lokalen System mit absoluten Werten vorgenommen.

Die Zeichnung beginnt in der linken oberen Ecke am Ende des Kreisbogens mit den Koordinaten `x=25, y=0` mit einer geraden Linie zum Punkt `x=225, y=0` gefolgt von einem Ellipsenbogen. Die Startposition des Bogens ist `x=225 y=0`, die Radien in `x` und `y` sind gleich, der Wert ist 25. Es folgen in der Beschreibung eines Ellipsenbogens drei Parameter: `x-axis-rotation, large-arc-flag, sweep-flag`. Wir setzen `0, 0, 1` und übergehen die Erläuterung dieser Werte zunächst. Der Linienzug wird vertikal weitergeführt bis zur Position `x=250, y=125`. Danach folgt wieder ein Kreisbogen mit anschließendem Linienzug, ein Kreisbogen in der linken unteren Ecke, wiederum eine Linie und der abschließende Bogen.

Die Bezeichnung für diesen Linienzug ist „Sprechblase". Damit die Grafik etwas vom Rand abgesetzt ist, benutzen wir hier bereits eine Transformation um 10 Pixel nach rechts und nach unten.

```
<svg width="640" height="480">
  <title>Unregelmaessige Formen in SVG</title>
  <desc> Beschreibung des path-Tags</desc>
  <!–Path-Attribute:
     id Attribut zur Identifikation
      d Attribut leitet den Datensatz ein
        Werte:
            M move to x,y
          L line to x,y
          A Ellipsenbogen
            x,y Startpunkt
            y-axis-rotation
            large-arc-flag
            sweep-flagg
            x,y Endpunkt -->
  <g transform="translate(10,10)">
    <path id="sprechblase"
```

```
        d="M25,0 L225,0
           A25,25,0,0,1,250,25
           L250,125
           A25,25,0,0,1,225,150
           L75,150 L35,195 L50,150
           L25,150
           A25,25,0,0,1,0,125
           L0,25
           A25,25,0,0,1,25,0"
           fill="rgb(192,192,192)"
           stroke="#000000"
           stroke- width="2px"/>
  </g>
</svg>
```

Listing 6.4: Elementare Grafikformen, *svg_datei_4.svg*

Zur allgemeinen Syntax eines Pfades kann festgehalten werden, dass auf den Befehlsbuchstaben unmittelbar die Werte durch Kommata getrennt folgen. Zwischen den Elementen ist ein Leerzeichen vorzusehen. Es ist noch die Bedeutung der Werte zur Orientierung der Ellipsenbögen zu erläutern.

Der erste Wert `x-axis-rotation-flag` bedeutet eine Drehung der Ellipse als ganzes gegenüber dem Koordinatensystem und wird als Winkel in Grad angegeben. Der Schalter `large-arc-flag` legt fest, welcher Teil der Ellipse gezeichnet wird, der Schalter ist 0 für den kürzeren Abschnitt, 1 für den längeren Abschnitt. Mit dem `sweep-flag` wird die Orientierung des Winkelfortschritts in der Ellipsengleichung festgelegt, 1 steht für positive Richtung des Winkels, 0 für negative Richtung des Winkels. Hieraus resultiert eine positive oder negative Krümmung. In aller Ausführlichkeit sind die Parameter in der Referenz *www.w3c.org/svg* beschrieben.

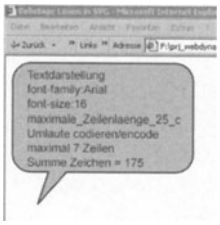
Text

SVG ist nicht für die Darstellung von langen Fliesstexten konzipiert. Es gibt aber eine Vielzahl von Textgestaltungsmöglichkeiten auch im Zusammenhang mit Filtern oder der Textausrichtung an Pfaden. Wodurch auch die notwendigen Gestaltungselemente zur Beschriftung von Karten und Plänen gegeben sind.

Wir füllen jetzt die Sprechblase mit Worten. Der Text wird linksbündig eingegeben. Bei einer Texthöhe von 16 Pixeln wird der Zeilenabstand hier mit 20 Pixel vorgegeben. Die konstruierte Sprechblase kann so 7 Zeilen mit einer Zeilenlänge von 25 Zei-

chen aufnehmen. Wie später gezeigt wird, können innerhalb dieses Textes auch Hyperlinks eingefügt werden.

Die Attribute des Text-Tags sind u.a. die Bezugsposition des Textes und Stilvorgaben für die Gestaltung. Der auszugebende Text erscheint zwischen dem einführenden Tag <text> und dem ausführenden Tag </text>.




```
<svg width="640" height="480">
  <title>Textdarstellung mit SVG</title>
  <desc>Anwendung des text-Tags </desc>
    <!-- die Sprechblase wird durch
         Text ergänzt.-->
  <g transform="translate(10,10)">
    <path id="sprechblase" d="M25,0 L225,0
      A25,25,0,0,1,250,25
      L250,125 A25,25,0,0,1,225,150 L75,150
      L35,195 L50,150 L25,150
      A25,25,0,0,1,0,125 L0,25
      A25,25,0,0,1,25,0"
      fill="rgb(192,192,192)" stroke="#000000"
      stroke-width="2px"/>
  <text id="t1" font-family="arial"
    font-size="16"
    x="20" y = "22">Textdarstellung</text>
    <!-- hier folgen die weiteren Texte-->
  </g>
</svg>
```

Listing 6.5: Textdarstellung mit SVG, *svg_datei_5.svg*

Jeder Text-Tag ist mit Gestaltungsattributen ausgezeichnet. Änderungen müssten an vielen Stellen im Quellcode vorgenommen werden. Aber glücklicherweise gibt es ja Stilvorlagen. Der Text-Tag wird um das Class-Attribut erweitert und an Stilvorlagen gebunden. Mittels des Style-Tags werden Textgestaltungen für die Klassen `sp` und `legende` festgelegt. Ebenfalls erhält die Sprechblase Stildefinitionen für eine eigene Klasse. Ergänzt wurde hier die Eigenschaft `opacity`, ein Wert für die Opazität (Transparenz). Die Gestaltungsattribute im Path-Tag und Text-Tag entfallen dadurch.

```
<svg width="640" height="480">
<title>Textdarstellung mit SVG und embedded StyleSheets</title>
  <desc>Anwendung des text-Tags svg_datei_6.svg</desc>
  <style  type="text/css">
  <![CDATA[
```

```
  text.sp {
    font-family:Arial; font-style :normal;
    font-size  :14px;  fill       :rgb(0,0,255)
  }
  text.legende {
    font-family :Arial;
    font-style  :italic;
    font-size   :14px;
    fill        :rgb(255,0,0)
  }
  path.sp {
    fill        :rgb(192,192,192);
    stroke      : #0000ff;
    stroke-width:2px;
    opacity:0.2;
  }
  ]]>
  </style>
  <!-- Grafik: Sprechblase mit Text -->
  <g id="sprechblase"
     tranform="translate(10,10)">
    <path class="sp" id="sprechblase_grafik"
       d="M25,0 L225,0 A25,25,0,0,1,250,25
          L250,125 A25,25,0,0,1,225,150 L75,150
          L35,195 L50,150 L25,150
          A25,25,0,0,1,0,125 L0,25
          A25,25,0,0,1,25,0"/>
    <text class="sp" id="t1" x="20" y= "22">
     Textdarstellung           </text>
    <text class="sp" id="t2" x="20" y= "42">
     font-family:Arial         </text>
    <text class="sp" id="t3" x="20" y= "62">
     font-size:16              </text>
    <text class="sp" id="t4" x="20" y= "82">
     maximale_Zeilenlaenge_25_c</text>
    <text class="sp" id="t5" x="20" y="102">
     Umlaute codieren/encode   </text>
    <text class="sp" id="t6" x="20" y="122">
     maximal7 Zeilen           </text>
    <text class="legende" id="t7" x="20"
     y="142">
     Summe Zeichen = 175       </text>
  </g>
</svg>
```

Listing 6.6: Textdarstellung mit Stilvorlagen, *svg_datei_6.svg*

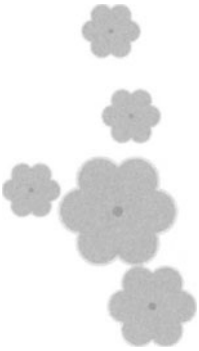
Wiederverwendung von Grafiken

Bereits definierte Grafiken können in SVG mittels des Use-Tags referenziert und wiederverwendet werden. Grundformen und auch Gruppenelemente, die mit <g> definiert und benannt wurden, können mit dem Use-Tag kopiert werden.

Use-Tag

Wir betrachten ein Lageplansymbol für einen Laubbaum, das mehrfach mit unterschiedlicher Skalierung abzubilden ist. Im Defs-Abschnitt der SVG-Datei ist die Geometrie mit den Eigenschaften formuliert. Zur Referenzierung des Mittelpunktes wurde die Translation benutzt. Dieser unrunde Wert ergibt sich aufgrund der Konvertierung aus CAD-Daten. Näheres hierzu weiter unten. Die weiteren Koordinatenangaben im Path-Tag sind in der Druckversion nicht wiedergegeben. Die vollständige Datei finden Sie im File *svg_datei_8.svg*.

```
<defs>
  <g id="lb" transform="translate
         (-489.65,-338.445)" >
    <desc>2D-Symbol fuer Laubbaeume in
      Lageplaenen</desc>
     <path   d="M491.13 338.45L491.11 338.22
       fill="lime" stroke="lime"
       stroke-  width="1"/>
     <path d="M489.65 321.55L488.74 320.55"
       fill="RGB(32,128,0)" stroke="lime"
       stroke-width="1"
       opacity ="0.5"/>
  </g>
</defs>
```



Im weiteren Verlauf der Datei kann jetzt das Element `lb` kopiert werden. Dabei ist `x, y` der Einfügepunkt und `xlink:href` referenziert das Element. Da hier unterschiedliche Größen des Symbols benutzt werden, ist für den Fall der Skalierung die Translation mit dem Referenzpunkt identisch. Der Einfügepunkt ist das Zentrum des Symbols. Die Skalierung ist unter `scale` angegeben.

```
<use x="361.55" y="483.06" xlink:href="#lb"/>
<use x="420.46" y="363.39" xlink:href="#lb"/>
<use x="435.91" y="427.09" xlink:href="#lb"/>
<use x="544.06" y="455.08" xlink:href="#lb"/>
<use x="0.0" y="0.0" xlink:href="#lb"
  transform="translate(452.32,569.62)
             scale(1.5)"/>
<use x="0.0" y="0.0" xlink:href="#lb"
  transform="translate(426.25,498.51)
```

```
scale(2.0,2.0)"/>
```

In der Sprachspezifikation von SVG ist auch die Referenzierung von Elementen in externer Dateien vorgesehen, die aber im Adobe SVG-Viewer 3.0 noch nicht implementiert ist.

Image-Tag

Bestehende Rastergrafiken können mit Vektorgrafiken kombiniert werden. Eine vorherige Datenkonvertierung ist nicht notwendig. Die externen Datenformate GIF, JPG und PNG können direkt in den SVG-Code über `xlink:href` eingebunden werden. Für das Grafikelement `image` gelten die gleichen Attribute wie für geometrische Elemente. Ebenfalls ist eine mit dem Image-Tag eingebundene Grafik animierbar.

Die Referenzposition einer Rastergrafik wird mit den Attributen `x,y` – und, wie oben bereits gezeigt, über die Transformation festgelegt. `Width` und `height` legen die Grafikgröße fest. Die externe Bilddatei *fritz.gif* wird im nachstehenden Code mehrfach mit unterschiedlicher Opazität und Größe eingefügt. Weiter unten wird diese Figur dann noch mit der bereits diskutierten Sprechblase ausgestattet.

```
<svg width="600.0" height="400.0">
  <title>Referenzieren von externen Grafiken </title>
  <desc>svg_datei_9.svg</desc>
  <g id="fritz">
    <image xlink:href="fritz.gif" x="10" y="10"
           width="138" height ="263">
    </image></g>
  <use x="0" y="0" xlink:href="#fritz"
    transform="translate(150,140) scale(0.5)"
    style="opacity:1"/>
  <use x="0" y="0" xlink:href="#fritz"
     transform="translate(300,140) scale(-0.5,0.5)"
     style="opacity:0.5"/>
</svg>
```

Listing 6.7: Einfügen von Bilddateien, *svg_datei_9.svg*

Die Originalgrafik hat eine Größe von 138px x 263px und ist unter dem Dateinamen *fritz.gif* im selben Verzeichnis mit der SVG-Datei abgelegt. Der Einfügepunkt ist (10,10). Mit dem Use-Tag wird die Datei wiederholt eingefügt. Dabei ist bereits von den nachfolgend noch zu beschreibenden Transformationen Gebrauch gemacht worden. Stilattribute, die in der Gruppendefinition noch nicht festgelegt wurden, können im Use-Tag individuell gesetzt werden. Bei der Bildauflösung ist die Zoomstufe zu

beachten. Mit niedriger Auflösung verliert die Grafik beim Einzoomen an Qualität. Ein Kompromiss zwischen Dateigröße und Bildqualität ist zu akzeptieren.

6.3 Transformationen

Abbildungstransformationen

SVG benutzt zwei unterschiedliche Koordinatensysteme: das Gerätekoordinatensystem, in dem das Gerätedarstellungsfenster (der sichtbare Bereich der Zeichenfläche) definiert wird und das Benutzerkoordinatensystem, in dem der Anwender seine Grafikelemente dimensioniert und positioniert.

Die Zeichenfläche (*canvas*) ist von unbegrenzter Größe. Der sichtbare Bereich wird im SVG-Tag mit den Attributen width und height begrenzt. Ist das SVG-Dokument in ein übergeordnetes Dokument eingebettet, z.B. in HTML mit dem Object-Tag, so steht die Begrenzung der sichtbaren Zeichenfläche in Abhängigkeit zu den übergeordneten Festlegungen.

Das Gerätedarstellungsfenster bezeichnet man auch als *viewport.* Ist keine spezielle Vereinbarung für das Benutzerkoordinatensystem vorgenommen, so erfolgt die Definition identisch zum Viewport mit einer 1:1 Abbildung. Der darauf abzubildende Bereich des Benutzerkoordinatensystems wird als *window* bezeichnet.

Der Grafikbereich wird im Root-Element durch die Attribute width und height im Gerätekoordinatensystem (Bildschirm) angegeben. Der Ursprung liegt in der linken oberen Ecke der Zeichenfläche, mit der positiven x-Achse nach rechts und der positven y-Achse nach unten. Mit dem Attribut viewBox wird der rechteckige Bereich des Benutzerkoordinatensystems definiert, der auf das Gerätedarstellungsfenster abzubilden ist.

Durch unterschiedliche Festlegungen dieser Systeme werden globale Transformationen und Skalierungen automatisch berücksichtigt. Alle Koordinatenangaben im weiteren Dokument beziehen sich danach immer auf das Benutzerkoordinatensystem. Sie finden die beschriebene Situation auch in der Abbildung 4.10 illustriert.

Werte für die Attribute `width` und `height` können in den Dimensionen `em`, `ex`, `px`, `pt`, `pc`, `mm`, `in` und Prozent angegeben werden. Die Dimensionen entsprechen den CSS-Vereinbarungen. Unterbleibt eine Festlegung der Dimension, dann gelten die Einheiten Pixel des Ausgabegerätes.

Die Funktionsweise der globalen Abbildungstransformation betrachten wir an einem Beispiel. Die darzustellende Gebäudeansicht hat eine Ausdehnung von etwa 920 Einheiten in x und etwa 195 Einheiten in y (im Benutzerkoordinatensystem positive y-Richtung nach oben).

Die Festlegung eines darzustellenden Bereichs im Benutzerkoordinatensystem von 1000 x 200 würde zur Gesamtdarstellung hinreichen. Das Seitenverhältnis ist 5:1. Zur Vermeidung von unterschiedlichen Maßstäben in x und y müsste der Viewport im gleichen Verhältnis definiert werden, z.B.: 500px x 100 px

Während die Attribute `width` und `height` des SVG-Tags die Größe der Zeichenflächen angeben, wird mit dem Attribut `viewbox` der darauf abzubildende Bereich mit folgenden Werten festgelegt:

`min-x, min-y`	Ursprung des Fensters im Benutzerkoordinatensystem, linke untere Ecke
`width, height`	Dimension des Fensters, Länge in x, Länge in y

Für das Beispiel der Abbildung 6.2 ist die Situation einfach zu erläutern. Ein Bereich von 1000 x 200 Einheiten des Benutzerkoordinatensystems, positioniert bei x=.0 und y= 250.0, soll auf einer Zeichenfläche von 750 x 150 Pixel dargestellt werden. Das führt zu keinen Abbildungsproblemen, da die Seitenverhältnisse des Darstellungsfensters und des Ausschnitts aus dem Benutzerkoordinatensystem gleich sind.

```
<svg width="750" height="150"
     viewBox="0.0 250.0 1000.000 200.000">
```

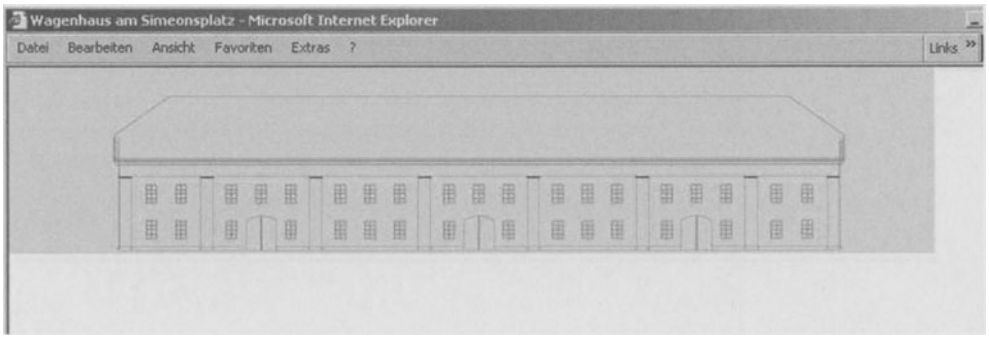


Abb. 6.2: Globale Transformation mit gleichen Seitenvehältnissen von Gerätefenster und Benutzerausschnitt

Bei veränderter Darstellungsfläche, quadratisch 750 x 750 Pixel, wird das Gebäude mit unterschiedlichen Maßstäben in den Koordinatenachsen dargestellt. Die Abbildung ist nicht mehr dem Original ähnlich, sondern weist affine Verzerrungen auf. Für das Verhältnis von Seitenlängen ist auch die Bezeichnung *AspectRatio* gebräuchlich.

```
<svg width="750" height="750"
     viewBox="0.0 250.0 1000.000 200.000">
```

An dieser Stelle sollte noch darauf hingewiesen werden, dass bei der Publikation eines SVG-Dokuments noch eine weitere Abbildungsvorschrift eintritt. Der Internet-Browser reserviert für das Dokument eine Fläche, in die es mit dem Embed-Tag oder dem Objekt-Tag eingebettet ist. Innerhalb dieses reservierten Bereichs leistet der SVG-Viewer seine Arbeit. Unsere Betrachtungen der globalen Abbildungstransformation berücksichtigen hier nur die für den SVG-Viewer anzugebenden Parameter.

Die Seitenverhältnisse einer Abbildung können mit dem Attribut `preserveAspectRatio` (Werte: `none`, `meet`, `slice`) erhalten werden. Zusätzlich ist auch noch die Position des Windows innerhalb der Viewbox anzugeben. Unabhängig von der Größe des Darstellungsfensters und den gewählten Seitenverhältnissen wird mit den Werten `xMidYMid` und `meet` immer eine korrekte Darstellung mittig in der Zeichenfläche erreicht.

```
<svg width="750" height="375"
     viewBox="0.0 250.0 1000.000 200.000
     preserveAspectRatio="xMidYMid meet">
```

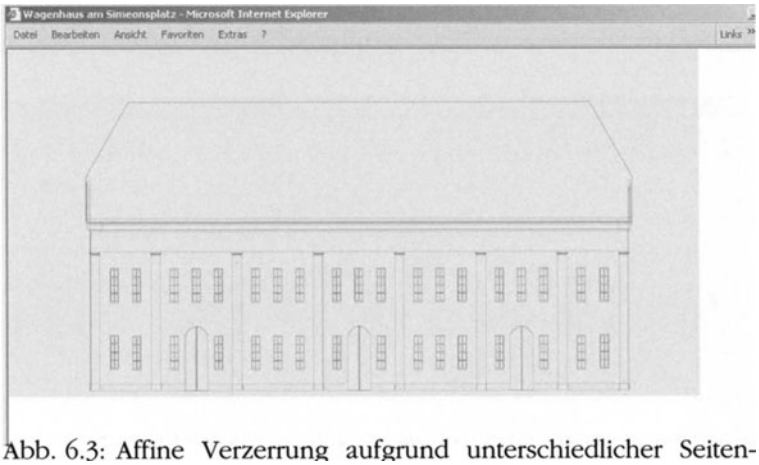


Abb. 6.3: Affine Verzerrung aufgrund unterschiedlicher Seitenverhältnisse desDarstellungsfensters und der ViewBox.

Es gilt die Empfehlung, obigen Code immer anzuwenden. Die weiteren Werte von `preserveAspectRatio` können der SVG-Referenz entnommen werden. Gewollte Manipulationen an Grafikelementen sollten aber nicht mit der globalen Abbildungstransformation durchgeführt werden. Zur gewollten Verformung von Grafikelementen benutzt man bessser das Attribut `transform`.

Alle Beispieldaten finden Sie auf der Webseite. Öffnen Sie die Datei mit einem Texteditor und manipulieren Sie die Daten der Abbildungsvorschrift. Nach Speichern folgt der Doppelklick im Explorer zur Ansicht. Machen Sie sich mit der globalen Abbildungstransformation vertraut und experimentieren Sie ein wenig mit den Werten.

Transformation der Grafikelemente

Zur Positionierung, Drehung und Formveränderungen der Grafikelemente (manchmal auch als Containerlemente bezeichnet) dient das Transform-Attribut. Transformationen können nach Translation (`translate`), Rotation (`rotate`), Skalierung (`scale`) oder Scherung (`skew`) getrennt vorgenommen werden. Geschachtelte Transformationen sind auch gemeinsam durch Angabe der Elemente der Transformationsmatrix durchführbar.

Die Werte des Transform-Attributs stellen eine Liste der einzelnen Transformationen dar, die in der angegebenen Reihenfolge durchgeführt werden. Als Trennzeichen dient ein Komma oder Leerzeichen, die Parameter stehen in runden Klammern:

`translate(tx,ty)`	tx Translation in x, ty Translation in y
`scale (sx[,sy])`	sx, sy Maßstabsfaktoren entlang der Achsen, sy optional bei ungleichen Werten
`rotate(winkel [,cx,cy])`	Drehwinkel in grad, optional Koordinaten des Drehpunktes
`skewX(winkel)`	Schieflage gegenüber der x-Achse
`skewY(winkel)`	Schieflage gegenüber der y-Achse

Zur Darstellung der Matrixoperationen sind mathematische Kenntnisse notwendig. Im Kapitel 4: Java Applets wurden hierfür die Grundlagen gelegt. Für weitere Informationen schauen Sie bitte in der SVG-Referenz unter *www.w3.org/svg* nach.

Zur Demonstration der Transformationen benutzen wir die schon bekannten Beispiele. Aus den zuvor bereits definierten Grafikelementen „Sprechblase“ und „Fritz“ wird eine Einheit, bezeichnet mit „Guide“, gebildet. Eine Gruppe mit mehreren Elementen bezeichnet man auch als Container. Dieser „Guide“ soll später einmal Fragen von Anwendern beantworten, dient also als grafisches Interface für ein Hilfesystem.

Die vollständige Datei finden Sie unter svg_datei_10.svg auf der Website, hier sehen Sie einen Auszug.

Die Stilvorlagen-Definitionen sind hinter dem SVG Root-Element wiederum eingebettet.

```
<svg width="640" height="480">
  <title>Transformationen von Grafikelementen und Containern
  <title>
  <desc>svg_data_10.svg</desc>
  <style  type="text/css">
    <![CDATA[
      <!-- hier stehen die Stylesheet Festlegungen -->
    ]]>
  </style>
```

Es folgen jetzt die Grafikelemente. Innerhalb des Containers „Guide“ sind die Sprechblase und Fritz beschrieben. Fritz wird um 112 Pixel nach unten positioniert. Die Sprechblase wird skaliert und um 80 Pixel nach rechts verschoben. Der gesamte Container wird an die Position 50, 50 gesetzt.

```
<!-- Der zusammengesetzte Museumsfuehrer  -->
<g id="guide" transform="translate(50,50)">
  <g id="fritz">
    <image xlink:href="fritz.gif"
      x="0" y="112"
      width="138" height ="263"></image></g>
  <g id="sprechblase"
     transform="scale(0.75,0.75) translate(80,0) ">
    <path class="sp" id="sprechblase_grafik"
    <!-- die Grafik ist aus vorhergehenden
         Beispielen bekannt-->
    <text class="sp" id="t1" x="20" y= "22">
        Guten Tag,</text>
    <!-- hier folgen die weiteren Texte-->
  </g>
</g>
</svg>
```

Die ursprünglich formulierten Koordinaten für den Pfad der Sprechblase und die Positionierung der Texte bleiben beibehalten. Bei Wiederverwendung der Grafik „guide“ ist nur noch die Positionierung zu beachten.

Abb. 6.4: Transformation von Grafikelementen und Containern

6. 4 Grafik-Effekte

SVG bietet eine Vielzahl von Grafikeffekten, die, sinnvoll eingesetzt, zu attraktiven Präsentationen beitragen. Farbverläufe und Füllmuster sind für Hintergründe geeignet. Schattenverläufe können durch Filter realisiert werden und geben den Objekten einer Grafik eine gewisse räumliche Anordnung. An drei ausgewählten Beispielen werden in diesem Abschnitt die Möglichkeiten der Grafikeffekte angedeutet.

Farbverläufe

Lineare Farbverläufe werden durch den LinearGradient-Tag beschrieben. Dieser Tag greift auf folgende Attribute zurück:

`<gradientUnits>` legt fest, wie ein Objekt oder die Grafikfläche gefüllt werden soll. Werte sind `userSpaceonUse` und `objectBoundingBox`, `<gradientTransform>` definiert Transformationen für den Verlauf.

`X1, y1` und `x2, y2` sind zwei Punkte eines virtuellen Pfades, an dem die Änderung der Farbintensität vorgenommen wird. Die

Punkte legen die Positionen der 0% und 100% Marke des Stop-Tags fest.

`SpreadMethod` definiert die Füllung der Restfläche mit den Werten `pfad`, `reflect` oder `repeat`. Pfad ist die Standardeinstellung. `Reflect` bewirkt eine Umkehrung, bei `repeat` wird der Verlauf wiederholt.

Zur Erzielung des Verlaufs muss mindestens ein Verlaufsabschnitt festgelegt werden. Dieser Abschnitt wird mit dem Stop-Tag, einem Child-Element von `linearGradient` festgelegt. Hiermit werden die Grenzen der Verlaufsabschnitte gekennzeichnet. Werte sind die Farbe und Opazität an den Grenzen.

```
<defs>
  <linearGradient id="hintergrundverlauf"
       gradientUnits="userSpaceOnUse"
       x1="0" y1="-480" x2="0" y2="960">
    <stop offset="50%" style="stop-color:black;
       stop-opacity:0"/>
    <stop offset="70%" style="stop-color:black;
       stop-opacity:1"/>
 </linearGradient></defs>
```

Das Hintergrundrechteck wird wie folgt angegeben:

```
<rect x="-640" y="-480"width="1280"height="960"
style="fill:url(#hintergrundverlauf)"/>
```

Abb. 6.5: Linearer Hintergrundverlauf, *svg_datei_11.svg*

Füllmuster

Mit Füllmuster werden Elemente bezeichnet, die ein weiteres durch eine Kontur begrenztes Element mit grafischem Hintergrund versehen. Ein Füllmuster wird zeilen- und spaltenweise wiederholt, bis die definierte Fläche vollständig mit dem Muster

gefüllt ist. Diese Vorgehensweise wird als „kacheln", engl. tiling, bezeichnet.

Zur Definition eines Füllmusters wird der Pattern-Tag im Defs-Abschnitt benutzt, <patternUnits> legt die Beziehung der angegebenen Koordinaten fest. Mit <width> und <height> wird die Kachelgröße des Füllmusters festgelegt, das Muster selbst wird mit <xlink:href> referenziert.

Im Beispiel wird das Attribut <fill> mit Verweis auf die id der Musteridefinition, *#hintergrund muster*, vorgenommen. Hier ist <fill> in die Style-Anweisung eingebunden.

```
<svg width="640" height="480"
     viewbox="0,0,640,480">
  <title>Hintergrundmuster</title>
  <desc>svg_data_12.svg</desc>
  <defs>
    <pattern id="hintergrund_muster"
       patternUnits="userSpaceOnUse"
       x="0" y="0"  width ="105" height="67">
    <image xlink:href="adler_relief.gif"
          width ="105" height="67"></image>
    </pattern>
  </defs>
  <rect x="-640" y="-480" width="1280" height="960"
    style="fill:url(#hintergrund_muster)"/>
</svg>
```

Listing 6.8: Hintergrundmuster, *svg_datei_12.svg*

Filter

Weitere Gestaltungshilfsmittel zur Erzielung von Grafikeffekten sind Filter. Filter werden beispielsweise zur Veränderung von Farben oder als Weichzeichner eingesetzt. Filter werden ebenfalls im Defs-Abschnitt einer SVG-Datei definiert und finden wieder durch Referenzierung im Fill-Attribut in der weiteren Datei Anwendung. Die Gestaltung von Filtern kann recht komplex ausarten.

Das hier zu betrachtende Beispiel benutzt den Gauß'schen Weichzeichner zur Erzielung eines Schatteneffekts. Da das eigentliche Bild von den Grauwerten her allein keinen entsprechenden Schattenwurf erzeugt, wird es mit einem Rechteck hinterlegt. Derartige Schatten geben der Gestaltung einer Web-Site ein plastisches Aussehen und sollten gezielt eingesetzt werden.

```
<svg width="640" height="480"
       viewbox="0,0,640,480">
```

```
<title>Bilder unterlegt mit Schatten</title>
<desc>svg_datei_13.svg</desc>
<defs>
  <pattern id="hintergrund_muster"
     patternUnits="userSpaceOnUse"
     x="0" y="0"  width ="105" height="67" >
    <image xlink:href="adler_relief.gif"
     width ="105" height="67"></image>
  </pattern>
  <filter id="schatten">
    <feGaussianBlur stdDeviation="2" result="wschatten"/>
    <feOffset in="SourceGraphic"
       dx="-10" dy="-10" result="original"/>
    <feMerge>
      <feMergeNode in="wschatten"/>
      <feMergeNode in="original"/></feMerge>
  </filter>
</defs>
<rect x="-640" y="-480" width="1280" height="960"
  style="fill:url(#hintergrund_muster)"/>
<rect x="140" y="120" width="320" height="240"
  style="filter:url(#schatten)"/>
<image xlink:href="plan_1873_3224.jpg"
   x="130" y="110" width="320" height="240" />
</svg>
```

Listing 6.9: Einsatz von Filtern, *svg_datei_13.svg*

6.5 Interaktion und Animation

SVG ist interaktiv. Das ist die Fähigkeit, auf vom Benutzer initiierte Ereignisse entsprechend zu reagieren. Vom Benutzer ausgelöste Ereignisse sind z. B. Mausbewegungen und Mausklicks. Der Benutzer kann Hyperlinks aufrufen, Animationen oder Skript-Programme aktivieren. In der Sprachspezifikation sind 26 Ereignisse definiert, die von SVG unterstützt werden. Im vorliegenden Zusammenhang sind onMouseover, onMouseout und onClick von Bedeutung, diese drei Ereignisse gehören zu den sog. Pointer-Events und werden in den nachstehenden Komponenten behandelt.

Die Veränderung des Abbildungsmaßstabes (zoom-in, zoom-out) und das Verschwenken des Bildausschnittes (panning) gehören zu den Standardfunktionen des Viewers, und sind daher nicht von den o. g. Ereignisssen abhängig.

Hyperlinks

Der von HTML bekannte Anchor-Tag wird auch von SVG unterstützt. Alle zwischen <a> und eingebundenen Elemente bilden die Schaltfläche. Bei Mausberührung verändert sich der Maus-Cursor, und bei Mausklick wird zu der mit `xlink:href` angegebenen URL verzweigt. Die URL kann auf eine Identifikation in einem beliebigen Dokument verweisen. Das Attribut `target` wird angewandt, wenn der Hyperlink in einem neuen Fenster oder einem MultiFrame HTML-Dokument geöffnet werden soll.

```
<a xlink:href="oekonomie.wrl" target="_blank">
   <path
      d="M510.56 205.18L512.12 189.24
         536.99 191.76 535. .... "/>
 </a>
```

Als Schaltfläche kann aber auch eine ganze Datengruppe ausgezeichnet werden:

```
<a xlink:href="oekonomie.wrl" target="_blank">
  <g id="oekonomie">
      <path d="M510.56 205.18L512.12 189.24
       .../>
      <!-- weitere Grafikdaten der Gruppe -->
  </g>
</a>
```

Animationen

Das Web ist ein dynamische Medium. SVG unterstützt Veränderungen einer Grafik während der Laufzeit, d.h. Grafikelemente verändern ihre Eigenschaften, ohne neu vom Server hochgeladen zu werden. Die Veränderungen der Grafik unterliegen zeitlich ablaufenden Modifikationen wie die Bewegung entlang von Pfaden oder das Ein- und Ausblenden, sogen. fade-in / fade-out-Effekte. Objekte können ihre Größe, Drehung oder Farbe ändern.

Der Zugriff auf die Elemente einer Grafik erfolgt über das SVG-eigene Animationsmodell mit dem Animate-Tag oder über das DOM und Skriptprogrammierung.

Den Unterschied werden wir an einem kartographischen Beispiel herausarbeiten. Der Zugang zu unterschiedlichen Epochen einer städtebaulichen Situation wird über einen einfachen Mausover-Effekt möglich. Die Situation wird durch Ein- und Ausblenden von Bildern erzeugt. Bei einem Mausklick wird dann zu einem weiteren Dokument verknüpft.

Abbildung 6.6 zeigt die Ausgangssituation. Rechts neben der Titelgrafik ist eine Zeitleiste angezeigt. Bei Mausbewegung über diese Zeitleiste wird links von der Zeitleiste der zugehörige Lageplan eingeblendet. Bei Mausklick auf ein Element der Zeitleiste wird der Link zu einer neuen Seite ausgeführt.

Sie erkennen an der Abbildung, dass die hier vorgestellten Komponenten dem Benutzerinterface einer umfangreichen Präsentation zuzuordnen sind. Die Programme sind aber durchaus auf andere Projekte portierbar und als Bausteine wiederverwendbar.

Die Zeitleiste besteht aus Rechtecken, die durch den Anchor-Tag mit externen Dokumenten verlinkt sind. Über diesen Rechtecken ist die Epoche durch einfachen Text notiert. Die Gestaltung der Texte erfolgt mittels Stilvorlage. Jeder Text und jedes Rechteck ist durch den Id-Tag eindeutig gekennzeichnet.

Abb. 6.6: Benutzerinterface für eine interaktive Präsentation

```
<!-
    Zeichnet eine senkrechte Zeitachse
    mit 5 Epochen
-->
<g id="zeitachse"
  transform="translate(822,201)">
  <!-- Rechtecke mit Links -->
  <a xlink:href="dummy.html" target="_self">
    <rect class = "epoche" id="go1820"
```

```
      x="0" y="0" width="50" height="30"
      rx="5" ry="5"/>
  </a>
  <a xlink:href="simeonsplatz_2004.svg"
     target="_self">
    <rect class="epoche" id="go2004"x="0" y="240" width="50"
      height="30" rx="5" ry="5"/>
  </a>
  <!-- Texte Jahreszahlen -->
  <text class="epoche" id= "1820" x="25 "y="20">1820</text>
  <text class="epoche" id= "2004" x="25" y="260">2004</text>
</g>
```

Die eigentliche Animation besteht nun aus dem Ein- und Ausblenden der den Epochen zugehörigen Lagepläne. Diese Lagepläne liegen als Bilder vor und sind mit dem Image-Tag eingefügt. Die Opazität für die Titelseite ist mit 1 festgelegt, alle darüber liegenden Pläne haben die Opazität 0, sind also völlig transparent.

Bei Mausberührung des Rechtecks mit der Jahreszahl wird die Opazität des entprechenden Bildes zeitgesteuert auf 1 verändert. Bei Verlassen der Schaltfläche wird dieser Vorgang wieder rückgängig gemacht. Beide Vorgänge werden durch den im Image-Tag eingebetteten Animate-Tag ausgelöst.

```
<g id="lageplaene">
<!-- die Titelseite wird als sichtbar
     deklariert -->
<image id="titel" xlink:href="titel_folie.jpg"
  x="152" y="92" width="640" height="480" opacity="1.0">
</image>
<!-- alle weiteren Grafiken werden als
    unsichtbar deklariert -->
<image id="Bild_1820"
  xlink:href="plan_1820_6448.jpg"
  x="152" y="92" width="640" height="480" opacity="0.0">
```

Es folgt nun die eingebettete Animation. Der Wert des Attributs `opacity` wird innerhalb einer Sekunde, Attribut `dur`, von 0.0 auf 1 gesetzt, ist also nach einer Sekunde voll sichtbar. Der Beginn dieser Aktion, Attribut `begin`, wird durch ein Ereignis ausgelöst. Das Ereignis ist definiert als `mouseover`, gültig für das Objekt mit der Identifikation *go1820*. Der Wert für das Attribut `fill` wird auf `freeze` gesetzt, damit ist der Animationseffekt zunächst beendet und verbleibt bis zum Eintreten einer erneuten Animation in diesem Zustand.

Der alternative Wert `remove` würde den ursprünglichen Zustand am Ende der Animation wieder herstellen. Dieser Vorgang wird aber im Beispiel durch eine weitere Animation ausgelöst. Die tritt bei Austritt des Mauszeigers aus der Schaltfläche ein. Die Schaltfläche ist wiederum identifiziert mit `go1820`, das Ereignis ist `mouseout`. Die Dauer der Animation beträgt eine Sekunde, die Opazität wird von 1 auf 0 gesetzt. Der Zustand wird durch den Wert `freeze` nach der Animation wieder beibehalten.

```
  <animate id="a1" attributeName="opacity"
    begin="go1820.mouseover" values="0.2;1"
    dur="1s" fill="freeze"/>
  <animate id="f1a" attributeName="opacity"
    begin="go1820.mouseout" values="1;0" dur="1s"
    fill="freeze"/>
</image>
<image
  <animate ... /><animate .../>
</image>
```

Nach dem gleichen Muster werden nun die weiteren Bilder eingefügt.

Das SVG-interne Animationsmodell beinhaltet die SMIL-Animation Spezifikation. SMIL ist die Abkürzung von *Synchronized Multimedia Integration Language*. Dieses Modell repräsentiert die generellen Möglichkeiten von XML-Animationen. Bearbeitet wird die Spezifikation von der W3C Synchronized Multi-Media-Arbeitsgruppe. SVG hat darüber hinaus einige spezielle Erweiterungen.

Allgemein wird mit dem Animate-Tag immer das zu verändernde Attribut definiert, die Veränderung des Wertes formuliert, die Dauer der Animation festgelegt und das auslösende Ereignis bestimmt. Dabei muss das Ereignis nicht auf das Animations-Objekt selbst ausgeübt werden. Es ist auch auf ein Fremdobjekt anwendbar, da das Ereignis an das ID-Attribut gebunden ist.

Eine weitere Anwendung von Animationen ist die Bewegung einer Grafik entlang eines Pfades. Am Beispiel der Legende eines Lageplans erarbeiten wir das Verfahren. Auf einem Lageplan befindet sich eine Legende mit einem Gebäudeverzeichnis, eingefügt als Gruppe mit dem Text-Tag. Alle Texte haben eine eindeutige `id` und sind in einen Anchor-Tag eingebettet.

```
<g id="texte" transform="translate(600,500)">
  <text id="txt1" x="0" y="10"style="fill:red">
```

```
    Wo bitte ist...?
  </text>
  <a>
    <text id="pos1" x="0" ="36">
      Wagenhaus</text></a>
  <a>
    <text id="pos2" x="0 y="54">
      Defensionskaserne</text></a>
<!--
   weitere Schaltflächen folgen-->
  <text class="legende" id="txt1"x="0" y="140">
    Klick auf den Text ...</text>
</g>
```

Neben den Texten befindet sich noch ein sichtbarer Wachsoldat:

```
<image x="560" y="520" width=" 32" height=" 96"
  xlink:href="wachsoldat.gif"
  style="visibility:visible">
</image>
```

Allerdings sind hinter dieser Grafik nochmals fünf weitere vorhanden, die aber erst an ihre Position zu bewegen sind. Wir betrachten eine dieser Grafiken:

```
<a xlink:href="wagenhaus.wrl" target="blank">
  <image id="wache_wagenhaus" x="560" y="520"
    width=" 32" height=" 96"
    xlink:href="wachsoldat.gif"
    style="visibility:visible">
    <animateTransform
      attributeName="transform"
      begin="pos1.click"
      dur="2s"
      fill="freeze"
      calcMode="linear"
      from="0 0" to="-360 -134"
      type="translate" additive="sum"/>
   </image>
</a>
```

Das Image-Element ist bezeichnet mit `wache_wagenhaus` und positioniert das Bild *wachsoldat.gif* auf den Punkt 560, 520. Das Bild ist sichtbar (`visible`). Eingebettet ist das Element in einen Anchor-Tag. Diesen Link trägt das Element mit sich, auch wenn es seine Position verändert. Auf der jetzigen Position kann es noch nicht angeklickt werden, da es von dem oben beschriebenen Image verdeckt ist.

Der Animations-Tag ist vom Typ `<animateTransform>`. Der Transformationstyp ist `translate`. Das Attribute `additive="sum"` bewirkt eine relative Translation zur augenblicklichen Position. Auf dem Transformationsweg vom Punkt (0,0) zum Punkt (-360,-134) wird eine lineare Interpolation durchgeführt, die Dauer der Bewegung beträgt 2 Sekunden. Ausgelöst wird die Animation durch einen Mausklick auf das Objekt mit der `id="pos1"` durch die Anweisung `begin="pos1.click"`. Die Animation eines Objekts wird also ausgelöst durch ein Ereignis, das auf ein anderes Objekt ausgeübt wurde.

Der Wachsoldat wird auf die Position bewegt und verbleibt dort. Wird der Mausklick wiederholt, verschwindet das Bild und der Animationsvorgang beginnt erneut. Im vorliegenden Fall können die Wachsoldaten auf ihre Position geschickt werden und zeigen dem Benutzer, wo sich das gewünschte Objekt befindet. Ist der Wachsoldat an seiner Zielpositition angekommen, dann führt ein weiterer Klick zu dem eingebundenen Link, der im Falle des Elements `id="pos1"` das interaktive 3D-Modell aufruft. Mit dem Attribut `target` können unterschiedliche Frames angesteuert werden. Nach Klick auf alle Positionsangaben haben die fünf Wachsoldaten ihre Stellung bezogen. Das Erscheinungsbild entspricht nunmehr dem der Abbildung 6.7

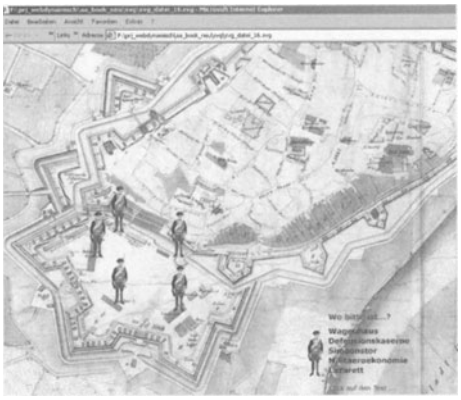


Abb. 6.7: Die Wachsoldaten haben Position bezogen. Die mit ihnen verknüpften Links sind jetzt verfügbar.

Scripting

SVG hat einen ECMAScript (standardisiertes JavaScript)-Interpreter integriert. Die Skriptsprache dient als Erweiterung des SVG-Viewers und stellt dem Anwender Befehle zur eigenen Programmierung zur Verfügung, die der Skriptinterpreter in Maschi-

nenbefehle umsetzt. Funktionen, die innerhalb des Script-Elements implementiert sind, können im gesamten Dokument benutzt werden.

Der Aufruf des Codes erfolgt ereignisgesteuert. Nutzereingriffe wie Tastatureingaben und Mausoperationen werden von Event-Handlern ermittelt und verarbeitet. Die Reaktion auf ein Ereignis ist üblicherweise der Aufruf einer JavaScript-Funktion.

Das *Document Object Model* (DOM) ist eine definierte Schnittstelle für XML- und HTML-Dokumente. XML hat die Einbindung allgemeiner Dateiformate zur Folge, also nicht nur Text. Das DOM ist eine abstrakte Schnittstellenbeschreibung, die zur Anwendung in der konkreten Scriptsprache implementiert sein muss. Alle Dokumente werden nach außen objektorientiert abgebildet. Die Objekte eines Elements sind die Knoten in einer hierarchischen Baumstruktur.

JavaScript kann intern oder extern in SVG eingebunden werden. Die interne Einbindung erfolgt mit dem Script-Tag

```
<script >
  <![cdata[
    /* JavaScript-Code
   ]]>
</script>
```

Eine Datei mit JavaScript-Code erhält die Erweiterungsbezeichnung js. Die externe Einbindung erlaubt die Mehrfachverwendung der Funktionen in beliebigen Dokumenten und erleichtert sehr die Programmpflege. Die Einbindung erfolgt mittels:

```
<script xlink:href="programm.js" type="text/javascript"/>
```

Zum Einstieg betrachten wir nun zunächst ein sehr einfaches Beispiel. Ein Kreis mit den Attributen `fill="red"` und `stroke="green"` tauscht bei Mausberührung die Farbwerte. Beim Verlassen der Kreisfläche wird der ursprüngliche Zustand wieder hergestellt. Diese Attributänderung soll mittels JavaScript und Zugriff auf die Objekte über das DOM erfolgen.

```
<svg width="200" height="200">
  <script type="text/ecmascript">
    <![CDATA[
      function objekt_farbwechsel
                (evt,stroke_farbe,fill_farbe){
        var svgDokument;
        var svgElement;
        svgDokument=evt.getTarget().getOwnerDocument();
```

```
        svgElement=svgDokument.getElementById("kreis");
        svgElement.setAttribute("fill",fill_farbe);
        svgElement.setAttribute("stroke",stroke_farbe);
      }
     ]]>
   </script>
<g id="daten" transform="translate(100,100)">
  <circle id="kreis"
    cx="0" cy="0" r="50"
    fill="green" stroke="red" stroke-width="10px"
    onmouseover="objekt_farbwechsel(evt,'green','red')"
    onmouseout ="objekt_farbwechsel(evt,'red','green')"/>
</g>
</svg>
```

Bei Eintritt der Ereignisse `onmouseover` und `onmouseout` wird die Funktion `objekt_farbwechsel()` aufgerufen. Die Parameter dieser Funktion sind das Ereignis selbst, sowie die Farbwerte für `fill` und `stroke`.

Innerhalb der Funktion werden die Variablen `svgDokument` und `svgElement` deklariert. Der DOM-Baum wird zunächst rückwärts verfolgt, und zwar in der Reihenfolge ausgelöstes Ereignis, Ziel des Ereignisses und das Dokument, in dem sich das Ziel befindet.

```
svgDokument=evt.getTarget().getOwnerDocument();
```

Dem Browser (Viewer) ist jetzt das Dokument bekannt. Nunmehr kann der DOM-Baum vorwärts verfolgt werden. Ausgehend vom Dokument wird das Objekt mittels der Methode `getElementByID()` identifiziert. Das gesuchte Element trägt die Bezeichnung `kreis`.

```
svgElement=svgDokument.getElementById("kreis");
```

Mit der Funktion `setAttribute()` werden die Werte entsprechend der eingehenden Funktionsparameter geändert.

Nach der gleichen Vorgehensweise verändern wir nun das Beispiel der Abbildung 6.4. Während die Figur „Fritz“ immer sichtbar ist, soll die Sprechblase nur bei Mausberührung erscheinen. Innerhalb des Textes wurde noch ein Link eingefügt. Die JavasScript-Funktion hat eingebettet folgende Codierung:

```
<script type="text/ecmascript">
  <![CDATA[
```

```
  function objekt_sichtbar(evt,sichtbarkeit){
    var svgDokument; var svgElement;
svgDokument=evt.getTarget().getOwnerDocument();
svgElement= svgDokument.getElementById("sprechblase");
svgElement.setAttribute("opacity",sichtbarkeit)
  }]]>
</script>
```

Ereignissteuerung und Aufruf erfolgt an zwei Stellen und zwar bei der Sprechblase selbst und bei der Figur Fritz. Dieses hat zum Ziel, auch den eingefügten Link im Text der Sprechblase zu erreichen.

```
<g id="guide" transform="translate(50,50)">
<g id="fritz">
  <image xlink:href="fritz.gif"  x="0" y="112"
         width="138" height ="263"
    onmouseover="objekt_sichtbar(evt,'1')"
    onmouseout ="objekt_sichtbar(evt,'0')">
  </image></g>
<g id="sprechblase" transform="scale(0.75,0.75)
   translate(80,0) "
   opacity="0"
   onmouseover="objekt_sichtbar(evt,'1')"
   onmouseout ="objekt_sichtbar(evt,'0')">
```

Sie finden den vollständigen Quelltext in der Datei *svg_datei_19.svg*.

6.6 SVG in HTML-Seiten einbinden

Die empfohlene Erweiterungsbezeichnung für SVG-Dateien ist `.svg`. Mit `gzip` komprimierte Dateien sollten die Erweiterungsbezeichnung `.svgz` verwenden. Das Programm `gzip` einschließlich Dokumentation kann von *www.gzip.org* heruntergeladen werden. Bei großen Dateien empfiehlt sich die Kompression zur Reduzierung der Dateigröße, die eine kürzere Übertragungsdauer zur Folge hat. Die Kompressionsrate beträgt um 80 %. Sie können aber auch eine SVG-Datei in den Adobe-Viewer laden und als komprimierte Datei wieder abspeichern. Ein SVG-Dokument kann als alleinstehende Web-Seite direkt von einem Web-Browser geladen werden.

Referenzieren von SVG in einer Web-Seite

Eine Parent-Webseite verweist auf ein separat gespeichertes SVG-Dokument und definiert dieses als Komponenten der Parent-Webseite. In HTML-Dokumenten erfolgt die Einbindung über die Elemente `<embed>` oder `<object>`. Wobei der Embed-

Tag kein Standard ist und der Object-Tag noch nicht in allen Browsern implementiert ist.

MIME-Typen (*Multipurpose Internet Mail Extensions*) beschreiben den Typ von Dateiinhalten und sind eine Spezifikation in der Kommunikation zwischen Browser und Server. Der Browser gibt bei der Anfrage an, welche Daten er bevorzugt verarbeiten kann, Der Webserver übermittelt die angeforderte Datei einschließlich der Information, um was für einen Dateityp es sich dabei handelt. Der Mime-Typ für SVG ist `image/svg-xml`.

Die Einbindung eines SVG-Dokuments in eine Tabelle mit <embed> kann wie folgt vorgenommen werden:

```
<html> <head>
<title>SVG in HTML einbetten</title></head>
<body  BGCOLOR="#dddddd">
<div align="center">
 <table width="480" border="0" cellspacing="0“ cellpadding="0">
  <tr><td>
  <embed type="image/svg-xml"
     name="SVG-Skripting" src="svg_datei_19.svg"
     width="640" height="480">
  </embed>
  <noembed>Embed-Tag nicht implementiert </noembed></td></tr>
  </table>
</div></body></html>
```

Alternativ ist der Object-Tag wie folgt zu verwenden:

```
<object data ="svg_datei_19.svg" type ="image/svg+xml"
        width="640px" height="480px">
</object>
```

Weitere Möglichkeiten zur Einbindung von SVG bestehen in der geplanten Erweiterung des Img-Tags, der Inline-Einbettung in XHTML oder der Referenzierung mit dem Anchor-Tag und Aufruf als alleinstehende Seite in einem eigenen Ziel-Frame.

SVG-Manipulation über HTML und JavaScript

Wie zuvor beschrieben, betten wir eine SVG-Seite in ein HTML-Dokument ein. Die Anweisung sieht wie folgt aus:

```
<embed width="480" height="480"
src="simeonsplatz.svg" name="Simplatz" type="image/svg+xml">
```

Die Datei Simeonsplatz.svg haben wir schon zu Beginn dieses Kapitels bei der Vorstellung des SVG-Viewers benutzt. Dort sind 3D-Ansichten von Gebäuden in einen Lageplan eingebunden. Diese Bilder sollen nun wahlweise durch Klick in eine Checkbox ausgeblendet oder angezeigt werden.

Die Checkboxen werden innerhalb eines Formulars definiert.

```
<form name="onoff" action="">
  <input type="checkbox" value=""
         onclick="zeige_elem(this,'DefKas')">
</form>
```

Der Name des Formulars ist onoff. Bei Klick in die Checkbox wird eine JavaScript-Funktion zeige_elem aufgerufen. Als Parameter werden übergeben: this als Identifikation, welche Checkbox ausgewählt wurde, dadurch ist in der Funktion der Zustand on/off und der Name des gewünschten Elements bekannt, so wie in der SVG Datei angegeben.

```
<image id="DefKas"
  xlink:href="defensionskaserne.gif" x="400" y="270"
  width="144" height="36" >
</image>
```

Das oben auszugsweise beschriebene Dokument wird in vollständiger Darstellung im Webbrowser wie in der Abbildung 6.8 angezeigt. Die Funktionalität des JavaScript-Programms ist wie folgt zu diskutieren:

```
<SCRIPT type="text/javascript">
  function zeige_elem(checkbox, element_name){
//
// Suche fuer das Element element_name das zugehoerige
// Stil-Objekt,
// setze dann die Sichtbarkeit entsprechend des Zustandes der
// Checkbox
//
  var svgobj= document.embeds['Simplatz'].getSVGDocument().
  getElementById(element_name);
  if (!checkbox.checked){
    // nicht sichtbar
    svgobj.setAttributeNS(null,
      'visibility','hidden');
  }
  else {
    // sichtbar
    svgobj.setAttributeNS(null,
       'visibility','visible');
  }
}
</script>
```

Dieses Beispiel zeigt sehr deutlich die Verknüpfung und Einbettung von HTML und SVG sowie die Anwendung von JavaScript und den Zugriff auf die Objekte über das DOM. Zur vollständigen Darstellung des Sourcecodes wird wieder auf die Website verwiesen.

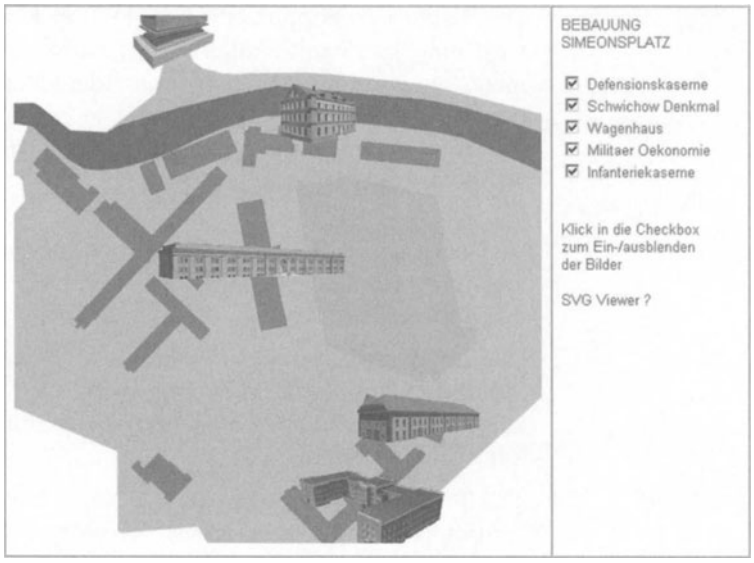


Abb. 6.8: SVG und HTML/JavaScript

6.7 SVG-Implementierungen und Konverter

Autoren-werkzeuge

Autorenwerkzeuge stellen ein Benutzerinterface zur Erstellung komplexer Anwendungen bereit. Die Sprachelemente bleiben dem Anwender unter der Benutzeroberfläche verborgen. Es ist aber ähnlich wie mit Fremdsprachen, man bekommt zu einem Land nur Zugang, wenn man dessen Sprache kennt. Auch bei Autorentools ist meistens etwas Handcodierung notwendig, insofern kann auf die Kenntnis von Sprachelementen nicht verzichtet werden.

Aus dem Angebot an Autorenwerkzeugen für SVG ist an vorderster Stelle *Adobe Illustrator* zu nennen, ein Grafikprogramm, in dem SVG in Kombination mit JavaScript und CSS vollständig unerstützt wird. Auch das weit verbreitete *CorelDraw* unterstützt das SVG-Format und ist ebenfalls als leistungsfähiges Autorenwerkzeug einzusetzen.

Eine interessante Alternative zu den großen professionellen Paketen kann *Jasc Web Draw* sein. Dieses Softwarepaket ist ein Werkzeug zur Erstellung und Manipulation von SVG-Grafiken, vgl. Abbildung 6.9, mit einer Lizenzgebühr, die im unteren Bereich von Shareware anzusiedeln ist.

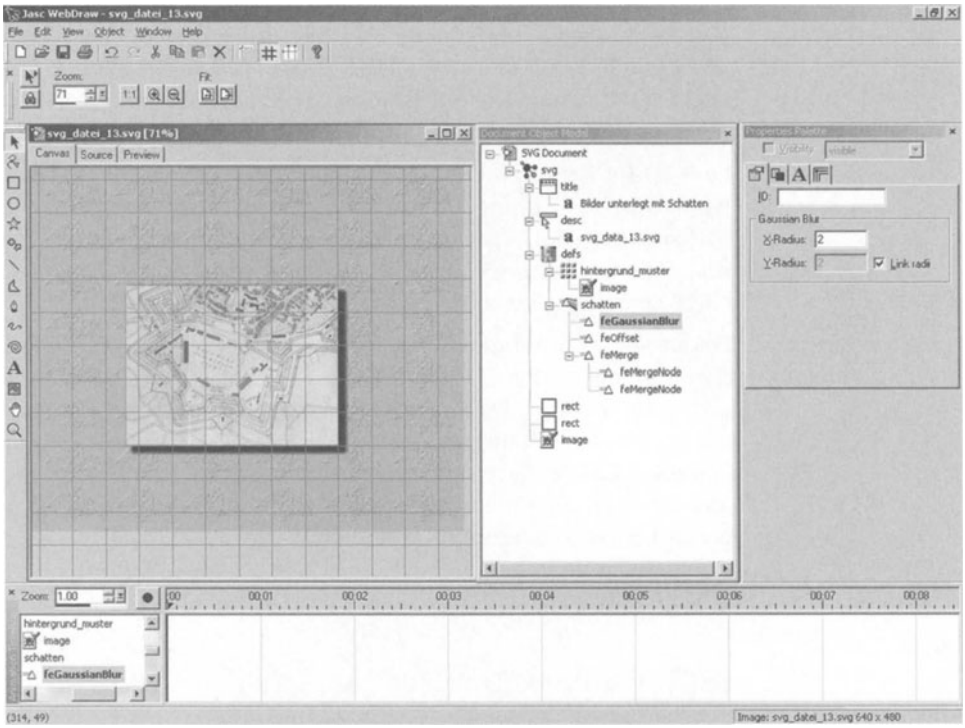


Abb. 6.9: Benutzerinterface von *Jasc WebDraw*

In mehreren Bearbeitungsfenstern werden die unterschiedlichen Objektinformationen bereitgestellt. Die eigentliche Bearbeitung erfolgt im grafischen Modus oder im alphanumerischen Modus mit Anzeige des Sourcecodes. Die Objekte können aus dem DOM-Baum ausgewählt werden. Attributwerte sind in einer Eigenschaftspalette einstellbar. Für Animationen ist der zeitliche Verlauf in einer Zeitleiste kontrollierbar. Editiervorgänge sind sofort grafisch überprüfbar.

Das Eingabe- und Ausgabeformat von Jasc WebDraw ist ausschließlich SVG mit Importfunktionen für Bilddateien. Die Bearbeitung bestehender Vektordaten aus CAD-Systemen kann nur nach vorhergehender Konvertierung erfolgen.

Konverter

SVG-Konverter haben die Aufgabe, existierende CAD-Daten in das SVG-Format zu überführen. Dabei ist zu beachten, dass SVG ein zweidimensionales Datenformat ist und die dritte Dimension nach der Konvertierung nicht mehr verfügbar ist. Gute Erfahrungen wurden mit dem Konverter *ACME CADConverter* gemacht. Diese Software liegt als eigenständiges ausführbares Programm vor, kann aber auch in einer DLL-Version in eigene Entwicklungen eingebunden werden. *ACME CADConverter* importiert DWG und DXF-Dateien. Die Software ist als Shareware im Internet verfügbar (Abb.6.10).

Die 3D-Information in CAD-Dateien wird bei der Ansichtsauswahl, die den AutoCAD-Ansichten weitgehend entspricht, berücksichtigt. Die Datenkonvertierung wird durch Layer-Selektion unterstützt. Auf die komplette Layerinformation der ursprünglichen Zeichnung besteht innerhalb des Konverters Zugriff.

Leider sind die Originalwerte der Koordinaten nach der Konvertierung nicht mehr verfügbar. Diese werden durch Bildschirmkoordinaten ersetzt. Daher ist bei schrittweisen Konvertierungen auf gleiche Abbildungsmaßstäbe zu achten. Ferner bleibt naturgemäß auch die Geometrie der Ursprungsdaten nicht mehr erhalten. Kurven der Originalzeichnung werden durch Pfade mit geraden Linienelementen approximiert.

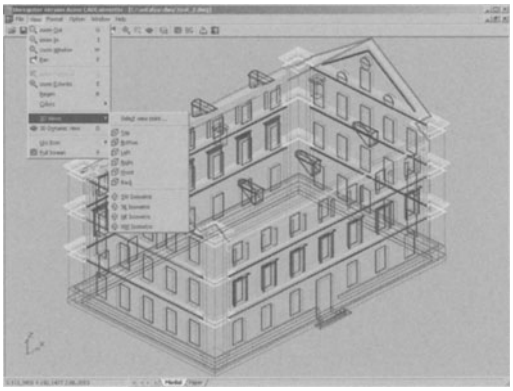


Abb. 6.10: ACME CADConverter, 3D-CAD Daten in 2D SVG-Formate konvertieren

Als Alternative zum Import von Bilddateien mit dem Image-Tag ist die vorhergehende Konvertierung eines Bitmaps in ein SVG-Format vorzusehen. Die Kompressionsrate ist abhängig von der Charakteristik des Bildmaterials., hohe Kompressionsraten sind mit Cartoon-Grafiken zu erzielen. Die Bitmapgrafik wird danach

unmittelbar in die SVG-Datei eingebettet. *SVGFactory*, ebenfalls eine im Web verfügbare Freeware, wurde ohne Probleme erprobt. Das Benutzerinterface ist denkbar einfach. Entweder werden Bilder aus dem Zwischenspeicher oder WMF- bzw. BMP-Dateien in ein Zielverzeichnis konvertiert. Neben Standard SVG kann auch das komprimierte SVG-Format erzeugt werden.

Aus einer 5 MByte großen Bitmap-Datei, RGB 1600x1200 Pixel Auflösung, wurde eine nur noch 339 KByte große SVG-Datei. Abbildung 6.11 ist ein Screenshot mit der angeführten Datei und zeigt das denkbar einfache Benutzerinterface. Das Konvertierungsergebnis kann im Web-Browser unmittelbar angezeigt werden.

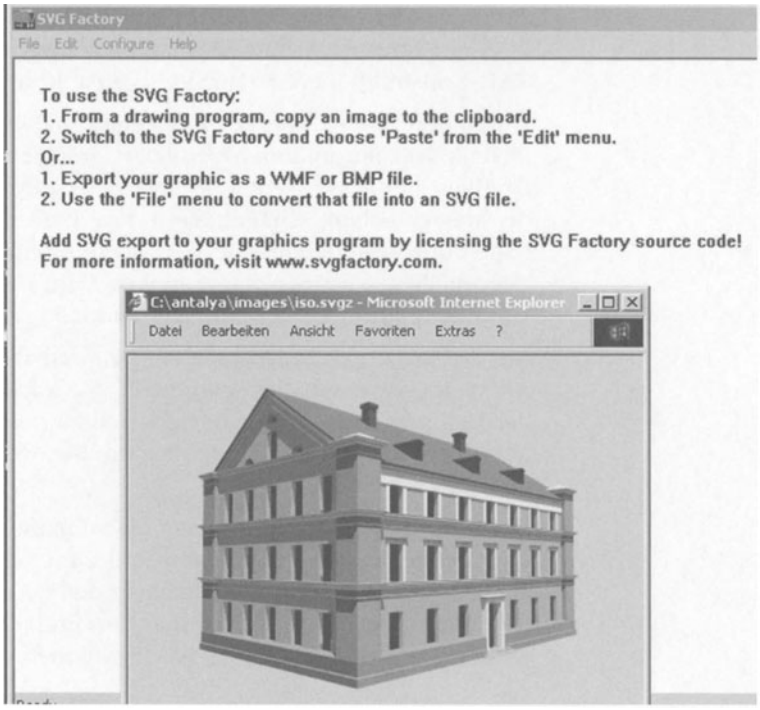


Abb. 6.11: Konvertierung von Bitmaps nach SVG mit der Freeware SVG Factory.

6.8 Zusammenfassung

Scalable Vector Graphics ist die Antwort der Web-Gemeinde auf Macromedias Flash. Durch das nicht proprietäre Format und als offener Standard hat SVG in vielen Anwendungsfeldern bereits weite Verbreitung gefunden.

Multimedia-Anwendungen wie Eingangsseiten, die den Hyperlink „Skip Intro" tragen, wurden hier nicht behandelt.

Es wurde gezeigt, dass die in XML formulierte Sprache durch Integration anderer Web-Standards wie XHTML, CSS, DOM und ECMA-Script ein mächtiges Werkzeug für interaktive Grafiken darstellt.

Aufgrund der Skalierbarkeit von Vektorgrafiken sind diese dem Rastergrafikformat vorzuziehen. Geometrische Daten können aus CAD-Konstruktionen in das SVG-Format konvertiert werden.

Durch den Einsatz von Rechnertechnik hat sich die Darstellungstechnik von Illustrationen verändert. Im Web werden interaktive Grafiken publiziert. Wenn Sie in einem gedruckten Stadtplan eine Strasse suchen, schauen Sie in das Straßenverzeichnis und erhalten dort das Planquadrat. Das Suchen beginnt. In einem interaktiven Plan wird der Name eingegeben, die in der Datenbank gefundene Position wird im Plan markiert.

Am Beispiel der „Wachsoldaten" wurden die Interaktions- und Animationseigenschaften vorgestellt .Sie klicken einen Namen in der Legende. Das Symbol bewegt sich zu der gesuchten Position, über den integrierten Link erhalten Sie weitere Informationen über das angefragte Objekt.

Der SVG-Viewer bietet Betrachtungsfunktionen zum Vergrößern und Verkleinern der Abbildung und zum Schwenken. Qualitätsverluste treten bei Vektorgrafiken in der Vergrößerung nicht auf. Die Abbildungen sind skalierbar für Endgeräte mit hochauflösenden Monitoren bis hin zu Mobiltelefonen und Printmedien.

Die serverseitige Generierung von SVG-Dokumenten aufgrund von Benutzeranfragen mit Zugriff auf umfangreiche Datenbestände oder Datenbanken ist eine Standardanwendungen in der Web-Kartographie.

7 Serverseitige Dynamik mit PHP

Mit serverseitiger Dynamik wird die Generierung von individuellen Webseiten durch ein Programm aufgrund von Benutzeranforderungen während der Laufzeit auf dem Server bezeichnet. Die angeforderte Webseite liegt demnach noch nicht vor, sondern wird erst nach Auswertung der Anfrage erstellt. Der Austausch von Daten zwischen Client und externem Programm wird durch die Serversoftware per CGI-Schnittstelle geregelt. Die CGI-Software ist nicht an eine Programmiersprache gebunden. Alternativen zum Einsatz von CGI-Programmen sind Scriptsprachen als Servererweiterung, vgl. auch Kapitel 1. Abbildung 1.5 erläutert das Prinzip beim Einsatz von PHP. Die Abkürzung PHP steht für *Hypertext Pre-Processor*. Die vom Client angeforderte Seite muss vom Script erst erstellt werden, bevor der Server die Daten an den Client übermittelt.

Die bisher im Buch verwendeten Entwicklungsumgebungen für XHTML/CSS, XML oder JavaScript beschränkten sich auf einen Webbrowser und einen einfachen Texteditor. Mehr Bedienungskomfort wurde durch Installation spezieller Editoren erreicht. Anspruchsvoller gestaltete sich die Umgebung bei der Entwicklung von Java-Applets. Im Kapitel 4 war das Java Software Developer Kit zu installieren und in eine integrierte Entwicklungsoberfläche einzubetten.

Die Entwicklungsumgebung für PHP unterscheidet sich von den bisher bekannten erheblich. Man kann sich zwar auf eine lokale Client-Server-Installation beschränken, es wird aber ein Zusammenwirken von Webserver, PHP, Browser und Editor auf dem lokalen System benötigt.

Wenn Sie es ganz einfach haben wollen, benutzen Sie eine vorkonfigurierte Version von Apache, MySQL und PHP. Unter Linux als LAMP bekannt, unter Windows mit der Bezeichnung WAMP. Komplette Distributionen mit Voreinstellungen bekommen Sie im Internet z.B. bei:

http://www.apachefriends.org/de/xampp-windows.html.

Füttern Sie Ihre Suchmaschine mit den Begriffen LAMP oder WAMP, und Sie werden fündig. Häufig finden Sie auch geeignete

Distributionen auf den CDs der Internet-Magazine. Wenn Sie auf den Begriff XAMP treffen, dann ist hierunter die Installation für Linux und/oder Windows möglich. Als ersten Schritt nehmen wir eine individuelle Installation des Apache Webservers mit PHP unter Windows vor und betrachten die wichtigsten Einstellungen.

7.1 Einrichten einer PHP-Entwicklungsumgebung

Installation und Konfiguration des Apache Webservers unter Windows

Als Open-Source-Projekt ist der HTTP-Webserver Apache in Quellcode oder binärer Version für Windows verfügbar. Den Quellcode müssen Sie selbst zur ausführbaren Datei compilieren. Wir nehmen den Download einer binären Version des Apache Webservers von der URL *http://www.apache.org* vor. Aus Kompatibilitätsgründen wird die Version 1.3.31 installiert (Abb. 7.1).

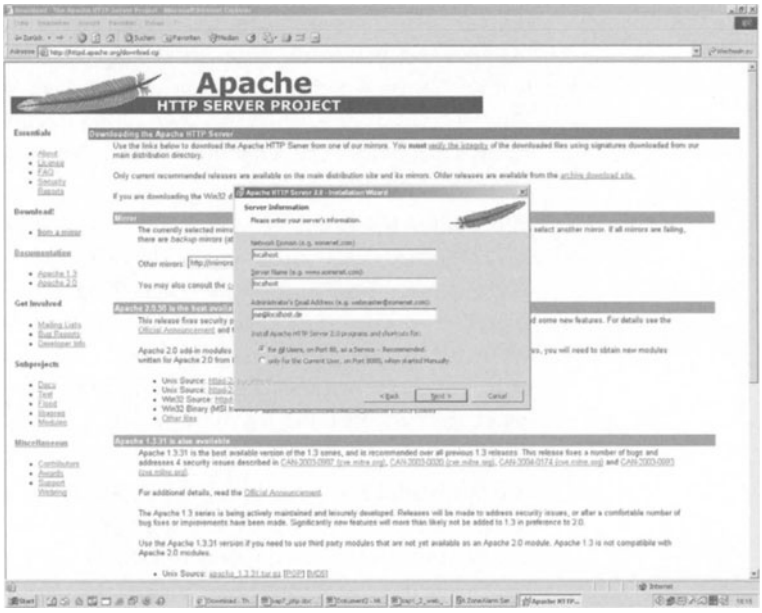


Abb. 7.1: Installation der binären Version von Apache 1.3.31 (August 2004) unter Windows

Folgen Sie dem Dialog der Installation. Geben Sie bei den Server-Informationen für die Network Domain und den Server-Namen `localhost` an. Die Email-Adresse können Sie mit `me@localhost.de` besetzen. Wählen Sie am besten gleich die Installation als *Dienst*. Sie können den Server im Kommando-Modus anhalten oder starten. Wechseln Sie dazu in das Ver-

zeichnis, in das Sie *apache.exe* installiert haben, z.B. `c:\web\Apache`. Mit `net stop apache` beenden Sie den Dienst, der Neustart erfolgt mit `net start apache`.

Testen Sie die Installation, indem Sie einen HTTP-Client aufrufen - ihren bevorzugten Webbrowser. Geben Sie in der Adresszeile `http://localhost` ein. Es öffnet sich die Index-Datei (Abb. 7.2)

Abb. 7.2: Adresseingabe http://localhost oder http:127.0.0.1 zeigt die erfolgreiche Installation des Apache-Servers

Die Installation erzeugte eine komplexe Verzeichnisstruktur. Ausführbare Dateien befinden sich im Verzeichnis *Apache/bin*. Im Verzeichnis *Apache/conf* befindet sich die kommentierte Konfigurationsdatei *httpd.conf*. Bevor Sie Änderungen an dieser Datei vornehmen, sollten Sie eine Sicherungskopie anfertigen und den laufenden Dienst anhalten, s. oben. In der Datei *httpd.conf* finden Sie die Einstellung der globalen Umgebung u.a. wie folgt:

```
### Section 1: Global Environment
ServerRoot "C:/web/Apache"
Listen 80
ServerName localhost:80
DocumentRoot "C:/web/Apache/htdocs"
```

Unter `ServerRoot` ist das Verzeichnis eingetragen, in dem der HTTP-Server installiert ist. Der Servername ist hier *localhost*.

Die Angabe `Listen` legt den ausgewählten Port zur Datenübermittlung fest.

Bei Eingabe von *http://localhost* oder *http://127.0.0.1* in der Adresszeile des Browsers wird die Datei

c:web/Apache/htdocs/index.html

angezeigt. Die Installation war erfolgreich. Unter `DocumentRoot` können Sie jedes beliebige Verzeichnis eintragen, z.B. das Verzeichnis, indem Sie Ihre PHP-Scripts vorhalten.

Installation von PHP

Im Anschluss an die Installation des Webservers erfolgt der PHP Download von der Website *www.php.net*. Den Link finden Sie auch auf der Apache-Homepage. Für die Beispiele dieses Buches wird die Version 4.3.8 benutzt, die zur Zeit der Programmentwicklung auch vom Web-Hoster unterstützt wurde. Unsere Programme sind zu kommenden Versionen von PHP aufwärtskompatibel. Wir wählen die binäre Version *PHP 4.3.8 zip package* und müssen die Konfiguration für den Apache-Server manuell durchführen. Nach Download steht die Datei *install.txt* mit umfangreichen Informationen zur Verfügung. Deutsche Sprachversionen hiervon finden Sie im Netz. Mit folgenden Schritten vervollständigen Sie die Installation:

1. Öffnen Sie die Datei *php.ini-dist* aus dem PHP-Installationsverzeichnis mit einem Texteditor. Suchen Sie `doc_root` und treffen Sie folgende Zuweisung entsprechend des von Ihnen gewähten Installationsverzeichnisses:
   doc_root ="c:\web\apache\htdocs"
2. Setzen Sie extension_dir
   extension_dir = "c:\web\php-4.3.8-Win32\extensions"
3. Speichern Sie die Datei unter *c:\WINNT\php.ini*
4. Kopieren Sie die Dateien *PHP.exe* und *php4ts.dll* in dasVerzeichnis *c:\WINNT* oder in das Verzeichnis, in dem sich *Apache.exe* befindet.
5. Dem Webserver muss noch die PHP-Umgebung mitgeteilt werden. Ergänzen Sie die Datei *Apache\conf\httpd.conf* am Ende mit den Zeilen:
   LoadModule php4_module c:/web/php-4.3.8-Win32/sapi/php4apache.dll
   AddType application/x-httpd-php .php

Zum Test der Installation wird ein minimales PHP-Script benötigt. Ein geeigneter Freeware-Editor für PHP-Sourcecode ist *Wea-*

verslave. Den „Weberknecht" finden Sie auf etlichen Download-Seiten in diversen Sprachversionen. Am besten gehen Sie über eine Suchmaschine. Nach Installation öffnen Sie Ihren Editor , geben Sie ein:

```
<?php
   phpinfo();
?>
```

und speichern Sie die Datei im Verzeichnis *Apache\htdocs\info.php*. Rufen Sie anschließend im Webbrowser `http://localhost/info.php` auf. Die Installationsbemühungen waren erfolgreich, wenn der Browser Ihnen die Abbildung 7.3 anzeigt.

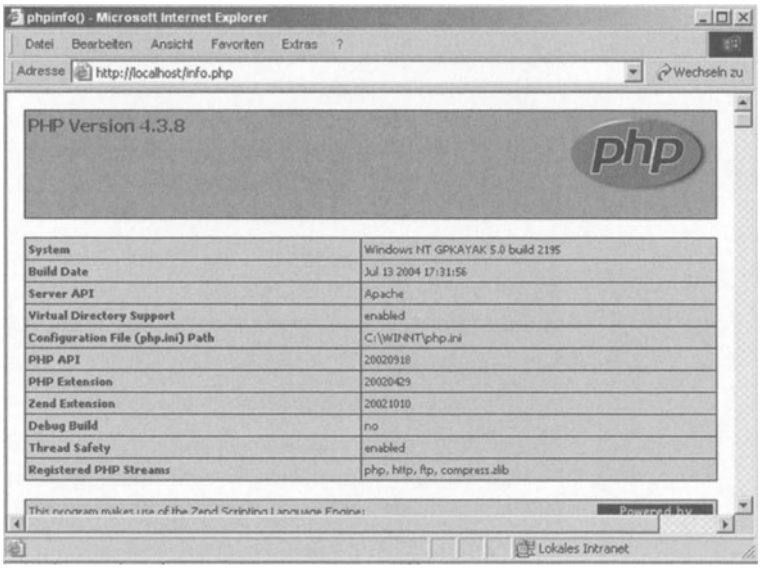


Abb 7.3: Installation von Apache 1.3.31 und PHP 4.3.8

PHP und HTML - HelloWorld

PHP Quellcode kann unmittelbar in HTML-Dateien eingefügt werden. Hierzu kann man unterschiedliche Notationen benutzen. Dem XML-Stil entsprechend wird hier wie folgt vorgegangen:

```
<html>
  <!--... html-Auszeichnungen -->
   <?php
      // PHP-Quelltext
```

```
  ?>
  <!--... html-Auszeichnungen -->
</html>
```

Kommentare werden in PHP mit // eingeleitet, längere Kommentarbereiche werden in /* *Kommentartext* */ eingeschlossen.

Anweisungen werden in PHP, wie auch schon bei Java und JavaScript erläutert, in der Reihenfolge ihres Auftretens abgearbeitet. Anweisungen sind Zuweisungen, bedingte Anweisungen, Kontrollstrukturen und Funktionsaufrufe. Anweisungsblöcke werden durch geschweifte Klammern {} begrenzt. Jede einzelne PHP-Anweisung wird durch ein Semikolon abgeschlossen.

PHP Programme können ausgelagert sein und mit den Anweisungen `include("datei");` oder `require ("datei");` in den Quelltext eingefügt werden. `Include` wird nur bei Bedarf eingebunden, während `require` immer ausgeführt wird.

Sonderzeichen werden in PHP durch einen vorangestellten Backslash markiert. Die Anweisung `echo();` ist ein PHP-Funktionsaufruf. Als Argument wird eine in Hochkommata eingeschlossene Zeichenkette erwartet. Sollen Hochkommata innerhalb der Zeichenkette ausgegeben werden, dann ist vor diese Sonderzeichen der Backslash zu setzen.

Bevor es zum ersten „HelloWorld"-Test kommt, sollten Sie sich für ein Arbeitsverzeichnis entscheiden und in der Datei *httpd.conf* darauf verweisen. Auf meinem System ist das

```
DocumentRoot
  "f:/prj_webdynamisch/programmierpraktikum/d/php"
```

Vor Änderung müssen Sie den Server anhalten, danach wieder starten, vgl. weiter oben *net stop/start apache.*

Das erste komplette PHP-HelloWorld-Programm, eingebettet in HTML, hat dann dann folgendes Aussehen:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
  <head><title>HelloWorld mit PHP</title>
</head>
<body>
  <?php
    echo (" ... endlich das erwartete
          \"HelloWorld-Programm\" mit PHP");
  ?>
```

```
</body>
</html>
```

Gespeichert unter *helloworld.php* und gestartet im Browser mit *http://localhost/helloworld.php* wird der an die Funktion echo() übergebene Textstring im Browserfenster angezeigt. Wenn Sie sich den Quelltext im Browser ansehen, wird erkennbar, das die PHP-Anweisungen nicht mehr vorhanden sind. Der Server gibt nur das vom Hypertext Pre-Processor erzeugte HTML zurück. Innerhalb des Arguments können Variable auftreten oder mehrere Zeichenketten durch einen Punktoperator verknüpft werden. Wir ergänzen den PHP-Teil um:

```
$n = 2;
echo ("<br />Das ist der $n. Versuch<h1>Hello World</h1>");
```

n ist eine Variable, der in PHP immer ein $-Zeichen vorangestellt wird. Der Inhalt von n wird mit 2 vorbesetzt. Das Argument des Funktionsaufrufs enthält mit <br /> jetzt einen HTML-Tag (Zeilenvorschub), der Wert der Variablen wird angezeigt, und Hello World ist mit <h1></h1> als Überschrift ausgezeichnet. Mit dem Inkrementoperator erhöhen wir den Inhalt von n und geben den Text in einer weiteren Programmzeile mt JavaScript in einer Alert-Box aus:

```
$n++;
echo ("<br />Der $n. Versuch jetzt mit
        <script language=\"JavaScript\">
                alert(\"HelloWorld\");
        </script> ");
```

Damit wären die ersten Schritte erfolgreich absolviert. Installation einer Client-Server-Umgebung mit PHP und Einbettung von PHP-Code in eine HTML-Seite mit Ausführung von JavaScript-Funktionen.

Zusammengefasst finden Sie das obige Listing im File *test_1.php*

7.2 Grundlagen von PHP

PHP ist eine mächtige, objektorientierte Programmiersprache für Web-Anwendungen mit Funktionalitäten zum Zugriff auf MySQL-Datenbanken und umfangreichen XML-Support. Die neueste Version ist derzeit das Release 5.0, auf die noch nicht alle Provider umgestellt haben. Im Rahmen unseres Grundkurses nutzen wir hauptsächlich die Möglichkeiten des Zugriffs auf das Filesystem des Servers. Alle Programmbeispiele sind mit PHP 4.38 lokal und online getestet und sind aufwärtskompatibel zu PHP 5

Nachfolgend werden Grundlagen der Sprache erarbeitet. Objektorientierte Konzepte von PHP und XML-Support werden im Rahmen dieser Einführung nicht behandelt. Wenn Sie mit MySQL und PHP Datenbankanwendungen bearbeiten wollen, dann ist der Grundkurs von Pollakowski, erschienen im Vieweg Verlag, zu empfehlen.

Auch in diesem Kapitel gilt, wie beim Erlernen jeder Programmiersprache, Programmieren lernt man nicht durch Lesen, sondern nur durch Programmieren. Erst verschafft man sich einen Überblick, dann vertieft man Details. Lassen Sie sich von Fehlermeldungen nicht entmutigen, versuchen Sie die Mitteilungen zu verstehen, korrigieren Sie den Sourcecode, speichern Sie die Datei und aktualisieren Sie die Browseranzeige. Verzweifeln Sie nicht – vor den Erfolg haben die Götter den Compiler bzw. Interpreter gesetzt.

Variablen, Konstanten, Operatoren, Ausdrücke

PHP ist eine so genannte typenlose Sprache. Der Typ einer Variablen wird nicht explizit deklariert, sondern ergibt sich aus dem Inhalt des zugewiesenen Wertes. Das erste Zeichen des Bezeichners einer Variablen muss ein $-Zeichen sein, danach folgt der Name. Die weiteren Zeichen des Namens sollen, vom Unterstrich ausgenommen, keine Sonderzeichen sein. Groß- und Kleinschreibung werden bei PHP unterschieden.

PHP kennt globale, statische und dynamische Variablen und Konstante. Variablen sind in dem Strukturblock sichtbar, in dem sie deklariert wurden. Mit den Angaben `global` oder `static` kann man die Sichtbarkeit auch in anderen Strukturen herstellen. Die praktische Programmierung kann aber darauf verzichten. Eine dynamische Variable entsteht durch doppelte $$-Zeichen. Der eigentliche Variablenname ensteht aus dem Inhalt einer zweiten Variablen.

```
<?php
  $variable="bezeichner";
  $$variable = "Neuer Inhalt";
  echo($bezeichner);?>
```

Der Variablen `$variable` wird der Wert *bezeichner* zugewiesen. Es wird eine dynamische Variable mit `$$variable` und Zuweisung der Zeichenkette *Neuer Inhalt* deklariert. Hierdurch wurde eine neue Variable mit dem Namen `$bezeichner` erzeugt. Die Ausgabe ist *Neuer Inhalt*.

Auf Variablen können Funktionen angewandt werden, mit denen das Bestehen der Variablen selbst, `isset()`, oder das Vorhan-

densein eines Wertes, `empty()`, geprüft werden kann. Nicht mehr benötigte Variable werden mittels `unset()` gelöscht. Das Argument ist immer die betreffende Variable.

Konstante sind innerhalb eines Programms unveränderlich. Die Zuweisung von Konstanten erfolgt durch Funktionsaufruf von `define()`, als Argumente werden der Name der Konstanten und ihr Wert übergeben:

```
define("Pi","3.1415"); echo(Pi);
```

Die Programmierung in PHP bringt aufgrund der mangelnden Typenstrenge gewisse Fehlerquellen mit sich. Benötigt man einen bestimmten Typ für eine Variable, dann muss man diese ggf. explizit umwandeln. Die Zahl Pi berechnet sich auch aus dem Ausdruck `2*asin(1)`. Wir wollen die Abweichung der Nachkommastellen von der oben erfolgten direkten Zuweisung feststellen.

```
<?php
define("lf","<br />"); // definiert lf als line feed
define("Pi","3.1415"); // definiert Pi als Konstante
echo(gettype(Pi) .lf); // bestimmt den Datentyp von Pi wieder
$a = 1; $b=2;
echo("Typ a = " .gettype($a) .lf);// Datentypen von a und b?
echo("Typ b = " .gettype($b) .lf);// wurden als integer erkannt
$a = 1.0; $b=2.0;
echo("Typ a = " .gettype($a) .lf);// Datentypen von a und b?
echo("Typ b = " .gettype($b) .lf);// wurden als double erkannt
$pi   = $b*asin($a);
$zahl =Pi;
settype($zahl,"double");
echo("Type von $zahl = " .gettype($zahl) .lf);
                    // bestimmt den Datentyp von Pi erneut
$diff = $pi - $zahl;
echo("differenz Typ = "  .gettype($diff) ." Wert = " .$diff);
?>
```

Listing 7.1: Datei *test_2.php* ; Bestimmung von Datentypen

Die Typbestimmung von `Pi` zeigt als Ergebnis `Pi` als Variable vom Typ *String* an. Die Zuweisung von `a` und `b` mit `1` bzw. `2` wird als Ganzzahl erkannt. Durch Zuweisung von `1.0` bzw. `2.0` wird der Datentyp zu `double`. Mit der Funktion `settype()` kann die explizite Umwandlung eines Datentyps erfolgen. Durch den letzten `echo()` Aufruf erfahren wir, dass unsere Festlegung von

Pi um 0.00009 ungenau ist. Das Argument von echo() wird durch Einsatz des Punktoperators aus mehreren Termen verknüpft.

PHP liefert bei der Anwendung unterschiedlicher Datentypen (mixed mode) häufig die gewünschten Werte, 1/2 wird 0.5, oder wollten Sie lieber 0? Bei der Auswertung mathematischer Ausdrücke ist etwas Vorsicht geboten.

Der Gebrauch von Operatoren und deren Rangfolge unterscheidet sich bei PHP kaum von anderen Programmiersprachen. Wir kennen:

- arithmetische Operatoren,
- Zuweisungsoperatoren,
- Vergleichsoperatoren,
- Logische Operatoren,
- Bit-Operatoren
- Cast-Operatoren.

Im weiteren Verlauf der Beispiele und Anwendungen werden wir den diversen Operatoren begegnen.

Ausdrücke setzen sich zusammen aus Variablen, Funktionsaufrufen und Operatoren, die deren Verknüpfung regeln. Die Priorität der Auswertung kann durch runde Klammern beeinflusst werden.

Hier einige Beispiel für Ausdrücke:

```
<?php
//
define("lf","<br />") // definiert lf als HTML-Tag neue Zeile
//
$pi = 2.0*asin(1.0);  // Ausdruck rechts wir ausgewertet,
                      //Inhalt wird $ pi zugewiesen
echo ("Pi = " .$pi .lf);
//
$a = sqrt((5 % 2));   // Modulo Operator liefert Rest einer
                      // Zahl nach Abzug der ganzen Vielfachen
                      // der zweiten Zahl
echo ("a = " .$a .lf);
//
$a = 3.0;                    // Zuweisung
$b = 4.0;                    // Zuweisung
$c = sqrt(($a*$a)+($b*$b)); // Auswertung der Klammern
                             //von innen nach außen a*a, b*b
```

```
                    // Quadratsumme wird Argument fuer sqrt
echo ("Satz des Pythagoras: " ."a = " .$a ., b = " .$b ., c
= " .$c);
?>
```

Listing 7.2: Datei test_3.php – Auswertung von Ausdrücken

Die Bildschirmausgabe des Scripts lautet:

```
Pi = 3.14159265359
a  = 1
Satz des Pythagoras: a = 3, b = 4, c = 5
```

Kontrollstrukturen Mit Kontrollstrukturen werden Anweisungen bezeichnet, die eine sequentielle Abarbeitung der Programmanweisungen unterbrechen. Darunter fallen bedingte Anweisungen, Schleifen, Fallunterscheidungen und auch Funktionsaufrufe.

Bedingte Anweisungen Bedingte Anweisung werden mit dem Schlüsselwort `if` eingeleitet. Der darauf folgende Ausdruck in runden Klammern, die Bedingung, wird logisch ausgewertet. Ist der Ausdruck wahr (true), erfolgt die Ausführung der unmittelbar auf `if` folgenden Programmanweisung oder des Anweisungsblocks. Ergibt die Auswertung des Ausdrucks den Wert falsch (false), wird die Anweisung bzw. der Anweisungsblock nicht ausgeführt, das Programm wird mit den Folgeanweisungen fortgesetzt. Hierzu ein Beispiel: Die PHP-Funktion `date()` gibt das Datum entsprechend des als Argument übergebenen Formats zurück. Den Variablen `$Tag`, `$Monat`, `$Jahr` wird der Rückgabewert der Funktion zugewiesen. Der Sonntag hat die Bezeichnung Sun. Wir fragen daher den Tag auf Gleichheit == oder Ungleichheit != mit Sun ab.

Ist der Tag Sonntag, dann soll die Ausgabe in rot erfolgen, bei anderen Tagen in blau. Es wird in jedem Fall nur eine der bedingten Anweisungen ausgeführt. Die Steuerung der Ausgabefarbe erfolgt über eine einfache Stilanweisung.

```
<?php
$Tag   = date("D");
$Monat = date("M");
$Jahr  = date("Y");
if ($Tag == "Sun") echo  ("<p style=\"{color:red;\">
                           $Tag $Monat $Jahr</p>");
if ($Tag != "Sun") echo ("<p style=\"{color:blue;\">
                           $Tag $Monat $Jahr</p>");
?>
```

Listing 7.3: Bedingte Anweisungen, *date_1.php*

Fallunterscheidungen

Man kann diese zweifache Abfrage auf eine Abfrage reduzieren, wenn man ein If–else-Konstrukt einbringt, Hierbei wird von beiden Anweisungsblöcken immer nur ein Block ausgeführt: entweder die Anweisungen hinter `if` oder hinter `else`. Die geschweiften Klammern begrenzen die Anweisungsblöcke. Da es sich im folgenden Beispiel um nur eine Anweisung handelt, könnte auf die {} Klammern hier verzichtet werden, diese erhöhen aber die Lesbarkeit.

```
<?php
  $Tag   = date("D");
  $Monat = date("M");
  $Jahr  = date("Y");
  if ($Tag == "Sun") {
    echo  ("<p style=\"{color:red;\">$Tag
   $Monat $Jahr</p>");
}
else {
    echo ("<p style=\"{color:blue;\">$Tag
    $Monat $Jahr</p>");
}
?>
```

Listing 7.4: Bedingte Anweisung mit Alternative, *date_2.php*

Das If-else-Konstrukt kann durch if-elseif an weitere Bedingungen gebunden werden. Mehrfachunterscheidungen werden aber besser mit der Switch-Anweisung ausgeführt.

Die Datumsfunktion gibt bei Übergabe des Formats `D` die englische Kurzbezeichnung für die Wochentage zurück. Der Ausdruck soll mit deutscher Bezeichnung erfolgen. Mit der Switch-Anweisung wird die Variable `$Tag` abgefragt. Mit `case` werden die möglichen Fälle unterschieden. Für Fälle, die nicht abgefragt sind, ist der Default-Bereich vorzusehen. Das Break-Kommando beendet die Switch-Struktur.

```
<?php
$Tag   = date("D");
$Monat = date("M");
$Jahr  = date("Y");
switch($Tag){
  case "Mon" : $Tag="Montag";
    break;
  case "Tue" : $Tag="Dienstag";
    break;
  case "Wed" : $Tag="Mittwoch";
```

```
    break;
  case "Thu" : $Tag="Donnerstag";
    break;
  case "Fri" : $Tag="Freitag";
    break;
  case "Sat" : $Tag="Samstag";
    break;
  case "Sun" : $Tag="Sonntag";
    break;
  default:
}
echo ("<p>$Tag $Monat $Jahr</p>");
?>
```

Listing 7.5: Bedingte Anweisung innerhalb einer Switch-Struktur, *date_3.php*

Schleifen

Anweisungsblöcke, die wiederholt auszuführen sind, werden in Schleifen eingebettet. Es wird unterschieden zwischen Schleifen, deren Abbruchbedingung sich während des Schleifendurchlaufs ändert, und Zählschleifen, bei denen die Anzahl der Durchläufe bereits vor Schleifenbeginn bekannt ist. Erstere sind die While- oder Do-while-Schleifen für Iterationen oder Eingaben mit Abbruchbedingung. Zählschleifen werden als For-Schleifen codiert.

Die Zählschleife wird mit dem Schlüsselwort `for` eingeleitet. Innerhalb der nachfolgenden runden Klammern wird ein Schleifenzähler initialisiert, die Abbruchbedingung formuliert und die Änderung des Schleifenzählers bestimmt. Der Schleifenkörper folgt eingebettet in geschweiften Klammern.

In Telefonrechnungen werden die Einzelverbindungen nicht mit voller Telefonnummer aufgelistet. Wir betrachten ein kleines Programm, das die letzten fünf Stellen einer Telefonnummer durch einen * markiert. Die Telefonnummer ist im String `$text` gespeichert. Das Zeichen * ist in der Variablen `$asterix` abgelegt. Die Länge des Textes kann man über die Funktion `strlen()` ermitteln. Es wird jetzt eine Zählschleife formuliert. Der Schleifenindex wird mit `$i` bezeichnet, der Anfangswert mit `$laenge -5` initialisiert. Die Schleife wird wiederholt, solange der Schleifenindex kleiner Stringlänge ist. Nach jedem Durchlauf wird der Schleifenindex um 1 erhöht. Im Schleifenkörper benutzen wir die Indizierung mit `[]` Klammern. Die Position `$i` im String `$text` wird durch das Zeichen `$asterix` ersetzt. Mit der

eingebundenen Ausgabezeile echo() können Sie den Ablauf verifizieren. Die Ausgabe lautet:

```
Text vor Schleifendurchlauf: 030-12 34 56 78
Schleifenindex i = 10 Text = 030-12 34 *6 78
Schleifenindex i = 11 Text = 030-12 34 ** 78
Schleifenindex i = 12 Text = 030-12 34 ***78
Schleifenindex i = 13 Text = 030-12 34 ****8
Schleifenindex i = 14 Text = 030-12 34 *****
Text nach Abarbeitung der Schleife: 030-12 34 *****
```

```
<?PHP
  $text = "030-12 34 56 78";
  $asterix="*";
  $laenge = strlen($text);
  echo("$text <br/>");

  for ($i=$laenge-5; $i<$laenge; $i++){
   $text[$i]=$asterix;
   echo("Schleifenindex i = $i Text
         $text<br/>");
  }
  echo("$text <br/>");
?>
```

Listing 7.6: Zählschleife, *for_loop.php*

Zählschleifen dieser Art werden besonders im Zusammenhang mit der Adressierung von Arrayelementen benutzt. Im weiteren Verlauf der Diskussion von Anwendungsprogrammen werden diese For-Schleifen bevorzugt eingesetzen.

Eine While-Schleife wird durch das Schlüsselwort `while` eingeleitet, es folgt die Bedingung zum Eintritt in den Schleifenkörper, der in geschweifte Klammern eingeschlossen ist. Nach Abarbeitung der Anweisungen wird die Programmfolge wieder bei while fortgesetzt. Was jeder Programmierer gerne vermeidet, ist eine Endlosschleife, hier ist sie gewollt:

```
<?php
  $n=0;
  while (TRUE){
    $n++;
    echo("Endlosschleife Durchlauf: $n <br/>");
  }
?>
```

TRUE ist eine PHP-Konstante, der Boolsche Wert ist wahr, die Bedingung ändert sich nicht. Der eingebaute Zähler läuft bis zum

Speicherüberlauf. Im Gegensatz zur While-Schleife wird bei der Do-while-Schleife die Abbruchbedingung am Ende abgefragt, d.h. die Schleife wird mindestens einmal durchlaufen. In ein Programm soll eine Wartezeit von 3 Sekunden eingebaut werden. Bei Aufruf der Funktion `gettimeofday()` wird ein assoziatives Array mit folgenden Elementen zurückgegeben:

```
sec          - Sekunden
usec         - Mikrosekunden
minuteswest  - Minuten westlich von Greenwich Mean Time
dsttime      - Korrektur durch Sommerzeit
```

Weiter unten werden die assoziativen Arrays im Detail erläutert. Zunächst genügt es zu wissen, dass die Sekunden der Startzeit in der Variablen `$SekundeStart` gespeichert sind. Die aktuelle Zeit liegt vor in `$SekundeAktuell`. Wir konstruieren eine While-Bedingung, die solange erfüllt ist, bis die Differenz dieser Variablen den in `$countdown` definierten Zeitunterschied erreicht. Innerhalb der Schleife wird ausschließlich die aktuelle Zeit abgefragt.

```
<?php
  $time = gettimeofday();
  $SekundeStart = $time[sec];
  $countdown=3;
  echo ("Startzeit: $SekundeStart<br/>");
  echo ("Bitte warten Sie $countdown
         Sekunden<br/>");
  do {
    $time = gettimeofday();
    $SekundeAktuell = $time[sec];
  } while (($SekundeAktuell-$SekundeStart)<$countdown);
  echo ("Endzeit: $SekundeAktuell<br/>");
  echo ("<br/>Danke fuer Ihre Geduld");
?>
```

Listing 7.7: Do-while-Schleife , *countdown.php*

Arrays

Arrays oder Felder sind Speicherbereiche, in denen unter der gleichen Variablenbezeichnung eine Vielzahl von Werten adressiert werden kann. Die Unterscheidung erfolgt über den Index. Im mathematischen Kontext ist ein eindimensionales Array ein Vektor, ein zweidimensionales Array stellt eine Matrix dar. Die Adressierung der Elemente erfolgt durch Angabe von Zeilen- und Spaltenindex. Im Kapitel 4.7 Einführung in die 3D-Grafik mit Ja-

va wurden Arrays ausgiebig benutzt. PHP bietet einige Besonderheiten. Es wird unterschieden zwischen indizierten Arrays und assoziativen Arrays. Für Array-Operationen stehen umfangreiche Funktionen zur Verfügung.

Indizierte Arrays benutzen als Schlüssel eine Ganzzahl. Alphanumerische Schlüsselnamen werden in assoziativen Arrays verwendet, vgl. auch `gettimeofday()` in Listing 7.7. Die Namen der Spieler einer Fußballmannschaft sollen in einem Array gespeichert und in sortierter Folge ausgegeben werden.

```
<?PHP
$Spieler = array ("Kuhn","Wurms","Hinken","Muller","Bein",
 "Holzmann","Babic","Tormeier","Ballermann",
 "Aussenlaeufer","Locke");
sort($Spieler);
$anzahl = count($Spieler);
for ($i = 0; $i<$anzahl; $i++){
  echo("Nr.: $i $Spieler[$i]<br/>");
}
?>
```

Listing 7.8: Eindimensionales indiziertes Array, *array_1.php*

Das Array wird mit `$Spieler` bezeichnet. Die Funktion `array()` vereinfacht die Eingabe der Daten. Mit `sort()` wird das Array sortiert, `count()` liefert als Rückgabe die Anzahl der Arrayelemente.

In einer For-Schleife werden die Spielernamen über den Arrayindex adressiert und wie folgt ausgegeben:

```
Nr.: 0 Aussenlaeufer
Nr.: 1 Babic
Nr.: 2 Ballermann
Nr.: 3 Bein
Nr.: 4 Hinken
Nr.: 5 Holzmann
Nr.: 6 Kuhn
Nr.: 7 Locke
Nr.: 8 Muller
Nr.: 9 Tormeier
Nr.: 10 Wurms
```

Es wird jetzt zusätzlich noch ein assoziatives Array mit den Rückennummern eingeführt:

```
$Rueckennummer = array(
"Kuhn" => "1","Wurms" => "2",
```

```
"Hinken"=> "5","Muller"=> "3","Bein"=> "6",
"Holzmann"=> "23","Babic"=> "9",
"Tormeier"=> "11","Ballermann" => "8",
"Aussenlaeufer" => "7","Locke" => "10");
```

Dem Arrayschlüssel (key) wird mittels des Operators => das Element zugewiesen. Die Schleife wird nun noch ergänzt durch die Ausgabe der Trikotnummern

```
for ($i = 0; $i<$anzahl; $i++){
  $index=$Spieler[$i];
  echo("$Spieler[$i] Trikot:
        $Rueckennummer[$index]  <br/>");
}
```

Der Index für die Rückennummer ist jetzt keine ganze Zahl mehr, sondern der Schlüssel, der sich aus dem Spielernamen ergibt. Die Ausgabe hat dann folgendes Bild:

```
Aussenlaeufer Trikot: 7
Babic Trikot: 9
Ballermann Trikot: 8
Bein Trikot: 6
Hinken Trikot: 5
Holzmann Trikot: 23
Kuhn Trikot: 1
Locke Trikot: 10
Muller Trikot: 3
Tormeier Trikot: 11
Wurms Trikot: 2
```

Wenn wir jetzt die Spieler sortiert nach Rückennummern ausgeben wollen, benutzen wir die Funktion `asort()` zum Sortieren des assoziativen Arrays. Mit der speziell auf Arrays anzuwendenden Schleife `foreach` wird jedes Feldelement abgearbeitet, Schlüssel und Element werden im Schleifenkörper als `$key` und `$element` verfügbar.

```
asort($Rueckennummer);
foreach ($Rueckennummer as $key => $element){
   echo ("$element $key<br/>");
}
```

Die Ausgabe des zweiten Teils nach Trikotnummern sortiert:

```
1 Kuhn
2 Wurms
3 Muller
5 Hinken
```

```
6 Bein
7 Aussenlaeufer
8 Ballermann
9 Babic
10 Locke
11 Tormeier
23 Holzmann
```

Hier das Listing noch einmal im Zusammenhang:

```
<?PHP
$Spieler = array ("Kuhn","Wurms","Hinken","Muller","Bein",
"Holzmann","Babic","Tormeier","Ballermann",
"Aussenlaeufer","Locke");
$Rueckennummer = array
("Kuhn" => "1","Wurms" => "2","Hinken"=> "5",
"Muller"=> "3","Bein"=> "6","Holzmann"=> "23",
"Babic"=> "9","Tormeier"=> "11",
"Ballermann" => "8","Aussenlaeufer" => "7",
"Locke" => "10");
$anzahl = count($Spieler);
//
// Ausgabe nach Spielernamen sortiert
sort($Spieler);
for ($i = 0; $i<$anzahl; $i++){
  $index=$Spieler[$i];
  echo("$Spieler[$i] Trikot:
        $Rueckennummer[$index]  <br/>");
}
// Ausgabe nach Rueckennummern sortiert
asort($Rueckennummer);
foreach ($Rueckennummer as $key => $element){
   echo ("$element $key<br/>");
} ?>
```

Listing 7.9: Indizierte und assoziative Arrays, *array_2.php*

Für den Einstieg mag das an dieser Stelle ein wenig kompliziert sein, wenn Sie sich zunächst mit dem Listing 7.8 vertraut machen, ist das hinreichend. Erfahrene C-Programmierer werden zugeben, dass PHP mit einigen Neuheiten und sehr vielseitigen Arrayfunktionen aufwartet.

Funktionen

Wiederkehrende Aufgaben werden in Funktionen oder Methoden gekapselt. Funktionen bestehen aus ihrem Namen, einer Parameterliste und dem Funktionskörper. Der Aufruf einer Funktion erfolgt durch Angabe des Funktionsnamens und den Argumenten, die in Reihenfolge und Typ mit der Parameterliste über-

einstimmen müssen. Gekennzeichnet wird eine Funktion durch das Schlüsselwort function. Das Ergebnis einer Funktion wird über den Rückgabewert dem aufrufenden Programm mitgeteilt. Ein einfaches Beispiel ist die Berechnung des Volumens eines Quaders:

```
function volumenQuader($laenge, $breite,
                       $hoehe){
 $volumen = $laenge * $breite * $hoehe;
 return $volumen;
}
```

Der Funktionsname ist volumenQuader, die Bezeichner in der Parameterliste und im Funktionskörper sind nur innerhalb der Funktion sichtbar. Die aktuellen Werte werden durch die Argumente beim Aufruf bestimmt. Die Variable $rauminhalt bekommt im rufenden Programm den Rückgabewert der Funktion zugewiesen. Neben der Kapselung von Code besteht der Vorteil von Funktionen in der Mehrfachverwendung, Übersichtlichkeit und Fehlerabsicherung.

```
<?PHP
  $a = 3.0; $b = 4.0; $c = 5.0;
  $raumInhalt = volumenQuader($a,$b,$c);
  echo ("Volumen : $raumInhalt");

function volumenQuader($laenge, $breite, $hoehe){
 $volumen = $laenge * $breite * $hoehe;
 return $volumen;
}
?>
```

Listing 7.10: Anwendung von Funktionen call-by-value

Der Datenaustausch zwischen Funktionen kann nicht nur über die Werte (call-by-value), sondern auch über die Referenz erfolgen. Sofern in der Parameterliste der Adressoperator & dem Bezeichner vorangestellt ist, wirken sich Änderungen an der Variablen auch im rufenden Programm aus. Die modifizierte Version unserer Volumenberechnung tauscht das Ergebnis jetzt über die Parameterliste aus (call-by-reference)

```
<?PHP
$a = 3.0;
$b = 4.0;
$c = 5.0;
$raumInhalt = 0.0;
volumenQuader($a,$b,$c,$raumInhalt);
```

```
echo ("RaumInhalt : $raumInhalt");

function volumenQuader($laenge, $breite, $hoehe, &$volumen){
 $volumen = $laenge * $breite * $hoehe;
 return;
}?>
```

Listing 7.11: Anwendung von Funktionen call-by-reference

Abschließend ist noch das Listing 7.12 zu betrachten. Die Variablen sind dort in einem Array $quader gespeichert, in der Funktion lokal mit $q bezeichnet, aber mit dem Adressoperator in der Parameterliste ausgezeichnet. Eine Änderung der Werte würde sich auch im rufenden Programm auswirken.

```
<?PHP
$quader = array(3.0,4.0,5.0,0.0);
volumenQuader($quader);
echo ("Volumen : $quader[3]");

function volumenQuader(&$q){
 $q[3] = $q[0] * $q[1] * $q[2];
 return;
}
?>
```

Listing 7.12: Funktionen call-by-reference mit Arrays

7.3 PHP-Anwendungen

Die erworbenen Kenntnisse über PHP werden nunmehr in praktische Anwendungen umgesetzt. Alle hier vorgestellten Lösungen müssen sich Beschränkungen u.a. hinsichtlich Ausschöpfung des Sprachumfangs und der Datenüberprüfung auferlegen, andernfalls würde die Übersichtlichkeit verloren gehen. Auch wird weitgehend auf die Diskussion von Programmalternativen und Optimierung verzichtet. Unsere Programme sollen robust sein und schrittweise entwickelt werden. Alle Applikationen finden Sie mit Testdaten lauffähig auf der Website zum Buch. Der komplette Quellcode steht den Lesern als Download zur Verfügung.

Online Dateiverwaltung

Als erfolgreicher Veranstalter eines Fortbildungsseminars über Web-Programmierung, haben Sie auch die Organisation zu übernehmen. Für die praktische Arbeit am Rechner steht Ihnen ein Raum mit 16 Rechnerarbeitsplätzen an zwei Terminen in der Woche zur Verfügung. Die maximal 32 Teilnehmer müssen sich

einen Rechnerarbeitsplatz reservieren – das soll natürlich online erfolgen.

Zunächst zur Datenstruktur: Wir wollen eine Datei erstellen, in der für jeden verfügbaren Arbeitsplatz der Name, Vorname und die Email-Adresse des Seminarteilnehmers sowie das Datum der Anmeldung eingetragen ist. Diese Datei wird zeilenweise organisiert und für jedes Seminar getrennt angelegt. In der ersten Spalte jeder Zeile befindet sich ein Zeichen, welches die Belegung kennzeichnet. Der Arbeitsplatz ist noch frei wird durch ein - angezeigt, eine bereits erfolgte Belegung wird durch * gekennzeichnet. Als Dateinamen wählen wir *seminar_nr.txt*, nr nimmt die Werte von 1 bis 2 an. Zunächst erfolgt in jeder Datei die Eintragung der Vorbesetzung, die 16 mal wiederholt wird:

```
-, niemand, keiner, nn@gmx.de,  datum
```

Am Ende der Zeile befindet sich ein Zeilenumbruch. Man bezeichnet eine Datei dieser Form mit CSV-Datei (*comma separated values*). PHP stellt mit der Funktion `fgetscsv()` eine komfortable Möglichkeit der Eingabe bereit. Aus diesem Grund fällt die Entscheidung zugunsten dieser Formatfestlegung hier leicht. Zur Aufbereitung der Datei mit den Vorbesetzungen wäre ein Texteditor hinreichend. Wir wollen die Datei aber per Programm erzeugen, da die Komponenten weiter unten ohnehin benötigt werden.

Starten Sie den PHP-Editor Weaverslave oder einen anderen Editor Ihrer Wahl und beginnen Sie mit folgender Codierung:

```
<html>
  <head>
  <title>PHP-Programme:FileGenarator</title>
  </head>
<body>

<?php

//  an dieser Stelle wird der PHP Programmtext eingefügt

?>
</body>
</html>
```

Speichern Sie die Datei in Ihrer Entwicklungsumgebung unter *file_generator.php*. Den Editor können wir während der gesamten Arbeit geöffnet lassen, ggf. durch Minimierung in die Taskleiste verweisen. Vorteilhaft wird sich später die gleichzeitige Bearbeitung mehrer Dateien im Editor auswirken.

Mit der Funktion `echo()` erzeugt PHP Ausgaben an den Browser. Hier zunächst eine Meldung zur Programmfunktionalität, gefolgt von der Deklaration der notwendigen Variablen.

Variablenbezeichnung / Symbole

Symbolische Namen für Variable beginnen mit dem $-Zeichen. Da PHP auf eine explizite Typendeklaration für Variable verzichtet, notieren wir nach dem $-Zeichen einen kleinen Buchstaben, gefolgt von der eigentlichen Variablenbezeichnung. Ein einzelnes Zeichen (*character*) wird durch c gekennzeichnet, eine Zeichenkette (*string*) durch s usw. Wortkombinationen werden mit beginnenden Großbuchstaben notiert.

```
echo("PHP-Programm: file_generator.php<br/ >");
echo("Generiert CSV-Dateien mit Voreinstellungen <br>");

$iDateien    = 2;   // Anzahl der Dateien
$iRechner    =16;   // Anzahl der Zeilen / Datei

$sName       ="niemand";    // Voreinstellungen
$sVorname    ="keiner";
$sMail       ="nn@gmx.net";
$sDatum      = date("Y-m-d");

$cMinus      = "-";   // erste Spalte AP frei
$cKomma   = ",";       // Komma
$cBlank   = " ";       // Leerzeichen
$sEOL     = "\n";      // Zeilenumbruch
```

Mit den folgenden Anweisungen wird für jeden Seminartermin eine Datei mit den Voreinstellungen erzeugt. Der Schleifenindex dient auch zur Generierung des Filenamens. Jede Datei benötigt im Programm einen sogenannten *Dateihandler*. Ohne diesen ist kein Schreib- oder Lesezugriff möglich. Das Ergebnis der Funktion `fopen()` mit dem Argument `w` zum Schreiben einer Datei initialisiert den Dateihandler `$hDatei`.

File I/O Dateihandler

```
for ($i=0; $i<$iDateien; $i++){
 $iFileNumber =$i+1;
 // Dateinamen generieren
 $sDateiname="seminar_".$iFileNumber .".txt";
 //
 // Datei mit Schreibzugriff oeffnen
 $hDatei = fopen ($sDateiname, "w");
 //
 // Bildschirmprotokoll
 echo("Neue Datei: " .$sDateiname ."<br>");
```

Innerhalb der For-Schleife, die hier zweimal durchlaufen wird, erfolgt die Zusammensetzung des Dateinamens, das Öffnen der Datei, die Ausgabe der Vorbelegungen und das Schließen der Datei. Mit der Funktion `echo()` wird der Programmablauf protokolliert. Zur Verknüpfung der Zeichenketten wird der Punktoperator ausgiebig eingesetzt. Das Markieren des Zeilenendes ist notwendig. Mit `fputs()` wird eine Zeile in die Datei ausgegeben.

Der Dateihandler wird mit `fopen()` der Variablen `$hDatei` zugewiesen und muss bei jeder Ein-/Ausgabeoperation angegeben werden. Es können auch mehrere Dateien gleichzeitig geöffnet sein, die Zuordnung der I/O-Operationen erfolgt über die unterschiedlichen Bezeichner der Dateihandler. Im nächsten Schritt wird eine weitere For-Schleife notiert, in der für jeden Arbeitsplatz eine Datenzeile in die Datei geschrieben wird. Der Schleifenindex ist `$k`, die Schleife wird 16 mal durchlaufen.

```
for ($k=0; $k< $iRechner; $k++){
  //
  // Datenzeile schreiben
  $sRecord = $cMinus .$cKomma .$sName .$cKomma
  .$sVorname .$cKomma .$sMail .$cKomma
  .$sDatum .$sEOL;
  // Bildschirmprotokoll
  echo($k ." " .$sRecord ."<br>");
  // $Record in die Datei schreiben
  fputs ($hDatei,$sRecord);
} // Ende der $k Schleife
```

Nach Ablauf der inneren Schleife muss die Datei mit `fclose()` geschlosssen werden, erst dann kann mit der äußeren Schleife fortgefahren werden.

```
  fclose($hDatei);        // Datei schliessen
  echo("Ende Datei<br>"); //Bildschirmprotokoll
 } // aeussere Schleife fortsetzen
// Datum und Uhrzeit anzeigen
echo ("Zeit:  " .date("H.i.s") ."<br>");
echo ("Datum: " .date("d.m.Y") ."<br>");
echo ("Ende Programm");
```

Bei der Gelegenheit haben wir auch gleich alternative Datumsformate angegeben. Je nach Formatangabe erzeugt die Funktion `date()` die Ausgabe von Uhrzeit und Datum.

Bei laufendem Webserver wird das Programm im Internetbrowser durch Eingabe von *localhost/file_generator.php* gestartet. Schauen Sie sich nach erfolgreichem Durchlauf mit dem Explorer das Inhaltsverzeichnis an, dort finden Sie die Einträge der Dateinamen.

Sie können die Dateien auch im Editor öffnen und die korrekte Eintragung überprüfen. Es sollte 16 Datenzeilen gleichen Inhalts geben:

```
-,niemand,keiner,nn@gmx.net,2004-08-29
 ...
-,niemand,keiner,nn@gmx.net,2004-08-29
```

Nach dieser erfolgreichen Vorbereitung geht es jetzt in die Anwendung. Ein Teilnehmer möchte sich für ein Seminar anmelden. Dazu muss er das Seminar auswählen und die persönlichen Daten in Seminarfelder eingeben. Das könnte zunächst ausschließlich mit HTML erfolgen. Wir möchten aber auch anzeigen, in welchem Seminar noch freie Arbeitsplätze vorhanden sind, bzw. wie viele Teilnehmer sich schon angemeldet haben. Daher findet die Kombination von HTML und PHP Anwendung.

Den HTML-Teil können wir als hinreichend bekannt ansehen. Im PHP-Programmteil erfolgt die oben beschriebene Aufgabe jetzt in umgekehrter Reihefolge. Die Dateien werden geöffnet, und die Eintragungen in der ersten Spalte jeder Zeile werden überprüft. Über eine Abfrage wird die Anzahl der noch freien Arbeitsplätze ermittelt.

Die Anzahl freier Plätze wird in einer Zeile mit dem Radio-Button und der Seminarnummer über die Echo-Funktion ausgegeben. Das weitere Formular wird wieder in einem HTML-Teil erstellt.

```
<?php
//
// Variablendeklaration
$iDateien    = 2; $iRechner  =16; $iPlatz    = 0;
$cMinus  = "-"; $cKomma   = ","; $cBlank  = " ";
//
// Schleife über die Anzahl der Seminare, hier 2
for ($i=0; $i<$iDateien; $i++){
 $iFileNumber = $i+1;
 //
 // Dateiname generieren und Datei mit Lesezugriff oeffnen
 $sDateiname="seminar_".$iFileNumber .".txt";
 $hDatei = fopen ($sDateiname, "r+");
```

```
  //
  // Zaehler initialisieren
  $iZaehler[$i] = 0;
  //
  // Datenzeilen lesen
  for ($k=0; $k< $iRechner; $k++){
    //
    // liest max 128 Zeichen bis Zeilenende
    // Feldwerte durch Komma getrennt
    $sRecord = fgetcsv ($hDatei, 128);
    //
    // Arbeitspaltz belegt?
    if ($sRecord[0] != $cMinus)
        $iZaehler[$i]++;
}
//
// alle Zeilen wurden überprueft, Datei schliessen
  fclose($hDatei);
//
//Anzahl freier Plaetze bestimmen
  $iAPfrei[$i] = $iRechner-$iZaehler[$i];
//
// HTML-Code ausgeben
echo ("<INPUT TYPE=\"RADIO\"
        NAME=\"semselect\"
        VALUE=\"" .$iFileNumber ."\"> Seminar "
        .$iFileNumber ." :Freie Arbeitsplätze "
        ."<b>".$iAPfrei[$i] ."</b><br>");
}
//
// Vorgang mit der nächsten Datei wiederholen
?>
```

Lesen von CSV-Dateien

Lesen von CSV-Dateien erfolgt mit der Funktion `fgetcsv()` mit den Parametern des Dateihandlers und der maximalen Länge der einzulesenden Zeichenkette. Das Ergebnis wird einem Array zugeordnet.

Innerhalb der Echo-Funktion ist darauf zu achten, dass Sonderzeichen, die für den Browser bestimmt sind, ggf. mit einem Backslash eingeleitet werden. Im Falle der Hochkommata würden sonst die Argumente für die Echo-Funktion verfälscht.

Der PHP-Programmteil ist nach Abarbeitung aller Dateien beendet. Das Eingabeformular wird per HTML-Code vervollständigt.

Abb.7.4: Anmeldeformular

Der Form-Tag der HTML-Datei hat folgende Gestalt:

```
<form name="select_sem" method = "post"
      action="neuer_eintrag.php">
```

Nach Ausfüllen der Teilnehmerangaben werden die Parameter mit der Methode `post` an den Server gesandt und die Datei *neuer_eintrag.php* angefordert.

Was passiert, wenn die Methode `get` eingesetzt wird? Die Aktion Aufruf des PHP Scripts wird in der Adresszeile des Browsers angezeigt:

```
http://localhost/neuer_eintrag.php?semselect=1&neu_name=Po-
maska&neu_vorname=G%FCnter+&neu_eMail=gp@imagefact.de&Submit1=
Anfrage+senden
```

Sie erkennen die Atrribute (Variablen) und deren Werte getrennt durch &. Die Codierung der Sonderzeichen ist ebenfalls erkennbar. Das Antwortscript gibt eine Bestätigung aus. An dieser Stelle kann im Browser die Ansicht-Aktualisieren-Funktion aufgerufen werden, und der Anmeldevorgang wird unkontrolliert wiederholt.

Unter Verwendung der Methode `post` wird hier zunächst eine Warnung zur Neuübertragung der Parameter ausgegeben. Geschützt ist eine weitere Eintragung aber auch nicht. Wir müssen also noch Maßnahmen treffen, um diesen gravierenden Mangel abzustellen, die Lösung wird weiter unten angegeben. Ferner unterscheidet sich die Post-Methode von der Get-Methode dadurch,

dass größere Datenmengen übertragen werden können und die Anzeige der Daten unterbleibt, da diese nicht in der URL mit übertragen werden, sondern sich im Übertragungsheader befinden.

Welche Anforderungen sind nun an das Script zur Eintragung des Anmeldenden in die Datei zu stellen? Das Script muss die ankommenden Daten auswerten, diese in die Datei eintragen und eine Bestätigung an den Absender übermitteln. Notwendig ist neben der Bestätigung am Bildschirm auch eine Mitteilung per Email, die im weiteren Verlauf der Projektbeschreibung vorgestellt wird. Hier zunächst die Diskussion des Daten empfangenden Programms.

Auswertung der empfangenen Daten

Die gesendeten Daten sind im PHP-Script unter den Bezeichnern, die beim Input-Tag im HTML-Formular benutzt wurden, verwendbar. In PHP muss aber das $-Zeichen vorangestellt werden. Im Anschluss an die einleitenden HTML-Tags werden im PHP-Abschnitt die eingehenden Daten geprüft. Sofern ein Datenfeld leer ist, wird keine Eintragung vorgenommen. Eine Warnmeldung wird angezeigt und das Programm mit `exit` wieder verlassen.

```
<html>
  <head>
  <title>PHP-Programm: Neuer Eintrag in Datei
  </title>
  </head>
<body bgcolor="#dddddd">
<?php
$iRechner    = 16; $cAsterix   = "*"; $cMinus     = "-";
$cKomma      = ","; $cBlank     = " "; $sEOL       = "\n";
$bAnmeldung=FALSE;
//
// Abfragen auf zulässige Werte
$len = strlen(trim($neu_name));
if ($len == 0) {
   echo ("Sie haben keinen Namen eingegeben");exit;}
$len = strlen(trim($neu_vorname));
if ($len == 0) {
   echo ("Sie haben keinen Vornamen eingegeben"); exit;}
$len = strlen(trim($neu_eMail));
if ($len == 0) {
   echo ("Sie haben keinen eMail-Adresse eingegeben"); exit;}
$len = strlen(trim($semselect));
if ($len == 0) {
```

```
    echo ("Sie haben kein Seminar ausgewählt"); exit;}
```

Sofern alle Daten geprüft sind, kann die Verarbeitung erfolgen. Die Eingabedatei wird vollständig eingelesen. Dabei wird die Anzahl belegter Arbeitsplätze gezählt. Der neue Eintrag wird dann beim Array-Schlüssel `$iZaehler` vorgenommen und die gesamte Datei wird nach Aufruf der Rewind-Funktion zurückgeschrieben. Mit `rewind()` wird der Dateizeiger auf den Anfang der Datei zurückgesetzt. Aufgrund der geringen Datenmenge ist diese Vorgehensweise vertretbar. Abschließend erfolgt die Ausgabe der Meldung, dass der Teilnehmer erfolgreich eingetragen wurde.

```
//
// oeffnet Datei mit Lesezugriff
 $sDateiname="seminar_" .$semselect .".txt";
 $hDatei = fopen ($sDateiname, "r+");
 $iZaehler = 0;
 for ($k=0; $k< $iRechner; $k++){
   // liest max 128 Zeichen bis Zeilenende
   // Feldwerte durch Komma getrennt
   $sRecord = fgetcsv ($hDatei, 128);
   if  ($sRecord[0] != $cMinus) {
    //zaehlt belegte Arbeitsplaetze
     $iZaehler++;
  }
    $iComputer[$k]= $sRecord[0];
   $sName[$k]    = $sRecord[1];
   $sVorname[$k ]= $sRecord[2];
   $sMail[$k]    = $sRecord[3];
   $sDatum[$k]   = $sRecord[4];
  } //Ende Dateneingabe
```

Es wird jetzt überprüft, inwieweit ein Teilnehmer bereits eingetragen ist. An dieser Stelle darf keine Identität von Vornamen und Namen bestehen. Bevor auf Gleichheit abgefragt wird, werden führende und nachfolgende Leerzeichen an den Eingabewerten entfernt.

```
for ($n=0; $n<$iRechner; $n++){
  if (
   (trim($neu_name)==trim($sName[$n]))
       &&
   (trim($neu_vorname)== trim($sVorname[$n]))
     ){
    echo("Ein Teilnehmer mit dem Namen: "
    .$neu_vorname  . $neu_name ."ist bereits
```

```
    eingetragen.
    <br>Benutzen Sie die Schaltfl&auml;che
    des Browsers zum Neueintrag.");
     exit;
  }
```

Jetzt wird noch einmal überprüft, ob es einen freien Platz gibt, dann erfolgt die Mitteilung an den Teilnehmer als Protokoll auf dem Bildschirm und als Email. Die Datei wird aktualisiert und geschlossen. Das aktuelle Datum der Anmeldung wird geschrieben !

```
//
//  Gibt es einen freien Platz ?
if ($iZaehler<16) {
 $iComputer[$iZaehler] = $cAsterix;
 $sName[$iZaehler]     = $neu_name;
 $sVorname[$iZaehler]  = $neu_vorname;
 $sMail[$iZaehler]     = $neu_eMail;
 $sDatum[$iZaehler]    = date("Y-m-d");
 $iTeilnehmerNr=$iZaehler+1;
 echo (
   "<div align=\"center\">
    <table
    width=\"400\" border = \"1\"
   bgcolor=\"#dddddd\">
   <tr>
     <td
       align=\"center\"><p>
       $neu_vorname $neu_name,  mail:
   $neu_eMail<br> Sie haben sich
   erfolgreich zum
   Seminar Nr.: $semselect angemeldet. <p>
   Der Rechner Nr.: $iTeilnehmerNr ist f&uuml;r
   Sie reserviert.</p></td></tr></table>");
```

Email mit PHP

Mit der Mail-Funktion von PHP wird automatisch eine Email an den Absender der Anmeldung versandt. Sie können beim Versand u.a. einen Absender, eine Betreffzeile und einen Header mit angeben. Die eigentliche Mitteilung ist im verknüpften String `$sMessage` enthalten. Die Mail wird durch Aufruf der Funktion `mail()` versandt. Wir beschränken uns hier auf eine reine Textmail.

```
// eMail absenden
$sAbsender="www.Programmierpraktikum.de"
$sBetreff="Seminaranmeldung";
```

```
$iArbeitsplatz = $iZaehler+1;
if ($semselect == 1){
// Zusammenstellen der Mitteilung für das Seminar 1
if ($semselect == 1){
 $sMessage  =  $neu_vorname ", ";
 $sMessage .= $neu_name;
 $sMessage .= "\nSie haben sich am "
 $sDatum[$iZaehler];
 $sMessage .= " zum Semiar Nr.: " .$semselect
 ." angemeldet.\n";
 $sMessade .= "\nRechner Nr.: "
 .$iArbeitsplatz  ." ist fuer Sie
 reserviert.\n";
 $sMessage .= "Seminarbeginn ist ...\n";
 $sMessage .= "\nBitte bringen Sie die
 Bestaetigung zum Seminarbeginn mit.\n";
 $sMessage .= "\nmfg\n\nDie Seminarleitung";
 }
 if ($semselect == 2){
  // Zusammenstellen der Mitteilung für das
  // Seminar 2 wie oben aber geaenderte Termine
 }
// die Headerdaten:
$header="From: www.Programmierpraktikum.de<" .$neu_eMail.">\n";
$header .= "Reply-To: gp@imagefact.de\n";
$header .= "X-Mailer: PHP/" . phpversion "\n";
$header .= "Content-Type: text/html";
// Aufruf der Mail-Funktion
mail($neu_eMail, $sBetreff, $sMessage,
     $header);
```

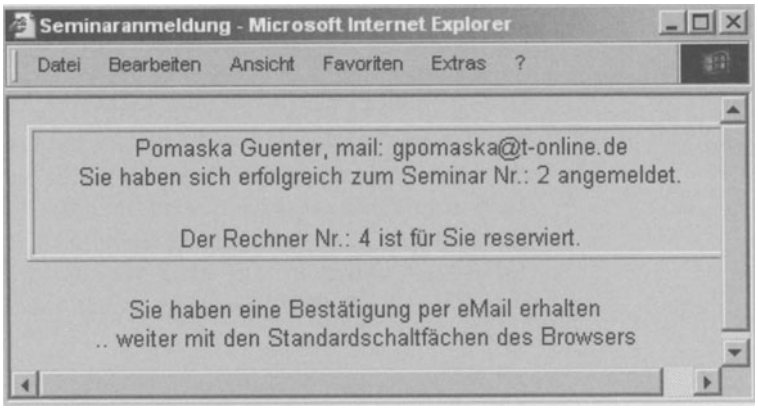


Abb. 7.5: Die Anmeldung war erfolgreich

Es folgt noch der verbleibende Code für die Bestätigung und das Aktualisieren der Daten.

```
echo("<p>Sie haben eine Best&auml;tigung per eMail
        erhalten<br/>")
echo("... weiter mit den Standardschaltf&auml;chen
        des Browsers</p></div>");
$bAnmeldung=TRUE;
// Daten aktualisieren
rewind($hDatei);
for ($k=0; $k< $iRechner; $k++){
    $sRecordOut = $iComputer[$k] .$cKomma
   .$sName[$k] .$cKomma
   .$sVorname[$k]  .$cKomma .$sMail[$k]
   .$cKomma .$sDatum[$k] .$sEOL;
   // Ausgabe Zeichenkette
   fputs ($hDatei,$sRecordOut);
  }// Ende Datenzeilen in Datei schreiben
fclose($hDatei); // Datei schliessen
// fuer den Fall, das etwas schief ging
if(!$bAnmeldung){
  echo ("<div align=\"center\"><table
  width=\"400\" border = \"1\"
  bgcolor=\"#dddddd\"><tr><td
  align=\"center\"><p>
  Ihre Anmeldung konnte nicht durchgefuehrt
  werden</p></td>
  </tr>
  </table>
  </div>");
  exit;
  }
?>
</body></html>
```

Mit den drei beschriebenen Skripten läuft das Online-Anmeldeverfahren. Jeder kann sich anmelden, Berechtigungen wurden nicht verteilt. Haben wir die Bestätigung zu früh rausgeschickt? Muss erst die Zahlung der Teilnehmergebühr abgewartet werden? Diese Verfahrensfragen sind nicht Gegenstand unserer Betrachtungen. Der File-Generator darf natürlich nicht öffentlich zugänglich sein. Die Besucher müssen die Scripte *freie_aps_form.php* und *neuer_eintrag.php* ausführen können. Das betreffende Verzeichnis muss auch Lese- und Schreibrechte für die Benutzer gestatten.

Mit dem FTP-Client Filezilla ist die Erteilung der Rechte kein Problem. Sie öffnen das Kontextmenü durch Klick mit der rechten Maustaste auf ein Verzeichnis bzw. eine Datei und tragen die Dateiattribute durch Markieren der Checkboxen ein, vgl. Abbildung 7.6.

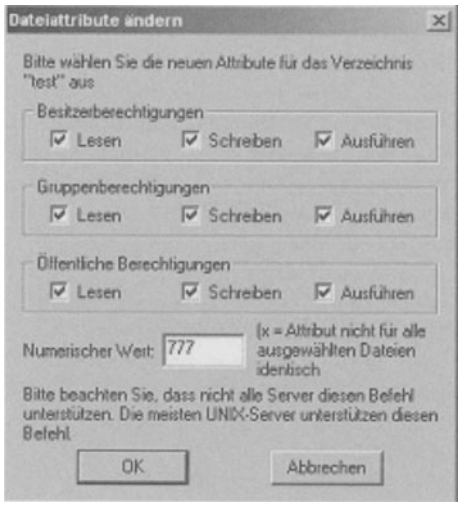


Abb. 7.6: Vergabe der Zugriffsrechte mit Filezilla

Wir können auch noch ein kleines Script gebrauchen, das eine sortierte Liste der Eintragungen in übersichtlicher Form ausgibt. Das sollte kein Problem darstellen. Sie finden es auf der Webseite in der Datei *auswertung_liste.php*. Die Ausgabe kann bei Bedarf entsprechend manipuliert werden.

Programmierpraktikum Teilnehmerliste

Rechner Nr.	Name	Vorname	Email	Datum
1	Fred	Freddy	gp@imagefact.de	2004-08-30
2	Karl	hans	gpomaska@t-online.de	2004-08-30
3	Letzte	Version	gp@imagefact.de	2004-08-30
4	Neuer Versuch	Hugo	gp@imagefact.de	2004-08-30

Abb. 7.7: Ausgabe der sortierten Anmeldeliste PHP-Script, *auswertung-liste.php*

VRML-Konfigurator

Virtual Reality Modeling Language (VRML) ist eine Formatbeschreibung zur Distribution von 3D-Grafikdaten im Internet. VRML in der Version 2.0 gilt als Vorgänger bzw. Untermenge von X3D, dem XML-konformen 3D-Grafikformat für das Web. In einer mit VRML beschriebenen Szene kann sich der Betrachter frei bewegen, sich durch die Szene navigieren. Hierbei findet die Größe des Betrachters, die Schwerkraft und die Kollision mit Objekten der 3D-Welt Berücksichtigung. Voraussetzung zur interaktiven Betrachtung ist ein VRML-Viewer. Derzeit gilt Cortona 4.2 als einer der leistungsfähigsten, zu beziehen als Freeware von *www.parallelgraphics.com*. Wenn Sie die hier vorzustellende Anwednung von der Webseite starten wollen, nehmen Sie vorher bitte die Installation eines VRML-Clienten vor.

VRML-Dateien tragen die Erweiterungsbezeichnung `.wrl`. Die Formate können aus CAD-Systemen heraus exportiert werden. Ein exportiertes Datenformat UTF-8 liefert scheinbar endlose Koordinatenlisten. VRML bietet aber auch geeignete Mechanismen zur Strukturierung der Daten. Eine Möglichkeit ist die Benutzung des Inline-Befehls. Die VRML-Datei muss in der ersten Zeile die Versionsinformation tragen, dann können Inline-Anweisungen den Import von diversen Dateien bewirken.

```
#VRML V2.0 utf8
Inline {url
"http://www.programmierpraktikum.de/vr_model_sp/navigation.wrl
"}
Inline {url
"http://www.programmierpraktikum.de/d/javascript/klavierspiele
r.wrl"}
```

Obige Codierung wäre ein Beispiel für eine VRML-Datei, in der weitere externe Dateien referenziert werden. Unser Problem liegt darin, ein städtisches Ensemble mit seiner Bebauung darzustellen. Das kann zu gewaltigen Datenmengen führen. Daher wird mit dem Begriff *Level of Detail* (LOD) die Strukturierung der Daten bezeichnet. Ein Bauwerk kann als Block, Struktur mit Dachformen, oder mit Wandöffnungen und Fassadenelementen in unterschiedlicher Detaillierung dargestellt werden. Die LOD werden von 0 bis 3 mit Zahlen gekennzeichnet.

Betrachtet man ein 3D-Modell mit einem VRML-Viewer, dann können dem Beobachter nahe Modelle mit hoher Detailtreue präsentiert werden, während weiter entfernte Objekte mit weniger Detaillierung gezeigt werden. In unserem Fall soll sich aber der Betrachter sein Modell vor Begehung selber konfigurieren.

Für jedes Gebäude der Szene liegt daher ein Modell unterschiedlicher Detaillierung vor. Zusätzlich soll die Anwendung auch noch die Darstellung zeitlich unterschiedlicher Epochen ermöglichen. Es liegt demnach eine Objektstruktur wie in der Abbildung 7.8 gezeigt vor. Aus dem Modellkubus mit den Achsen Objekte, Epochen und Detaillierung stellt sich der Web-User selbst eine virtuelle Welt zusammenstellen.

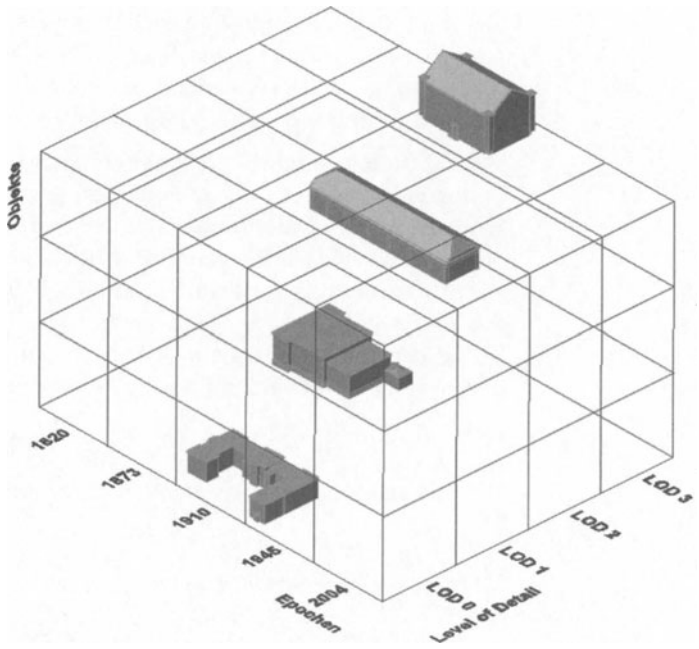


Abb. 7.8: Modellkubus, Auswahl von Objekten, Epochen und Modellstruktur

Ein geeignetes Interface kann unter Nutzung der Formulareigenschaften von HTML bereitgestellt werden. Die Auswahl der Objekte erfolgt durch Markieren von Checkboxen, das Level of Detail ist alternativ mit Radiobuttons einzuschalten. Da immer nur eine Epoche abzubilden ist, erfolgt die Epochenauswahl aus einer Vorgabeliste. Das Benutzerinterface im Webbrowser mit entsprechenden Formatierungen und Bildern zeigt Abbildung 7.9. Das PHP-Programmierproblem ist nicht sonderlich komplex., beschränkt es sich doch weitgehend auf den Zugriff auf das Dateisystem des Servers.

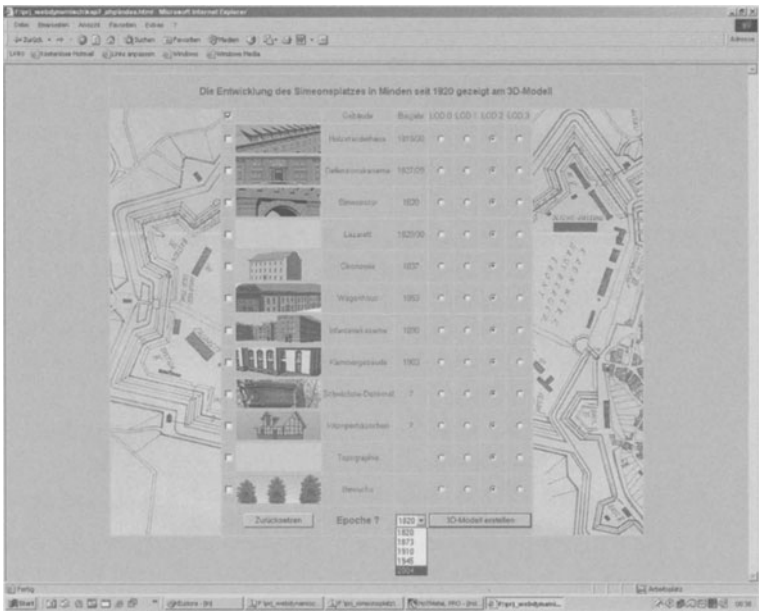


Abb. 7.9: HTML-Formular als Benutzerinterface zur Auswahl von VRML-Komponenten

Die HTML-Codierung wird hier nur hinsichtlich der Formularauswertung diskutiert. Alle Bedienelemente werden mit Tabellen positioniert, die Gestaltung erfolgt mit Stilvorlagen. Alle Input-Elemente befinden sich innerhalb des Formulars mit dem Namen *config*, bei Klick auf den Submit-Button wird die angegebene `action` ausgelöst. Die Variablen werden mit der Methode `post` an die PHP-Anwendnung weitergeleitet.

Für den Programmtest empfiehlt sich zunächst die lokale Installation mit `http://localhost`.

```
<form name="config" method="post"
   action= "http://localhost/config_wrl.php"
   id="wrl_file">
</form>
```

Eine einzelne Checkbox wird mit

```
<input type="checkbox"
       name="hst_check"
       checked="checked" >
```

deklariert. Der Name der Checkbox bekommt in unserem Programm als Präfix einen drei Zeichen langen Code, der objektbe-

zogen einem Gebäude zugeordnet ist. Die Variable hat den Inhalt on, wenn der Status checked ist, andernfalls wird ein Leerstring übergeben.

Radiobuttons, die mit dem selben Namen bezeichnet sind, bilden eine Gruppe. Aus dieser Gruppe kann immer nur ein Button angeklickt sein.

```
<tr>
   <td><input type="radio" name="hst" value="10"></td>
   <td><input type="radio" name="hst" value="11"></td>
   <td><input type="radio" name="hst" value="12"
                  checked= "checked"></td>
   <td><input type="radio" name="hst" value="13"></td>
</tr>
```

Der Radiobutton *hst* kann, wie alle anderen Radiobuttons auch, die Werte `lo,11,12` oder `13` annehmen. Voreingestellt ist hier `12`, der Wert für eine mittlere Detailtreue der Grafikdaten.

Es folgt noch zur Auswahl einer Epoche das Element `<select>`. Der Name ist *epoche*, angezeigt wird nur eine Zeile der Liste. Der Wert ist für jede ausgewählte Epoche unterschiedlich.

```
<select name="epoche" size="1">
   <option value="1820">1820</option>
   <option value="1873">1873</option>
   <option value="1910">1910</option>
   <option value="1945">1945</option>
   <option value="2004">2004</option>
</select>
```

Jetzt fehlen nur noch die Schaltflächen mit einer Reset-Funktion zum Zurücksetzen aller Parameter auf die Anfangswerte und der Submit-Button zum Aufruf des weiterverarbeitenden Programms mit der gleichzeitigen Parameterübergabe. Der Submit-Button erhält ein individuelles Label, zugewiesen im Attribut `value`.

```
<input type="RESET" name="Reset1">
<input type="SUBMIT" name="Submit1"
     value="3D-Modell erstellen">
```

Zusammenhängend und komplett finden Sie den Quelltext in der HTML-Datei

www.programmierpraktikum.de/vr_model_sp/config_wrl_web.html

Nach Klick auf den Button Submit wird der lokale Webserver aufgerufen und die Datei *config_wrl.php* angefordert. Der Ser-

ver erkennt an der Endung PHP des weiterverarbeitenden Programms, dass der Hypertext Pre-Prozessor einzuschalten ist. PHP nimmt die Auswertung der Daten vor und stellt das Ergebnis in einer HTML-Datei dem Client zur Verfügung.

Im Verlauf einer Programmentwicklung ist es empfehlenswert, zunächst alle ankommenden Werte zu überprüfen. Die Echo-Anweisungen, Stringliterale mit eingebetteten Variablen, können später wieder auskommentiert werden. Es werden die Werte von 12 Checkboxen, 12 Gruppen Radiobuttons und ein Wert der Auswahlliste angezeigt.

```
<?php
 // Protokoll Eingangsparameter
  echo "Holszstaenderhaus:  $hst hst_check: $hst_check<br/>";
  echo "Defensionskaserne:  $def def_check: $def_check<br/>";
  echo "Simeonstor:         $str str_check: $str_check<br/>";
  echo "Lazarett:           $laz laz_check: $laz_check<br/>";
  echo "Oekonomie:          $oek oek_check: $oek_check<br/>";
  echo "Wagenhaus:          $wgn wgn_check: $wgn_check<br/>";
  echo "Infanteriekaserne:  $inf str_check: $inf_check<br/>";
  echo "Kammergebaeude:     $kam str_check: $kam_check<br/>";
  echo "Schwichow-Denkmal:  $shw str_check: $shw_check<br/>";
  echo "Kruemperhaeuschen:  $krh krh_check: $krh_check<br/>";
  echo "Topographie:        $top top_check: $top_check<br/>";
  echo "Bewuchs:            $bme bme_check: $bme_check<br/>";
  echo "epoche:             $epoche <br/>";
?>
```

Bevor es jetzt weiter mit der PHP-Programmierung geht, wollen wir noch mal auf unsere Problemformulierung zurückkommen. Es wird eine VRML-Datei mit der Referenzierung von wiederum VRML-Dateien benötigt, wie eingangs dargestellt. Alle Dateien liegen in einer Dateistruktur entsprechend des in der Abbildung 7.10 dargestellten Verzeichnisbaums vor.

Die Dateinamen setzen sich jeweils zusammen aus der Objektkennung, der Epoche, einer Kennung für das Level of Detail und der Erweiterungsbezeichnung .wrl. Es gilt jetzt entsprechend dieser Struktur und den Benutzereingaben von der HTML-Seite, eine neue Datei auf dem Server anzulegen und diese dann dem VRML-Browser zur Anzeige zu übermitteln.

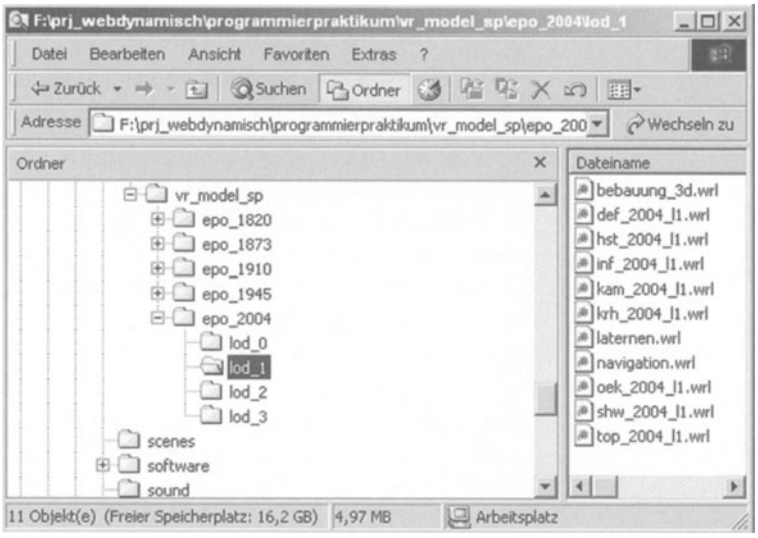


Abb. 7.10: Verzeichnisbaum der VRML-Dateien

Im einleitenden PHP-Programmteil werden die Eingangsvariablen zunächst Feldern zugewiesen, da diese innerhalb von Schleifen geeignet zu verarbeiten sind. Die Zuweisungen werden gekürzt dargestellt, entfernte Zeilen sind durch ... markiert. Das komplette Programm entnehmen Sie bitte wieder dem Leserbereich der Website.

```
//
// Array Checkboxen
$obj_on[0]= $hst_check; ...$obj_on[10]=$bme_check;
//
// Array Objektbezeichner
$obj_bez[0]= "hst"; ... $obj_bez[10]= "bme";
//
// Array Level of Detail
$obj_lev[0]= $hst; ... $obj_lev[10]= $bme;
```

Es folgt die Deklaration von Variablen zur Ermittlung der Referenzen und einiger Sonderzeichen.

Die Variable $subdir_epoche wird aus der Zeichenkette epo_ mit dem Inhalt der Variablen $epoche gebildet. Die Verknüpfung erfolgt mit dem Punktoperator.

```
$url="http://www.programmierpraktikum.de";
$dir="vr_model_sp";
$subdir_epoche="epo_" .$epoche;
```

```
$path="http://www.programmierpraktikum.de/vr_model_sp";
$k=0;
$sEOL="\n";
$sBlank=" ";
$sSlash="/";
//
// Zielfile oeffnen und Kopfzeile schreiben
$sDateiname="d3modell_sp.wrl";
$hDatei = fopen ($sDateiname, "w");
fputs ($hDatei,"#VRML V2.0 utf8" .$sEOL);
fputs ($hDatei,"Inline {url \"" .$url ."/" .$dir
."/navigation_sp.wrl\"}" .$sEOL);
```

Das Zielfile erhält einen konstanten Namen, hier *d3modell_sp.wrl*, und befindet sich im selben Verzeichnis wie das PHP-Programm.

Mit der Funktion `fopen()` wird der Dateihandler `$hDatei` definiert. Die Datei wird mit Schreibzugriff geöffnet. Der Funktionsaufruf von `fputs()` bewirkt die Ausgabe einer Zeichenkette in die Datei. Ein Zeilenvorschub wird mit `$sEOL` erzeugt.

Die Datei *navigation_sp.wrl* enthält Informationen zur Navigation in der 3D-Welt und wird standardmäßig immer zugefügt. Hiernach folgt ein erster Block mit HTML-Anweisungen, u.a. auch Style Sheets, ein Form-Tag und der Anfang eines Select-Tags.

```
//
// HTML-Ausgabe
echo ("<html><head><title>3D-Modell Konfigurator</title>");
echo ("<style type=\"text/css\">
        td{font-family:\"\"Courier New\"\", arial;
        font-size:10pt;color:#5a460b;
        font-weight:normal;
        }
        select, option {
        font-family : \"Courier New\", Arial;
        background:#cccc99; color:##5a460b;
        font-size:8pt; font-weight:normal;
      }
     </style>
      </head>");
echo ("<body margintop= \"128px\"
        bgcolor=\"#cccc99\">
```

```
      <div   align=\"center\">");
echo ("<form>");
echo ("<table width=\"640\" align=\"center\">");
echo ("<tr><td>URL: " .$url ."<br/></td></tr>");
echo ("<tr><td>Project: " .$dir ." <br /></td></tr>");
echo ("<tr><td>Epoche:  " .$subdir_epoche
      "<br /></td></tr>");
echo ("<tr><td>Files:<select  size=\"1\">");
```

In einer Schleife über alle Objekte wird jetzt für jedes in der Checkbox markierte Objekt eine Ausgabezeile zusammengestellt und in die .VRML-Datei geschrieben.

Innerhalb dieser Schleife erfolgt auch die Ergänzung des Select-Tags mit den `<option> ... </option>`. In einer Listbox werden die ausgewählten Dateien angezeigt. Wir bestimmen auch noch die Größe der zu übertragenden VRML-Daten mit der Funktion `filesize()`. Nach Abarbeitung der Schleife wird die Datei mit `fclose()` geschlossen. Abfragen auf Existenz und Gültigkeit der Dateien konnten unterbleiben, da ein VRML-Browser falsche Angaben in der Inline-Anweisung überliest. Außerdem gehen wir hier mal von der korrekten Installation der Basisdaten auf dem Server aus.

```
$groesse=0;
for ($i=0; $i<11;$i++){
  if ($obj_on[$i]=="on") {
    //
    // Level of Detail abfragen
    if ($obj_lev[$i]=="l0") $subdir="lod_0";
    if ($obj_lev[$i]=="l1") $subdir="lod_1";
    if ($obj_lev[$i]=="l2") $subdir="lod_2";
    if ($obj_lev[$i]=="l3") $subdir="lod_3";
    // relative Filereferenz
    // und URI ermitteln
    $file = "epo_" .$epoche ."/" .$subdir
            ."/"
             .$obj_bez[$i] ."_" .$epoche ."_"
             .$obj_lev[$i] ".wrl";
    $download=  $path ."/" .$file;
    $groesse += filesize($file);
    //
    // Ausgabe <option> ... </option>
    echo ("<option>$download</option>");
    //
```

```
    // Ausgabezeile in VRML-Datei schreiben,
    $outline="Inline {url \"" .$download
             ."\"}" .$sEOL;
    $k++;
    //
    // Ausgabezeile schreiben
    fputs ($hDatei,$outline);
   } // Ende ausgewaehltes Objekt
}
fclose($hDatei);
```

Das VRML-File ist nun konfiguriert und kann vom Browser aufgerufen werden. Die HTML-Datei ist noch dementsprechend zu vervollständigen.

Zur Einbettung der VRML-Datei benutzen wir hier noch den Embed-Tag. Der Webbrowser reserviert einen Bereich von 640 x 360 Pixel für die Datei *d3modell_sp.wrl*. Positioniert wird dieser Bereich in einer Tabelle. Es wird auch noch ein kleines JavaScript zur Datumsanzeige eingebunden und die HTML-Datei abgeschlossen.

Der Server sendet die generierten Daten jetzt an den Webbrowser, dieser stellt den HTML-Code dar, der VRML-Browser rendert die Grafikdaten. Viel Spaß beim Navigieren durch die Szene.

```
//
// .... weiter mit HTML-Code
echo("</select></tr></td>");
echo("
 <tr>
   <td>
     <embed type=\"model/vrml\"
            name=\"VRML-Datei\"
            src=\"d3modell_sp.wrl\"
            VRML-DASHBOARD=\"false\"
            VRML-BACKGROUND-COLOR=\"#ffffcc\"
            VRML-SPLASHSCREEN=\"true\"
            width=\"640\"
            height=\"360\">
      </embed></td></tr>
   <tr>
     <td>
        3D-Modell Simeonsplatz konfiguriert
        am: ");
        echo("
         <Script language=\"JavaScript\">
         var heute = new Date();
```

```
                document.write(
                  heute.toLocaleString());
              </Script>");
          echo(" Modellgroesse: "
          .$groesse ." Byte");
          echo("<br />
            &copy; gp www.imagefact.de");
          echo("</td></tr>
</table>
</form>
</div>
</html>");
?>
```

Listing 7.13: Das PHP-Script zur Zusammenstellung der VRML-Komponenten in Auszügen

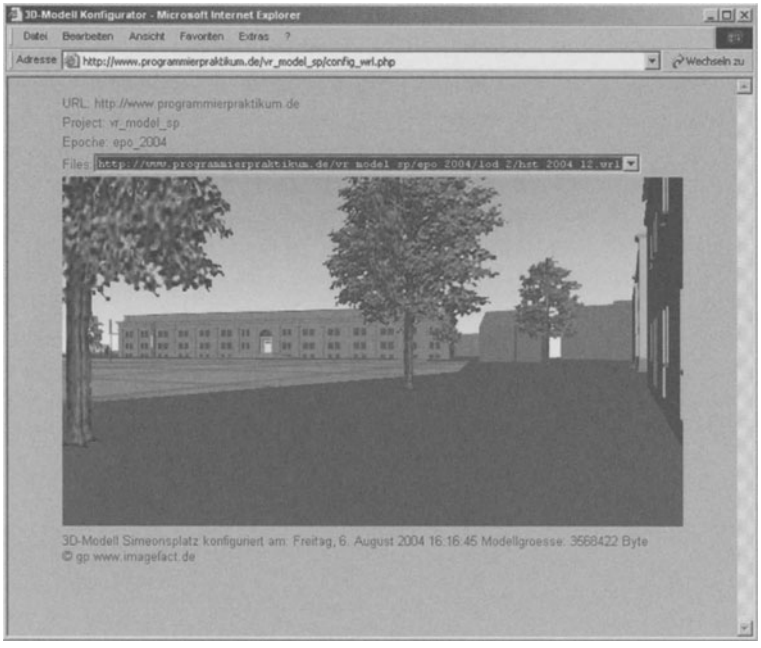


Abb. 7.11: Die vom PHP-Script generierte HTML-Seite.

Die vom PHP-Script generierte HTML-Seite zeigt die ausgewählten Dateien in der Listbox über dem Embed-Bereich, Datum und Uhrzeit aus der JavaScript-Anweisung und die Summe der übertragenen Bytes. Sie können jetzt auch die Auswirkung der Parameter im Embed-Element erkennen. `VRML-Dashboard ="false"`

unterdrückt die Anzeige der Navigationselemente. Den Splash-Screen sehen Sie aufgrund der langen Ladezeit von 3.5 MByte. Die Hintergrundfarbe wird nicht sichtbar, da wir einen eigenen Background in der VRML-Datei erzeugt haben. Das Kontextmenü des Cortona VRML-Client öffnet sich bei rechtem Mausklick im Fenster. Dort können Sie dann Viewports auswählen und eigene Einstellungen vornehmen.

Das formulierte Problem ist hier auf einen speziellen Fall zugeschnitten, kann aber durch Modifikation auch allgemein Gültigkeit bei der Formulierung von Dateischnittstellen haben und ist nicht auf die 3D-VRML Applikation beschränkt.

7.4 Hochladen von Dateien mit PHP

Das Hochladen von Dateien auf einen Webserver erfolgt üblicherweise mit einem FTP-Client. Filezilla wurde im Kapitel 2 als Beispiel angeführt und dort auch angewandt. Der Nutzer von Filezilla muss natürlich die Zugangsdaten zum Webserver kennen. Da wir diese aber nicht preisgeben wollen, können wir mit PHP-Unterstützung Besuchern der Website das Hochladen einzelner Dateien ermöglichen. Serverseitige Programme können dann mit diesen Daten arbeiten. Ein Verzeichnis auf dem Webserver mit Schreibrechten ist eingerichtet. Wir beschränken uns hier exemplarisch auf VRML-Dateien, das sind 3D-Grafikdaten mit dem Mime-Typ `model/vrml`.

Zunächst benötigen wir ein HTML-Formular. Im Attribut `enctype` wird dem Server die Dateiübermittlung signalisiert. Im Beispiel wird der lokale Server aufgerufen. Im Verzeichnis *htdocs* des Apache Webservers befindet sich die Datei *upload.php*. Mittels der Methode `post` wird der Name der Datei (hier in der Variablen `fn` gespeichert) an das PHP-Programm übermittelt. Die Datei selbst wird temporär gespeichert.

Das Attribut `type` des Input-Elements erhält den Wert `file`, damit wird das Durchblättern der clientseitigen Verzeichnisse möglich. Zusätzlich gibt es noch einen Submit- und einen Reset-Button.

```
<html>
<head><title>Upload</title></head>
<body><div align="center"><p>Hochladen von Dateien</p>

  <form name="up"
        action="http://localhost/upload.php"
```

```
        method ="post"
        enctype="multipart/form-data">
  <p>Filereferenz:
  <input type="file" name="fn" size="40"/></p>
  <p>Upload:
  <input type="submit" value="go" /> .....
  <input type="reset" value="reset" />
  </p>
  </form></div></body></html>
```

Listing 7.14: HTML-Eingabeformular zum Datei-Upload

Das PHP-Script hier zunächst in einer Kurzform. `$path` enthält das Verzeichnis, in das die temporäre Datei kopiert werden soll. Wir bestimmen Dateigröße, Name und Mime-Typ. Sofern der Typ nicht `model/vrml` ist, wird die Kopie nicht zugelassen. Die Referenz der Zieldatei wird aus dem Pfad und dem Filenamen gebildet. Mit `copy(quelldatei,zieldatei)` erfolgt die permanente Speicherung der temporären Datei auf dem Server.

```
<?php
  $path="f:/prj_webdynamisch/kap7_php/";
  echo "Datei  : $fn <br />";
  echo "Groesse: $fn_size<br />";
  echo "Name   : $fn_name<br />";
  echo "Type   : $fn_type<br />";
  if ($fn_type != "model/vrml"){

    echo "Dateityp ist nicht model/vrml:
          Upload fehlerhaft";
  }
  else {
    $ziel = $path.$fn_name;
     copy ($fn,$ziel);
     echo "$fn nach $ziel kopiert";
     echo "<br />Upload erfolgreich";
  }
 ?>
```

Listing 7.15: PHP Script zur Handhabung einer hochgeladenen Datei

Das Verfahren des Dateiuploads wird in der nächsten Anwendung zum Hochladen von Messdaten Verwendung finden. Auf der Webseite zum Buch ist die Funktion auskommentiert. Da ein allgemeiner Zugriff möglicherweise das Verzeichnisvolumen der Hompage sprengen könnte.

7.5 Dynamische Grafik mit PHP

Test der Grafikumgebung

PHP bietet mit der GD2-Grafikbibliothek etwa 100 Funktionen zur Erzeugung dynamischer Grafiken an. Die Grafikbibliothek kann in der php.ini-Datei durch Aktivieren (Kommentarzeichen; entfernen) der Zeile `extensions=php_gd2.dll` oder durch direkten Aufruf als erste Scriptzeile mit `dl("php_gd2.dll");` eingebunden werden.

Eine PHP-Bilddatei enthält einen Header, der den Browser auf den Typ der Bilddatei hinweist. Als zweite Anweisung wird eine Variable deklariert, die den Handler auf die Grafik enthält. Dieser wird mit `@ImageCreate(width,height)` initialisiert. Es folgen die PHP-Anweisungen zur Erzeugung der Grafik und die Anweisung `imagejpeg()`, oder `imagepng()` je nach Dateityp, mit der das Bild an den Browser gesandt wird bzw. in einer Datei auf dem Server gespeichert wird. Sie können die PHP-Grafikbibliothek mit folgendem kleinen Programm testen:

```
<?php
//Test der PHP-Grafikumgebung grafik_test.php
//PNG Format alternativ
//header("Content-Type: image/png");
//imagepng($img);
 header("Content-Type: image/jpeg");
// Bildgroesse 240px x 180 px
 $img       = imagecreate (240,180);
 // Farbwerte definieren, erste Farbe wird
 // Hintergrundfarbe
 $bg_color  = imagecolorallocate ($img,225,225,225);
 $txt_color = imagecolorallocate ($img,0,0,255);
  // Textausgabe: Interner Zeichensatz 5,
 //              Position x = 60, y = 100 Farbe $txt_color
 imagestring   ($img, 5, 30, 80, "PHP Grafik
                                  HelloWorld", $txt_color);
 // Bild an Browser zur Ausgabe senden
 imagejpeg($img);
 imagedestroy($image);  // Speicher wieder
 // freigeben
 // Alternativ: Ausgabe in eine Datei mit
 // Qualitaet 100
 // imagejpeg($img,"datei.jpg",100);
 //
 //
?>
```

Listing 7.16: Test der PHP-Grafikumgebung

Grafikfunktionen können in einem PHP-Script nicht mit HTML-Befehlen vermischt sein, da der Browser die unterschiedlichen Datentypen aufgrund der bereits empfangenen Header-Informationen dann nicht mehr interpretieren kann. Innerhalb einer PHP-Datei können Sie über die Echo-Anweisung einen Image-Tag erzeugen, der im Attribut src das PHP-Script aufruft.

```
echo("<img src=\"grafik_test.php\">");
```

Wenn Sie die Alternative der Dateispeicherung benutzen, können Sie anschließend mit folgender Anweisung im PHP-Script die Ausgabe der Datei veranlassen:

```
echo("<img src=\"datei.jpg\">");
```

Berechnung und Darstellung von Ausgleichungspolynomen

Wenden wir uns nach dieser Einführung dem Problem der Datenanalyse mit dem Ziel der Einführung in die dynamische Grafikprogrammierung mit PHP zu. Eine Funktion y = f(x) ist durch diskrete Punkte gegeben. Durch diese Punkte soll ein ausgleichendes Polynom vom Grad n gelegt werden. Das Programm soll die Koeffizienten des Polynoms und die grafische Darstellung des Polynoms durchführen, vgl. Abbildung 7.13. Zunächst betrachten wir ausschließlich die Erzeugung der Grafik. Später werden wir dann eine Schnittstelle für die interaktive Bearbeitung formulieren. Den umfangreichen Quelltext des Programms finden Sie im Zusammenhang auch wieder im Leserbereich auf der Website.

Vorbereitung der Grafik, Deklarationsteil:

Die Stützpunkte des Polygons liegen in zwei eindimensionalen Feldern vor:

```
$pt_x = array (-100.0, -75.0, -50.0, -25.0,
        0.0,  25.0,   50.0,   75.0,  100.0);
$pt_y = array (   4.06,  6.78,  9.49, 16.27,
       40.67, 97.62, 146.43, 151.85, 162.70);
```

mit

```
array_multisort ($pt_x, $pt_y, SORT_ASC, SORT_NUMERIC);
```

wird sichergestellt, dass die Werte nach aufsteigendem $x numerisch sortiert sind. Änderungen der Reihenfolge im Feld $pt_x würden auch auf $pt_y übertragen.Die Anzahl der Punkte wird mit der Array-Funktion count() bestimmt.

```
$pt_anz = count($pt_x);
```

Es werden die Extremwerte in x und y bestimmt. Mit $xmin, $xmax, $ymin und $ymax liegt das umhüllende Rechteck der

Messpunkte vor. Die Daten werden zur nächsten Zahl auf- bzw. abgerundet:

```
$xmin = floor ($xmin); $xmax= ceil ($xmax);
$ymin = floor ($ymin); $ymax= ceil ($ymax);
```

Der Grafikbereich soll noch eine Polsterung `$padding` als Raum für Randbeschriftungen bekommen. Mit `$precision` wird eine weitere Rundung der Zahlenwerte möglich. Der Grafikbereich in Koordinaten des Benutzers bzw. in den Dimensionen der Messwerte ist daher wie folgt bestimmt:

```
$xmin=round($xmin,$precision)-$padding;
$xmax=round ($xmax,$precision)+$padding;
$ymin=round($ymin,$precision)-$padding;
$ymax=round ($ymax,$precision)+$padding;
```

Die Bildgröße in Pixeln wird in der Weite vorgegeben, zur Vermeidung von unterschiedlichen Maßstäben in x und y wird die Höhe des Bildes berechnet und auf volle Pixel gerundet.

```
$img_x = 480;
$dx    = $xmax - $xmin;
$dy    = $ymax - $ymin;
$ratio = $dx/$dy;
$img_y =  ceil ($img_x *$ratio);
```

PHP-Grafikfunktionen erwarten Positionsangaben für Zeichenbefehle in der Dimension Pixel im Koordinatensystem des Bildschirms. Die obere linke Ecke ist 0,0 die positive x-Achse zeigt nach rechts, die positive y-Achse nach unten, vgl. auch Abbildung 4.2.

Alle Eingaben im Benutzerkoordinatensystem sind vor der Zeichnungsausgabe auf den durch `$img_x` und `$img_y` festgelegten Bildschirmbereich zu reduzieren. Das sind zwei lineare Transformationen, die in der Funktion `transform_to_image ()` durchgeführt werden. Eingangsparameter sind die Koordinaten im Benutzersystem `$x, $y`; Ausgangsparameter sind `&$x_img` und `&$y_img`. Beachten Sie den Adressoperator.

Die zur Transformation benötigten Variablen müssen für die Funktion noch sichtbar gemacht werden. Mit dem Schlüsselwort `global` wird dem Programm mitgeteilt, dass die Werte aus dem aufrufenden Programm zu übernehmen sind.

```
function transform_to_image ($x, $y, &$x_img, &$y_img){
  global $xmin, $ymin, $dx, $dy, $img_x,$img_y;
  $x_img =($x-$xmin)/$dx *$img_x;
```

```
  $y_img = $img_y - (($y-$ymin)/$dy *$img_y);
}
```

Die Grafik kann jetzt mit der Headerinformation und der Zuweisung des Grafikhandlers inititalisiert werden

```
header("Content-Type: image/jpeg");
$imgm= imagecreate ($img_x,$img_y);
```

Es folgt die Definition von Farben und weiteren Grafikparametern wie Schrittweite des zu zeichnenden Polynoms, Zeilenabstände für Beschriftung u.a.

```
$bg_color       = imagecolorallocate ($img,196,196,196);
$pt_color       = imagecolorallocate ($img,0,0,255);
$poly_color     = imagecolorallocate ($img,0,0,255);
$line_color     = imagecolorallocate ($img,0,0,0);
$grid_color     = imagecolorallocate ($img,255,255,255);
$pt_size        = 8;
$lineHeight     = 8;
$poly_thickness = 1;
$line_thickness = 1;
$grad           = 4;
$step           = 1;
```

Alle weiteren Aufgaben werden nunmehr in Funktionen formuliert. Es wird ein beschriftetes Gitternetz gezeichnet und die Punktmarkierung für Ellipsen vorgenommen.

```
function drawGrid($img,$xmin,$xmax,$ymin,$ymax,
                  $padding,$dist, $color){
  // Grid-Bereich
  $dx =  ($xmax-$xmin)- 2.0*$padding;
  $dy =  ($ymax-$ymin)- 2.0*$padding;
  // Anzahl Gitterlinien
   $nx =  $dx/($dist*$padding);
  $ny =  $dy/($dist*$padding);
  // zeichne Linien mit x = const
  $y_low =  $ymin + $padding;
  $y_up  =  $y_low + $dy;
  for ($i= 0; $i<=$nx; $i++){
    $x_const = ($xmin+$padding)+$i*($dist*$padding);
    transform_to_image ($x_const, $y_low, $x1, $y1);
    transform_to_image ($x_const, $y_up,  $x2, $y2);
    imageline($img, $x1, $y1, $x2, $y2, $color);
  }
// zeichne Linien mit y = const
  $x_left  =  $xmin + $padding;
```

```
  $x_right =  $x_left + $dx;
  for ($i= 0; $i<=$ny; $i++){
    $y_const = ($ymin+$padding)+$i*($dist*$padding);
    transform_to_image ($x_left, $y_const, $x1, $y1);
    transform_to_image ($x_right,  $y_const,$x2, $y2);
    imageline($img, $x1, $y1, $x2, $y2, $color);
  }
   transform_to_image ($xmin+$padding,
                       $ymax-$padding, $x1, $y1);
   transform_to_image ($xmax-$padding,
                       $ymax-$padding, $x2, $y2);
   imageline($img, $x1, $y1, $x2, $y2, $color);
}
```

Der Abstand der Gitterlinien ergibt sich aus der Randbreite, `$padding`, multipliziert mit einem Faktor `$dist`. Innerhalb zweier For-Schleifen werden die Linien mit x = const bzw y = const mit der Funktion `imageline()` gezeichnet. Imageline zeichnet eine Linie von einem Punkt P1 zu einem Punkt P2 in der angegebenen Farbe, der Grafikhandler ist wie bei allen Ausgabefunktionen mit zu übergeben. Die Funktionen zur Achsbeschriftung sind für die x-Achse und die y-Achse getrennt angelegt. Die Ausgabe der Zahlenwerte erfolgt mit `imagestring()`. Das Argument für `$font` legt von 1 bis 5 einen internen Zeichensatz fest.

```
function drawTextXAchse($img, $y, $xmin, $xmax,
  $padding, $dist, $color,$font){
  // Grid-Bereich
  $dx = ($xmax-$xmin)- 2.0*$padding;
  // Anzahl Gitterlinien
  $nx =  $dx/($dist*$padding);
  $y  =  $y+$padding -$padding/5;
  for ($i= 0; $i<=$nx; $i++){
     $x = ($xmin+$padding)+$i*($dist*$padding);
     transform_to_image ($x, $y, $x1, $y1);
     $text = (string)round($x,0);
     imagestring ($img,$font, $x1, $y1, $text, $color);
   }
 return;
}
function drawTextYAchse($img, $x, $ymin, $ymax, $padding,
$dist, $color,$font){
 $dy =  ($ymax-$ymin)- 2.0*$padding;
// Grid-Bereich
 $ny =  $dy/($dist*$padding);
```

```
// Anzahl Gitterlinien
 $x  =  $x+$padding /5;
 for ($i= 0; $i<=$ny; $i++){
   $y = ($ymin+$padding)+$i*($dist*$padding);
   transform_to_image ($x, $y, $x1, $y1);
   $text = (string)round($y,0);
   imagestring ($img,$font, $x1, $y1, $text, $color);
 }
 return;
}
```

Es werden die Stützpunkte der Funktion markiert mit `drawPoints();`

```
function drawPoints ($img,&$x,&$y,$n,$size,$color){
for ($i=0; $i<$n; $i++){
 transform_to_image($x[$i],$y[$i],$x1,$y1);
 imagefilledellipse ($img, $x1, $y1, $size, $size, $color);
}
return;
}
```

Die Funktion `imagefilledellipse()` gibt eine Ellipse der Abmessungen `$size` an der Stelle `$x1,$y1` aus. Der Sonderfall Kreis wird erreicht durch gleiche Abmessungen `$size` in x und y.

Wir berechnen nun für das Beispiel die lineare Regression und die Polynomkoeffizienten. Auf die Darstellung der Algorithmen wird an dieser Stelle verzichtet.

Aufzurufen sind die Funktionen `calcRegression()`, `calcPolynomCoeff()`und `solveCholesky()`. Der Sourcecode liegt dokumentiert zum Download vor. Die Berechnungsergebnisse werden über `imagestring()` ausgegeben.

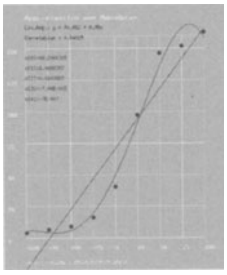
Die Zeichnung der Regressionsgraden erfolgt durch Aufruf von `drawRegression()`. Als Argumente werden u.a. der Start- und Endwert in x-Richtung und die Parameter der Geradengleichung übergeben. Sofern eine Linienstärke mit `imagesetthickness()` definiert wird, ist diese am Ende der Funktion wieder auf die Voreinstellung von 1 zurückzusetzen.

```
function drawRegression ($img, $xmin,
$xmax,$a,$b,$color,$thickness){
  imagesetthickness ( $img, $thickness);
  $y_left  = $a + $xmin *$b;
  $y_right = $a + $xmax *$b;
  transform_to_image ($xmin,  $y_left,  $x1, $y1);
```

```
  transform_to_image ($xmax,  $y_right, $x2, $y2);
  imageline($img, $x1, $y1, $x2, $y2, $color);
  imagesetthickness ( $img,1);
  return;
}
```

Das Polynom ist eine glatte Kurve, die näherungsweise durch gerade Linienabschnitte dargestellt wird. Da wir uns hier auf eine Funktion y=f(x) beschränken, können für zwei benachbarte Punkte aus den Stützstellen, x1 und x2, die Funktionswerte, y1 und y2, bestimmt werden. Der Abstand zwischen x1 und x2. ist die Schrittweite. Wird diese hinreichend dicht gewählt, kommt es zur gewünschten glatten Darstellung. Innerhalb der Funktion `drawPolynom()` werden die Grafikfunktionen `imagesetthickness()` und `imageline()` aufgerufen. Es wird wieder `transform_to_image()` benötigt. Der Funktionswert y an der Stelle x wird durch Aufruf der Funktion `calcPolynom()` berechnet. Die Polynomkoeffizienten sind in dem eindimensionalen Array `$a[]` abgelegt. Die Anzahl der Koeffizienten beträgt `grad+1`.

```
function drawPolynom ($img, $start, $end,
                      $step,$a,$grad,$color,$thickness){
//
// Grafikausgabe eines Polynoms
// $img ..... Grafikhandler
// $start ... Anfangswert x
// $end ..... Endwert x
// $step .... Schrittweite
// $a[] ..... Koeffizienten des Polynoms
// $grad .... Polynomgrad
// $color ...Farbe
// $tickness Linienstaerke
//
  imagesetthickness ($img,$thickness);
  $anz = ($end-$start)/$step;
  $x1 =  $start;
  $y1 =  calcPolynom ($x1,$a,$grad);
  transform_to_image ($x1,$y1,$xa,$ya);
  for ($i=0; $i<$anz; $i++){
    $x2 = $start + $step*$i;
    $y2 = calcPolynom($x2,$a,$grad);
    transform_to_image ($x2,$y2,$xe,$ye);
    imageline ($img,$xa,$ya,$xe,$ye,$color);
    $xa = $xe;
    $ya = $ye;
  }
```



```
  $x2 = $end;
  $y2 = calcPolynom($x2,$a,$grad);
  transform_to_image ($x2,$y2,$xe,$ye);
  imageline ($img,$xa,$ya,$xe,$ye,$color);
  imagesetthickness ( $img, 1);
  return;
}
function calcPolynom ($x,$a,$grad){
//
// Berechnet den Funktionswert y eines Polynoms
// vom Grad $grad mit den Koeffizienten $[a]
//
  $y = 0.0;
  for ($i=0; $i<=$grad; $i++){
    $y = $y+ $a[$i]* pow($x,$i);
  }  return $y;
}
```

Am Ende wurde noch das Bearbeitungsdatum erzeugt und mit `imagestring ()` ausgegeben.

```
$datum = date("d-m-Y G:i:s");
transform_to_image ($xmin+$padding,$ymin
                    +$padding/3,$x1,$y1);
imagestring   ($img,2, $x1, $y1,
     "gp polynom.php / " .$datum, $grid_color);
```

Es besteht jetzt noch die Aufgabe der Ausarbeitung einer interaktiven Schnittstelle zur Eingabe und Auswahl von Daten und zur Steuerung der Programmparameter, wie in der Abbildung 7.12 dargestellt.

Zum Hochladen von Daten kann auf dem Clientrechner eine Datei mit der Erweiterungsbezeichnung `*.dat` ausgesucht und auf den Server geladen werden. Dateien mit der Dateiendung `*.dat` können zur Polynomapproximation auf dem Server ausgewählt werden. Der Serverzugriff ist auf das zugewiesene Verzeichnis beschränkt. Die Stützpunkte eines Polynoms sind zeilenweise, durch Komma getrennt, in der Form x, y gespeichert. Alle Parameter werden über eine Dateischnittstelle ausgetauscht.

Das Formular zum Upload erhält als `action`-Attribut die Angabe des PHP-Scripts, das die Daten auf den Server schreibt, `enctype` muss in der angegebenen Form benutzt werden. Der Input-Typ `file` gestattet das Öffnen einer Dialogbox zur Auswahl einer Datei auf dem Clientrechner, vgl. Abschnitt 7.4.

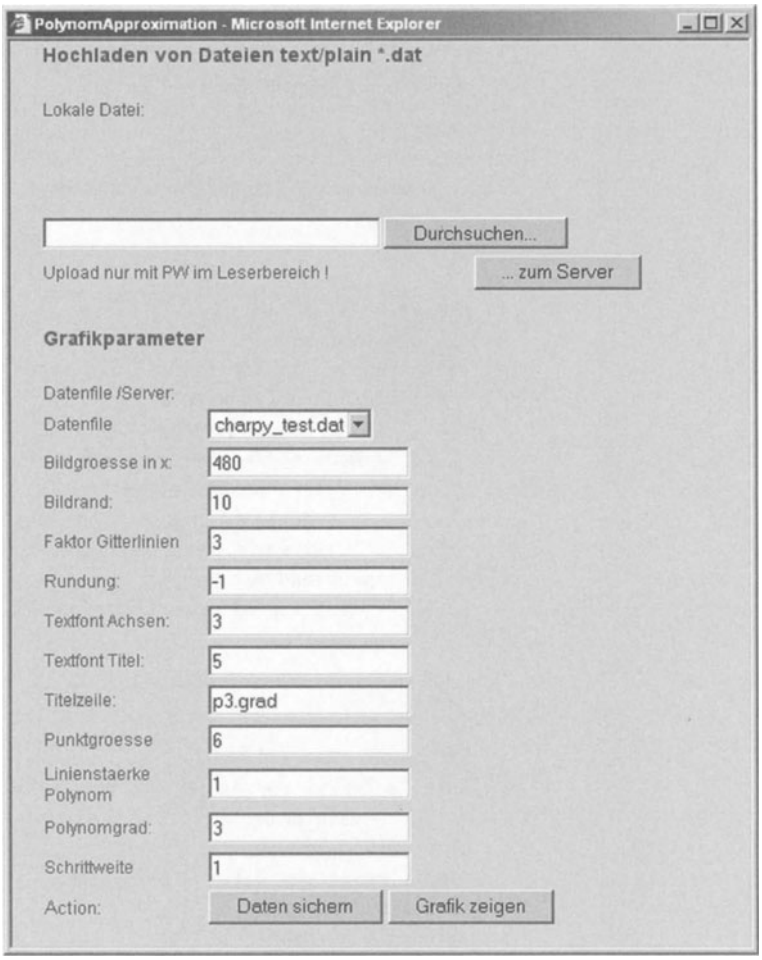


Abb. 7.12: Hochladen der Messwerte und Eingabe der Steuer parameter zur Polynomausgleichung

```
<form name="up"  ac-
tion="http://www.programmierpraktikum.de/d/php/up_file.php"
method ="post"
enctype="multipart/form-data"><p>Lokale Datei:</p>
<input type="file" name="fn" size="36" />
```

Das PHP-Script überprüft den Dateityp. Zugelassen ist hier `text/plain` oder `application/octet-stream` (wird bei Beaerbeitung mit Weaverslave erzeugt). Die Fehlermeldung wird in ein Inputfeld geschrieben. Bei einem neuen Upload-Versuch wird

auf die ursprüngliche HTML-Codierung zurückgegangen. Die HTML-Tags zur Positionierung wurden hier aus Gründen der besseren Übersicht entfernt.

```
<form name="up"
action=
"http://www.programmierpraktikum.de/d/php/
 up_file.html">
<?php
  //
  // Fehler beim Upload
  echo ("$fn_type <br>");
  if ($fn_type != "text/plain" && $fn_type !=
      "application/octet-stream"){
    echo (" <p>Server Datei</p>
            <input style=\"{color:red;}\"
               type=\"text\" name=\"fn\"
                size=\"36\" value=\"Fehler:Datei
                ist nicht text/plain\"></td>
            <p>Upload:</p
            <input type=\"submit\" value=\"... neuer Versuch\"
/>");
   }
  // Upload erfolgreich
 else {
     $fn;
     $ziel = "xxxxx Server-Verzeichnis xxxxxx"
              .$fn_name;
     copy   ($fn,$ziel);
     echo ("<p>Server Datei:</p><p>Temp:
      $fn<br>Groesse:
      $fn_size<br>Name   :
      $fn_name<br>Type   :
      $fn_type</p>
      <input type=\"text\" name=\"fn\"
         size=\"36\"  value=$fn_name>
         <p>Upload erfolgreich:</p><input
            type=\"submit\"
            value=\"... neuer Versuch\" />  ");
 }
 ?>
```

Abb. 7.12: Hochladen der Messwerte und Eingabe der Steuer parameter zur Polynomausgleichung

Das eigentliche Hochladen der Datei ist vollzogen, wenn die temporäre Datei mit der Copy-Anweisung auf das Zielverzeichnis

gespeichert wurde. Die Angabe des Zielverzeichnisses muss den Vorgaben des Providers entsprechen.

Das Datenformat für die Stützpunkte des Polynoms ist als CSV-Datei ausgelegt. Innerhalb einer Zeile befindet sich ein x-Wert und ein y-Wert getrennt durch Kommata. Das Dateiende muss unmittelbar hinter dem letzten Wert sein (kein CR, LF).Gelesen werden die Daten mittels:

```
$sDateiname=$daten_file;
 $hInput= fopen ($sDateiname, "r");       // oeffnet Datei mit
Lesezugriff
 $iZaehler  = 0;
 while ($sRecord = fgetcsv($hInput, 128)){// liest max 128
Zeichen bis
                                          // Zeilenende
   $pt_x[$iZaehler] = $sRecord[0];
   $pt_y[$iZaehler] = $sRecord[1];
   $iZaehler ++; // zaehlt belegte Arbeitsplätze
 }
 fclose($hInput);
```

Listing 7.18: Einlesen von CSV-Dateien

`$daten_file` ist der variable Dateiname. Mit `fgetcsv()` wird eine Datenzeile der maximalen Länge von 128 Zeichen gelesen, die durch Kommata getrennten Werte werden einem Array zugewiesen. Die Auswertung dieses Arrays erfolgt entsprechend der Bedeutung und Reihenfolge der Werte. `$hInput` ist der Dateihandler.

Werfen wir noch einen Blick auf die Parameterdatei und das Programm *settings.php*. Die Parameter werden mit `fscanf()` aus der Datei gelesen.

```
$sDateiname="polynom_parameter.asc";
  $hDatei = fopen ($sDateiname, "r"); // oeffnet Datei mit Le-
sezugriff
  fscanf ($hDatei,"%s",$daten_file);
  fscanf ($hDatei,"%s",$img_x);
  ...
  fscanf ($hDatei,"%s",$step);
  fclose ($hDatei);
```

Listing 7.19: Einlesen von Strings mit fscanf

`Fscanf()` liest Strings aus einer Datenzeile, hier jeweils einen. Entsprechend der Reihenfolge und Bedeutung muss auf den korrekten Variablennamen geachtet werden.

Das Dateiende wird nicht abgefragt, da die Anzahl der Datenzeilen bekannt ist. Das PHP-Script generiert die Inputfelder des Formulars und nimmt entsprechend der Variablen die Vorbesetzung vor.

Wir betrachten den Teil zur Auswahl einer Datei vom aktuellen Verzeichnis des Servers. Der HTML-Tag ist `<select>`, die Listenelemente sind mit `<option>` ausgezeichnet. Zunächst wird das Datenfile aus der Parameterdatei eingetragen, anschliessend wird in einer While-Schleife mit `readdir()` das Zielverzeichnis gelesen.

Die letzten vier Zeichen der Filereferenz werden mit `substr()` auf `.dat` überprüft. Nur diese Dateien werden in die Auswahlliste eingetragen. Das mit `opendir()` geöffnete Verzeichnis wird mit `closedir()` wieder geschlossen.

```
echo ("<select name=\"daten_file\"><option value =\""
.$daten_file ."\">" .$daten_file ."</option>");
   $verz=opendir ('.');


   while ($file = readdir ($verz)) {

     if (strlen($file) > 4){
       if( substr($file,-4)== ".dat"){
       // nur Dateien akzeptieren, deren
       Erweiterungsbezeichnung .dat ist
       echo ("<option value =\"" .$file ."\">"
       .$file ."</option>");
       }
     }
   }
   closedir($verz); echo("</select>");
```

Listing 7.20: Listenauswahl einer Datei

Das gesamte Programm besteht aus etwa 700 Programmanweisungen. *Index_poly.html* ist die Startdatei, ein Frameset, in den *upload_file.html* und *settings.php* geladen werden. Die Auswertung dieser Seiten erfolgt mit *up_load.php* und *interface.php*. Die Grafik erscheint in einem eigenen Fenster durch Aufruf von *batch_polynom.php*. Zusammen mit Beispieldaten finden Sie die Dateien zum Download in einem separaten Unterverzeichnis. Beachten Sie beim Upload von Dateien die Einräumung der entsprechenden Nutzerrechte.

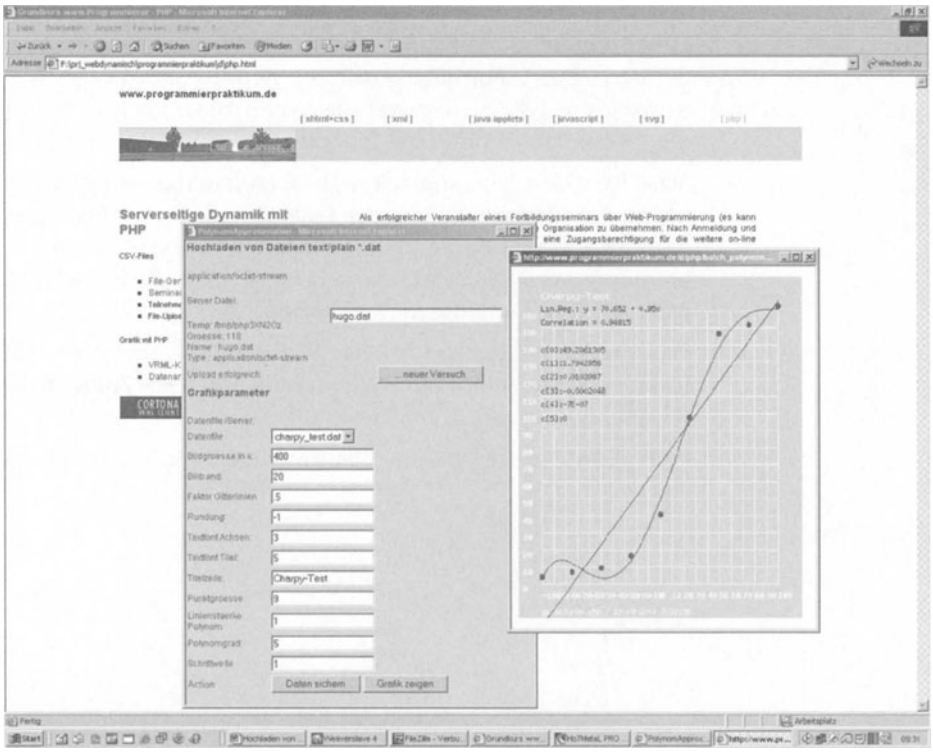


Abb. 7:13: *www.programmierpraktikum.de* - das laufende PHP-Script zur Polynomapproximation

7.6 Zusammenfassung

Nach intensiver Befassung mit dem vorliegenden Kapitel sollten Sie von PHP überzeugt sein, obwohl wir die Mächtigkeit der Sprache noch nicht ausgereizt haben.

Die Basissyntax ist von anderen Programmiersprachen nicht groß unterschiedlich. Mit den Array-Funktionen konnte PHP am Beispiel der Fußballmannschaft glänzen.

Der Aufwand zur Online-Dateiverwaltung mit exemplarischer Seminaranmeldung und Email-Bestätigung war gering, das erzielte Ergebnis schon recht brauchbar.

Mit der Zusammenstellung von VRML-Komponenten in einer Datei auf dem Server konnte das Zusammenwirken von HTML und

PHP einmal an einer ungewöhnlichen Anwendung vorgestellt werden. Das doch recht komfortable Benutzerinterface besteht aus Input-Tags, die in einem HTML-Formular eingebettet sind. Der Programmieraufwand zur Fertigstellung der Oberfläche ist gering. Das Formular-Attribut Action startet eine PHP-Anwendung, die sich zuerst die Daten zusammenstellt und dann die eigene HTML-Seite mit Einbettung der VRML-Datei schreibt.

Dass PHP den Umgang mit SQL-Datenbanken unterstützt, ist bekannt. Die Möglichkeiten der GD-Bibliothek zur Erzeugung dynamischer Grafiken ist aber noch nicht so populär. Mit der Berechnung von Ausgleichungspolynomen hat PHP auch den mathematischen Eignungstest bestanden.

Das lokale Aufsetzen eines Webservers sowie die Online-Installation bereiten keine Schwierigkeiten. Das Zeitverhalten der Programme ist gut.

PHP ist in der Tat eine im Web universell einsetzbare Scriptsprache.

Literaturhinweise, Internet-Quellen

Kapitel 1 / 2: Internet / HTML / XHTML / CSS

Klaus Dembowski

Das eigene PC-Netz, Heyne/Markt+Technik Verlag, 2002

Preston Gralla

So funktioniert das Internet , Markt+Technik Verlag, 1999

Werner Mellis (Hrsg.)

Web-Programmierung, GWV Fachverlage GmbH, Wiesbaden, 2003

Günter Pomaska

Internetpräsentation von Bauprojekten, Bauwerk-Verlag, Berlin, 2002

Peter Winkler

M+T computerlexikon, Markt + Technik Verlag, 2002

Technische Informationen zur Hardware und Software

http://www.howstuffworks.com

http://www.html-world.de

Organisationen / Administration

http://www.w3c.org

http://www.denic.de

http://www.nic.com

http://www.isoc.org

Validierung

http://validator.w3.org

http://jigsaw.w3.org/css-validator

Freeware HTML-Editor Phase V

http://www.qhaut.de

Referenz Selfhtml

http://selfhtml.org

CSS-Werkzeuge

http://www.bradsoft.com/topstyle

http://www.styleassistant.de

FTP-Client Filezilla

http://sourceforge.net/projects/filezilla

Webbrowser

http://www.mozilla.org/products/firefox

http://www.opera.com

http://www.ms.com

Blindtexte

http://www.newmediadesigner.de

Farbgestaltung

http://www.webmart.de

www.visibone.com

Meta-Tag Generator

http://www.metatag-generator.de

Internet-Provider

http://www.allink.com

Kapitel 3: XML Extensible Markup Language

Günter Born

Jetzt lerne ich XML, Markt und Technik Verlag, 2001

Natanya Pitts

XML Black Book, The Coriolis Group, 2001

Peter's HTML Editor, Freeware

http://www.iol.ie/~pxe

Validierung und XSLT-Transformation online:

http://www.cogsci.ed.ac.uk/~richard/xml-check.html

www.cherrypy.org/demo/xmlXslOnline

http://www.rgagnon.com/javadetails/java-0407.html

Kommandozeilentool zur Bearbeitung von XML-Dateien

xmlstarlet.sourceforce.net

Kapitel 4: Programmieren von Java-Applets

Judith Bishop

Java Lernen, Addison-Wesley Longman Verlag, 2001

Guido Krüger

Goto Java 2, Addison-Wesley Longman Verlag, 1999

Laura Lemnay & Rogers Cadenhead

Tech Yourself Java in 21 Days, Sams Publishing, USA, 1999

Dirk Louis, Peter Müller

Jetzt lerne ich Java, Markt + Technik Verlag, 2002

Herbert Schild

Java 2 IT Tutorial, mitp-Verlag,.Bonn, 2002

The Source for Developers

http://java.sun.com

Kapitel 5: JavaScript

Robert Rotter

JavaScript, Franzis' Verlag, Poing, 2002

Referenz und Beispiele

http://de/selfhtml.org/javascript/index.html

Lösungen, Tutorials

http://www.internet.com

http://www.meb.uni-bonn.de/java

http://www.htmlgoodies.com

Kapitel 6: Scalable Vector Graphics SVG

Iris Fibinger

SVG – Scalable Vector Graphics, Praxiswegweiser und Referenz für den neuen Vektorgrafikstandard, Markt + Technik Verlag, München, 2002

Helma Spona

SVG Webgrafiken mit XML, Verlag Moderne Industrie Buch AG & Co. KG, Landsberg, 2001

SVG-Spezifiktion / Referenz

http://www.w3c.org/svg

SVG Viewer

http://www.adobe.com/svg

SVG-Einführung, Beispiele

http://www.scale-a-vector.de

http://www.datenverdrahten.de

Kartographie und SVG

http://www.carto.net

CAD-Converter

http://www.firestudio.com

Bitmap to SVG Converter

http://www.svgfactory.com

Kapitel 7: Serverseitige Dynamik mit PHP

Sven Letzel, Robert Gacki

MySQL und PHP, Markt und Technik Verlag, 2001

Kai Seidler, Kay Vogelsang

Apache für Dummies, Mitp-Verlag, Bonn, 2002

Dieter Staas

PHP4, Franzis' Verlags GmbH, Poing, 2002

Susanne Wigard

PHP 4 – Das Einsteigerseminar, Verlag Moderne Industrie Buch AG & Co KG, Landsberg, 2001

Werkzeuge, Downloads

http://www.apache.org

http://www.php.net

http://www.weaverslave.ws

http://www.apachefriends.org

PHP-Referenz

http://www.selfphp.de

Grafikbibliothek – Diagramme

http://www.aditus.nu/jpgraph/jpdownload.php

Sachwortverzeichnis

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z