# Visualisierung von Informationsräumen

## Dissertationsschrift

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

vorgelegt an der

Fakultät für Informatik und Automatisierung
Institut für Praktische Informatik und Medieninformatik
der Technischen Universität Ilmenau

von

Dipl.-Inf. Martha Argentina Barberena Najarro

Berichterstatter:

1. Gutachter: Univ.-Prof. Dr.-Ing. habil. D. Reschke, TU Ilmenau
2. Gutachter: Prof. Dr.-Ing G. Schade, FH Erfurt
3. Gutachter: Prof. Dr.-Ing R. Böse, FH Schmalkalden

Tag der Einreichung: 06.06.2003

Tag der öffentlichen wissenschaftlichen Aussprache: 16.12.2003

# Vorwort

Als ich diese Arbeit begann, ahnte ich nicht, welches Ausmaß sie annehmen würde und wieviele Fragen und Detail-Probleme für ca. 200 Seiten beantwortet und aufbereitet werden mußten. An dieser Stelle sei deshalb allen ganz herzlich gedankt, die mich in den vergangenen Jahren unterstützt haben.

Mein besonderer Dank gilt meinem Doktorvater, Herrn Prof. Dr.-Ing. habil. Dietrich Reschke für die Unterstützung und das Vertrauen, mit dem er mich während der Arbeit begleitet hat. In vielen Diskussionen mit ihm erhielt ich wertvolle Hinweise, die mir sehr hilfreich waren bei der Anfertigung dieser Arbeit.

Über die Tatsache, daß ich Frau Prof. Dr. Gabriele Schade und Herrn Prof. Dr. Ralf Böse als Gutachter gewinnen konnte, habe ich mich sehr gefreut und danke Ihnen für ihre sofortige Bereitschaft.

Bei meinen Kolleginnen und Kollegen bedanke ich mich für die angenehme Arbeitsatmosphäre, die anregenden Gespräche und die gemeinsamen Unternehmungen.

Ganz besonders danke ich meinem Mann, Rainer Barth. Sein Verständnis und seine Hilfe bei Problemen jeglicher Art haben es mir ermöglicht, diese Arbeit fertigzustellen.

Ein unschätzbarer emotionaler Halt war mir in den vergangenen Jahren meine Stieftochter Juliane Barth.

Ilmenau, den 01.06.2003

Martha Argentina Barberena Najarro

## Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer Prüfungsbehörde vorgelegt.

Ich bin darauf hingewiesen worden, dass die Unrichtigkeit der vorstehenden Erklärung als Täuschungsversuch angesehen wird und den erfolglosen Abbruch des Promotionsverfahrens zur Folge hat.

(Ort, Datum) (Martha Argentina Barberena Najarro)

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

## 1.1 Motivation und Umfeld

Das *World Wide Web* (WWW) ist ein Teildienst des Internet. Es stellt eine einheitliche grafische Oberfläche für die Darstellung von Informationen zur Verfügung und bietet Schnittstellen für verschiedene Internetdienste wie FTP, TelNet, Gopher, Email usw. zur Verfügung. [Pot98]. Das Auffinden des *gewünschten* Webdokumentes bzw. der *richtigen* Information im WWW kann sich als schwierig und mühsam erweisen. Die Lage wird sich weiter verschlechtern, da die Menge der Informationen und Informationsanbieter ständig wächst. Ein wichtiges Hilfsmittel bei der Suche nach relevanten Informationen sind die Internet-Suchdienste (Suchmaschinen). Im Jahr 1994 wurden die ersten Internet-Suchmaschinen vorgestellt, seit dem hat ihre Anzahl und Vielfalt kontinuierlich zugenommen. Die gegenwärtig bedeutendsten sind öffentlich über das WWW zugänglich und arbeiten mit automatisch oder manuell erzeugten Indexen. Innerhalb dieser Gruppen lassen sich wiederum globale und regionale Dienste, generelle und fachspezifische sowie kommerzielle bzw. von öffentlichen Organisationen entwickelte und betriebene unterscheiden. Auf die Suchmaschinen-Technologie wird im Kapitel 2 genauer eingegangen.
Die verbreitetsten und größten Suchwerkzeuge sind den roboterbasierten, globalen Suchmaschinen zuzurechnen. Sie decken die meisten Ressourcen ab und bieten in der Regel den größten Umfang an Funktionalität im Bereich des Information Retrieval an. Damit wird ein gezielter und umfassender Zugriff auf gesuchte Ressourcen möglich. Ein Teil der roboterbasierten Suchmaschinen stellt Retrievalfunktionen zur Verfügung, die iterative Suchprozesse, welche schrittweise zu Verfeinerungen der Suche führen, unterstützen [Bek97].
Fast alle Suchmaschinen bieten mehr oder weniger ausführliche Dokumentationen ihrer Suchfunktionen an. Die meisten Suchmaschinen bieten die Möglichkeit, zwischen einem einfachen und einem erweiterten Suchmodus mit zusätzlichen Operationen zu wählen, damit die Suche effektiver wird und zu einem gezielten Ergebnis führt.
Trotz der Fortschritte bei der Technologie der Suchmaschinen ist zu sehen, daß die bisherige Form der Präsentation der Suchergebnisse den Ansprüchen an eine optimale Suche nach relevanten Informationen nicht gerecht wird. Obwohl die inhaltlich orientierte Suche nach Informationen mit Suchmaschinen auf einfache Weise möglich ist und schnell durchgeführt wird, tritt eine Reihe von Problemen auf, die noch nicht gelöst sind.
Wird eine Suche durchgeführt, so wird das Ergebnis dieser Suche dem Benutzer in einer nach Relevanz geordneten Liste der gefundenen Treffer präsentiert. Diese Liste ergibt sich aus einer umfangreichen Ranking-Berechnung, bei der die „Nähe“ eines Dokumentes zu einem Suchbegriff aber verloren geht. Dies ist nachteilig, wenn mit mehreren verknüpften Begriffen gesucht worden ist. Für diesen Fall existieren keine Hilfsmittel zur Extraktion der wirklich relevanten Dokumente aus der Ergebnisliste. Wenn Optionen zur Verfeinerung der Suche vorhanden sind, dann müssen diese *vor*

dem Suchprozeß eingestellt werden. Eine Unterstützung zur Evaluierung und Weiterverarbeitung der Ergebnisliste gibt es nicht.
Ein weiteres Problem ist die Art und Weise, in der die Suchergebnisse dargestellt werden. Es ist eine Text-Liste, in der sich der Benutzer von Dokument zu Dokument vorwärts bewegt. Dabei beinhalten Links weitere Links, der Benutzer „springt“ zu anderen Dokumenten. Von diesen weiß er nicht, ob sie ebenfalls von der Suchmaschine gefunden wurden und in einer weiter hinten liegenden Position der Trefferliste verzeichnet sind, oder ob er auf ein Dokument gestoßen ist, das für ihn zwar wichtig ist, aber vom verwendeten Suchdienst nicht gefunden wurde (nicht erreichbar, noch nicht angemeldet, ...). Dabei besteht die Gefahr, daß angesichts der Länge der Trefferliste (von der immer nur ein kleiner Ausschnitt sichtbar ist), und des Begutachtens vieler Dokumente der Überblick verloren geht. Nach einer bestimmten Zeit weiß der Benutzer in Bezug auf die Ergebnisliste nicht mehr:

- wo er sich gerade befindet,
- auf welchem Weg er dorthin gekommen ist,
- wo er sich vorher befunden hat,
- wie er zu seiner vorherigen Position zurückkehren kann.

Diese Situation wird in der Literatur als „Problem der Desorientierung“ bezeichnet [Mar93]. Es ist typisch für die Navigation durch eine Liste von Dokumenten, da die Navigation nicht in einem wirklichen Raum stattfindet. Durch die Dokumentverweise gibt es keine Hilfe zur Orientierungsrichtung, und nicht in jedem Fall führt vom Ziel eines Dokumentverweises ein Weg zurück zum Ausgangspunkt. Der Benutzer ist damit nicht mehr in der Lage, die für ihn relevanten Informationen durch *exploratives Browsing* (Navigieren anhand vorgegebener Links) aufzufinden und zu nutzen. Durch diese Umstände verliert er schnell die Übersicht bei der Handhabung der Ergebnismenge.
Im Vordergrund steht die Forderung nach neuen Techniken zur Informationsaufbereitung und Recherche – auch, da inzwischen technische Fortschritte bei grafischer Hardware und der Verarbeitung großer Datenmengen gemacht wurden. Umfangreiche und komplexe Daten können anschaulich mit Hilfe einer computergestützten Visualisierung dargestellt werden. Eine grafische Informationsdarstellung ermöglicht dem Betrachter, verschiedene Aspekte und Beziehungen mühelos zu erfassen und aufzunehmen. Im Gegensatz dazu ist die listenorientierte Darstellung von Suchmaschinen eine unzureichende Unterstützung. Der Umgang damit erweist sich als unübersichtlich, unhandlich und zeitintensiv.
In [Sch99] wird die grafische Informationsdarstellung als eine Methode dargestellt, die sowohl eine einheitliche Sicht auf verschiedene Informationsquellen eröffnet, als auch die Möglichkeit bietet, große Daten- und Informationsmengen in ihrer Gesamtheit zu analysieren und zu bewerten. Das Erfassen einer grafische Darstellung ist eine besondere menschliche Fähigkeit. Nur mit einer visuellen Darstellung gelingt es, so viele verschiedene Aspekte und Beziehungen auf einen Blick sichtbar zu machen [Dä99]. Durch die Visualisierung von Informationen wird der Benutzer beim Erkennen von Inhalten, z.B. beim Sortieren von Ergebnissen oder Auffinden von Unterscheidungsmerkmalen, entlastet [Sch99].

## 1.2 Bisherige Entwicklung

Der vermehrte Einsatz von Computern im täglichen Gebrauch war eine der wichtigsten Entwicklungen im Laufe der letzten Jahrzehnte. Die zu verarbeitende Datenmenge allgemein, und die Erzeugungsrate von neuen Daten sind extrem angestiegen. Als logische Konsequenz entwickelte

sich die Visualisierung zu einer Schlüsseltechnologie zur Aufbereitung und Analyse von großen Datenmengen und Prozessen. Die dabei gemachten Erfahrungen können in gewisser Weise auf die Analyse von Suchergebnissen übertragen werden. Einige Meilensteine dieser Entwicklung werden hier aufgeführt, die Problematik an sich wird in den Kapiteln 2 und 3 ausführlich behandelt.

- Am Anfang der 90er Jahre wurden innerhalb des Prototyp-Systems *Information Visualizer* diverse interessante Visualisierungstechniken entwickelt. Am bekanntesten sind Cone Tree und die Perspective Wall [Rob93, Rob91b]. Diese Entwicklung fand bei Xerox Parc statt, dort wurde auch der Begriff *Informationsvisualisierung* geprägt. Die genannten Verfahren werden in den Abschnitten 3.2.1 und 3.2.3 analysiert.

- Ebenfalls Anfang der 90er Jahre wurde auch in Deutschland eine Reihe von Projekten begonnen. Ein Beispiel dafür ist der Prototyp *LyberWorld* der GMD (Gesellschaft für Mathematik und Datenverarbeitung), welcher die explorative Suche und Navigation in Dokumentenbeständen durch eine dreidimensionale interaktive Benutzungsschnittstelle realisiert [Hem93a]. Diesem System ist der Abschnitt 3.2.5 gewidmet.

- Ein weiteres Projekt ist der Prototyp *KOAN* (**KO**ntext**AN**alysator) der SIEMENS AG, wobei ein Verfahren zur Visualisierung von Informationsbeständen entwickelt wurde.

- Bereits 1991 wurde in der Universität von Minnesota die visuelle Oberfläche *GopherVR* für den Informationsdienst „Gopher“ entwickelt.

- Im Jahr 1992 hat die Firma Silicon Graphics das Konzept einer abstrakten Informationslandschaft mit dem *File System Navigator* (FSN) entwickelt. Der File System Navigator erzeugt aus hierarchischen Verzeichnisstrukturen eine dreidimensionale Darstellung.

- Nach den ersten Fortschritten des World Wide Web im Jahr 1994 wurden erste Applikationen mit Navigationsoberflächen für die Informationsvisualisierung veröffentlicht – Applikationen zur Web-Visualisierung. Eines davon ist das System *Hyperspace* der „School of Computer Sciencie“ der University of Birmingham. Dieses System stellt die Hyperlinksstruktur von Teilbereichen des Web dar.

- Das System *Hyperbolic Tree* ist einer der wenigen Prototypen, die tatsächlich Einsatz in der Informationsvisualisierung gefunden haben. Hyperbolic Tree wurde ebenfalls vom Xerox Palo Alto Research Center entwickelt und im Jahr 1995 vorgestellt. Dieser Prototyp und seine Anwendungsgebiete werden im Abschnitt 3.2.2 bechrieben.

## 1.3 Aufgabenstellung

Diese und andere Ansätze wurden bei ihrer Entwicklung in erster Linie für die *reine Darstellung von Suchergebnissen* entwickelt und enthalten daher auch entsprechend hoch entwickelte und komplexe Visualisierungsverfahren. Dabei wird versucht, strukturelle Beziehungen und Zusammenhänge aufzuzeigen.
In dieser Arbeit wird ein neuer Ansatz vorgestellt, der als Ziel haben soll, eine Architektur zu entwerfen, die das schnelle und zielgerichtete Auffinden von relevanten Informationen mittels grafischer Darstellung, Filtermechanismen und interaktivem Management der Dokumente gestattet. Ausgangspunkt ist auch dabei die von einer beliebigen Suchmaschine oder Wissensbasis ermittelte Menge von Dokumenten, die den angegeben Suchbegriffen und Kriterien entspricht. Die wichtigsten Komponenten einer solchen Architektur sind:

- geeignete grafische Darstellungen, die auf intuitive Weise einen guten Überblick des Suchergebnisses vermitteln und Interaktionen zur Analyse ermöglichen,
- eine wohldefinierte Schnittstelle zum Informationstransfer von der Suchmaschine zum Visualisierungstool,
- Filter, die eine effiziente Reduktion der großen Dokumentmenge erlauben,
- eine Komponente zur grafischen Darstellung der verbliebenen Dokumentmenge, wobei die Darstellungsverfahren über eine definierte Schnittstelle austauschbar sind,
- Navigation im durch die Suchbegriffe aufgespannten Informationsraum,
- Mechanismen zur direkten Interaktion mit den Dokumenten über die grafische Darstellung,
- Möglichkeiten zum Auffinden von Dokumenten vermutlich ähnlichen Inhalts,
- Datenstrukturen zum Management der Dokumente (z. B. Markieren von interessanten Dokumenten, Löschen irrelevanter Dokumente, ...)
- Schnittstellen zum Laden/Speichern von Zwischenergebnissen der laufenden Recherche, bzw. zum Austausch der Ergebnisse

Das Ziel besteht also nicht nur in einer visuell erfaßbaren bedeutungstragenden Form der grafischen Darstellung von Informationen, sondern in einem System zur Informationsaufbereitung, das über offene Schnittstellen nach außen verfügt, erweiterbar, anpaßbar und auch plattformunabhängig ist. Dieser Ansatz schließt auch die Auswertung vorhandener (aber zu wenig verwendeter) HTML-Features ein – den Meta-Tags. So können Deskriptoren, die im Bibliothekswesen erprobt sind, zur Klassifizierung von Dokumenten herangezogen werden. Die Verwendung solcher Möglichkeiten könnte vor allem im wissenschaftlichen Bereich zum Hervorheben von Publikation beitragen.
Begleitend dazu erfolgt die Implementierung eines Prototypen für die Visualisierung von Informationsräumen. Dieser Prototyp hat zum Ziel, daß der Benutzer eine Menge von relevanten Informationen erhält, die er gleichzeitig handhaben kann. Dies kann mit Hilfe der dreidimensionalen grafischen Darstellung der Informationen realisiert werden. Sie erlaubt, relevante Dokumente schnell visuell zu erfassen und herauszufiltern und dabei gleichzeitig nicht die Übersicht zu verlieren. Außerdem unterstützt sie erheblich die Möglichkeit der Navigation innerhalb eines Informationsraumes und die Suche nach ganz bestimmten Dokumentinhalten.

## 1.4 Der Aufbau dieser Arbeit

Die vorliegende Arbeit ist in sieben Kapitel gegliedert. Diese Kapitel bauen inhaltlich aufeinander auf.

- Das folgende Kapitel 2 wird in die Informationsgewinnung im Hypertext-Informationssystem WWW einführen. Dazu werden verschiedene Arten von Suchmaschinen kurz beschrieben und die Probleme, die bei der Nutzung von Suchmaschinen-Ergebnissen auftreten, erläutert. Der Begriff „Visualisierung“ wird definiert und die technischen Hintergründe erläutert. Die Adaption von Prozessen aus der wissenschaftlich-technischen Visualiserung wird als Ansatz zur Lösung der genannten Probleme begründet.
- Das Kapitel 3 präsentiert und beschreibt eine Anzahl von innovativen Ansätzen zur Informationsvisualisierung als Ergebnis einer Literaturrecherche. Es wird ein Katalog von Kriterien

zur Beurteilung der Visualisierungsverfahren hinsichtlich ihrer Anwendungsmöglichkeiten aufgestellt und jedes einzelne Verfahren dementsprechend evaluiert. Daraus werden die derzeit offenen Probleme extrahiert.

- Im Kapitel 4 werden die Phasen der objektorientierten Softwareentwicklung und ihre Methoden kurz beschrieben. Insbesondere wird dabei auf die *Objektorientierte Analyse* eingegangen. Anschließend wird die grundlegende Anforderungsanalyse zum Visualisierungssystem durchgeführt.

- Das Kapitel 5 bildet mit Kapitel 4 den eigentlichen Kern dieser Arbeit. Es wird auf das *Objektorientierte Design* eingegangen und das Konzept für ein System zur Informations-Visualisierung vorgestellt.

- Der vorgestellte Entwurf wurde fortlaufend durch die Implementation eines Prototyp verifiziert, der im Kapitel 6 beschrieben ist. Im Zusammenhang damit werden die Anforderungen an das zu verwendende Grafiksystem definiert und die beiden Grafik-Schnittstellen *OpenGL* und *Java3D* sowie die Beschreibungsprache *VRML* hinsichtlich ihres Einsatzes im Visualisierungssystem untersucht.

- Das Kapitel 7 faßt die Arbeit und ihre Ergebnisse kurz zusammen. Es wird ein Ausblick auf mögliche weitere Entwicklungen gegeben und Vorschläge zur einer effizienteren Extraktion von Informationen gemacht.

Am Schluß der Arbeit befinden sich Anhänge, das Literaturverzeichnis, das Abbildungs- und Tabellenverzeichnis, ein Abkürzungsverzeichnis sowie ein Index zum Nachschlagen von wichtigen Stichwörtern.

# Kapitel 2

# Visualisierung von Informationen

## 2.1 Einführung

Nach einer kurzen Einführung in die Problemstellung wird in diesem Kapitel zunächst auf das WWW-Informationssystem eingegagen. Anschließend werden die verschiedenen Arten von Suchmaschinen erläutert. Die Betrachtung der Suchwerkzeuge ist wichtig, da sie für die Visualisierung von Informationen eine große Rolle spielen. Weiterhin werden die Probleme aufgeführt, die bei der Darstellung der Ergebnisse von Suchanfragen auftreten und es wird eine Lösungsalternative vorgestellt.

## 2.2 Das Hypertext-Informationssystem WWW

Das WWW (**W**orld **W**ide **W**eb)[1] ist einer von vielen Internetdiensten. Das WWW ist ein Informationssystem, das einen bequemen Zugriff auf Informationen, die auf vielen verschiedenen Computern gespeichert sind, in Form von Hypertext- und Hypermedia-Links ermöglicht [Sch94]. Der Zugriff erfolgt nach dem Client-Server-Prinzip über das Internet mit dem Protokoll *HTTP* (**H**yper**T**ext **T**ransfer **P**rotocol). Text-Informationen werden auf den WWW-Servern in der Form von HTML-Files, WML-Files und XML-Files gespeichert. Außerdem können auch Bilder, Töne, Videos und beliebige sonstige Files über das WWW übertragen werden, und es können Programme gestartet und Benutzer-Eingaben verarbeitet werden. Bevor weiter auf dieses Thema eingegangen wird, soll zunächst der Begriff *Dokument* definiert werden. Aus der Literatur [Rö01] stammt dazu die folgende Definition:

> Ein Dokument ist ein Behälter für Informationen, die dem Anwender immer als eine vollständige Einheit präsentiert werden. Dokumente können aus Papier, gescannten Papieren, elektronischen Textdokumenten, Ausdrucken usw. bestehen. Alle Informationen welche sich in eine digitale Form bringen lassen, können in Form von Dokumenten verwaltet werden. Dazu gehören auch Videos, Audios, Fotos, Röntgenbilder, WEB-Seiten, usw.

Das WWW, als „multimediales Internet-Informationssystem“, bietet eine Reihe von Vorteilen gegenüber anderen Internetdiensten. Die Arbeit mit dem WWW erfolgt über sogenannte *Browser*. Die zur Zeit bekanntesten Browser sind der *Microsoft Internet Explorer* und der *Netscape Navigator*. Sie [Zol96] sind spezielle grafische Benutzeroberflächen, die die Navigation durch das Internet ermöglichen. Mit ihrer Hilfe können Dokumente angefordert und angezeigt werden.

[1] WWW wurde am Europäischen Kernforschungszentrum CERN in Genf entwickelt und wird vom W3-Consortium weiter entwickelt.

Leider ist die Suche nach Dokumenten mittels eines Browser nicht leicht, da das Internet keine zentrale Organisationsstruktur aufweist. Das unkontrollierte Angebot von Informationen macht es problematisch, Dokumente zu einem bestimmten Thema zu finden. Es gibt keine zentrale Koordination oder Kontrolle was, von wem, und in welcher Form im Web veröffentlicht wird. Auch den Dokumenten selbst mangelt es häufig an Strukturierung. Deshalb wurden verschiedene Suchwerkzeuge darauf spezializiert, beim Auffinden von Dokumenten behilflich zu sein. Diese Suchwerkzeuge werden hier kurz erläutert.

## 2.3 Die Suchwerkzeuge

Die Suchwerkzeuge erschließen die Internetressourcen mit verschiedenen Methoden. Allen gemeinsam ist die Benutzung in einem WWW-Browser über die Eingabe von Suchwörtern und die Zusammenstellung von Ergebnislisten.

### 2.3.1 Thematisierte Suchmaschinen (Themenkatalog)

Themenkataloge [Phi00, Bek01] sind von menschlichen Redakteuren zusammengestellte Sammlungen von Hyperlinks, welche in einer hierarchischen Struktur oder in Kategorien und Unterkategorien aufgeteilt sind. Es gibt geografisch, chronologisch und sachlich geordnete (systematische) Verzeichnisse. Außerdem können diese Verzeichnisse national, regional oder fachlich begrenzt sein. An der Wurzel befindet sich eine erste grobe Themenauswahl, mit der der Benutzer durch Verfeinerung der Themenauswahl schnell zu der Liste von Seiten übergehen kann, die für seine Fragestellung am interessanten sind.
Mit Hilfe der Themenkataloge wird die Suche über Stichwörter angeboten. Die Suche basiert nicht auf Dokumenten im Volltext. Suchbar sind vielmehr lediglich die Inhalte des Titels und der Dokumentadresse und die ersten Wörter der aufgenommenen WWW-Seiten. Bei vielen Themenkatalogen werden darüber hinaus nur die Hauptseiten von Informationsanbietern aufgenommen. Außerdem sind die Suchmöglichkeiten zumeist wesentlich eingeschränkter als bei robotbasierten Suchmaschinen.

### 2.3.2 Robotbasierte Suchmaschinen

Robotbasierte Suchmaschinen erschließen Internetressourcen maschinell mittels automatisch arbeitender Programme (robots oder spider), indem sie Indizes der durchsuchten Seiten in Form von Datenbanken anlegen [Phi00, Bek01]. So wird die Suche in der Datenbank und nicht direkt im Internet durchgeführt.

### 2.3.3 Meta-Suchmaschinen

Bei Meta-Suchmaschinen(Multi-Search Engine) handelt es sich eigentlich nicht um Suchmaschinen, sondern um Benutzeroberflächen, welche hinter einer einheitlichen Eingabemaske andere Suchdienste wie Themenkataloge oder Suchmaschinen gleichzeitig befragen und die empfangenen Ergebnisse aufarbeiten und gleichartig darstellen [Phi00, Bek01].

## 2.4 Probleme bei der Ergebnisdarstellung von Suchmaschinen

Der Erfolg des „Internet“ entsteht vor allem durch die Verbesserung und Vereinfachung des Zugriffs auf das Internet – auch für den Privatnutzer. Das Medium Internet existiert aber schon viel länger, der Zugriff war jedoch zunächst insbesondere wissenschaftlichen Einrichtungen vorbehalten. Wenn heute auch Anwendungen wie Home-Banking, Internet-Shopping und Unterhaltung im Mittelpunkt der Aufmerksamkeit vieler Benutzer stehen, bringt das Internet im wissenschaftlichen Bereich immer noch den größten Nutzeffekt. Täglich werden weltweit tausende wissenschaftliche Abhandlungen, Forschungsberichte, Artikel für Konferenzen oder Bücher verfaßt. Kein Mensch ist in der Lage, diesen Informationszuwachs nur mit Hilfe von Zeitschriften, Büchern oder persönlichen Kontakten zu verfolgen. Dazu kommt, daß auch die aktuellsten Printmedien eine gewisse Herstellungszeit benötigen und bei einigen Forschungsgebieten eine Information tatsächlich schon veraltet sein kann, wenn sie beim Empfänger eintrifft.
Wie gelangt man zu einer bestimmten Information, zum Beispiel zu einer wissenschaftlichen Abhandlung? Auf konventionelle Weise geht man in eine Bibliothek und benutzt einen Katalog, in dem alle verfügbaren Bücher und Zeitschriften mit „Schlüsselwörtern“ verzeichnet sind. Sie sollen den Inhalt der Veröffentlichung charakterisieren. Mit ihrer Hilfe sucht man die Publikationen, die interessant sein könnten. Obwohl die Kataloge mittlerweile meistens auf CD-ROM verfügbar sind, bekommt man die Veröffentlichungen nur, wenn sie auch darin verzeichnet sind. Das schränkt die Aktualität ein, außerdem ist der Zugriff auf Bücher oder Zeitschriften oft mit Wartezeiten verbunden.
Die Suche im Internet ist damit im Prinzip vergleichbar. Für diesen Zweck werden die *Suchmaschinen* benutzt. Obwohl es bereits viele Arten von Suchwerkzeugen im Internet gibt, mit denen man brauchbare Ergebnisse erzielen kann, ist die Suche nach Internetressourcen nicht immer ganz einfach. Ein Defizit besteht in der Schwierigkeit, relevante Informationen aus einer riesigen Menge von weniger relevanten Informationen herauszufinden. Dazu sind nicht nur verbesserte Suchwerkzeuge nötig, sondern auch differenzierte Angaben über die Internet-Dokumente selbst.
Nachdem über einen Dialog die Schlüsselwörter eingegeben werden können, nach denen gesucht werden soll, werden *alle* Dokumente (Artikel, Bücher, Vorträge, ...) ausgegeben, die im gesamten Internet unter den eingegebenen Schlüsselwörtern auf WWW-Seiten verzeichnet sind.
Obwohl die Suchmaschinen einer ständigen Verbesserung der Funktionen und Suchoptionen unterliegen, ist das Ergebnis bei allen diesen Tools in einer sequentiellen Liste von Dokumenttiteln dargestellt, wobei die Dokumente als *Verweise (Links)* in der Liste verzeichnet sind. Das bedeutet, daß ein Dokument durch einfaches Anklicken angefordert und betrachtet werden kann.
Wie man sieht, geht die Suche nach Dokumenten wesentlich schneller als mit einem Katalog in einer Bibliothek. Ein Katalog enthält nur eine Teilmenge aller Veröffentlichungen, so erhält man bei der Suche eine überschaubare Anzahl von Büchern und Publikationen. Bei der Suche im Internet ist diese Zahl nach oben offen. Es ist keine Seltenheit, daß die Liste, die die Suchmaschine ausgibt, mehrere hundert oder sogar tausend Einträge enthält, so daß eine Darstellung der gesamten Ergebnismenge in der Regel auf mehrere Seiten verteilt werden muß. Durch diesen Umstand verliert der Benutzer schnell den Überblick über die Ergebnismenge.
Viele Retrievalsysteme treffen eine Auswahl von besonders relevanten Dokumenten, um die Größe der Ergebnismenge zu begrenzen. Die Reihenfolge in der Titelliste kann aber bestenfalls einen schwachen Anhaltspunkt über das Relevanzmaß bestimmter Dokumente liefern. Die Suchwerkzeuge zeigen nicht, wie die Dokumente zueinander in semantischen Beziehungen stehen. Durch die Listendarstellung werden keine weiteren Hilfen zur Verbesserung der Anfrage angeboten.
Die derzeit bestehenden Probleme seien in drei Punkten zusammengefaßt:

- Bei der Suche nach Informationen hat es gigantische Fortschritte gegeben - aber nur rein technisch gesehen. Bei der Vorgehensweise hat sich nichts geändert, der Ablauf ist immer

noch analog zum Arbeiten in der Bibliothek: Mit Hilfe von Schlüsselwörtern wird eine Liste von Veröffentlichungen zusammengestellt, die dann mühsam durchgearbeitet werden muß. Der Vorgang läuft lediglich wesentlich schneller ab.

- Genau wie nach der Suche in einem Katalog weiß man nicht, ob die gefundenen Dokumente auch wirklich genau die Informationen enthalten, die man braucht, da es keine festen und bindenden Regeln zur Vergabe von Schlüsselwörtern gibt.
- Weiterhin ist es bei Verwendung mehrerer Schlüsselwörter schwierig, der Textliste die Beziehung der gefundenen Dokumente zu den einzelnen Schlüsselwörtern zu entnehmen. Das heißt, der Benutzer kann kaum einschätzen, wie gut die gefundenen Dokumente wirklich seinen Anforderungen entsprechen.

Es wird keine Chance gesehen, die Situation auf Basis einer rein textuellen Ausgabe verbessern zu können. Es ist nicht möglich, die bei einer Suche anfallende Menge von Text gleichzeitig, lesbar strukturiert und übersichtlich auf einem Bildschirm darzustellen. Daher müssen andere Wege und Mittel gesucht werden, eines davon ist die grafische Darstellung von Suchergebnissen.

## 2.5 Die Visualisierung von Suchergebnissen als Lösung

### 2.5.1 Grundlagen

Es ist notwendig, ein Hilfsmittel zur Verfügung zu stellen, das den Benutzer dabei unterstützen kann, eine Vorstellung von der Informationsstruktur zu entwickeln, also den Informationsraum, den er von seinem Suchdienst als Ergebnis erhält, zu verstehen und für sich nutzbar zu machen. Da – wie oben beschrieben – eine große Menge von Informationen (die Menge der gefundenen Dokumente) aufbereitet werden muß, ist es notwendig die „Bandbreite“ bei der Aufnahme von Informationen gegenüber der Textliste wesentlich zu erhöhen. Dies ist mit grafischen Darstellungen möglich. Mehrere Gedanken haben zur Idee der Visualisierung von Suchergebnissen als Ersatz (oder zusätzliches Hilfsmittel) der unübersichtlichen Textlisten geführt:

- der Mensch nimmt die meisten Informationen über das Auge auf, indem er Formen und Farben erkennt,
- Visualisierungstechniken sind in vielen Bereichen zu einem üblichen Arbeitsmittel geworden,
- mittlerweile verfügt selbst ein aktueller Personal Computer über genügend Performance um sogar dreidimensionale Darstellungen zu generieren, sowie Interaktionen zuzulassen,
- der Umgang mit virtuellen dreidimensionalen Umgebungen ist dem heutigen durchschnittlichen Benutzer nicht mehr fremd, sondern aus diversen Anwendungen (auch aus Computerspielen) bekannt.

Das Auffinden von Dokumenten mit Hilfe von Suchmaschinen ist ein schnelles, praktisches und bewährtes Verfahren, das sicher auch beibehalten wird. Der Schwerpunkt dieser Arbeit liegt in der Entwicklung von Methoden zur Überführung der Suchergebnisse in geeignete grafische Darstellungen. Es soll gezeigt werden, daß eine Visualisierung der erhaltenen Dokumentenmenge dem Benutzer helfen kann, die wirklich interessanten Dokumente wesentlich schneller und sicherer zu finden als in der konventionellen textbasierten Liste.
Die Visualisierung von Informationsräumen als Ansatz zur Lösung der beschriebenen Probleme bietet die Möglichkeit, große Daten- und Informationsmengen in ihrer Gesamtheit zu analysieren, sowie Strukturen und Kontexte zu erfassen und zu verstehen:

- Der Benutzer erhält einen Überblick über alle gefundenen Dokumente. Das ist bei einer Textliste nicht möglich, da immer nur ein kleiner Ausschnitt auf dem Bildschirmfenster Platz findet.
- Er kann auf einen Blick erkennen, wie gut die Dokumente zu den eingegebenen Schüsselwörtern passen.
- Er kann durch Interaktion sorgfältig eine Teilmenge der interessanten Dokumente finden und diese dann anfordern.
- Er sieht, wie häufig seine gewählten Schlüsselwörter auftreten und kann diese variieren und sofort die Auswirkung erkennen. Auf diese Weise kann er Schritt für Schritt seine Suchanfrage verfeinern
- Die Gestaltung der Benutzungsoberfläche eines Informations Retrevial System spielt eine wichtige Rolle für die Erfüllung der Wünsche des Benutzers und die Güte eines Anfrageergebnisses.

Die dreidimensionale Visualisierung ist bereits aus dem technischen Bereich bekannt und wird vielfach eingesetzt (Medizin: Computertomografie; Entwurf: CAD; Chemie: Molekülmodellierung; ...) Dort haben die dargestellten Daten aber eine Herkunft, die bereits eine dreidimensionale Analogie hat – die *nur noch* grafisch umgesetzt werden muß. Die Struktur der gefundenen Dokumente hat jedoch keinen räumlichen Charakter. Hier müssen Mittel und Methoden entwickelt werden, diese abstrakte Struktur für das menschliche Vorstellungsvermögen aufzubereiten – das ist ein Ziel dieser Arbeit.

### 2.5.2 Begriffsbestimmung: „Visualisierung“ und „Information“

Zum Begriff „Visualisierung“ existieren eine Reihe von Definitionen. Zur Zeit wird die Bezeichnung im täglichen Sprachgebrauch für nahezu jede Software verwendet, die irgend eine Form der grafischen Darstellung eines Sachverhaltes hervorbringt. Einige Beispiele für verschiedene Defnitionen seien hier in Form von Zitaten aufgeführt:

> In der Enzyklopädie [Aut94] ist der Begriff Visualisierung als eine Bezeichnung für bildliche Formulierung und Kommunikation definiert, d.h. für Aufbereitung von Information mit vor allem bildlichen Mitteln wie auch für die visuelle Wahrnehmung.
>
> In [Cha94] ist Visualisierung als die Umwandlung von Informationen, die ursprünglich nicht in Bildform vorliegen, in eine meist grafische Darstellung definiert.
>
> Visualisierung [Dä99] ist ein spezieller Bereich der Computergrafik, der sich vorrangig mit der Darstellung von physischen Parametern oder Objekten befaßt. Visualisierung wird vornehmlich im wissenschaftlichen Bereich bei der Interpretation massenhafter und komplexer Daten eingesetzt.
>
> Visualization [McC87] is a method of computing. It transforms the symbolic into the geometric,enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights. In many fields it is already revolutionizing the way scientists do science.
>
> Visualization [Dom00] is more than a method of computing. Visualization is the process of transforming information into a visual form, enabling users to observe the information. The resulting visual display enables the scientist or engineer to perceive visually

> features which are hidden in the data but nevertheless are needed for data exploration and analysis.

Aus den obengenannten Definitionen wird ersichtlich, daß der Begriff der Visualisierung, in Abhängigkeit der verschiedenen Forschungsinteressen durchaus unterschiedlich interpretiert wird. In jeder Definition des Begriffes Visualisierung wird von „Information“ gesprochen, wobei aber selten der Versuch einer Definition dieses Begriffes unternommen wird. Die hier angeführte Definition ist aus der Literatur [H.97] entnommen .

> Unter Information wird die Bedeutung, die durch eine Nachricht übermittelt wird, verstanden. Nachrichten können gesprochene oder geschriebene Texte, Bilder, Geräusche oder sonstige Daten sein, die wir mit unseren Sinnesorganen wahrnehmen und mit technischen Hilfsmitteln speichern und übertragen können. Welche Information einer solchen Nachricht entnommen wird, ist vom Empfänger dieser Nachricht abhängig und somit subjektiv.

Bei der Visualisierung von Informationsräumen handelt es sich um eine komplexe Aufgabe, die nur mit Hilfe wissenschaftlicher Methoden und Konzepte gelöst werden kann. Im letzten Jahrzehnt hat sich eine eigenständige Forschungsrichtung herausgebildet, die über solche Konzepte verfügt – die wissenschaftlich-technische Visualisierung. Im Folgenden soll untersucht werden, inwiefern diese Konzepte für die Visualisierung von Informationsräumen genutzt oder adaptiert werden können.

## 2.6 Wissenschaftlich-technische Visualisierung

### 2.6.1 Begriff und Historie

Der Begriff „Wissenschaftlich-technische Visualisierung“ ist eine Bezeichnung, die in der Mitte der achtziger Jahre in der Computertechnologie geprägt wurde und großes Interesse in der Computerwissenschaft gefunden hat. Ursprünglich wurde die Visualisierung eingeführt, um große Datenmengen besser handhabbar zu machen und die Suche nach Phänomenen in den Daten zu vereinfachen. Wissenschafliche Visualisierung [Mor98] ist eine interdisziplinäre Methodik, die mehrere, weitgehend unabhängige, aber sich näher kommende Arbeitsgebiete verbindet. Sie ermöglicht durch die Interpretation von Meß- oder Simulationsdaten das bessere Verständnis dieser Größen. Weitere übliche Bezeichnungen sind *Scientific Visualisation* und *Visualisation in Scientific Computing* (ViSC). Eingeschlossene Arbeitsgebiete sind: generative Computergrafik, Bildverarbeitung, Bilderkennung, computerunterstützte Konstruktion, Signalverarbeitung und die Entwicklung von Benutzeroberflächen [Mor98].
Visualisierung [Dom00] wird auch als ein kognitiver Prozeß gesehen, der von Menschen durchgeführt wird, indem sie ein mentales Bild einer Informationsdomäne bilden. In der Computertechnologie und in der Informationswissenschaft [Dom00] ist es die Benutzung von Grafiken, Bildern, animierten Sequenzen usw., mit deren Hilfe Daten, Strukturen, das dynamische Verhalten von großen und komplexen Datenmengen, Ereignisse, Konzepte u.a. auf verständliche Weise dargestellt werden können.
Die Visualisierung dient neben der visuellen Verarbeitung von Simulationsergebnisssen auch als Analyseinstrument bei der Auswertung von Meßdaten. Beispiele dafür sind u.a. in den Bereichen Medizin (Computertomografie) und Satellitenbildanalyse (Aufbereitung der Daten von Planeten) zu finden [Ort93]. Ziel der Visualisierung ist es, die hochentwickelten Fähigkeiten des menschlichen Sehsystems zur Informationsvermittlung zu nutzen.

### 2.6.2 Inhalte

Die wissenschaftliche Visualisierung ist ein wachsender Forschungszweig der Informatik. Ihre Aufgabe ist es, wissenschaftliche Daten sichtbar zu machen, um eine visuelle Auswertung komplexer Datensätze zu ermöglichen und ihre zugrundeliegende Struktur besser zu verstehen [Kon96, Rö00]. Die Daten können aus Experimenten, der Theorie oder – am häufigsten – aus numerischen Simulationen stammen. Diese darzustellenden Daten besitzen im allgemeinen eine physikalisch, mathematische Struktur und sie sind numerisch oder symbolisch repräsentiert. Typische Informationsobjekte sind Flächen, Gitter, Graphen, Skalar-, Vektor-, und Tensorfelder usw. Für solche Objekte sind visuelle Darstellungen zu entwickeln, die den kognitiven Fähigkeiten des Menschen besonders gut entsprechen. Dazu werden Methoden aus der Computergrafik, Bildverarbeitung und algorithmischen Mathematik verwendet, sowie Technologien wie Grafik- und Display-Hardware, Video, Multimedia oder Virtual Reality eingesetzt, um die Qualität der Interpretation wissenschaftlicher Daten zu erhöhen.
Die Grundlagen der wissenschaftlich–technischen Visualisierung sind die grafische Datenverarbeitung speziell mit den Teilbereichen generative Computergrafik und Bildverarbeitung sowie das Bilderverstehen (Computer–Vision) [McC87].
Bei der Entwicklung wirkungsvoller Visualisierungsverfahren sind auch Belange des jeweiligen Anwendungsgebiets zu berücksichtigen und sinnesphysiologische, wahrnehmungspsychologische und ästhetische Aspekte zu beachten.
Die Menge der numerischen Daten aus den Simulationen wächst rasant. Jedoch sind diese Daten wenig nützlich, solange aus ihnen nicht Information extrahiert wird. Das bedeutet, die Visualisierung ist oft die einzige Möglichkeit, Ergebnisse zu interpretieren. Das Ziel der wissenschaftlichen Visualisierung besteht also darin, Konzepte für die visuelle Repräsentation der Daten zur Verfügung zu stellen, um ein besseres Verständnis für physikalische Prozesse, mathematische Konzepte und andere reale Phänomene zu vermitteln [Dä98]. Die Techniken der wissenschaftlichen Visualisierung werden in einer Reihe von Natur- und Ingenieurwissenschaften verwendet, von der Physik bis hin zur Medizin. Die grundlegenden Prinzipien werden im folgenden Abschnitt erläutert.

### 2.6.3 Die Visualisierungs-Pipeline

Ausgehend von zugrundeliegenden Meß- oder Simulationsdaten müssen die interessierenden wissenschaftlich-technischen Phänomene in geeigneter Weise sichtbar gemacht werden. Dies erfordert problemspezifische Visualisierungstechniken. Simulationen produzieren z.B. im Bereich des *Scientific Computing* große Mengen von Ergebnisdaten, da häufig mit Punktmengen in hochdimensionalen Räumen gearbeitet wird. Deswegen müssen die Daten zuerst aufbereitet und entsprechenden grafischen Grundelementen zugewiesen werden, bevor sie unter Berücksichtigung der gewählten Visualisierungsmethode grafisch dargestellt werden.
Das Ziel im Visualisierungsprozeß ist es, visuelle verständliche Bilder von abstrakten Daten zu erzeugen. Der Visualisierungsprozeß besteht aus den Verarbeitungsschritten, die bei der Erzeugung eines Bildes aus den Ausgangsdaten durchlaufen werden. Diese Verarbeitungsschritte lassen sich in einer so bezeichneten *Visualisierungs-Pipeline* [Ort93, SH00] anordnen. Konzeptuell kann der Prozess der Datenvisualisierung in eine Abfolge von drei wichtigen Schritten oder Transformationsstufen unterteilt werden. Die Abbildung 2.1 zeigt diese drei Stufen: *Filtering, Mapping und Rendering*. Zusammen bilden sie die sogenannte *Visualisierungs-Pipeline (visualization pipeline)*, nach der die meisten heute erhältlichen Visualisierungstools organisiert sind. Das Modell in der Abbildung 2.1 konkretisiert die Aufgabenverteilung bei einer visuellen Analyse von Simulationsdaten. Eine Datenquelle bildet den Anfang der Visualisierungs-Pipeline [Are00, SH00, Ort93]. Sie stellt einen Rechenprozeß zur Auswertung eines physikalischen Modells dar. Im Vordergrund steht generell die Repräsentation von Modellen (Phänomenen). So müssen die Rohdaten in geeigneter

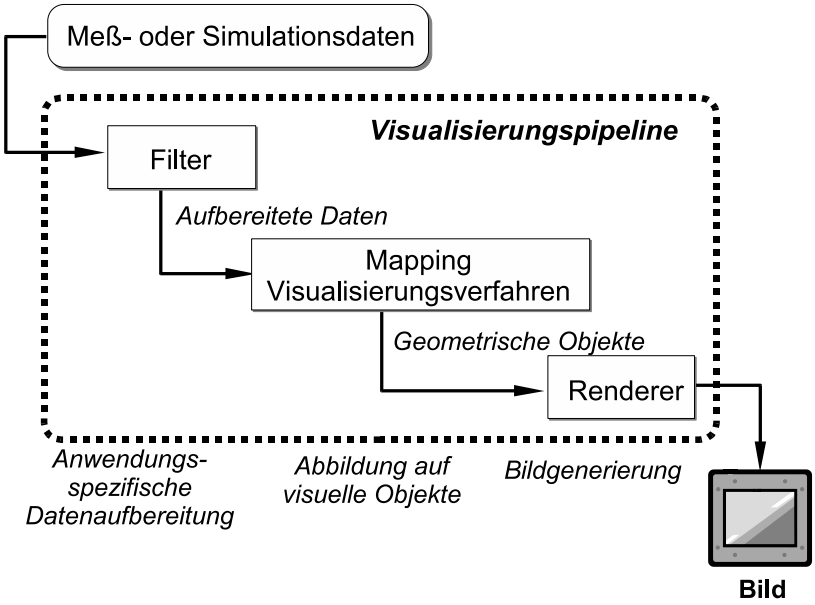



Abbildung 2.1: Visualisierungs-Pipeline bei der grafischen Auswertung von Daten [Ort93]

Weise aufbereitet werden. Diese Aufbereitung erfolgt mit Hilfe von Filtern.

### Die Filter-Stufe

Rohdaten können unvollständig oder für die nachfolgenden Visualisierungsschritte zu umfassend sein. Darum werden Datenfilterungsmethoden eingesetzt, wodurch die Rohdaten in die für die Visualisierung verwendbaren Daten umgeformt werden. Dieser Prozeß erfolgt in der ersten Phase, die *Filtering* genannt wird. Die Datenaufbereitung beschränkt sich in dieser Phase auf die Konstruktion eines empirischen Modells, das die geometrische Struktur des interessierenden Phänomens charakterisiert. So können diskrete Punktmengen in den zwei- oder dreidimensionalen Raum abgebildet werden.

### Die Mappingstufe

Die zweite Phase und entscheidende für den Visualisierungsprozeß ist der *Mapping-Prozeß*. Er ist für die Komposition des Bildes, das erzeugt wird, verantwortlich. Hier wird die eigentliche Visualisierungstechnik festgelegt. Der Mapping-Prozeß bildet die gefilterten Daten auf abstrakte visuelle geometrische Objekte und ihre Attribute ab. Die visuellen Objekte sind allgemeine geometrische Primitive (Basiselemente von Grafiksystemen wie Linie, Kreis, Kugel, Zylinder, ...), Oberflächengitter oder andere Darstellungselemente wie Volumenelemente (Voxel). Die Attribute der geometrischen Objekte beinhalten geometrische oder topologische Eigenschaften wie z. B. Skalierung, Ausrichtung, Deformation, Position oder auch Parameter, die die Erscheinung der Objekte beeinflussen, wie z. B. Farbe, Transparenz oder Textur.

### Die Renderingstufe

Die letzte Phase ist die eigentliche Darstellung. Diese Phase heißt *Rendering-Prozeß*. Hier wird aus der Geometrie der Objekte ein zweidimensionales Bild auf der vorgesehenen Ausgabefläche (Bildschirm, Drucker, ...) generiert. Bei der Berechnung dieses Bildes werden unter Berücksichtigung der Kameraposition und der Begrenzungen des sichtbaren Bereiches verschiedene Transformationen

durchgeführt, sowie Reflektionsparameter und vorhandene Lichtquellen ausgewertet. Der Renderer erzeugt also aus dem rechnerinternen Modell der Mappingstufe das Endresultat – eine visuell auswertbare Repräsentation der Ausgangsdaten. Um eine eingehende Untersuchung zu ermöglichen, können in der Regel bestimmte Parameter vom Benutzer direkt an der Renderingstufe angepaßt werden – z. B. die Position der Kamera oder die Beleuchtungssituation.

### 2.6.4 Der Visualisierungszyklus

Die vorgestellte modulare Aufbauweise der Visualisierungs-Pipeline suggeriert eine sequentielle Arbeitsweise des Visualisierungsprozesses. Die Parameter der Filtering-, Mapping- und Rendering-Transformationsstufe können jedoch beispielsweise interaktiv geändert oder gegen andere, besser geeignete, ausgetauscht werden.
Wird die Abbildung 2.1 in der Form erweitert, daß die Eingriffe in den Visualisierungsprozeß interaktiv sind, wird man einen zirkulären Visualisierungsablauf erhalten, der in Abbildung 2.2 gezeigt ist. Die Simulationsdaten werden von einem Visualisierungsprogramm gelesen, gefiltert und visuell

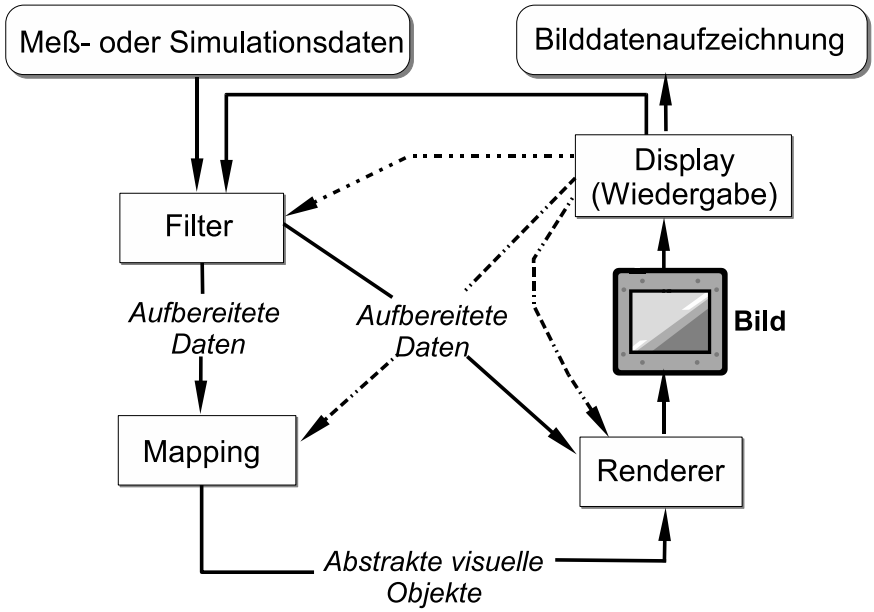



Abbildung 2.2: Visualisierungszyklus [Ort93]

dargestellt. Die gestrichelten Linien deuten mögliche Eingriffe des Benutzers in den Visualisierungszyklus an. Im Prinzip sind interaktive Eingriffsmöglichkeiten in allen Transformationsstufen möglich und erforderlich.

## 2.7 Informations-Visualisierung

Der Begriff Informations-Visualisierung umfaßt hingegen alle Konzepte, Methoden und Werkzeuge zur visuellen Darstellung abstrakter Informationen, wie sie in verschiedenen Dokumentsammlungen wie Datenbanken oder digitalen Bibliotheken auftreten. Die abstrakten Informationen haben bei der Visualisierung keine physikalische Datenbasis. Informations-Visualisierung beinhaltet die computergestützte Aufbereitung und die visuelle Repräsentation abstrakter Informationen mit dem Ziel, den kognitiven Zugang zu elektronisch gespeicherten Daten zu erleichtern [Dä99].

### 2.7.1 Begriffsbestimmung: „Informationsraum“

Zunächst muß die Frage nach dem eigentlichen Gegenstand der Visualisierung geklärt werden – dem *Informationsraum*. Dieser enthält die abstrakten Informationen, d.h. Dokumente, Begriffe oder andere textbasierte Informationen aus Datenbanken, Retrievalsystemen oder dem WWW. In der Literatur ist nur schwer eine Definition des Informationsraumes zu finden. In [Wü97] ist dieser Begriff wie folgt definiert:

> Ein Informationsraum ist durch eine Informationsmenge und eine Informationsstruktur definiert, wobei eine Informationsstruktur eine Relation ist, die die Beziehungen zwischen den Informationsobjekten beschreibt. Eine spezifische Information kann auf diese Weise auch in unterschiedlichen Informationsräumen repräsentiert werden.

Die Überführung einer solchen abstrakten Struktur in eine geeignete dreidimensionale Struktur ist ein wichtiger Teil der vorliegenden Arbeit. Eine solche Struktur muß:

- übersichtlich,
- intuitiv,
- mit angemessenem Aufwand visualisierbar sein.

Der Benutzer sollte sofort erkennen, wie gut die gefundenen Dokumente den spezifizierten Suchkriterien entsprechen. Die Computergrafik bietet dazu eine Vielzahl von Methoden wie Icons, Abbildung auf Farbe und Form, Position, Größe, usw. Weiterhin sollte es möglich sein, die Wichtung der Suchkriterien interaktiv zu verändern und im dargestellten Informationsraum zu navigieren. Da vom Benutzer keine Erfahrungen auf diesem Gebiet erwartet werden können, ist die Gestaltung einer entsprechenden Benutzerschnittstelle eine anspruchsvolle Aufgabe. Veröffentlichungen haben gezeigt, daß eine Visualisierung von Informationsräumen grundsätzlich möglich ist, eine Reihe von entsprechenden Ansätzen wird im Kapitel 3 besprochen. Die Visualisierung bildet eine wichtige Grundlage zur Sichtung großer Informationsbestände, wobei neben der Veranschaulichung der Informationsobjekte und ihrer Beziehungen untereinander dem Benutzer auch die Möglichkeit gegeben wird, im Datenraum zu navigieren und mit den Objekten zu interagieren.

### 2.7.2 Der Abfragezyklus der Informationssuche

Der oben beschriebene Zyklus der wissenschaftlich-technischen Visualisierung hat durchaus Ähnlichkeit mit dem Abfragezyklus einer Informationssuche und beides kann verglichen werden. Eine adaptierte Version ist in der Abbildung 2.3 dargestellt. In dieser Darstellung der Informationssuche formuliert der Benutzer seine Anfrage, auf deren Basis die Suchmaschine eine Liste von relevanten Dokumenten zusammenstellt. Diese Liste kann durch eine Filterstufe aufbereitet werden, so können z.B. Dokumente unter einer bestimmten Relevanzschwelle entfernt und damit von der Visualisierung ausgeschlossen werden. Auf diese Weise wird eine Untermenge von Informationen gebildet. Eine eingehende Erklärung wie die Informationen (Dokumente) von der Filterstufe aufbereitet werden, wird im Punkt 5.4.4 vorgenommen.
Die verbleibende Untermenge von Informationen wird dann der Mapping-Stufe zugeführt. In dieser Stufe ist zunächst ein entsprechender Informationsraum mit geeigneten Verfahren auf einen dreidimensionalen Raum abzubilden. Der Informationsraum hat eine spezifische Raumstruktur, in die die Objekte und Objektbeziehungen abgebildet werden (Mapping). Eine ausführliche Beschreibung ist in Punkt 5.2.1 zu finden.
Nach der geometrischen Abbildung der Informationen wird letztendlich die Renderingstufe durchlaufen. Diese Stufe stellt praktisch auch eine Benutzerinterface für den Umgang mit der Ergebnismenge zur Verfügung. Sie ermöglicht die Navigation durch die Dokumentenmenge, das genauere

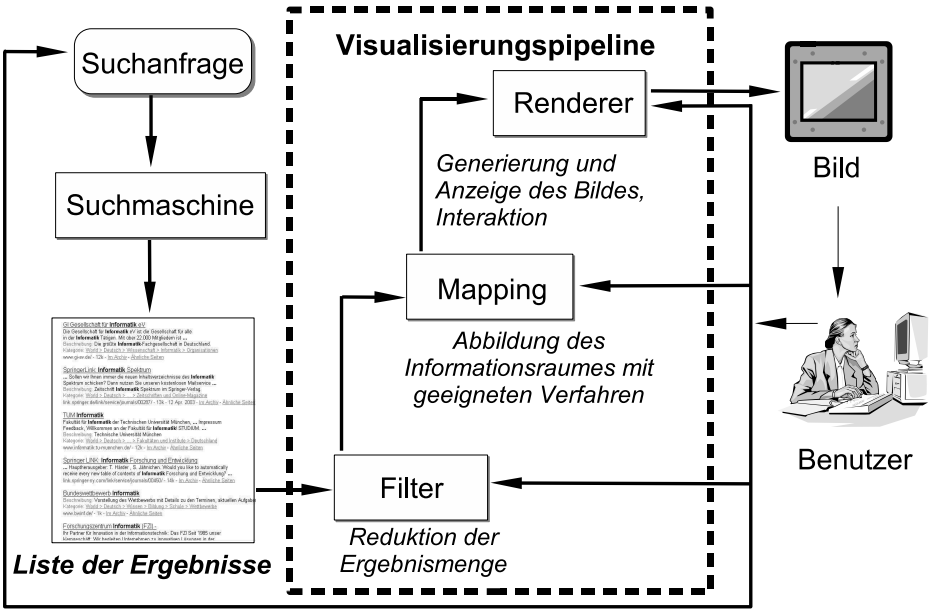



Abbildung 2.3: Der Zyklus der Informationssuche

Untersuchen einzelner Dokumente und gewährt einen Überblick über die gefundenen Dokumente. Außerdem kann der Nutzer – ausgehend von der grafischen Darstellung – auf fast alle Parameter des Zyklusses Einfluß nehmen. Er kann z.B. andere Mapping-Verfahren in der Mappingstufe wählen, die Anzahl der angezeigten Dokumente weiter reduzieren, oder seine Suchanfrage aufgrund der gewonnenen Erkenntnisse verbessern.

## 2.8 Der Unterschied: Wissenschaftliche und Informations-Visualisierung

Die Visualisierung stellt eine starke Abstraktion des Datenmodells dar. Das Gebiet der Informations-Visualisierung ist noch komplexer als das der wissenschaftlichen Visualisierung, aber die Informations-Visualisierung bietet die Möglichkeit, durch frei definierbare visuelle Metaphern die Informationsräume optimal den Erfordernissen des Benutzers anzupassen.
Die wissenschaftliche Visualisierung wurde früher als Anwort auf das Bedürfnis von Wissenschaftlern entwickelt, Daten in grafischem (anschaulichem) Format anzusehen [Her00].

- Die vorliegenden auszuwertenden Daten sind bei der wissenschaftlichen Visualisierung an eine Geometrie (ein-, zwei-, oder dreidimensional) gebunden. Diese Geometrie beschreibt die Lage eines Punktes, der mit Parametern versehen ist, im Raum. Diese Kombination von geometrischem Punkt und dazugehörigen Parametern wird als *Knoten* bezeichnet. Ein Parameter kann folgende Typen haben:

  – Skalar
  – Vektor
  – Tensor

  Die Anzahl und die Reihenfolge der Parameter auf einem Knoten ist beliebig. Der Parametertyp und die Dimension der Knoten sind maßgeblich für die Auswahl eines geeigneten

Mappingverfahrens. Bei der Informations-Visualisierung ist das nicht der Fall. Für solche Daten muß die geometrische Darstellung erst „erfunden“ werden [Her00].

- Im Allgemeinen unterliegen die zu visualisierenden Daten einer Topologie, die entweder dem zugrundeliegenden Phänomen entspringt oder künstlich erzeugt wurde. Diese Topologie ist eine Grundlage für die meisten eingesetzten Mappingverfahren der wissenschaftlich-technischen Visualisierung. Im Gegensatz dazu liegen in der Informations-Visualisierung lediglich abstrakte Datenstrukturen (z. B. Hypermedia-Strukturen) ohne räumliche Analogie vor.

- Eine weitere wichtige Rolle spielt die Mensch-Maschine-Interaktion. Während die Benutzer in der wissenschaftlichen Visualisierung normalerweise Experten sind, können die Benutzer in der Informations-Visualisierung aus dem gesamten Spektrum der heute aktiven Computerbenutzer stammen.

- Die Anwendungen der wissenschaftlichen Visualisierung laufen normalerweise auf Graphic-Workstations oder gar Supercomputern. Dementsprechend aufwendig können die eingesetzten Visualisierungsverfahren sein. Im Gegensatz dazu müssen Informations-Visualisierungssysteme auf heute gebräuchlichen PCs lauffähig sein, wenn sie in der Praxis einsetzbar sein sollen.

## 2.9 Fazit

Die dreidimensionale Visualisierung ist im wissenschaftlich-technischen Bereich ein erprobtes Mittel zur Verdeutlichung komplexer Zusammenhänge. In diesem Bereich sind es jedoch Phänomene, die von Natur aus dreidimensional sind, die auf grafische 3D-Verfahren abgebildet werden. Alle genannten Voraussetzungen bestehen bei der Informationsvisualisierung nicht.

> Der Informationsraum ist ein abstraktes Gebilde, das keine Analogie zur realen Welt und zum menschlichen räumlichen Vorstellungsvermögen besitzt.

Die Abbildung (Mapping) der Information in einem Informationsraum ist daher komplizierter als in der wissenschaftlichen Visualisierung. Eine Teilaufgabe bei dieser Art der Visualisierung besteht darin, erst eine geometrische Struktur mit Knoten und mathematisch auswertbaren Parametern zu erzeugen, die für die Auswertung von Informationen geeignet ist. Es müssen visuelle Metaphern definiert werden, die den Informationsraum optimal den Erfordernissen und den Erfahrungswelten des Benutzers anpassen.
Da die Informationsmenge und die Anzahl der Informationsanbieter im Web ständig wächst, ist die Visualisierung von Informationsräumen eine Herausforderung der Computertechnologie. Es werden neue grafisch-visuelle Methoden für den Umgang mit Informationen gesucht.

# Kapitel 3

# Ansätze zur Visualisierung von Datenstrukturen

Am Anfang der 90er Jahre wurde eine Reihe von Ansätzen für die Visualisierung von Informationsräumen entwickelt. Viele dieser Projekte [Dä99, You96] hatten als Ziel, den kognitiven Zugang zu Datenbanken und anderen Informationsressourcen im Internet mittels neuer grafischer Nutzeroberflächen zu verbessern und den Benutzer beim Erfassen von Informationen zu unterstützen. In diesem Kapitel werden ausgewählte Ansätze für verschiedene Anwendungen aus den Bereichen Dokumentretrieval und Dateiverwaltung beschrieben und mit Hilfe eines Kriterienkataloges hinsichtlich Darstellung, Benutzerführung und Möglichkeiten zum schnellen Erreichen des Zieles bewertet.

## 3.1 Die Bewertungskriterien

Es wird eine Liste von relevanten Kriterien aufgestellt, um die Ansätze zur Visualisierung von Informationen zu bewerten. Einige Kriterien wurden aus dem Katalog „Kriterien zur Reduzierung erzwungener Sequentialität“ [Kei90] als Grundlage für diese Arbeit ausgesucht. Die Kriterien werden in drei Gruppen eingeteilt: Benutzeroberfläche, Interaktion und Funktionalität. Im Folgenden wird nach einer kurzen Begriffserklärung auf die einzelnen Kriterien eingegangen.

### 3.1.1 Einschätzung des Aufbaus der Arbeitsoberfläche

In der Gruppe „Benutzeroberfläche“ sind die Kriterien enthalten, die sich mit dem statischen Aufbau der Oberfläche befassen. Generell müssen bei der Bildschirmarbeit die Auswahlmöglichkeiten für die durchzuführenden Operationen sowie die dadurch erreichbaren Zielzustände für den Benutzer präsent gemacht werden. Dies ist wichtig, damit der Benutzer bei einer grafischen Oberfläche die Befehle und ausführbaren Aktionen schnell erkennen kann und nicht aufwendig erlernen muß.

**Strukturiertheit**
Es soll untersucht werden, ob die Oberfläche so gestaltet ist, daß der Benutzer schnell einen Überblick der Gesamtheit erhält, die Zusammenhänge erfassen und die Informationen effektiv erschließen kann [Bau99]. Diese Untersuchung bezieht sich nicht nur auf die Benutzungsoberfläche im herkömmlichen Sinne (Menüs, Dialoge, etc.), sondern auch auf die grafische Präsentation der Informationen, sofern mit dieser interagiert werden kann. Dies ist notwendig, da über die grafischen Objekte häufig direkt Aktionen ausgeführt oder Informationen extrahiert werden können. In diesen Fällen dient die grafische Darstellung des Informationsraumes also direkt der Arbeit mit dem zugrundeliegenden Programm und wird daher im Folgenden als Bestandteil

der Benutzungsoberfläche betrachtet.

**Erkennbarkeit**
Unter diesem Kriterium wird die Forderung nach der Veranschaulichung der jeweils erreichbaren Zielzustände bzw. Operationen zusammengefaßt. Bei diesem Kriterium werden folgende Fragen untersucht:

- Wie sind die Elemente der Arbeitsoberfläche gestaltet, unterstützen oder beeinträchtigen sie die Wahrnehmungen des Benutzers?
- Wie ist das Verhältnis der Objekte zueinander?
- Wie gut sind die Objekte auffindbar und selektierbar?
- Erfolgt von der Visualisierung eine Rückmeldung, welches Objekt betrachtet bzw. dargestellt wird?
- Können unter bestimmten Umständen Informationsverstopfungen oder Informationsausdünnungen aufreten?

**Lokalität:**
Die Ausführung einer Aktion und die Wahrnehmung durch den Benutzer sollten immer möglichst eng gekoppelt sein. In der Darstellung ist sicherzustellen, daß die häufigsten Funktionen unabhänging von der Position des Betrachters im direkten Zugriff erreichbar sind. Die Ausgaben dieser Aktionen sollen möglichst unmittelbar am Ort der Ausführung erfolgen, damit der Benutzer sie durch seine Position und Blickrichtung wahrnehmen kann [Kei98]. Wie gut diese Anforderungen umgesetzt sind, wird bei diesem Kriterium untersucht. Dies ist von besonderer Bedeutung, da bei den hier vorgestellten Verfahren unter Umständen relativ komplexe Aktionen im dreidimensionalen Raum durchzuführen sind.

**Orientierung:**
Die zeitlich und räumlich gegebenen Handlungsalternativen wie zum Beispiel Zielzustände, ausführbare Operationen oder erreichbare Positionen u.a. sollen möglichst vollständig visualisiert und zugänglich gemacht werden. Der aktuelle Ort oder Zeitpunkt soll im Verhältnis zum Ganzen dargestellt werden. Bei den gängigen Benutzeroberflächen wird das z. B. mit Hilfe von Icons realisiert, die für viele Aktionen standardisiert sind. Hier geht es jedoch darum, Handlungsalternativen im dreidimensionalen Raum, die in Abhängigkeit von Ort, Ansicht und Zeit angeboten werden müssen, in einer für den Benutzer verständlichen Form anzuzeigen.

### 3.1.2 Untersuchung des Interaktionsverhaltens

In der Gruppe „Interaktion“ sind die Kriterien zusammengefaßt, die die dynamischen Aktivitäten eines interaktiven Systems beschreiben. Das betrifft in erster Linie die Interaktionsmechanismen, die bereitgestellt werden um z. B. die Ansicht des Informationsraumes zu verändern oder bestimmte Elemente genauer zu betrachten.

**Wechselwirkung:**
Es soll untersucht werden, welche Mechanismen zur Interaktion mit dem System dem Benutzer angeboten werden und wie hoch der Aufwand an Manipulationen zum Erreichen eines bestimmten Zustandes ist. Bei einer dreidimensionalen Präsentation häufig benötigte Mechanismen sind u.a.:

- Rotation der Ansicht (z. B. um die Sicht auf verdeckte Elemente zu gestatten),

- Veränderung von Kamerastandpunkt und Blickrichtung, um die darzustellende Region des Informationsraumes auszuwählen oder eine „Kamerafahrt“ durchzuführen,
- Veränderung des Öffnungswinkels der Kamera, um in ein interessantes Gebiet „hineinzoomen“ zu können,
- das Auswählen von Elementen, um z. B. zusätzliche Informationen anzeigen zu können.

**Ausführungsminimalität:**
Es sollten flexible Möglichkeiten zur Verfügung gestellt werden, die einen einfachen Umgang mit der Struktur des Informationsraumes und somit verschiedene Sichtweisen auf die grafische Szene erlauben. Es sind Mechanismen notwendig, die es dem Benutzer erlauben, schnell das gewünschte Objekt zu erreichen oder in den Fokus zu bringen. Zu untersuchen ist, ob die Aktionen verständlich sind, sich evtl. gegenseitig bedingen, und ob sich Handlungssequenzen zu einer Aktion zusammenfassen lassen.

### 3.1.3 Untersuchung der Funktionalität

In dieser dritten Gruppe von Kriterien werden die Verfahren hinsichtlich der gebotenen Möglichkeiten zur Untersuchung von Dokumentenmengen verglichen. Das betrifft sowohl die Art und Weise in der der Informationsraum grafisch repräsentiert wird, als auch den Einsatz von Techniken zur Vor- und Nachbereitung der Menge der Elemente des Informationsraumes.

**Darstellung:**
Die Verfahren werden hinsichtlich der grafischen Ausgabemöglichkeiten untersucht: Art der Darstellung (2D/3D), Freiheitsgrade bei der Navigation im Informationsraum, Einstellungsmöglichkeiten bezüglich der Art der Darstellung (z.B. Auswahl von Objekttypen für die Dokumentenrepräsentation), um weitere Informationen zu den Objekten zu erhalten.

**Datenreduktion:**
Besonders beim Einsatz von dreidimensionalen Darstellungsvarianten ist trotz der gewachsenen Leistungsfähigkeit der Hardware die Anzahl der darzustellenden Objekte noch immer ein wichtiger Faktor für die Ausgabegeschwindigkeit. Dies macht sich besonders bei Interaktionen bemerkbar. Bei großen Dokumentenmengen ist es daher sinnvoll, die Datenmenge bereits *vor* der grafischen Darstellung auf das notwendige Maß zu reduzieren.

**Erweiterbarkeit:**
Nicht jedes Darstellungsverfahren ist für jede Art von Informationsraum geeignet und auch Benutzer können hinsichtlich der grafischen Ausgaben völlig unterschiedliche Preferenzen haben. Daher kann es sinnvoll sein, ein Verfahren durch weitere Visualisierungsarten, andere Ausgabeelemente für die Informationsobjekte oder Filter für eine Vorauswahl der Dokumentenmenge zu erweitern.

**Details:**
Die Entscheidung, ob ein Dokument wirklich interessant ist, kann der Benutzer oft erst nach genauerer Betrachtung fällen. Unter diesem Kriterium werden Möglichkeiten dazu eingeschätzt.

## 3.2 Bewertung der Visualisierungsansätze

Die verschiedenen Ansätze werden zuerst im einzelnen vorgestellt und gleichzeitig mit Hilfe der oben erläuterten Kriterien bewertet. Die Einschätzung, in welchem Maß diese Kriterien erfüllt werden, erfolgt durch den Vergleich der einzelnen Ansätze zueinander, um eine direkte Hervorhebung etwaiger Gleichheiten oder wichtiger Unterschiede zu ermöglichen. Die Untersuchung der Ansätze beschränkt sich auf eine Betrachtung der Oberfläche und des Interaktionsverhaltens, was aber dem Rahmen dieser Arbeit genügt.

### 3.2.1 Perspective Wall

Das Verfahren *Perspective Wall* [Rob91a] ist ein Abbildungsverfahren zur Darstellung linear strukturierter Informationen. Es wird als *Fokus+Kontext*-Visualisierungsverfahren bezeichnet [McG01]. Das bedeutet, während der Fokus den relevanten Ausschnitt einer Informationsmenge enthält und dieser Teil in höchstmöglichem Detail gezeigt wird, enthält der Kontext einen groben Überblick über die restlichen Informationen, die angezeigt werden können. Diese Information wird verkleinert (verzerrt) dargestellt. Das Verfahren wurde am PARC-Laboratory der Firma Xerox entwickelt.
*Perspective Wall* entfaltet ein 2D-Layout auf einer dreidimensionalen Wand, in der eine sequentielle Darstellung von Dateien über eine Zeitachse repräsentiert wird. Die Dateien werden auf die Oberfläche der perspektivischen Wand projiziert. Die Wand wird vertikal in drei Teilflächen unterteilt. Auf der mittleren Wandfläche werden die Dateien, die zu einem bestimmten Zeitpunkt erstellt wurden, dargestellt. Gleichzeitig sind andere Dateien mit älterem und neuerem Datum überblicksartig zu sehen. Die mittlere Wand ist für die Betrachtung von Details vorgesehen und die Dateien, die sich auf dieser Wand befinden, werden relativ größer gesehen als die auf den anderen Wänden. Die Wand kann interaktiv verschoben werden, so daß jeweils andere Zeitabschnitte in den Vordergrund gelangen.
Die anderen beiden Wandflächen an der linken und rechten Seite der mittleren Wand dienen zur Betrachtung des Kontextes. Diese beiden Wände sind nach hinten perspektivisch verzerrt, so daß der Eindruck einer dreidimensionalen Abbildung entsteht. Auf diese Weise wird eine gute Ausnutzung der vorhandenen Darstellungsfläche auf dem Bildschirm erreicht. Das wird deutlich, wenn man die perspektivische Darstellung des 2D-Layouts mit den Ausmaßen einer „ungefalteten“ Darstellung desselben Layouts vergleicht. Diese Eigenschaft kommt besonders bei Strukturen zum Tragen, deren Layout zu einem hohen Breiten/Höhenverhältnis führt. Die Abbildung 3.1 zeigt eine Darstellung dieser Struktur.
Jede einzelne Wand ist in mehrere, in Reihen und Spalten angeordnete Flächen unterteilt. Auf diesen kleinen Flächen sind die Informationsdateien dargestellt. Sie sind nach Datum sortiert in den vertikalen Spalten und nach Arten der Dateien sortiert in horizontalen Reihen angeordnet. [Wie98]. Die Abbildung 3.2 zeigt eine Ansicht mit zwei verschiedenen Anordnungen einer Wand [Rob91a]:

- Die gerade Wand (in der Abbildung auf der rechten Seite) befindet sich vollständig im Blickfeld. Die Entfernung zum Betrachter ist dadurch groß und führt zu einer kleinen Darstellung der Details (und damit auch unter Umständen zur Unkenntlichkeit).

- Die zweite gerade Wand (in der Abbildung links, nahe der Kamera) befindet sich im gleichen Abstand zum Betrachter wie die Detailansicht der *Perspective Wall* (fett eingezeichnet). Dabei würden sich viele Detailinformationen außerhalb des Gesichtsfeldes des Betrachters befinden.

Beide Fälle zeigen also Nachteile, die eine übersichtliche und aussagekräftige Darstellung der Informationen stark einschränken. Daher entstand der Ansatz, einen Teil der Anzeigefläche mit einem

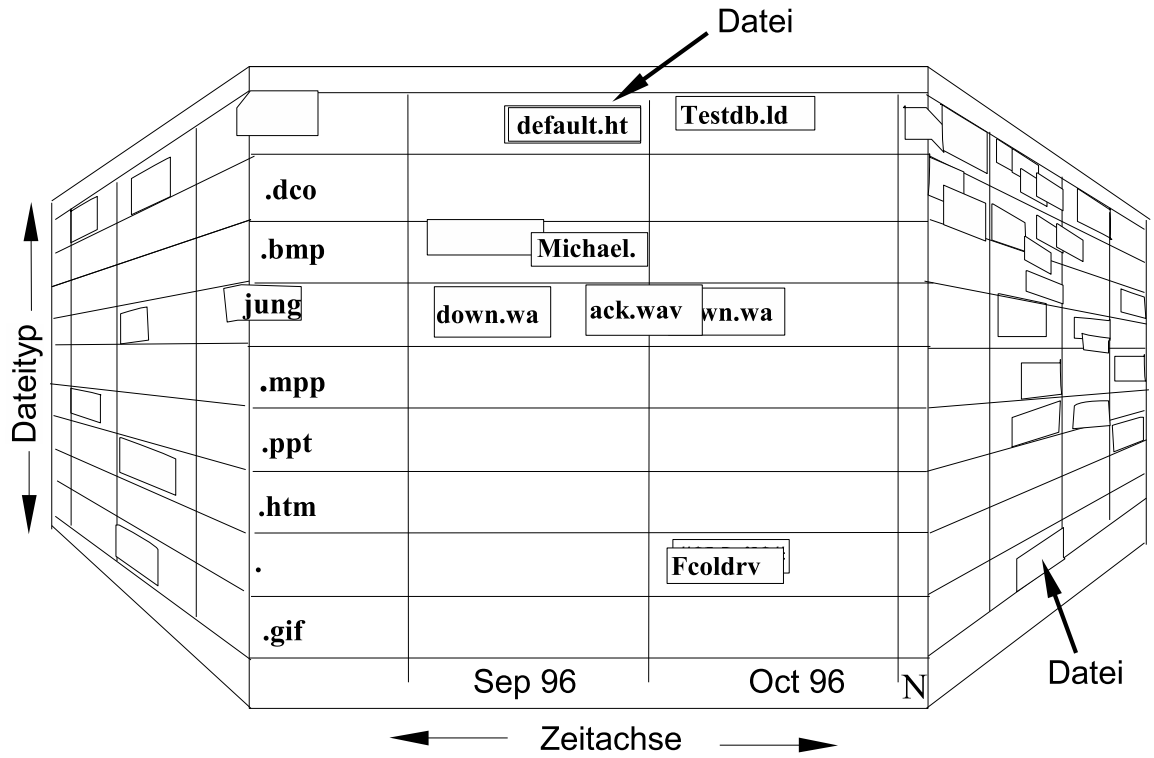



Abbildung 3.1: Informationsdarstellung mittels Perspective Wall [Dä99]

entsprechenden Layout für die Informationen perspektivisch darzustellen.
Folgende Parameter werden für die Berechnung des Layouts auf einer *Perspective Wall* verwendet:

- Der Winkel $\theta_f$ beschreibt die Faltung der perspektivisch dargestellten Seitenwände.
- Die Breite $w$ entspricht der Breite einer Seitenwand, wobei die Breite der mittleren Wand der *Perspective Wall* mit dem Wert 1.0 angenommen wird.
- Der Winkel $\theta_e$ entspricht dem halben Öffnungswinkel der Kamera. Typische Werte für diese Parameter sind [Rob91a]: $\theta_f = 60°$, $w = 5.0$, und $\theta_e = 17.5°$ .

Die relative Größe der Detailinformationen kann bestimmt werden, indem man ihre Abstände zur Kamera vergleicht. Dieser Abstand $d_1$ zur entfernten geraden Wand ergibt sich aus:

$$d_1 = \frac{w + 0.5}{sin(\theta_e)} * cos(\theta_e)$$

Für die oben genannten typischen Werte ergibt das einen Abstand $d_1 = 17.4$. Der Abstand $d_2$ der mittleren Wand der *Perspective Wall* zur Kamera wird wie folgt berechnet:

$$d_2 = \left(\frac{w * cos(\theta_f) + 0.5}{sin(\theta_e)}\right) cos(\theta_e) - w * sin(\theta_f)$$

Daraus resultiert für die als typisch angesehenen Parameter ein Wert $d_2 = 5.2$. Das bedeutet, daß bei Darstellung der gleichen Informationsmenge in Form eines 2D-Layouts die Details auf der mittleren Wand der *Perspective Wall* mindestens drei mal größer als auf einer umgeklappten Wand angezeigt werden können.

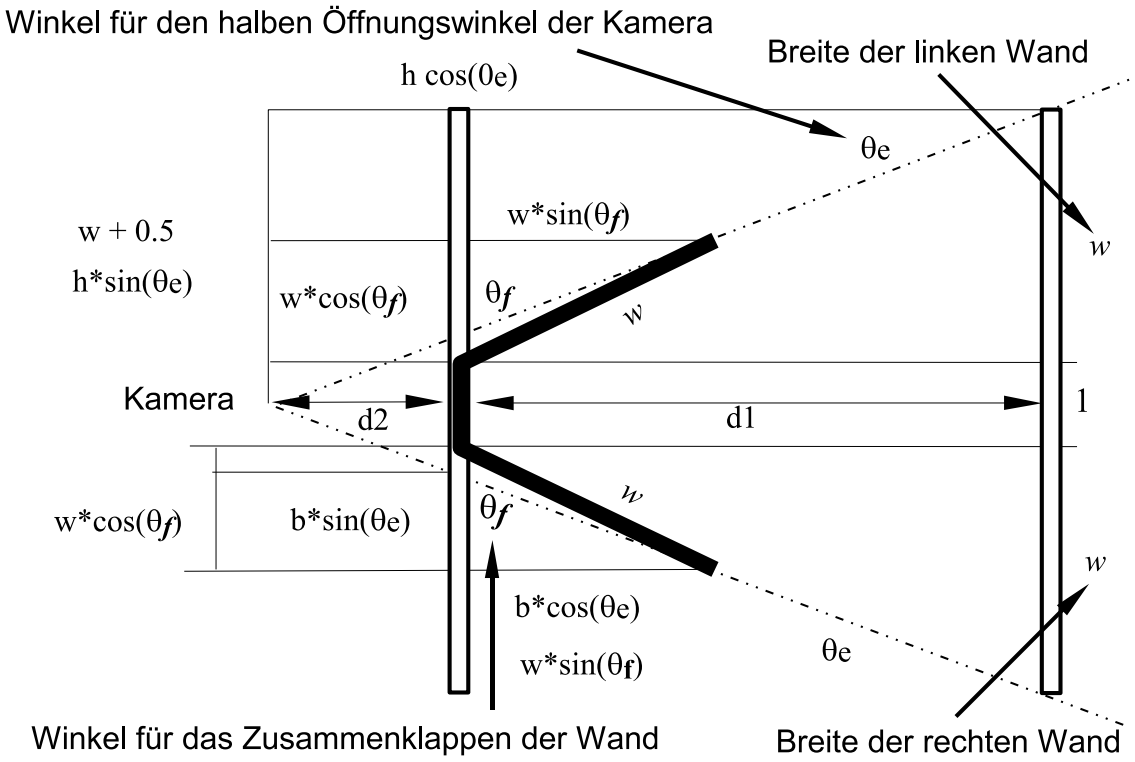



Abbildung 3.2: Analysis [Rob91a]

**Bewertung des „Perspective Wall“ - Verfahrens**

*Benutzeroberfläche*

- *Strukturiertheit:*
  Man bekommt auf den ersten Blick einen guten Einblick in die Struktur und Anordung der Informationsmenge. Die Informationen sind übersichtlich in Reihen und Spalten angeordnet. Die Anordung der Informationen nach Datum, Menge von Dokumenten und ihrer Beziehungen kann sowohl im Detail als auch in der Entfernung erkannt werden.
  Probleme können bei sehr großen Dokumentmengen auftreten. Die Dokumente werden sich überlappen und der Überblick der gesamten Struktur geht verloren. Die Informationen, die sich auf den umgeklappten Wänden befinden, erscheinen verkürzt dargestellt, so daß sie nicht mehr lesbar sind. Ferner können Diskontinuitäten an den Wandecken auftreten.

- *Erkennbarkeit:*
  Bei der dargestellten Struktur gibt es eine klare Trennung zwischen Fokus- und Kontextansicht. Mehrere Selektionen können relativ gut erkannt werden, da die Gesamtstruktur permanent sichtbar ist. Deutlich zu erkennen sind die Datensätze innerhalb der Fokussierung, da sie größer dargestellt sind. Aufgrund der nach Dateiarten und nach Datum sortierten Anordnung der Informationen sind die Datensätze schnell in der Detailansicht zu finden und zu identifizieren. Die Dokumente werden wie kleine Karteien dargestellt. Durch ihre Position, Form, Textur oder Farbe werden Informationen über sie vermittelt.
  Die Datensätze in den hinteren Bereichen der Seitenwände sind schwer selektierbar. Sie erscheinen perspektivisch verkleinert und können nicht genau von den anderen unterschieden werden. Deutlich erkennbar werden sie erst durch eine Veränderung des Fokusses. Durch diese Manipulation kann es für den Benutzer schwierig sein, den Überblick zu behalten. Attribute

wie Textur und Text sind auf die kleinen Symbole im hinteren Bereich nur schwer anzuwenden und zu erkennen.

- *Lokalität*
  Es werden keine Interaktionsmöglichkeiten innerhalb der Symbole und keine Fokussierung mehrerer Bereiche unterstützt.

- *Orientierung:*
  Während ein Informationsbereich im Detail zu betrachten ist, sind in der Nähe andere Symbole mit weniger Details noch gut sichtbar. Diese Eigenschaft erlaubt dem Benutzer, das Verhältnis von Detail und Kontext zu regulieren und einen gewissen Sinn für Orientierung in der Struktur zu behalten.

*Interaktion*

- *Wechselwirkung:*
  Ein ausgewähltes Symbol wird auf der Wand entlang einer definierten Bahn in das Zentrum des Fokus verschoben. Dieser Prozeß wird mittels einer leichten Animation vollzogen, die dem Benutzer hilft, das Symbol immer in seiner konstanten Form während der Bewegung zu betrachten. Eine Selektion der weit entfernten Symbole kann vorgenommen werden, wenn sie sich auf der mittleren Wand oder zumindest in ihrer Nähe befinden. Bei diesem Verfahren sind die Detailinformationen ohne Zoom erkennbar.

- *Ausführungsminimalität:*
  Die Übergänge zwischen Fokus und Kontext kann der Benutzer leicht regulieren, da sich die Wand einfach strecken läßt. Mit Hilfe von Dialogfunktionen kann der Benutzer mit der Visualisierung interagieren. Mit einer fließenden Animation erscheint das selektierte Element vergrößert im Zentrum des Blickfeldes.

*Funktionalität*

- *Darstellung:*
  Das Layout ist in einer zweidimensionalen Form dargestellt. Die Repräsentation der Daten ist deswegen auf Textflächen beschränkt. Diese Art der Präsentation eines Informationsraumes ist leicht verständlich. Außerdem sind keine komplexen Interaktionen zum Erforschen der Informationsmenge erforderlich.

- *Datenreduktion:*
  Die *Perspective Wall* ist Teil eines größeren Versuches der Entwicklung effektiver Raum- und Zeit-Techniken, um auf große Informationsräume zuzugreifen und diese zu verwalten. Eine Datenreduzierung ist nicht das Ziel.

- *Erweiterbarkeit:*
  Das Verfahren arbeitet „nur“ mit einem zweidimensionalen Layout. Das Layout besteht aus 2D-Vektoren (Datum-Achsen und Dateiarten-Achsen), wobei Textsymbole positioniert sind. Es ist in sich abgeschlossen und kann nicht erweitert werden.

- *Details:*
  In der Detail-Ansicht ist es möglich, mehr Informationen über die Dokumente im Fokusbereich zu bekommen. Dokumentinhalte können aus dem Informationsraum heraus abgerufen und in einem externen Editorfenster angezeigt werden.

**Fazit**
Eine Version der *Perspective Wall* ist im *Information Visualizer*[1] implementiert und integriert worden [Rob92]. Das Verfahren ist für verschiedene Typen von Informationen genutzt worden, wie auch für Mitteilungen und Berichte [Rob93]. Es wird u.a. als Visualisierungstool der Visualisierungssoftware *VizControls*[2] eingesetzt. Nach der „User Interface Research (UIR)“[3] sind die letzten Veröffentlichungen von *Perspective Wall* 1991 erschienen. Aktuelle Informationen über dieses Verfahren sind nicht verfügbar.
Die intuitive Verzerrung des Layouts erlaubt eine wirkungsvolle Nutzung des vorhandenen Raumes und außerdem einen leichten Übergang zwischen den Wänden oder Ansichten. Untersuchungen dieses Verfahrens haben gezeigt, daß diese Technik eine dreifache Verbesserung der Erkennbarkeit gegenüber einer einfachen 2D-Visualisierung bringt. Die Daten werden aber dennoch in einer linearen Struktur dargestellt, die zu besagtem 2D-Layout führt und demzufolge ist die Darstellung in Abhängigkeit von „nur“ zwei Parametern angeordnet.

### 3.2.2 Hyperbolic Browser

Der Hyperbolic Browser [Lam95] ist ein System zur Visualisierung und Manipulation großer Hierarchien mittels hyperbolischer Geometrie. Er wurde am Xerox Parc (Palo Alto Research Center) entwickelt und 1995 auf der Conference on Human Factors in Computing Systems (CHI95) vorgestellt.
Das Verfahren gehört zu den *Fokus+Kontext*-Visualisierungsverfahren [McG01]. Die hyperbolische Ebene ist eine nicht-euklidische Geometrie, in der parallele Linien voneinander weg laufen. Dies führt zu der interessanten Eigenschaft, daß der Umfang eines Kreises auf der hyperbolischen Ebene exponential mit seinem Radius wächst. Das heißt, daß mehr Abstand exponential mit zunehmender Entfernung verfügbar ist.
So können die Hierarchien im hyperbolischen Raum in einer konstanten Weise ausgebreitet werden, damit der Abstand (wie in der hyperbolischen Geometrie gemessen wird) zwischen Eltern, Kindern und Geschwistern überall in der Hierarchie ungefähr gleich ist [Lam95, Fer00].
Wie die Abbildung 3.3 zeigt, stellt ein *hyperbolischer Browser* zunächst einen Baum mit seiner Wurzel im Fokusbereich des Anzeigefelds dar, wobei sich die restliche, an das Wurzelelement hierarchisch verknüpfte Struktur nach allen Seiten (im Kontextbereich) sternförmig und verzerrt ausbreitet. Der ganze Baum wird auf eine zweidimensionale kreisförmige Fläche projiziert. Die Knoten stehen für einzelne Dokumente und die Linien für Verbindungen (Links) zwischen den Dokumenten. Alle Knoten sind beschriftet. Den Informationsknoten in der Nähe des Ursprungs wird mehr Platz als den anderen Knoten in der Fläche zugewiesen. Der Kontext schließt mehrere Generationen von Eltern, Geschwistern und Kindern ein. Die Kinder werden mit gleicher Entfernung in einem Kreisbogen um diesen Verzweigungspunkt herum angeordnet. Anschließend erhält jedes Kind einen Verzweigungspunkt für seine eigenen Nachfolger. Die Anzeige kann dann in alle Richtungen frei verschoben werden, wodurch andere Knoten ins Fokusfeld gelangen. Je weiter sie an den Rand des kreisförmigen Anzeigefelds gelangen, desto kleiner werden sie und schaffen dadurch Platz für die Knoten, welche nun weiter in das Zentrum gerückt sind.

[1] *Information Visualizer* ist ein experimentelles System für Information Retrieval und wurde am Xerox Parc (Palo Alto Research Center) entwickelt. Das System basiert auf den drei Hauptkomponenten *Information Visualization*, *3D Room* und *Kognitive Koprozessor*

[2] VizControls ist eine Informations-Visualisierungssoftware von InXight Software zur Visualisierung großer Hierarchien.

[3] User Interface Research ist eine Forschungsgruppe der Informationswissenschaft des Palo Alto Research Center (Parc). Informationen sind unter der Adresse http://www2.parc.com/istl/projects/uir/ zu finden.

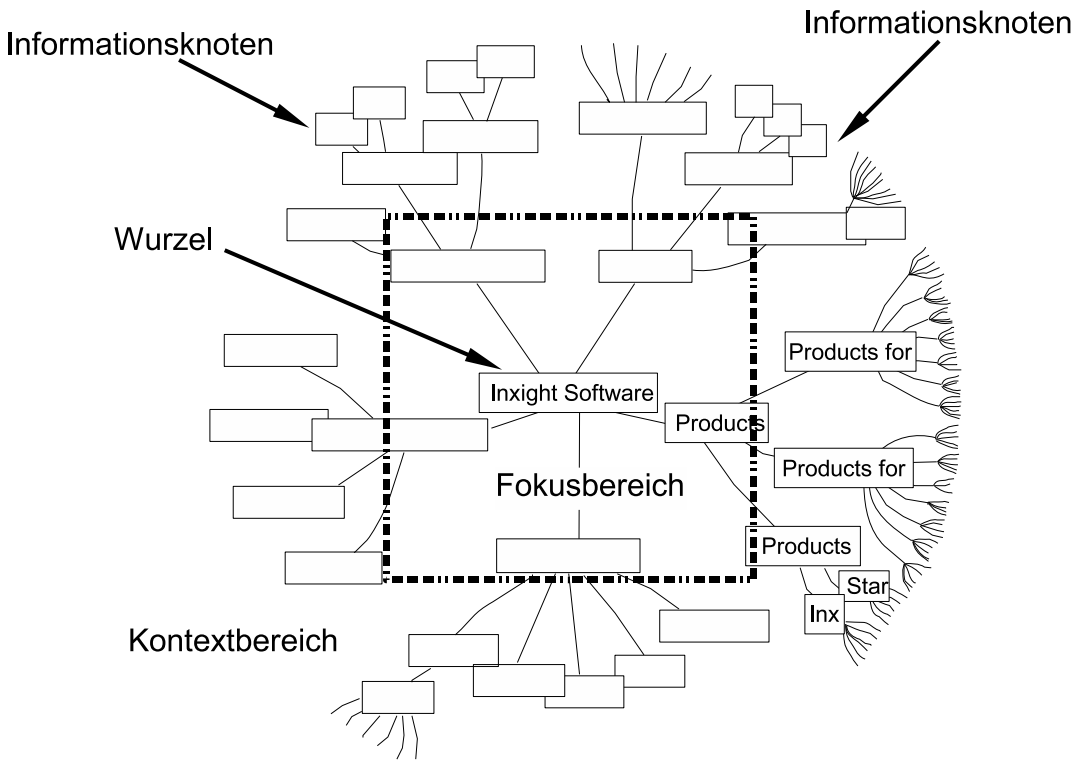



Abbildung 3.3: Darstellung einer Informationsstruktur mittels Hyperbolic Browser [Inx00]

***Bewertung des „Hyperbolic Browser" – Verfahrens***

*Benutzeroberfläche*

- *Strukturiertheit:*
  Besonders hervorgehoben wird bei diesem Verfahren das *Fokus+Kontext*-Prinzip. Dabei kann jedes Dokument im Detail angesehen werden, wobei die restlichen Begriffe als Kontext verbleiben und Muster erzeugen, die einerseits für ein besseres Verständnis des gesamten Datensatzes fungieren und andererseits die Zusammenhänge besser erkennen lassen sollen. Das Prinzip soll dem Betrachter zu einem schnellen Überblick der ganzen Darstellung verhelfen. Die gesamte vorhandene Fläche wird für die Darstellung gut ausgenutzt und es können verschiedene strukturelle Merkmale abgebildet werden. Nicht alle Knoten der Struktur können deutlich gesehen werden. Die Knoten im Fokusbereich erhalten mehr Platz und sind dadurch deutlich sichtbar, während die Knoten am Rand des Kreises klein oder überhaupt nicht zu sehen sind.

- *Erkennbarkeit:*
  Am Anfang einer Sitzung befindet sich die Wurzel im Mittelpunkt. Die Knoten und Merkmale im Fokusbereich sind deutlich zu erkennen. Am Rand des Anzeigefelds sind Zweige ohne Knoten zu sehen. Um diese zu erkennen, müssen die gewünschten Elternknoten interaktiv gezogen werden, so daß die versteckten Kinderknoten sichtbar werden. Mit der Anwendung von Farben können Knoten nach Merkmalen gruppiert werden, so daß die Suche erleichtert und zielgerichtet durchgeführt werden kann.
  Die Knoten haben unterschiedliche Größen, die im Extremfall sogar ein Pixel betragen können. Der Unterschied der Objektgrößen kann auf bestimmte Weise einen negativen Einfluß auf die Wahrnehmung des Benutzers haben. Eine Verbesserung dafür ist in einer möglichen

Skalierung der Knoten zu sehen. Das kann jedoch dazu führen, daß der durch eine perspektivische Ansicht erzeugte Fokus zerstört wird.

- *Lokalität:*
  Ein Maximum an Lokalität ist bei diesem Verfahren schon durch das Fokusprinzip gegeben. Generell erfolgt die Interaktion mit dem Knoten, der sich im Fokusbereich der Darstellung befindet. Es besteht beispielsweise die Möglichkeit, in dem Baum nach bestimmten Knotentexten zu suchen.

- *Orientierung:*
  Durch das Bewegen eines Knoten ins Zentrum werden die anderen Knoten in der Anzeige einer Rotation unterzogen. Die am Rand der Anzeige befindlichen Knoten werden davon stärker betroffen. Diese Art der Animation ist einigermaßen überschaubar. Beim interaktiven Verschieben von Knoten (durch Ziehen mit der Maus) werden alle anderen Knoten in der Nähe der Start- und Zielposition auf kaum nachvollziehbare Art gedreht. Die Sicht auf die Struktur wird dabei komplett verändert, was für den Benutzer die Notwendigkeit der Neuorientierung zur Folge hat. Wird ein Knoten allein auf einer Kreisbahn auf seine ursprüngliche Position zurückverschoben, so führt dies zu einer Rotation der Gesamtansicht. So hat der Benutzer, der einen Knoten nach gewisser Zeit wieder in den Fokus gebracht hat, einen anderen Blickpunkt auf diesen, obwohl er die Hierarchie nicht gedreht hat. Diese Inkonsistenz kann beim Benutzer Verwirrung hervorrufen. Bei der Untersuchung von mehreren hervorgehobenen Knoten, muß die gesamte Darstellung jedesmal neu aufgebaut werden. Dadurch geht die Übersichtlichkeit zum Teil verloren und die Richtung zur Wurzel der Hierarchie ist nicht mehr zu erkennen.

*Interaktion*

- *Wechselwirkung:*
  Eine Fokusveränderung erfolgt durch Anklicken eines sichtbaren Knotens, um diesen in die Mitte zu rücken, oder durch das Ziehen eines Knoten in eine andere Position. Daraufhin wird die gesamte Darstellung neu geordnet und aufgebaut und somit der Fokus verändert. Mit Hilfe von Schaltflächen oder mit der Maus können die Abstände zwischen den Knoten und die Schriftgröße verändert werden. Die Knotentexte können lang oder verkürzt angezeigt werden. Es besteht die Möglichkeit, in dem Baum nach einem bestimmten Text zu suchen und durch einen Klick auf die Home-Schaltfläche gelangt man zum Ausgangspunkt zurück.

- *Ausführungsminimalität*
  Die gesamte Struktur ist visualisiert und zu jedem Zeitpunkt sichtbar, das Scrollen in der Darstellung ist also nicht nötig. Grund dafür ist die angebotene Interaktivität, die das Umherschieben der einzelnen Knoten erlaubt, sowie die verzerrte (hyperbolische) Anordnung der Elemente. Die vom Benutzer auszuführenden Aktionen sind wenig komplex und wurden übersichtlich umgesetzt.

*Funktionalität*

- *Darstellung*
  Die Darstellung ist so konzipiert, daß sie „fast“ eine Gesamtübersicht gewährt. Die Baumhierarchie wird in einer hyperbolischen Ebene ausgebreitet und auf eine zweidimensionale, kreisförmige Fläche projiziert. Der Kontext ist in Form von Vorgänger-, Nachfolger- und Nachbarknoten dargestellt.

- *Datenreduzierung*
  Eine Reduktion der darzustellenden Datenmenge ist bei diesem Verfahren nicht vorgesehen.

Da es sich um eine zweidimensionale Darstellung mit geringer Komplexität handelt, ist der Ressourcenbedarf eher gering. Weiterhin wird der größte Teil der Elemente ohnehin sehr stark verkleinert (und damit unkenntlich) am Rande des Kreises dargestellt. Eine Verringerung der Anzahl der auszugebenden Elemente ist daher nicht notwendig, die Reduktion findet durch die Darstellung selbst statt.

- *Erweiterbarkeit*
  Der *Hyperbolic Browser* kann nicht um weitere grafische Verfahren oder Darstellungsformen erweitert werden, es handelt sich um ein in sich geschlossenes Werkzeug.

- *Details*
  Zu jedem Knoten im Fokus, bzw. zu allen durch direkte Interaktion erreichbaren Knoten kann das gesamte Dokument in einem separaten Browser-Fenster angezeigt werden. Dazu muß das Dokument jedoch komplett verfügbar sein. Eine Anzeige zusätzlicher Vorabinformationen existiert nicht.

**Fazit**
Der *Hyperbolic Browser* wurde für Anwendungen entworfen, die den Zugriff, die Handhabung und die Organisation von Daten gestatten sollen, damit das Untersuchen von großen Datenhierarchien wie z. B. Produktkatalogen, Dokumentsammlungen oder einer Verweisstruktur von einem Web-Host erleichtert wird [Inx00]. Eine Implementierung des Hyperbolic Browsers gewann auf der CHI97 einen Wettbewerb als effizientestes Navigationswerkzeug für hierarchische Strukturen [Mul97]. Das Verfahren ist in vielen kommerziellen Websites und Produkten wie MapIt und Hyperbolic Reader eingesetzt worden. In der Site von Inxight können die Produkte mit *hyperbolic Browser* betrachtet werden, aber kaum in akademischen Recherchen.
Das Verfahren ist durch seine große Vielfalt an visuellen Parametern ein flexibles Navigationswerkzeug mit einem ansprechenden Design. Es ermöglicht eine effiziente Navigation in der Gesamtstruktur und kann mengenmässig viel Information pro Fläche darstellen. Problematisch dabei ist, daß nur ein geringer Teil der Information deutlich sichtbar (im Fokusbereich) dargestellt wird. Schwierigkeiten sind bei breiten Hierarchien zu beobachten. Die Darstellungen der Knoten können sich überschneiden und Knotentitel sind nur im Fokusbereich zu sehen.
Das Verfahren erlaubt dem Benutzer, effizient und leicht durch große Mengen von Informationen zu steuern. Es verkörpert eine leistungsfähige „sehen und gehen“–Interaktionsart, die lediglich das Anklicken eines einzigen Punktes erfordert. Nachteile bestehen in den Schwierigkeiten bei der Einschätzung der Richtung von Verweisen und bei der Einordnung eines Knotens in die Gesamtstruktur.

### 3.2.3 Cone Tree und Cam Tree

Das erste Konzept dazu wurde 1991 vom Xerox Parc (Palo Alto Research Center) vorgestellt. Dann wurde es Bestandteil des *Information Visualizer*, eines *information workspace*[4] zur Untersuchung großer Datenmengen verschiedener Typen [Rob91b]. Diese Verfahren zählen zu den *Fokus+Kontext*-Verfahren.
Cone Tree und Cam Tree bilden hierarchisch gegliederte Informationen in Form von verknüpften durchsichtigen Kegeln auf eine Baumstruktur ab, welche die räumliche Tiefe des Bildschirmbereichs zur Darstellung nutzt [Rob91b]. Die grafischen Darstellungen können eine vertikale hierarchische Struktur (*Cone Tree*) bzw. eine horizontale Struktur (*Cam Tree*) aufweisen. In der Abbildung 3.4 wird eine Darstellung mit *Cone Tree* gezeigt. Die Wurzel ist an der Spitze eines Kegels nah der

[4]Ein *information workspace* ist eine Arbeitsumgebung, die zur Visualisierung von Informationen dient

Oberseite der Anzeige plaziert. Der Ausgangspunkt jedes Kegels wird als „Knoten“ bezeichnet. Beim *Cam Tree* wird die Wurzel am linken Bildschirmrand dargestellt. Alle Kinderknoten werden dann an gleichen Abständen entlang der Basis des Kegels verteilt. Dieser Prozeß wird für jeden Knoten in der Hierarchie mit einem geringeren Durchmesser der Kegel wiederholt, der sich auf jedem Nievau weiter verringert, während die Hierarchie absteigt. Dies ist erforderlich, um genügend Raum für die Anzeige aller Blattknoten bereitzustellen.
Alle Hierarchieebenen der Struktur haben die gleiche Höhe. Das Größenverhältnis des Baumes wird anhand des verfügbaren Raumes berechnet, so daß jeweils die gesamte Hierarchie auf dem Bildschirm dargestellt werden kann. Ebenso wird auch der Durchmesser jedes einzelnen Kegels berechnet, damit die unterste Ebene auf die Grundfläche des Raumes paßt. Das Abbildungsverhältnis von Breite und Tiefe des Baumes wird dabei entsprechend angepaßt.

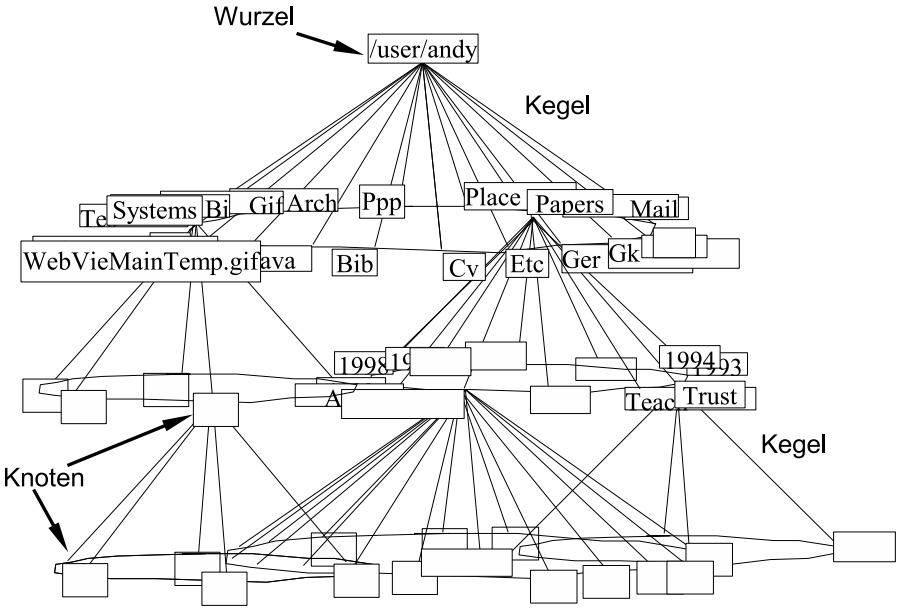



Abbildung 3.4: Informationsdarstellung mittels Cone Tree [Rob91b]

Der folgende mathematische Ansatz beschreibt, wie der Radius $R(h)$ eines Baumes bei den Höhenniveaus $h$, berechnet werden muß, um alle Elemente des Baumes im zur Verfügung stehenden Raum unterzubringen [Car95].

$$R(h) = \frac{R_i}{h+1}$$

Dabei ist $R_i$ eine festgelegte Initialgröße. Dies ist in gewissem Maße ausreichend, bei größeren Hierarchien werden jedoch Überlappungen der Kegel auftreten. Um dies zu vermeiden, müssen sich die Radien der Kegel automatisch anpassen. Dazu sind Ansätze in [Fur86, Koi93] beschrieben, auf die hier jedoch nicht näher eingegangen werden soll.

***Bewertung des „Cone Tree“- bzw. „Cam Tree“ – Verfahrens***

*Benutzeroberfläche*

- Strukturiertheit
  Durch die Anwendung der dritten Dimension ist es möglich, den Platz auf dem Bildschirm sinnvoll auszunutzen und die vollständige Visualisierung der Struktur zu realisieren. Eine Visualisierung mittels dieses Verfahrens wird bis zu einer bestimmten Hierarchiegröße ein konstantes Abbildungsverhältnis ergeben. Wenn das nicht der Fall ist, kann es möglich sein, daß der Benutzer den Überblick über die Struktur verliert. Eine Baumstruktur kann ein günstiges Abbildungsverhältnis für die Bildschirmdarstellung erreichen, wenn die Baumtiefe (bis zu 10) und die Baumbreite (von ca. 30) eine Darstellung der Objekte in gut wahrnehmbarer Größe gestatten [Rob91b]. Um die Sichtbarkeit der weiter hinten im Raum befindlichen Kegel zu gewährleisten, ist die Hülle der Kegel transparent. Dadurch bleibt der Überblick über die Struktur erhalten.

- *Erkennbarkeit*
  Die Knoten, die vom Benutzer betrachtet werden, befinden sich im Fokuspunkt und aufgrund der Perspektive erscheinen sie größer als die anderen Knoten. Der Fokuspunkt wird als „Fisheye-Effekt“[5] bezeichnet. Den Knoten werden Form, Farbe und Textur zugewiesen. Die horizontale Struktur (*CamTree*) bietet mehr Platz für die Beschriftungen als die vertikale Struktur (*Cone Tree*). Die Einordnung einer Selektion ist durch die Hervorhebung des Pfades von der Wurzel bis zu dem selektierten Knoten zu erkennen. Probleme können bei der Selektion mehrerer Knoten eines Kegels auftreten.

- *Lokalität*
  Eine Interaktion ist lokal begrenzt. Sie kann nur mit den Objekten, die sich an der vorderen Seite befinden, realisiert werden. Ohne eine Drehung der Kegel ist eine Interaktion mit den hinteren Objekten nicht möglich. Das Ergebnis einer Handlung ist sofort ersichtlich. Es ist leider keine gleichzeitige mehrfache Fokussierung möglich.

- *Orientierung*
  Ein selektierter Knoten kann in einer nachvollziehbaren Geschwindigkeit und durch eine geeignete Rotation in den Vordergrund gedreht werden, so daß eine Orientierung bezüglich der Lage des Knotens in der Baumstruktur möglich wird. Auf die Basisebene werden semitransparente Schatten projiziert, um die Orientierung innerhalb der Baumstruktur zu erleichtern und den Eindruck einer dreidimensionalen Form zu verstärken.
  Beim *Cam tree* soll dem Benutzer die Orientierung innerhalb der Baumstruktur durch die Projektion der Struktur auf die Basisebene zusätzlich erleichtert werden.
  Wie beim *Cone Tree* können die selektierten Knoten interaktiv in den Vordergrund gedreht werden. Der Pfad von der Wurzel zu diesem Knoten wird hervorgehoben. Die Rotation wird animiert, so daß die Veränderung der Ansicht vom Benutzer gut nachvollzogen werden kann.

*Interaktion*

- *Wechselwirkung:*
  Die Kegel sind um ihre Achse drehbar. So können alle Knoten mit Hilfe des Mauszeigers betrachtet und selektiert werden. Ein selektierter Knoten wird mit den anderen Knoten im Pfad in den Vordergrund gerückt und hervorgehoben. Die Rotation der Substruktur erfolgt gleichzeitig entlang des kürzesten Rotationsweges.

[5] Mit dem Fisheye-Prinzip lässt sich eine Art der Präsentation erreichen, bei der der Detaillierungsgrad der dargestellten Objekte mit der Nähe zum Kontext, des im Zentrum befindlichen Objektes, kontinuierlich zunimmt

- *Ausführungsminimalität:*
  Mittels Suchfunktionen kann man nach Beschriftungen im Baum suchen. Die einzelnen Kegel können gedreht werden. Sie können ebenfalls auf- und zugeklappt werden. Diese Prozesse werden vom Benutzer direkt und intuitiv durchgeführt.

*Funktionalität*

- *Darstellung:*
  Die hierarchische Struktur wird in Form von verknüpften Kegeln im dreidimensionalen Raum dargestellt, wobei die Objekte mit verschiedenen Formen repräsentiert werden können. Bei diesem Verfahren kann der Benutzer eine Rotation ausführen, eine Zoomfunktion ist nicht vorhanden.

- *Datenreduzierung:*
  Eine Reduktion der Datenmenge durch Auswahlverfahren ist nicht vorgesehen.

- *Erweiterbarkeit:*
  Die Darstellung ist nicht erweiterbar.

- *Details:*
  Eine Anzeige detaillierter Informationen innerhalb der Darstellung ist nicht vorgesehen.

**Fazit**
Dieses Verfahren wurde in verschiedenen Anwendungen eingesetzt. Ein Unix File System ist mit *Cone Tree* dargestellt worden. Die Hierarchie des Systems enthielt ungefähr 600 Verzeichnisse und 10,000 Files [Cha98]. Ein anderes Beispiel ist die organisatorische Struktur der Xerox Corporation, wobei die längste Hierarchie 650 Körperschaftsgruppen enthielt. Weiterhin wurden Funktionsplanungen von Unternehmen mit diesem Verfahren visualisiert. Es gibt nach [Rob91b] andere zahlreiche potentielle Anwendungen wie Software-Modul-Management, Dokumentenmanagement bis hin zu lokalen Netzen.
Es bestehen Einschränkungen bei der Anzahl der darstellbaren Knoten. Wie in der Bewertung des Verfahrens beschrieben wurde, liegt die maximale Anzahl von Knoten für eine geeignete Darstellung bei ungefähr 1000, bei einer Baumtiefe von ca. zehn Hierarchie-Stufen und nicht mehr als 30 Nachfolgern auf der untersten Hierarchiestufe. Das heißt, die Verfahren können nicht zur Wiedergabe großer Mengen von Informationen genutzt werden, sonst ist ein günstiges Abbildungsverhältnis nicht zu erreichen.

### 3.2.4 Informationswürfel (Information Cube)

Der Informationswürfel ist ein dreidimensionales Abbildungsverfahren zur Visualisierung großer hierarchischer Informationsstrukturen mittels verschachtelter durchsichtiger Würfel. Er wurde von Rekimoto und Green [Rek93] entwickelt und 1993 vorgestellt.
Das System zeigt die Würfel in einer dreidimensionalen Darstellung. Die hierarchischen Informationen werden durch ineinander geschachtelte Würfel dargestellt, die die Baumstruktur repräsentieren. Dabei wird mit Hilfe von Semitransparenz die Schachtelungstiefe und damit der Informationsgehalt festgelegt und gesteuert. Der äußerste Würfel symbolisiert die Informationen der obersten Ebene, der alle weiteren Würfel enthält. Jede zweite Ebene eines Würfels enthält eine dritte Ebene von Würfeln und so weiter. Den Würfeln, die keine weiteren Verschachtelungen enthalten, kann ein Titel zugewiesen werden, der auf der Oberfläche dargestellt wird. Jeder Würfel hat einen Namen auf seiner Oberfläche. Das System zeigt diese Würfelhierarchie in dreidimensionaler Darstellung. Der Informationswürfel kann beliebige Informationen enthalten und ist nicht darauf beschränkt,

nur weitere Würfel zu enthalten, andere 3D-Visualisierungen oder Informationen können ebenfalls im Inneren dargestellt werden [Rek93, You96].
In diesem System ist eine *Virtual-Reality*–Schnittstelle implementiert. Es kann ein HMD (Head-mounted display)[6] oder ein herkömmliches CRT[7] benutzt werden, um das Bild anzuzeigen oder die Würfel in der Hierarchie zu manipulieren.
Es wird eine automatische Skalierung durchgeführt, damit der Benutzer die kleinen Würfel leicht sehen kann. Die Translation und Skalierung werden nicht plötzlich vollzogen. Das System kontrolliert die Geschwindigkeit der Animation, um die Übergänge natürlich zu präsentieren, so daß der Benutzer durch die Änderung des Fokus nicht verwirrt wird. Die Geschwindigkeit wird so gesteuert, daß die Ausgangs- und abschließende Geschwindigkeit geringer als die Zwischengeschwindigkeit ist. Der Benutzer soll den Eindruck gewinnen, daß die Bewegung durch natürliche Beschleunigung und Geschwindigkeitsverminderung gesteuert wird.
Die Größe eines Würfels wird berechnet, indem man die Größe seiner Nachfolger in der Hierarchie rekursiv aufsummiert. Um die Gesamtgröße der Würfel angemessen zu halten, wird der Normierungsfaktor schrittweise für die inneren Ebenen verringert. Wird beispielsweise der äußere Würfel mit 100% eingestuft, so wird der Würfel der folgenden Ebene mit 90% skaliert, der in der dritten Ebene mit 81% und so weiter. Wenn der Benutzer auf eine spezifische Ebene fokussiert, skaliert das System wieder automatisch die Würfel , damit optisch eine gleichmäßige Abstufung der Ebenen erreicht wird. In der Abbildung 3.5 ist eine Darstellung mit dem *Informationswürfel* gezeigt.

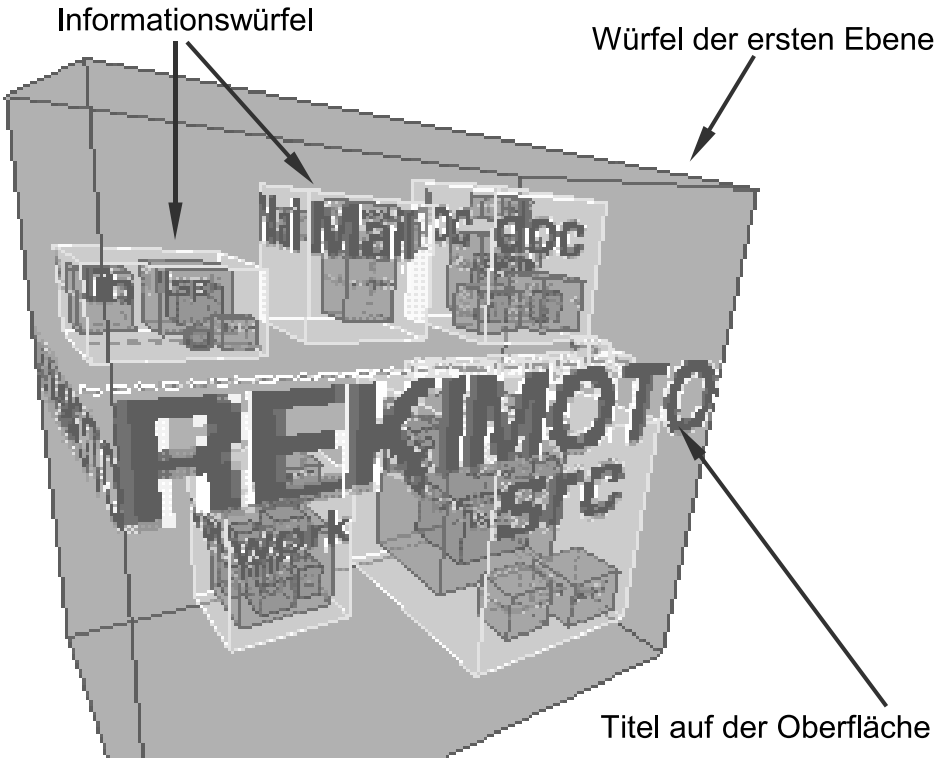



Abbildung 3.5: Informationsdarstellung mittels Informationswürfel (Information Cube) [Rek93]

[6]Computerendgerät, das auf dem Kopf getragen wird und eine annähernde Verbindung über Sehen, Hören oder Sprechen mit der virtuellen Realität erlaubt.
[7]CRT(Cathode Ray Tube – Kathodenstrahlröhre ) erzeugt die Bilder z.B. in TV-Geräten oder Computermonitoren

**Bewertung des „Informationswürfel“ (Information Cube) – Verfahrens**

*Benutzeroberfläche*

- *Strukturiertheit*
  Bei diesem Verfahren steht ein Fokuspunkt zur Verfügung, somit sind die Informationen aus verschiedenen Teilen der Hierarchie schwer in Beziehung zu setzen. Die visualisierte Struktur von ineinander geschachtelten Würfeln stellt geringe Anforderungen an das räumliche Vorstellungsvermögen des Benutzers. Zusammengehörende Elemente werden von einer einfachen geometrischen Form umschlossen, wodurch die visuelle Komplexität der Visualisierung verringert wird, ohne die Informationsdichte zu reduzieren. Die Struktur ist schwer als Ganzes wahrzunehmen, wenn es sich um eine tiefere Hierarchie handelt. Eine große Hierarchie, die besonders viele Dateien in den inneren Würfeln der höheren Ebenen enthält, kann eine Informationsverstopfung zur Folge haben.

- *Erkennbarkeit*
  Jeder Würfel ist mit einer halbtransparenten Farbe gerendert, so daß man das Innere des Würfels sehen kann. Es sind Probleme bei der Selektion und Wahrnehmung von Objekten zu erwarten, wenn die Hierarchie sehr groß ist. Insbesondere Objekte mit vielen Nachfolgern können zu unübersichltichen Bereichen führen. Die Einordnung von Objekten in die Gesamtstruktur kann problematisch sein. Bei Veränderung der Ansicht kann die Position eines vorher ausgewählten Objektes aus der Visualisierung nicht entnommen werden. Texte sind an der Oberfläche von Würfeln anzubringen, wobei die Länge auf die Größe des Würfels beschränkt ist. Unter Umständen verdecken die Texte die Sicht auf die weiter im Inneren liegenden Würfel, dadurch wird die Erkennung oder die Selektion eines Objekts schwierig.

- *Lokalität*
  Die verschiedenen Interaktionselemente aus dem Bereich der virtuellen Realität (VR) ermöglichen ein freies Bewegen in der Szene und eine direkte Interaktion mit ihren Objekten unabhänging von der Position des Betrachters.

- *Orientierung*
  Die Orientierung des Würfels ist direkt an die Orientierung des Datenhandschuhs gekoppelt. Der Würfel kann vom Benutzer durch Drehung der Hand frei gedreht werden. Ebenso kann die Position des Würfels durch Handbewegungen verändert werden. Ist die Bewegung der Hand beendet, werden die Orientierung und die Position festgestellt.

*Interaktion*

- *Wechselwirkung*
  Der Benutzer kann einen Würfel unter Verwendung des Datenhandschuhs (DataGlove) bewegen und rotieren und für eine ausführliche Betrachtung zu ihm hin navigieren.
  Mit einer Handgeste kann der Benutzer einen Würfel selektieren. Das System wählt den fokussierten Würfel aus und ändert die Blickrichtung des Benutzers zur Position des Würfels hin. So kann der Benutzer die gesamte Form des fokussierten Würfels vor sich sehen. Daher arbeitet diese Methode auch als Navigationsmethode. Die Skalierung wird ebenfalls automatisch vollzogen, damit der Benutzer den kleinen Würfel leicht sehen kann.
  Bei der Rotation kann der Benutzer den Würfel frei drehen, indem er die Hand dreht. Die Rotation des Würfels ist auf eine Achse beschränkt, um beim Benutzer Orientierungsschwierigkeiten zu vermeiden. Genauso kann der Benutzer die Position des Würfels durch Bewegung der Hand ändern.

- *Ausführungsminimalität*
  Die Interaktionselemente befinden sich in direktem Kontakt mit dem Benutzer. Die Interaktion (Rotation und Selektion) mit dem Datenhandschuh erscheint sehr natürlich, da das Ergreifen der Würfel mit der Hand simuliert wird. Es soll an dieser Stelle erwähnt werden, daß dieses Verfahren im Jahre 1993 entwickelt wurde – in der Zeit eines „VR-Booms“. Aus heutiger Sicht haben sich die bekannten VR-Interaktionsmittel nicht auf breiter Front durchgesetzt, ihr Einsatz ist auf einige wenige Bereiche begrenzt. Die Ursachen dafür liegen unter anderem an den nach wie vor hohen Kosten, begrenzter Genauigkeit der Geräte sowie ergonomischen Problemen bei der Handhabung.

*Funktionalität*

- *Darstellung*
  Die Informationen werden vor allem als Würfel dargestellt, es sind jedoch auch andere Formen wie Kugel, Quadrat usw. für die Darstellung der Informationen in der Hierarchie möglich.

- *Datenreduzierung*
  Eine Reduktion der Datenmenge vor dem Darstellungsprozeß ist nicht vorgesehen. Die Menge der dargestellten Objekte wird zwar durch die Semitransparenz geregelt, trotzdem durchlaufen alle Objekte der Hierarchie die Renderingpipeline.

- *Erweiterbarkeit*
  Dies ist ein in sich geschlossenes Verfahren, es existiert keine Schnittstelle für Erweiterungen.

- *Details*
  Die Anzeige von Detailinformationen ist auf das Anbringen eines Textes auf einem Würfel begrenzt. Hier sei nochmals angemerkt, daß sich die dazu notwendige Technologie damals erst in der Entwicklung befand. Auf dem heutigen Stand der Technik sollte die Anwendung der hier genannten VR-Interaktionsmittel kein Problem darstellen.

**Fazit**
Der *Informationswürfel* ist mit dem „Mimimal Reality (MR) Toolkit[8]“ [Gre92, SC92] auf Basis der IRIS GL Graphics Library implementiert worden [Sil91]. Die Implementation des *Informationswürfel* benutzt (für die damalige Zeit) anspruchsvolle Grafikfähigkeiten wie *Texture Mapping* und *Alpha blending* (Transparenz-Berechnung). Es sind mit diesem Verfahren Unix-Verzeichnisse, C++ Klassen-Hierarchien und die Usenet news group dargestellt worden [Rek93].
Das Verfahren ist für hierarchische Darstellungen geeignet. Bei einer kleinen Anzahl von Objekten ist es möglich, ein klares Bild der Struktur des Informationsraumes zu erhalten, wobei der Benutzer sich auch nach der Durchführung von Interaktionen noch gut darin orientieren kann. Bei einer großen Anzahl von Objekten wird aufgrund der hohen Kompaktheit besonders in den inneren Ebenen das Auffinden von benachbarten Objekten mit jedem Interaktionsschritt schwieriger, da nach einer Drehung oder Neufokussierung die alte Ansicht auf die Struktur nicht wieder restauriert wird.

[8]MR Toolkit ist ein Plattformsystem für Anwendungen virtueller Realität (VR). Es wurde in der Universität von Alberta entwickelt

### 3.2.5 LyberWorld

*LyberWorld* [Eng95, Wol96] ist eine interaktive, dreidimensionale Benutzerschnittstelle für Information Retrieval Systeme[9], die am Institut für Integrierte Publikations- und Informationssysteme (IPSI) der Gesellschaft für Mathematik und Datenverarbeitung (GMD) in Darmstadt entwickelt wurde. Es konzentriert sich auf die Visualisierung eines abstrakten Informationsraumes.
Das System *LyberWorld* besteht aus drei Visualisierungstools, die hier weiter beschrieben werden. Bei Beginn einer Suche werden die drei Werkzeuge als 3D-Symbol [Hem97] angezeigt. Die Suche nach Informationen beginnt mit der Eingabe eines Begriffs oder eines bereits bekannten Dokuments. Sofort nach der Eingabe wird der Navigationskegel aktiviert, der die Ergebnisse darstellt.

1. Der *Navigationskegel* visualisiert den Inhalt der Information in Form von Kegeln. Die Oberfläche des Kegels kann eine Menge von Informationen in Form von Textzeilen aufnehmen. Diese spiegeln die Antwortmenge einer Suchanfrage wieder. Dabei können in abwechselnder Reihenfolge Begriffsebene oder Dokumenttitelebene vom System zurückgeliefert werden. In der Abbildung 3.6 ist eine solche Darstellung gezeigt. In der Begriffsebene befinden sich alle Begriffe, die eine Beziehung zu dem Dokument haben und die sich in der Datenbank befinden [Hem94, Hem93b]. Mit einer Auswahl von Begriffs- und Dokumentebenen werden neue Ebenen erzeugt, und so die entprechenden Ebenen weiter expandiert [Hem97].

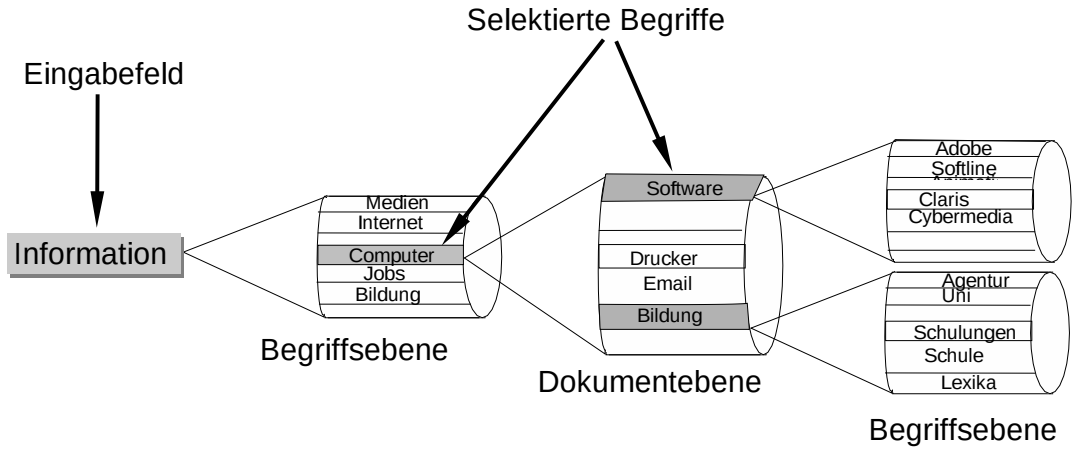



Abbildung 3.6: Navigationskegel mit dem Ergebnis einer Suche [Hem97]

2. Die *Relevanzkugel* ist eine halbtransparente Kugel, wie die Abbildung 3.7 zeigt. In ihrem Inneren befinden sich Informationsobjekte, die als Würfel repräsentiert sind. Die Relevanzkugel setzt sich aus einem Manipulator, Informationsobjekten im Inneren, Referenzobjekten auf der Hülle und einer halbtransparenten mit kreisförmigen Bändern umspannten Kugel zusammen. Der *Manipulator* besteht aus einer kleinen Drahtgitterkugel und vier orthogonal zueinander um die Kugel herum angeordneten 3D-Pfeilen. Eine Verschiebung der gesamten Relevanzkugel erfolgt durch Anklicken und Ziehen der Pfeilsymbole. Bei der Verschiebung in der x–y–Ebene werden alle Komponenten und der Manipulator selbst mit verschoben. Eine Rotation dagegen wird durch Anklicken des Manipulators über der Drahgitterkugel mit der Maus ausgeführt. Die Relevanzkugel mit allen visuellen Komponenten wird frei im Raum um ihren Mittelpunkt rotiert. Die Suchbegriffe werden in Form kleiner Kugeln auf der Oberfläche

[9]Ein Information Retrieval System ist eine Datenbank für wenig strukturierte Informationen. Ein solches System besteht typischerweise aus einer Komponente zur Dateiorganisation, einer Retrieval-Maschine zum Auffinden der Daten und einer Benutzungsschnittstelle zur Interaktion mit den beiden anderen Komponenten.

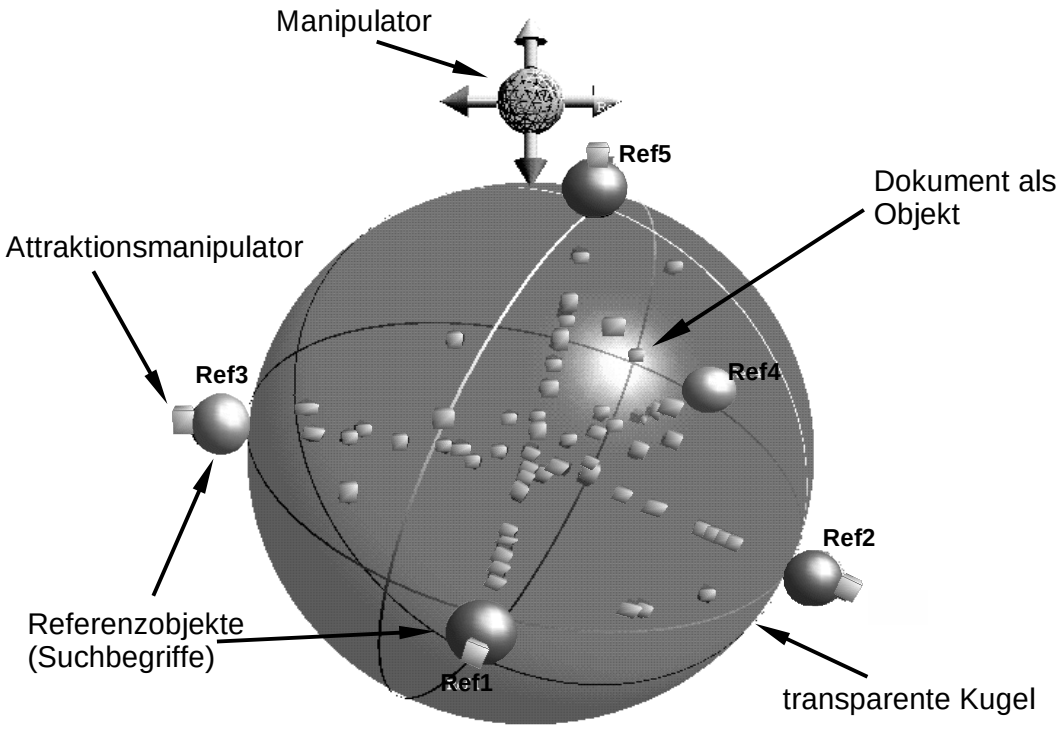



Abbildung 3.7: Relevanzkugel-Visualisierung [Hem97]

der *Relevanzkugel* dargestellt und als *Referenzobjekte* definiert. Die Referenzobjekte haben eine Funktion ähnlich einer Gravitationsquelle, deren Größe die Stärke der Anziehung auf die von ihr abhängigen Informationsobjekte widerspiegelt.
Die *Informationsobjekte* werden als einfache kleine farbige Quader dargestellt. Sie befinden sich im Inneren der Relevanzkugel. Die Position der Informationsobjekte wird durch den Bezug zu den Referenzobjekten, von denen sie abhängen, bestimmt. Es existieren $n$ Referenzobjekte für jedes Informationsobjekt. Die Abhängigkeit zwischen dem $i-ten$ Referenzobjekt und dem Informationsobjekt wird durch die Gewichtung $w_i$ ausgedrückt. Eine Anziehungskraft wird von jedem $i-ten$ Referenzobjekt proportional zu der Gewichtung $w_i$ in Richtung des Vektors aktiviert. Dabei ist $a_i$ eine zusätzliche Gewichtung des $i-ten$ Referenzobjekts. Die folgende Berechnungsvorschrift definiert die Position der Informationsobjekte [Rom97]:

$$\bar{P} = \sum_i normal(\vec{r_i}) * w_i * a_i$$

Es gibt einen *Attraktionsmanipulator*, der durch einen kleinen Würfel auf der Oberfläche des Referenzobjektes dargestellt wird. Wenn dieser Würfel mit der Maus zum Relevanzkugelmittelpunkt hin- oder von ihm weggezogen wird, wird das Referenzobjekt skaliert bzw. gewichtet. Die von einem Referenzobjekt abhängigen Informationsobjekte werden zum Mittelpunkt der Relevanzkugel wandern, wenn das Relevanzobjekt verkleinert wird, ansonsten nähern sich die Informationsobjekte dem Referenzobjekt [Hem97]. Die *transparente Kugel* dient lediglich der visuellen Unterstützung des Benutzers. Die Abbildung 3.7 zeigt eine vollständige Relevanzkugel im Einsatz.

3. Der *Dokumentenraum* ermöglicht die erweiterte Exploration von Dokumenten der Kontextmenge, indem der textuelle Inhalt der Dokumente angezeigt wird. Die anderen beiden Visua-

lisierungswerkzeuge können mit diesem gleichzeitig aktiv bleiben. Es ist ebenfalls möglich, die Dokumente entweder während der Suchphase mit dem Navigationskegel oder während der Auswahl in der Relevanzkugel zu lesen.

#### *Bewertung des „LyberWorld“ – Verfahrens*

*Benutzeroberfläche*
Dieses Abbildungsverfahren besteht aus drei Komponenten. Hier werden jedoch nur der Navigationskegel und die Informationskugel bewertet, da der Dokumentenraum lediglich zur Anzeige des Inhaltes der Dokumente dient.

- Strukturiertheit
  Der *Navigationskegel* ähnelt der Visualisierungsstruktur *Cam Tree* (siehe Abschnitt 3.2.3). Mit dem Navigationskegel kann die Gesamtheit aller verfügbaren Informationsobjekte angezeigt werden, wobei einige Informationsobjekte sichtbar sind und andere versteckt im Hintergrund bleiben.
  Die Informationszweige können gelöscht werden, so daß die Struktur des Gesamtbaumes auf eine Menge mit besserem Überblick reduziert wird. Dadurch könnten jedoch ungewollt interessante Teile gelöscht werden, da sie ohne ein bestimmtes Kriterium entfernt werden können.
  In der *Relevanzkugel* ist es möglich, die Gesamtstruktur zu überblicken, da diese Form kompakt ist. Auf der Kugel sind Gitterlinien ähnlich der Längen- und Breitengrade auf einem Globus abgebildet, so daß der Benutzer bei einer Rotation der Kugel optisch unterstützt wird und die räumliche Wahrnehmung der Kugel verstärkt wird.

- Erkennbarkeit
  Durch die Expansion des Navigationkegels in mehrere Titel- und Suchbegriffsebenen benötigt die Darstellung sehr viel Platz. Obwohl Scroll-Komponenten zur Verfügung stehen, mit deren Hilfe alle Bereiche erreichbar sind, werden Schwierigkeiten hinsichtlich des Überblicks der Darstellung auftreten und die Navigation daher schwerfallen. Die Überlagerung (Informationsverdichtung) bei einer großen Menge von Dokumenten ist mit hoher Wahrscheinlichkeit ein größeres Problem für eine dreidimensionale Darstellung. In dem *Navigationskegel* lassen sich die selektierten Titel oder Begriffe farblich hervorheben. Es ist schwierig, mehrere Selektionen in der gesamten Darstellung zusammen zu erfassen, da einige unter Umständen auf der hinteren Seite des Kegels liegen. Die Titel und Begriffsebenen lassen sich voneinander deutlich durch farbige Hervorhebung unterscheiden.
  Die Selektion von Informations- und Referenzobjekten bei der *Relevanzkugel* ist anspruchsvoll. Eine direkte Manipulation (Skalierung oder Verschiebung) wird entweder mit den sich an der Oberfläche der Relevanzkugel befindlichen Referenzkugeln oder mit der Komponente (Pfeilsystem oder Drahtgitterkugel) der Relevanzkugel durchgeführt. Sowohl die Komponenten als auch die selektierten Informationsobjekte werden farbig gekennzeichnet. Dadurch werden diejenigen Informationsobjekte farblich hervorgehoben, die von der gerade manipulierten Referenzkugel thematisch abhängen. Es können sich beliebig viele Informationsobjekte in selektiertem Zustand befinden und ebenso in ihren ursprünglichen Zustand zurückversetzt werden. In das Innere der Relevanzkugel wird eine zweite Kugel hineinprojiziert, so daß die wenig relevante Informationsobjekte aufgenommen und abgeschirmt werden. Die für den Suchkontext relevanten Informationsobjekte schweben leicht erkennbar außerhalb dieser neue Kugel im Raum herum.
  Bei den verschiedenen Manipulationen können Probleme auftreten, da es nicht möglich ist, die Menge der Suchergebnisse zu reduzieren. Besonders tief im Inneren der Relevanzkugel können

einzelne Objekte daher schlecht wahrnehmbar oder hinter anderen Objekten sein, was auch eine Selektion erschwert. Die Halbtransparenz der Relevanzkugel ist als visuelle Unterstützung des Benutzers vorgesehen, damit er die Informationsobjekte von außen betrachten kann.

- Lokalität
  In beiden Visualisierungswerkzeugen erfolgen die Reaktionen zeitlich unmittelbar. Ereignisse werden nach dem Lokalitätsprinzip immer zuerst genau dort behandelt, wo sie auftreten und der Benutzer davon in Kenntnis gesetzt.

- Orientierung
  Ein generelles Problem des rotierbaren *Navigationskegels* ist die Verdeckung von Informationen und der Verlust der Raumorientierung, wenn der Radius der Zylinder mit der Anzahl der Informationseinheiten auf der Zylinderoberfläche anwächst und die verschiedenen Ebenen expandiert werden. Es ist möglich, einen gesuchten Begriff einzugeben, um zu erfahren, ob er vorhanden ist. Ein kleiner Kegel wird an sämtliche Dokumentflächen gehängt, um anzudeuten, wo der Begriff möglicherweise enthalten ist.
  Problematisch ist es auch, daß alle selektierten Objekte wegen der Navigationskegelsform nicht zusammen betrachtet werden können.
  Durch die direkte Manipulation der *Relevanzkugel* mit Hilfe des Manipulators ist eine sofortige visuelle Rückmeldung der Aktionen gewährleistet. Mit dem Manipulator wird die Relevanzkugel gedreht oder verschoben. Damit wird ein neuer Ansichtpunkt vorgegeben, was zu Verwirrung führen kann, da das räumliche Vorstellungsvermögen bei dieser Aktion stark beansprucht wird.

*Interaktion*

- *Wechselwirkung:*
  Man kann mit dem *Navigationskegel* direkt interagieren. Durch Drehung der einzelnen Ebenen können Dokumenttitel oder Begriffe, die sich auf der Rückseite befinden, sichtbar gemacht werden. Nach Anklicken einer Dokument- oder Begriffsebene mit der Maus oder mit Hilfe der Cursortasten wird eine neue Ebene erzeugt. Genauso können die Ebenen entfernt werden.
  Zur Interaktion mit der *Relevanzkugel* können mit der Maus unabhängige Funktionen ausgelöst werden. Die linke Maustaste ist für Selektion und Manipulation von Werkzeugen zuständig. Die anderen Maustasten können für die restlichen Manipulationen benutzt werden, wie Verschiebung oder Rotation verschiedener Elemente der Kugel [Hem97, Mü98]. Auch die Manipulation der gesamten Relevanzkugel mit allen Komponenten erfolgt duch Anklicken der Drahtgitterkugel des Manipulators.
  Mit einer Mausmanipulation und mit Hilfe der Cursortasten können die Referenzkugel und die Informationsobjekte in alle Richtungen verschoben werden. Diese Art der Interaktion erscheint kompliziert, da sowohl Maustasten als auch Cursortasten gleichzeitig betätigt werden müssen – andererseits ist diese Art der Interaktion auch in anderen grafischen Systemen üblich.
  Die Anziehungskräfte der Referenzkugeln können mit Hilfe von Slidern auf den Informationsobjekten variiert werden. Alle abhängigen Informationsobjekte werden von einer großen Referenzkugel stark angezogen, wärend sie von einer kleinen Referenzkugel weit entfernt bleiben. Ein weiterer Slider hilft die Anziehungskraft der Relevanzkugeloberfläche zu verändern. Die Informationsobjekte bewegen sich weiter zur Oberfläche, wenn diese erhöht wird. So ist es möglich, die Dichte der Informationsobjekte zu verringern und eine bessere Selektierbarkeit herzustellen.

- *Ausführungsminimalität*
  Der *Navigationskegel* ähnelt dem Verfahren *Cone Tree* (siehe Abschnitt 3.2.3). Er eröffnet die Möglichkeit zum Navigieren (vorwärts und rückwärts) im Inhaltsraum der Datenbank, und unterstützt die Suche mit Begriffen, die mit den Dokumenten der Datenbasis verknüpft sind. Diese Interaktion wird direkt und intuitiv realisiert.
  In der *Relevanzkugel* sind die Informationsobjekte direkt erreichbar. Wünschenswert wäre angesichts der hohen Komplexität eine Undo-Funktion für den Fall, daß dreidimensionale Aktionen versehentlich ausgeführt wurden und die ursprüngliche Ansicht wiederhergestellt werden soll.

*Funktionalität*

- Darstellung:
  Die Informationsobjekte werden als einfache kleine farbige Quader dargestellt. Diese Darstellung wurde aus Gründen der Performanz beim Rendering gewählt. So wird vermieden, daß das System unbrauchbar wird, wenn komplexe Geometrien bei einer großen Anzahl von Informationsobjekten verwendet werden. Eine Zoomeinstellung ist nicht vorhanden. Eine Kamerafahrt ist nur bei einer Drehung der *Relevanzkugel* möglich, aber nicht in die Kugel hinein.

- Datenreduzierung:
  Im *Navigationskegel* können Informationszweige gelöscht werden. Eine Reduzierung der Objektmenge ist bei der *Relevanzkugel* nicht möglich.

- Erweiterbarkeit:
  Das entworfene direktmanipulative Werkzeug *Relevanzkugel* wurde erweitert. Die Funktionalität der Relevanzkugel ist so weit verbessert, daß sie nach [Hem97] in jeder Anwendung des LyberWorld-Systems zu finden ist. Diese starke Verflechtung macht es schwer, das Werkzeug in einer anderen Anwendungsumgebung zu verwenden.

- Details:
  Es sind keine speziellen Details der Informationsobjekte aus der Darstellung ersichtlich. Der Benutzer muß dazu das Werkzeug „Dokumentenraum“ erst aktivieren, dann kann er überprüfen, ob die von ihm und dem System als relevant eingestuften Dokumente auch wirklich sein konkretes Informationsbedürfnis befriedigen.

**Fazit**
Nach der Entwicklung des LyberWorld-Prototyps wurde die Relevanzkugelmetapher erweitert. Das erste Redesign wurde auf Rechnern der Firma Silicon Graphics Inc. mit einem Betriebssystem IRIX 5.3 durchgeführt. Die Architektur der Relevanzkugel wurde dabei in Module aufgegliedert und verändert. Danach wurde ein zweites Redesign der Relevanzkugel auf einem PC mit WindowsNT-Umgebung implementiert. Die Funktionalität wurde dabei erweitert [Rom97].
Beim *Navigationskegel* wird nicht mehr als nur eine Verarbeitung des Suchergebnisses durchgeführt, wo entschieden wird, was in der *Relevanzkugel* dargestellt werden soll.
In der Darstellung der Informationsobjekte mit der *Relevanzkugel* ist die Anwendung von Attributen fast nur auf die Farbe beschränkt. Dadurch wird es erschwert, die interessanten Dokumente aufzufinden, ohne jedes einzelne Informationsobjekt vorher zu untersuchen. Es wäre hilfreich, wenn für die Informationsobjekte eine zusätzliche und direkte Information zum Inhalt vorhanden wäre, um zu den interessanten Dokumenten zu gelangen, bevor man den Dokumentenraum betritt.
Für eine effektive Arbeit mit dem System sind ein gutes räumliches Vorstellungsvermögen sowie eine gewisse Einarbeitungszeit Voraussetzung. Die *Relevanzkugel* wurde an das objektrelationale

Datenbaksystem Illustra[10] angebunden. Dafür wurde eine Beispieldatenbank mit einer Anzahl von Daten implementiert, um ein Test zu realisieren. Die Relevanzkugel besitzt nach [Rom97] keine Anbindung an eine reale Datenbasis, sie arbeit bisher mit simulierten Daten.

### 3.2.6 SQWID

Das *SQWID*-Tool (Search Query Weighted Information Display) [McC97] ist ein System, welches eine dynamische grafische Darstellung des Ergebnisses einer Web-Anfrage erzeugt. Die erzeugte grafische Darstellung ist ein Knotenlink-Graph, der zwei Arten von Knoten enthält. Das sind die Begriffs- und die Dokumentknoten.
Die Abbildung 3.8 soll veranschaulichen, wie das SQWID Visualisierungs- und Schnittstellensystem arbeitet und erklärt die grundlegenden Details. In dieser Abbildung sind drei Begriffsknoten in einem Dreieck angeordnet. Sie sind durch die Farben Rot, Grün und Blau gekennzeichnet. Die anderen sind die Dokumentknoten, die in Form von Textzeilen einen Titel oder eine Hostadresse anzeigen. So repräsentieren sie jedes einzelne Dokument. Die Position eines Dokumentknotens wird durch die Bewertung des Dokuments für alle drei Begriffe bestimmt. Ein Dokument, das für einen bestimmten Begriff hoch bewertet wird, ist näher an diesem Begriffsknoten positioniert.
Dokumentknoten mitten in dem durch die Begriffsknoten aufgespannten Dreieck sind an alle drei Begriffsknoten gebunden. Die Dokumentknoten entlang einer Kante an zwei Begriffsknoten, und die Dokumentknoten, die um einen Begriffsknoten herum (aber außerhalb des Dreiecks) positioniert sind, gehören nur zu diesem Knoten.

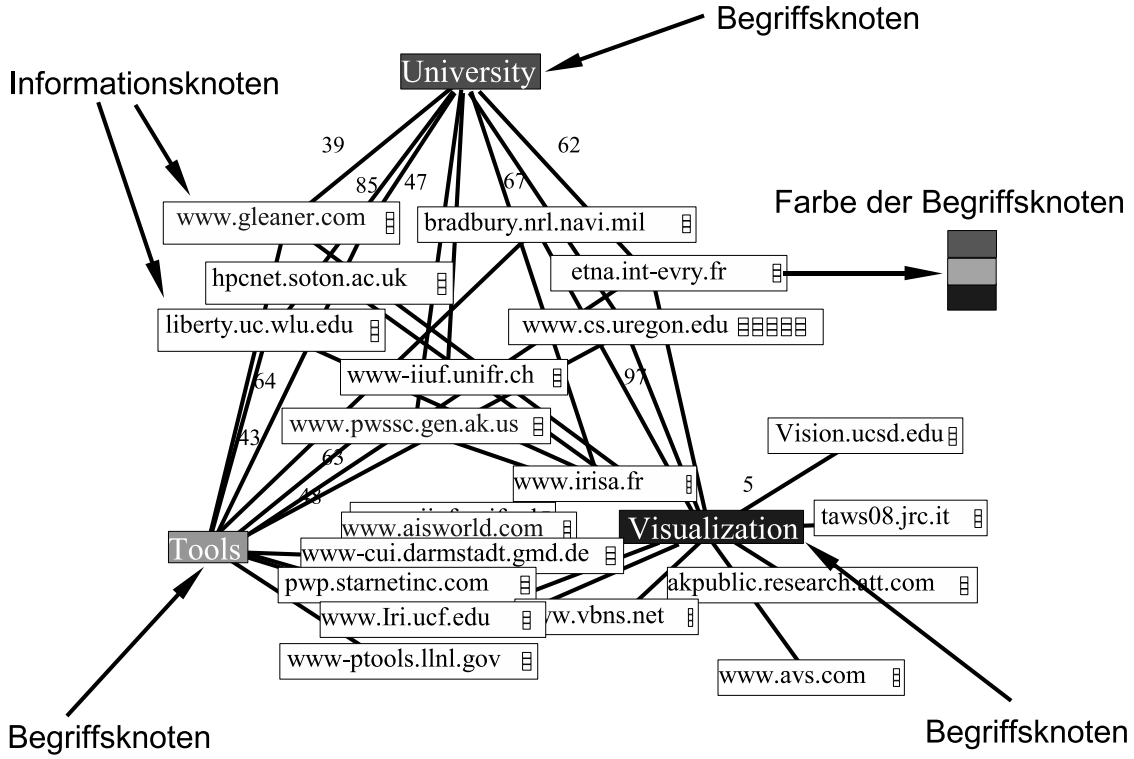



Abbildung 3.8: Ergebnisdarstellung mittels *SQWID* [McC97]

Am rechten inneren Rand jedes Dokumentknotens befindet sich ein Kästchen mit drei Farben

[10]Illustra ist ein erweiterbares Datenbankmanagementsystem (DBMS), das auch als objektrelationale Datenbank bezeichnet wird. An den Datenbank-Kern können sogenannte "DataBlades", Module für spezielle Datentypen, angeschlossen werden.

übereinander. Die drei Farben – Rot, Grün und Blau – entsprechen den Farben für jeden Begriffsknoten, wobei die Intensität der Farbe dem Maß für die Häufigkeit des Auftretens des Begriffs im Dokument entspricht. Die Dokumente, die zu einem Hostknoten gehören, können visuell zu dem Hostknoten zusammengefaßt werden. Der Hostknoten enthält dann eine Reihe von dreifarbigen Kästchen, die parallel nebeneinander stehen und jeder Seite eines Dokuments entsprechen. Mit Hilfe der Kästchen kann man visuell erfassen, wieviele Seiten in einem Host enthalten sind und wie hoch die Seiten bezüglich des Begriffs bewertet sind.

#### *Bewertung des „SQWID“ – Verfahrens*

*Benutzeroberfläche*

- *Strukturiertheit*
  Das Layout ist ein Spannungsmodell, wobei Suchbegriffe und Dokumente schweben. Durch eine Einstellung können sie zur Ruhe gebracht werden, wobei ihre Position klar die relative Anziehungskraft der drei festgelegten Begriffsknoten reflektiert. Die Begriffs- und die Dokumentknoten können sich leicht in der Darstellung durch ihre Informationen und Farben unterscheiden lassen. Bei *SQWID*-Verfahren enthält die Darstellung eine große Anzahl von Merkmalen, die die Interpretation des Ergebnisses zu einer komplexen Aufgabe werden lassen: die Position der Dokumentknoten bezüglich der Begriffsknoten, die Anzeige der Anzahl von Dokumentseiten, die Farbe für die Begriffsknoten und die Intensität der Farbe als Maß für die Häufigkeit des Auftretens der Begriffsknoten.

- *Erkennbarkeit*
  Anhand der Position läßt sich schnell die Beziehung der Dokumentknoten zu den Begriffsknoten identifizieren. Durch eine breite Verteilung der Begriffsknoten mit der Maus bekommen die Dokumentkonten neue Positionen im Raum, und somit kann die Beziehung untereinander besser erkannt werden. Eine mögliche Selektion ist dadurch erleichtert. Etwas problematisch ist die Selektion bei einer Überlappung von Dokumentobjekten. Dafür müssen Dokumentobjekte verschoben werden, so daß das untere Dokumentobjekt gesehen und selektiert werden kann. Das ist nicht so einfach, da dieses Dokument für kurze Zeit sichtbar bleibt und die anderen Dokumente schnell in ihre alte Position zurückkehren werden. Wurde das Dokument selektiert, wird es farbig hervorgehoben. Die Kästchen für die Anzahl des Auftretens der entsprechenden Suchbegriffe im Dokument sind klein, und damit schlecht erkennbar. Genauso klein ist die Anzeige der Häufigkeit der Suchbegriffe als Farbintensität und so stellen sie zu hohe Anforderungen an die farbliche Wahrnehmungsfähigkeit des Benutzers. Weiterhin wird diese Anzeige häufig durch andere Dokumente verdeckt, so daß ein optischer Vergleich mit anderen Dokumenten kaum möglich ist.

- *Lokalität*
  Die wichtigsten Funktionen zur Interaktion sind direkt erreichbar. Sie beziehen sich zumeist auf Positionsänderungen der Begriffs- und Dokumentobjekte und sind damit klar, verständlich und in ihren Auswirkungen nachvollziehbar.

- *Orientierung*
  Die Handlungsalternativen ergeben sich aus der Darstellung selbst und der Zugänglichkeit der Dokument- und Begriffsknoten. Die Interaktion ist somit intuitiv gestaltet. Zur besseren Unterscheidung werden Farben eingesetzt. Die Lage der Objekte im aufgespannten Informationsraum ist klar ersichtlich, nachteilig sind die häufigen Überlappungen der Icons.

*Interaktion:*

- *Wechselwirkung:*
  Der Benutzer hat die freie Wahl, die Begriffe mit Hilfe einer definierten Liste in den Begriffsknoten durch andere zu ersetzen. Genauso kann er die Anzahl und einen Datumsbereich der Dokumente mit Hilfe von Slidern einstellen. Im Menü befindet sich eine Liste von hilfreichen Funktionen, die die Verarbeitung des Suchergebnisses unterstützen. Die Seitenanzahl der Dokumente kann eingegeben werden, so werden automatisch die Kästchen in Zahlen umgewandelt. Die Dokumente können „eingefroren“ werden, so daß sie ohne Bewegung sind und in Ruhe betrachtet werden können. Die Dokumentknoten können zu einer Art Explosion gebracht werden, damit der Dokumenttitel dargestellt wird. Die Position der Begriffsknoten kann vom Benutzer mit der Maus beliebig geändert werden. Dadurch werden die Dokumentknoten neu positioniert.

- *Ausführungsminimalität:*
  Die grafische Darstellung kann vom Benutzer dynamisch und unkompliziert eingestellt werden, um die relative Bewertung der Ergebnisse schnell sichtbar zu machen. Weniger intuitiv ist der Umgang mit Suchergebnissen, die mehr als drei Suchbegriffe enthalten. In einem Menü wird eine Liste der Suchbegriffe angeboten, woraus drei Begriffe auszuwählen sind. Nach jeder Auswahl wird die Darstellung in ihrer Ausgangsform neu aufgebaut. Der Zusammenhang zu den vorher verwendeten Begriffen geht dabei verloren.

*Funktionalität*

- *Darstellung*
  Die Darstellung der Dokumente ist „nur“ auf eine zweidimensionale rechteckige Fläche begrenzt. Es werden genau drei Suchbegriffe für das Ergebnis benutzt und dargestellt. Bewegungen und Interaktionen laufen zügig ab.

- *Datenrezudierung:*
  Dokumente können durch eine Einschränkung des Datumsbereiches oder Dokumentanzahl nicht mehr in der Darstellung angezeigt werden.

- *Erweiterbarkeit:*
  Das Verfahren ist in sich geschlossen und verfügt über keine Erweiterungsmöglichkeiten.

- *Details:*
  Durch doppeltes Klicken auf einen Dokumentknoten erscheint ein Browser mit einer HTML-Beschreibung des Hosts. Dazu muß das Dokument jedoch zuerst vollständig verfügbar sein. Eine Anzeige von Vorabinformationen über die Dokumente existiert nicht.

**Fazit**
Das *SQWID* ist in Java implementiert worden und läuft lokal am *Georgia Tech College of Computing*[11] als Java-Applet unter dem HotJava Browser. Wegen der Sicherheitsbeschränkungen kann es nicht mit voller Funktionalität auf remote sites laufen, aber eine begrenzte Version des Programms steht auf der SQWID Web Site[12] zur Verfügung.
Das *SQWID*-System bietet vom Ansatz her durch die Darstellung der Dokumente in der Nähe der Suchbegriffe eine übersichtliche und intuitive Möglichkeit zum Auffinden der relevanten Dokumente. Nach Angabe von [Keh97] können mit *SQWID* ungefähr 200 Dokumente visualisiert

[11] http://www.cc.gatech.edu/
[12] http://www.cc.gatech.edu/grads/m/Scott.McCrickard/sqwid/

werden. Das ist nicht ausreichend für die Darstellung von großen Dokumentenmengen, wie sie bei Suchanfragen häufig entstehen. Nachteilig ist weiterhin die Beschränkung auf die Auswertung von drei Suchbegriffen. Für diese maximale Anzahl von drei Suchbegriffen wird die Mehrdeutigkeit vermieden, die in einer zweidimensionalen Darstellung vorkommen kann.

### 3.2.7 Sphärenvisualisierung (Sphere Visualisation)

Die *Sphärenvisualisierung* wird innerhalb des *VizNet*[13]-Visualisierungssystems [Fai93b] verwendet, um assoziative Beziehungen zu betrachten.
Assoziative Beziehungen werden mit Hilfe der Sphärenstruktur visualisiert. Die Sphärenstruktur wird für die Darstellung aller Beziehungen, die mit einem Objekt „OOI“ (*object of interest*) assoziert sind, angewendet. Das OOI befindet sich stets auf der Außenschale. Das Layout der Sphäre ist der Struktur einer Zwiebel ähnlich, es ist aus mehreren in Schichten angeordneten Kugeln aufgebaut. Dies gestattet die Darstellung unterschiedlicher Niveaus von Informationen.
Die Informationen werden in Form zweidimensionaler Flächen und auf den Oberflächen der verschachtelten Sphären dargestellt. Die Anordung der Objekte erfolgt entsprechend einer vorher definierten Beziehungsrelevanz. Die Objekte, die direkt auf das Objekt OOI bezogen sind, werden in der äußersten Sphäre und nah am OOI angezeigt. Nachfolgende Objekte, die durch andere Objekte mit dem OOI verwandt sind, werden als Objekte in der niedrigen Ebene eingestuft und daher in den tiefer liegenden Sphären dargestellt und erscheinen weniger gut sichtbar.
Dadurch wird ein natürlicher „Fisheye-View“[14] erzeugt, die die interessanten Objekte hervorhebt und die weniger verwandten Objekte verdrängt [Fai93a] (siehe Abbildung 3.9). Die Farbe der Sphären wird mit zunehmender Tiefe der Verschachtelung dunkler, so daß der Benutzer seinen gegenwärtigen Standort innerhalb der Visualisierung abschätzen kann.
Die Abbildung 3.9 zeigt eine Sphärendarstellung. In der Darstellung sind die assoziierten Links zusammen mit einer Menge von Bildern, die sich auf ein Flugzeug beziehen. In der Mitte der Sphäre liegt das Informationsobjekt, das gerade untersucht wird. Um dieses herum befinden sich die anderen Informationsobjekte, die strahlenförmig angeordnet sind.
Durch Rotation und Verschiebung der Kugel kann ein bestimmtes Objekt ins Sichtfeld bewegt werden. Der Betrachter wird beim Überlaufen eines Objektes, das weitere Assoziationen in die darunterliegende Schicht anzeigt, auf diese Unterschicht „herunterfallen“. Bezüglich Assoziationen zu Elementen höherer Relevanz wird der Betrachter auf die nächst höhere Schale transportiert.

#### *Bewertung des „Sphärenvisualisierung“ – Verfahrens*

*Benutzeroberfläche*

- *Strukturiertheit*
  Ein bedeutendes Merkmal für die Objektdarstellung durch Sphärenvisualisierung ist der Grad der Assoziation. Objekte, die eine starke Beziehung zu dem Objekt *OOI* haben, befinden sich näher an diesem als die weniger stark assoziierten Objekte. Aufgrund der kompakten Struktur bekommt der Benutzer keinen guten Eindruck von der Gesamtheit der Sphäre und der darin enthaltenen Objekte. Einige Objekte sind schwer zu erkennen, wenn sie sich auf einer der

[13] VizNet ist ein multimediales Visualisierungssystem für unterschiedliche Typen von Daten. Es wurde am Institut für Wissenschaftssysteme in Singapur entwickelt [Fai93a].

[14] Fisheye Views sind Filtermechanismen, die ein verzerrtes Abbild zeigen, analog einer Linse mit sehr großem Winkel (Fischauge). Es wird sowohl die nahe als auch die weitere Umgebung dargestellt, wobei die nahe Umgebung sehr detailliert und die vom aktuellen Betrachtungspunkt weiter entfernt liegenden Objekte weniger detailliert dargestellt werden.

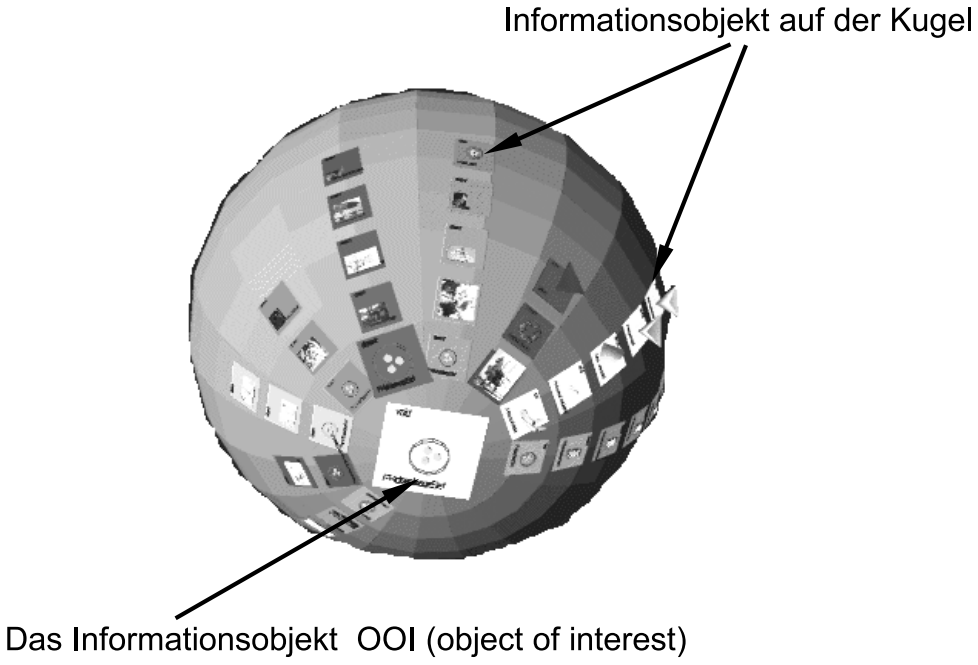



Abbildung 3.9: Informationsdarstellung mittels *Sphärenvisualisierung* [Fai93a]

tieferen Schalen befinden. Die Objekte und die Beziehung zwischen den Objekten können für den aktuellen Fokuspunkt gut erkannt werden. Die Objektgröße ist wegen der Objektanzahl und der Struktur der Sphäre beschränkt.

- *Erkennbarkeit*
  Das Verfahren Sphärenvisualisierung hat nur einen Fokuspunkt, in dem bestimmte Objekte sichtbar werden. Die Attribute wie Farbe und Textur sowie zusätzliche Texte werden angewendet, wobei einige auf die Größe der Objekte beschränkt sind.
  Ein selektiertes Objekt ist gut sichtbar, da es sich auf der sichtbaren Seite der Sphäre befindet. Aufgrund der Struktur der Sphäre ist es schwer, dieses Objekt in die Gesamtstruktur einzuordnen und zu erkennen. Andere Objekte sind schwer zu selektieren oder wahrzunehmen, da sie sich auf tieferen Schalen befinden.

- *Lokalität*
  Die möglichen Benutzeraktionen sind auf das Bewegen der Kugel begrenzt, wobei die Wirkungen unmittelbar erfolgen. Das Prinzip, nach dem die Objekte beim Wechsel der Selektion neu angeordnet werden bleibt für den Benutzer unklar. Somit ist das Ergebnis einer solchen Interaktion nicht vorhersehbar und nur schwer nachvollziehbar.

- *Orientierung*
  Durch die Auswahl eines neuen Fokuspunktes werden die assoziierten Objekte um das *OOI* neu angeordnet. Das bedeutet, daß der Benutzer eine Vorstellung der Gesamtstruktur entwickeln und gleichzeitig auf die jeweils aktuelle Ansicht der assoziierten Objekte reagieren muß. Dies führt mit hoher Wahrscheinlichkeit zu Orientierungsschwierigkeiten bezüglich der Darstellung.

*Interaktion*

- *Wechelwirkung:*
  Bei einer Objektmanipulation wird die Kugel entweder gedreht oder verschoben. Die gezielte Interaktion mit Objekten auf unteren Schichten ist in den zur Verfügung stehenden Quellen nicht hinreichend beschrieben.

- *Ausführungsminimalität:*
  Die räumliche Manipulation der Kugel erfolgt mit der Maus und ist leicht verständlich.

*Funktionalität*

- *Darstellung:*
  Die Informationsobjekte können unterschiedliche Formen annehmen, die aber in jedem Fall zweidimensional sein müssen. Die Berechnung von Transparenzen ist ein relativ aufwendiger Bestandteil der Renderingpipeline von Grafiksystemen und ist auch nur theoretisch in beliebiger Tiefe durchführbar. Praktisch läßt die Erkennbarkeit schon nach wenigen halbdurchlässigen hintereinanderliegenden grafischen Objekten stark nach.

- *Datenreduzierung:*
  Bei diesem Verfahren ist es nicht vorgesehen, die Anzahl der Objekte bereits vor der grafischen Darstellung zu reduzieren.

- *Erweiterbarkeit:*
  Das Verfahren kann nicht in seiner Funktionalität erweitert werden.

- *Details:*
  Möglichkeiten zur Anzeige detaillierter Informationen sind nicht Bestandteil des Verfahrens.

**Fazit**
Informationen über das Verfahren *Sphärenvisualisierung* sind kaum verfügbar. In der Literatur ist nicht beschrieben worden, ob und wo das Verfahren eingesetzt worden ist. Die Sphärenvisualisierung ist aufgrund ihrer kompakten Struktur kein geeignetes Verfahren für die Darstellung von Suchergegbnissen. Außerdem bildet sie rein assoziative Beziehungen ab. Eine genaue Aussage über die Menge von darzustellenden Objekten ist in der Dokumentation über das Verfahren nicht gemacht worden. Es ist zu erwarten, daß wegen der Sphärenstruktur die Menge nicht groß sein darf.

### 3.2.8 Tree-Map

*Tree-Map* ist ein Visualisierungsverfahren für die Darstellung großer Mengen von hierarchisch strukturierten, gegliederten Informationen in zweidimensionaler Form auf dem Bildschirm [B.92]. Das Konzept von *Tree-Map* wurde von B. Shneiderman an der Universität von Maryland Mitte der 90er Jahre entwickelt [B.92]. Die Motivation für die Entwicklung dieses Verfahrens war der Mangel an angemessenen Werkzeugen für die Visualisation großer Verzeichnisstrukturen auf Festplattenspeichern.
Die traditionellen Methoden für die Darstellung von hierarchisch strukturierten Informationen können als Listen, Verzeichnisbaumansichten und Baumdiagramme klassifiziert werden. Es ist schwer, mit diesen Methoden Informationen aus großen Strukturen zu extrahieren [Vic87].
Durch die Liste kann zwar viel Informationsinhalt vermittelt werden, aber sie ist gewöhnlich sehr schlecht geeignet, um strukturelle Informationen darzustellen. Jeder Informationsknoten einer Hierarchie kann unabhängig aufgeführt werden, aber man muß manuell die Hierarchie durchlaufen, um die Struktur zu erkennen. Als Beispiel dafür kann die Ausgabe des Befehls „dir“ unter DOS oder

die Ausgabe des Befehls „ls“ unter UNIX angeführt werden. Mit dem Befehl wird die aktuelle Verzeichnisebene des Datenträgers am Bildschirm angezeigt, aber nicht die gesamte Struktur. Verzeichnisbäume, die u. a. aus dem Windows Explorer bekannt sind, können gut Struktur und Inhalt wiedergeben. Da diese Darstellung bei „aufgeklappten“ Knoten aber sehr lang wird, kann immer nur ein Ausschnitt der Struktur in einem Fenster betrachtet werden. Die Informationen, die für den Überblick einer Hierarchie notwendig sind, sind häufig unzugänglich [Chi94]. Die Baumdiagramme benötigen ebenfalls viel Fläche für die Darstellung von Informationen. In der Struktur werden viele Knoten und Informationen unübersichtlich, so daß es besonders bei großen Hierarchien schwer ist, den Überblick zu behalten.

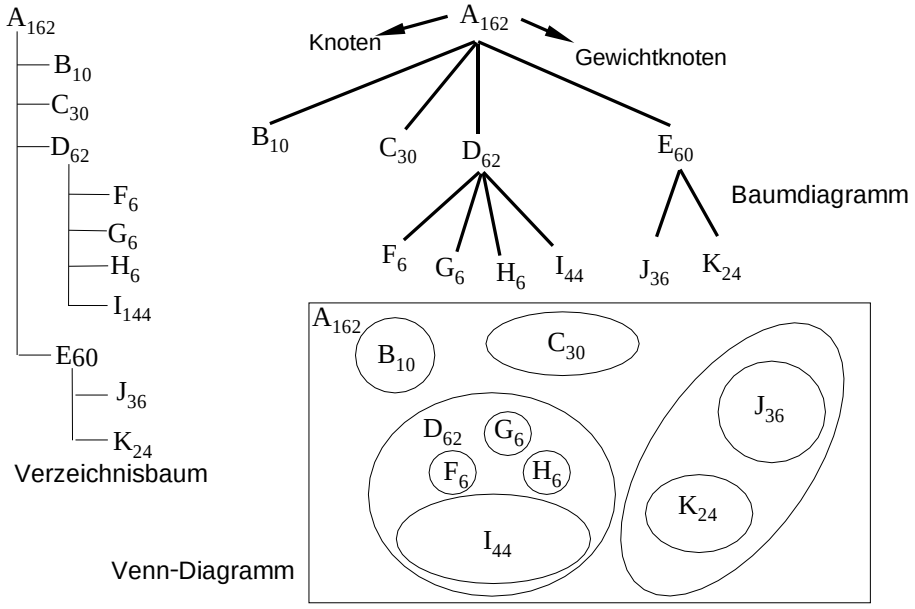



Abbildung 3.10: Verzeichnisbaum und Baumdiagramme nach [Joh91]

Die Methode *Tree-Map* ermöglicht eine völlig neue Darstellung von großen Daten- und Informationsmengen. Dabei wird die zur Verfügung stehende Fläche des Bildschirmes bzw. Fensters effizient genutzt, da alle Informationen zweidimensional in rechteckiger Form dargestellt werden. Die Rechtecke sind jeweils ineinander geschachtelt, wobei sie die komplette Hierarchie widerspiegeln. Die Arten der Informationen werden mit Farben kodiert, um sie optisch voneinander unterscheiden zu können.

### Von der Liste zum Tree-Map

Die folgenden Abbildungen zeigen einfach dargestellt den Übergang eines normalen Baumes in einen *Tree-Map*. Auf der linken Seite der Abbildung 3.10 ist ein Verzeichnisbaum zu sehen, während auf der rechten Seite ein typisches Baumdiagramm dargestellt ist. In der Darstellung sei eine Hierarchie mit elf Knoten gegeben, drei davon sind Verzeichnisse mit Kindknoten und acht sind Blattknoten. Jeder Knoten hat einen Namen und eine dazugehörige Größe. Im unteren Teil der Darstellung ist diese Struktur als *Venn-Diagramm*[15] dargestellt. Dies ist eine gute Möglichkeit zur Visualisierung von Hierarchien, die aber einen Nachteil hat: durch die kreis- bzw.

[15]Die Venn-Diagramme sind eine Technik zur Darstellung logischer Zusammenhänge, die 1880 von John Venn eingeführt wurde. Sie bestehen aus einem „Rahmen“, der das „Universum“ darstellt, und darin ein beliebiges geschlossenes Gebilde (z.B. ein Kreis) für jedes auftretende Objekt.

ellipsenförmige Darstellung der Elemente bleibt ein großer Teil der Darstellungsfläche ungenutzt und die grafischen Objekte werden mit zunehmender Hierarchietiefe sehr schnell kleiner und damit schwer erkennbar. Die Abbildung 3.11 zeigt ein kastenbasiertes Venn-Diagramm. Die Hierarchie wird hier durch verschachtelte Rechtecke, statt Kreise, repräsentiert. Das Venn-Diagramm ist ein gutes Werkzeug für die Visualisierung von kleinen und flachen Hierarchien. In der Abbildung wird veranschaulicht, daß durch die Verschachtelungen Leerraum entstanden ist, der keinen Nutzen für die grafische Darstellung hat. In der Abbildung 3.12 sind Balken zwischen den Verzeichnissen

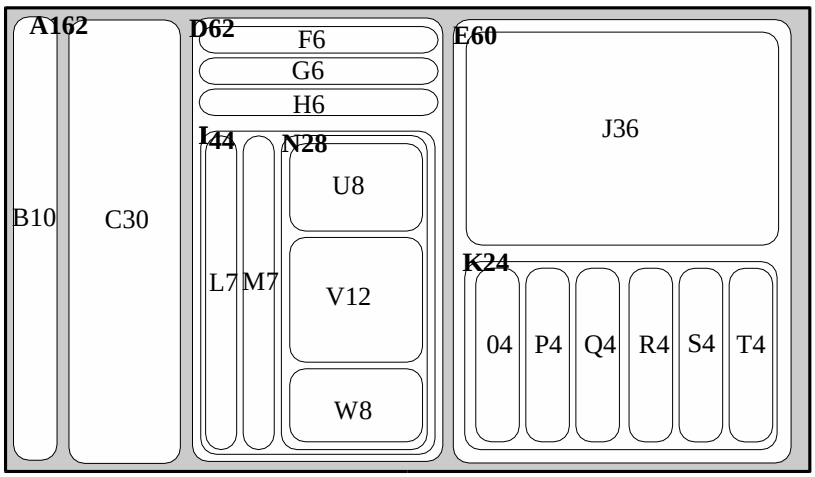



Abbildung 3.11: Verschachtelter Tree-Map [Joh91]

eingefügt worden, so daß die Trennung (Nebenlinien) der Flächen beseitigt werden konnte. Die Rechtecke stellen die Blätter (Elemente) des Baumes dar.

**Die Tree-Map Methode**

Für jedes Rechteck wird eine Gewichtung benötigt, um das Element relativ zur eigenen und zur Größe der anderen Elemente zu zeichnen. Diese Gewichtung kann aus einem oder mehreren Faktoren bestehen.

- **Strukturelle Information: Bildschirmaufteilung**
  Shneiderman hat bei der Zeichnung der Rechtecke folgende grundsätzliche Richtlinien entwickelt [B.92].
  - Ist ein Element „A“ Vorgänger von Element „B“, dann ist das Rechteck von Element „A“ vollkommen eingeschlossen, oder entspricht genau dem Rechteck von „B“.
  - Die Rechtecke zweier Elemente schneiden sich, wenn ein Element Vorgänger des anderen ist.
  - Jedem Element wird eine Fläche mit relativer Größe in Abhängigkeit von der Gewichtung zugeordnet.
  - Das Gewicht eines Knoten ist größer oder gleich groß der Summe der Gewichte der Kinder/Nachfolger des Knotens

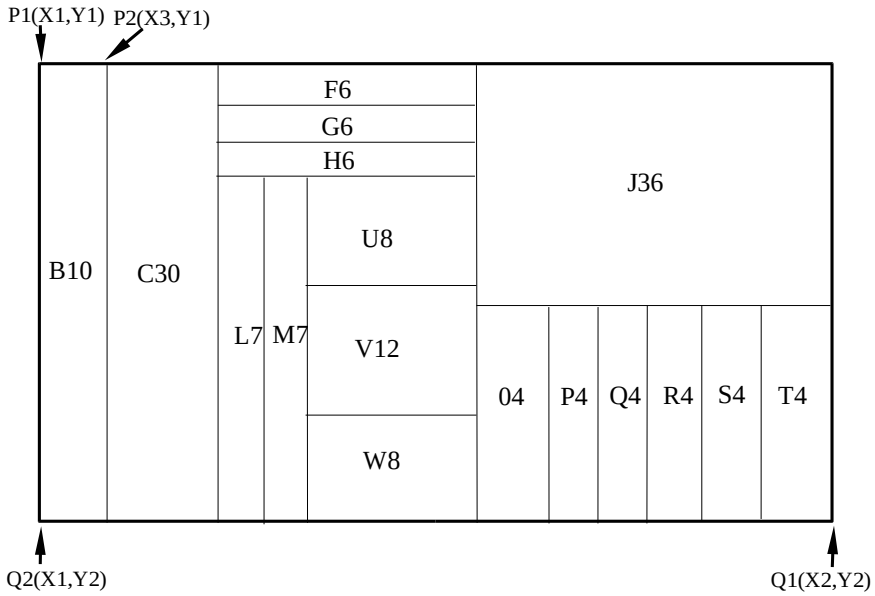



Abbildung 3.12: Tree-Map [Joh91]

- **Inhalt der Information: Darstellungsmerkmale**
  Sobald ein Rechteck eines Knotens berechnet ist, werden die visuellen Eigenschaften für diesen Knoten hinzugefügt. Zur besseren optischen Abgrenzung können Eigenschaften wie Farbe (spezifizierbar durch Komponenten nach dem HSV-Farbmodell: Farbton, Sättigung, Helligkeit), Textur, Form, Grenze, Bewegung usw. angewendet werden. Dabei ist die Farbe die wichtigste dieser Eigenschaften beim Suchen und Erkennen von Dateien.

**Tree-Map Algorithmus**
Der Treemap Algorithmus besteht aus zwei Teilen:

- **Das Zeichnen der Rechtecke**
  Die Rechtecke ergeben sich, indem eine Baumwurzel und eine Rechteckfläche ausgewählt wird, die durch die obere linke Koordinate $P1(x1, y1)$ und die untere rechte Koordinate $Q1(x2, y2)$ definiert ist. Die Anzahl der direkten Nachfahren des Wurzelknotens ist ausschlaggebend dafür, in wieviele Rechtecke das äußere Rechteck horizontal (im Bereich $[x1, x2]$) aufgeteilt wird.
  Jedes dieser Rechtecke muß eine Fläche einnehmen, die der Gewichtung des jeweiligen Unterbaumes entspricht. Bei der Visualiserung eines Dateisystems ergibt sich die Gewichtung aus der Größe in Bytes. Für die Berechnung der Positionen der vertikalen Trennungslinien werden die Größen der Unterverzeichnisse zur Gesamtgröße ins Verhältnis gesetzt und anschließend auf die Breite des äußeren Rechteckes abgebildet. Die Position der Linie des ersten Unterbaumes $x3$ wird also wie folgt berechnet:

$$x3 = x1 + (Size(child[1])/Size(root)) * (x2 - x1)$$

  Enthält der nun gezeichnete Unterbaum weitere Knoten, „dreht“ der Algorithmus seine Abarbeitungsrichtung um 90° und führt die Berechnung mit den Koordinaten $P2(x3, y1)$ und $Q2(x1, y2)$ entlang der y-Achse aus. Die Abarbeitungsrichtung wird jeweils beim Auffinden einer neuen Ebene in der Hierarchie geändert. Auf diese Weise wird die gesamte Struktur rekursiv auf Rechteckflächen abgebildet. In der Abbildung 3.12 ist die Erläuterung zu sehen. Die Knoteninformationen werden durch die Größe und Farbe des Rechtecks dargestellt.

- **Der Trackingalgorithmus für MouseEvents**
  Der Pfad von der Wurzel des Baums zum Knoten, der mit einem gegeben Punkt in der Darstellung assoziiert ist, kann zur Laufzeit proportional zur Tiefe des Knotens ermittelt werden. Das ist möglich, da der Tree-Map beim Zeichnen eines Knotens dessen „Bounding Box“ (ungebendes Rechteck) speichert. Zu jedem Punkt in der *Tree-Map*-Darstellung kann später eine solche „Bounding Box“ und damit der zugehörige Knoten der Hierarchie ermittelt werden.

***Bewertung des „Treemap“ – Verfahrens***

*Benutzeroberfläche*

- *Strukturierheit*
  Die Informationen werden in einer hierarchischen Struktur unter optimaler Ausnutzung der verfügbaren Bildschirmfläche auf Rechtecke abgebildet. Die Struktur der Hierarchie wird implizit durch die Beziehung der Rechtecke zueinander dargestellt. Je größer die hierarchische Struktur ist, desto enger und kleiner werden die Flächen sein. Bei vielen kleinen Rechtecken verringert sich die Übersichtlichkeit und somit werden die Details und die Struktur selbst nicht mehr erkennbar. Hier muss dann eine Zoom-Funktion weiterhelfen. In der Struktur sind die Maße *Höhe* und *Breite* der Rechtecke unterschiedlich, deswegen ist es schwierig einen Vergleich der Größe durchzuführen.

- *Erkennbarkeit*
  Die Tree-Maps ermöglichen dem Benutzer einen schnellen Überblick und schnellen Informationsgewinn bezüglich der Größe der Knoten. Zur visuellen Unterstützung können Eigenschaften wie Farbe, Textur, Form und Begrenzung angewendet werden. Die Farbfüllung der Rechtecke symbolisiert unterschiedliche Dateitypen, wie z. B. Text, Bild, Programmdatei. Weiterhin können zusätzliche Informationen als Text angezeigt werden.
  Die Selektion von Elementen ist von der Art der Hierarchie abhängig. Bei einer flachen Hierarchie ist es möglich, daß das Element schnell zu erfassen ist. Je mehr Ebenen in der Hierarchie vorhanden sind und je mehr Rechtecke demzufolge verschachtelt sind, desto kleiner müssen die Rechtecke gezeichnet werden. Sehr kleine Rechtecke sind entsprechend schlecht erkenn- und selektierbar.

- *Lokalität*
  Eine Visualisierung der kompletten Hierarchie findet unter Verwendung von Farben zur schnelleren Lokalisierung von Informationen und der Gewichtung von Knoten statt. Die Suche von Informationen wird direkt realisiert.

- *Orientierung*
  Mit Hilfe der verwendeten Eigenschaften *Farbe* und *Größe* kann der Benutzer schnell die Unterschiede zwischen den Knoten erkennen.
  Bei großen Hierarchien werden die Rechteckflächen unter Umständen extrem klein, was zu einer verwirrenden Darstellung führen kann. Diesem Problem kann mit einer Zoom-Funktion begegnet werden, dabei geht allerdings der Überblick über die gesamte Hierarchie verloren.

*Interaktion*

- *Wechselwirkung:*
  Bei einer Berührung der Informationsfläche erscheinen detaillierte Informationen zu dem betreffenden Knoten (bei der Visualisierung eines Dateisystems z. B. Dateiname, Pfad, Erstellungsdatum usw.).

- *Ausführungsminimalität:*
  Das Auffinden von Dateien gelingt auch in großen Hierarchien direkt mit Hilfe der Maus.

*Funktionalität*

- *Darstellung:*
  Die Darstellung der Informationen beschränkt sich auf farbige zweidimensionale rechteckige Flächen, die mit Linien abgegrenzt werden sowie zusätzliche textuelle Anzeigen.

- *Datenreduzierung:*
  Eine Reduzierung der Menge der Informationen ist nicht vorgesehen, das Verfahren ist für die Darstellung der gesamten Informationsmenge ausgelegt.

- *Erweiterbarkeit:*
  Dies ist ein in sich geschlossenes Darstellungsverfahren ohne Erweiterungsmöglichkeiten.

- *Details:*
  In den Veröffentlichungen [Joh91, B.01] sind den einzelnen Elementen der Struktur lediglich Farben, und Texturen zuzuordnen. Auch textuelle Informationen wie der Dateiname können im Rechteck angezeigt werden, diese sind jedoch bei sehr kleinen Rechtecken nur schwer erkennbar. Zusätzlich können Informationen zu den Dateien (z.B. in einem Pop-Up-Fenster) angezeigt werden. So liefert ein Element Information über Größe (Gewichtung) und letzte Änderungen (Farbsättigung/Saturation). Mit einer Zoom-Funktion können die Elemente genauer betrachtet werden.

**Fazit**
Die Visualisierung mittels TreeMap ist für viele Einsatzbereiche geeignet. Einige Unternehmen haben begonnen, kommerzielle Software zur Darstellung von Dateisystemen auf dieser Basis zu entwickeln und zu vertreiben (z. B. *TreeViz* und *WindSurfer*).
Ein anderes Anwendungsbeispiel ist die aktuelle Darstellung eines Aktien-Portfolios (*SmartMoney*). Dabei werden die Marktsegmente auf die Rechtecke der ersten Hierarchieebene aufgeteilt. In diesen werden die dazugehörigen Aktienwerte als Rechtecke mit der ihrer Marktkapitalisierung entsprechenden Größe dargestellt. Der aktuelle Gewinn/Verlust beeinflußt die Farbe der Rechtecke. Auf diese Weise gelingt es, einen Gesamteindruck des Marktes zu vermitteln und zugleich das Verhalten der Einzelwerte sichtbar zu machen.
Das Konzept des *Tree-Map* läßt sich auf jegliche Art hierarchischer Struktur adaptieren. Die verschiedenen Anwendungen unterscheiden sich leicht in der Art der Darstellung und Navigation. Die Qualität einer TreeMap-Darstellung ist in jedem Fall von der verfügbaren Bildschirmfläche abhängig. Durch die Farbanwendung und Gewichtung von Knoten findet die Lokalisierung von Informationen schnell statt. Da es sich nicht um eine Fokus+Kontext- Technik handelt, wird es schwer, in die kleinen dargestellten Rechtecke zu gelangen, um Information detaillierter darstellen zu lassen als in der restlichen Struktur.

### 3.2.9 Bewertung der dargelegten Visualisierungsverfahren

Die Verfahren sind unabhängig von ihrer konkreten Einsatzmöglichkeit zur Visualisierung von Suchergebnissen bewertet worden. Die meisten der vorgestellten Programme sind monolithisch aufgebaut, Erweiterungen oder die Ausdehnung auf andere, ähnliche Aufgabengebiete sind nicht vorgesehen.
Von den vorgestellten Systemen sind lediglich LyberWorld und SQWID speziell für die Visualisierung von Dokumenten in einem Informationsraum entwickelt worden.

Dem System LyberWorld kommt eine gewisse Sonderrolle zu, da es aus mehreren Verfahren besteht. Obwohl seine einzelnen Komponenten hinsichtlich einiger Kriterien Defizite aufweisen, gestattet ihr Zusammenspiel eine relativ effiziente und zielgerichtete Suche nach den interessanten Dokumenten einer vorgegebenen Menge. Der Umgang mit diesem System ist allerdings sehr anspruchsvoll, die komplexen Interaktionstechniken verlangen eine gewisse Einarbeitungszeit.
Das zweite System, das auf die Visualisierung von Suchergebnissen ausgerichtet wurde, ist SQWID. Auch dieses System bietet eine gute Darstellung, die die Nähe der einzelnen Dokumente zu einem Suchbegriff sofort ersichtlich werden läßt. Allerdings ist die Anzeige auf drei Suchbegriffe begrenzt. Werden mehr Suchbegriffe verwendet, müssen drei davon für die Darstellung ausgewählt werden. Eine Übersicht der Nähe der Dokumente zu allen verwendeten Begriffen ist in diesem Fall also nicht möglich.
Ein Problem, das beide Systeme betrifft, ist die extrem große Menge von Dokumenten, die gewöhnlich bei einer Internet-Suche gefunden werden. Die daraus resultierende grafische Darstellung dürfte in der Praxis zu überladenen Darstellungen führen, die sich aufgrund der Menge grafischer Objekte auch nur schwer manipulieren lassen. Hier fehlen effiziente Mechanismen, die es erlauben, die Dokumentenmenge bereits *vor* ihrer Abbildung auf grafische Objekte zu reduzieren.
Bei beiden Systemen muß der Benutzer die Entscheidung zwischen *interessant* und *uninteressant* nur aufgrund der grafischen Darstellung fällen. Zur genaueren Inspektion eines Dokuments muß zuerst der Download erfolgen. In dieser Hinsicht gibt es keinen Unterschied zur Auswertung eines Suchergebnisses auf Basis der konventionellen Textliste. Eine Möglichkeit, vorher zusätzliche Informationen wie Erscheinungsdatum oder einen Abstract zur Entscheidung heranzuziehen gibt es nicht. So müssen unter Umständen viele Dokumente aus dem Internet geladen werden, die aber dann letztendlich doch als *uninteressant* zu klassifizieren sind.
Die vorgestellten Programme sind monolithisch aufgebaut. Sie bieten genau eine Sicht auf den Informationsraum. Inwieweit diese Sicht geeignet ist, hängt stark von der Dimension des Informationsraumes ab, die sich aus der Anzahl der verwendeten Suchbegriffe ergibt. Ein zweidimensionales Suchergebnis muß z. B. nicht unbedingt aufwendig in eine dreidimensionale Darstellung abgebildet werden. Weiterhin werden bei einigen Verfahren hohe Anforderungen an das räumliche Vorstellungsvermögen des Benutzers gestellt. Dieses kann jedoch sehr verschieden ausgeprägt sein. Eine Möglichkeit des „Umschaltens“ der Darstellungsart bieten beide vorgestellten Systeme nicht an.
Die Systeme LyberWorld und SQWID bieten bereits gute Lösungen für eine große Zahl von Anwendungsfällen und Benutzergruppen. Um die obengenannten Probleme zu lösen bedarf es jedoch neuer Ansätze, die das Ergebnis dieser Arbeit sein sollen.
Das Ergebnis des Vergleichs der vorgestellten Visualisierungsansätze ist in Tabelle 3.1 zusammengefaßt.

Die Bedeutung der in der Tabelle verwendeten Symbole ist wie folgt:

| | | | | | |
|---|---|---|---|---|---|
| $\oplus\oplus$ | sehr gut | $\oplus$ | gut | $\odot$ | befriedigend |
| $\ominus$ | nicht ausreichend | $\ominus\ominus$ | nicht vorhanden | k. A. | keine Angaben möglich |

## 3.3 Experimentelle Ansätze

In diesem Abschnitt sind Methoden beschrieben, die in der Literatur als Denkansatz formuliert oder experimentell als Prototyp implementiert wurden, aber bisher nicht zum praktischen Einsatz bei der Visualisierung von Informationsräumen gekommen sind. Zu diesen Ansätzen sind wesentlich weniger Informationen verfügbar als zu den im Abschnitt 3.2 beschriebenen Verfahren. Dementsprechend wird hier nur die grundsätzliche Arbeitsweise beschrieben und keine Bewertung vorgenommen. An dieser Stelle werden nur Ansätze vorgestellt, die für Endanwendungen geeignet sind. Methoden,

| Merkmale | PW | HB | CT | IC | LW | SQ | SV | TM |
|---|---|---|---|---|---|---|---|---|
| **Benutzeroberfläche** | | | | | | | | |
| Strukturiertheit | ⊙ | ⊕ | ⊕ ⊕ | ⊖ | ⊕ ⊕ | ⊕ | ⊖ | ⊕ ⊕ |
| Erkennbarkeit | ⊙ | ⊙ | ⊖ | ⊖ | ⊙ | ⊙ | ⊖ | ⊕ |
| Lokalität | ⊖ ⊖ | ⊕ ⊕ | ⊕ | ⊕ | ⊕ | ⊕ | ⊖ | ⊖ ⊖ |
| Orientierung | ⊕ ⊕ | ⊖ | ⊕ ⊕ | ⊕ | ⊙ | ⊙ | ⊖ | ⊕ ⊕ |
| **Interaktion** | | | | | | | | |
| Wechselwirkung | ⊕ | ⊕ ⊕ | ⊕ | ⊕ | ⊕ ⊕ | ⊕ | ⊙ | ⊕ |
| Ausführungsminimalität | ⊕ | ⊕ ⊕ | ⊕ ⊕ | ⊙ | ⊙ | ⊕ | ⊙ | ⊕ ⊕ |
| **Funktionalität** | | | | | | | | |
| Darstellung | ⊙ | ⊕ | ⊕ ⊕ | ⊕ | ⊕ ⊕ | ⊕ | ⊖ | ⊕ |
| Datenreduktion | ⊖ ⊖ | ⊖ ⊖ | ⊖ ⊖ | ⊖ ⊖ | ⊕ | ⊙ | ⊖ ⊖ | ⊖ ⊖ |
| Erweiterung | ⊖ ⊖ | ⊖ ⊖ | ⊖ ⊖ | ⊖ ⊖ | ⊖ ⊖ | ⊖ ⊖ | ⊖ ⊖ | ⊖ ⊖ |
| Details | ⊙ | ⊖ | ⊖ ⊖ | ⊖ ⊖ | ⊖ | ⊖ | ⊖ ⊖ | ⊕ |

**Legende:**
PW=Perspective Wall HB= Hyperbolic Browser CT= Cone Tree/Cam Tree
IC= Information Cube SQ= SQWID LW= LyberWorld
SV= Sphere Visualisation TM= Tree Map

Tabelle 3.1: Visualisierungstechniken

die größeren Aufwand erfordern, wie z.B. neuronale Netze (*WebSOM*[Lag96b, Lag96a, Hon97]), werden hier nicht betrachtet.

### 3.3.1 TheBrain Technology

TheBrain[16] Technologies Corporation hat ein visuelles System entwickelt, das die Suche in großen Verzeichnissen, Katalogen usw. anschaulicher gestalten soll. Es ist ein java-basiertes System, das es ermöglicht, die Dateien in einer Baumstruktur darzustellen. Es ähnelt in seiner Darstellungsstruktur dem Hyperbolic Browser (siehe Abschnitt 3.2.2). Die Texte repräsentieren Files, Webseiten oder Datenbanken. Im Fokuspunkt befindet sich ein Begriff. Um diesen Begriff herum sind alle anderen Begriffe angeordnet, die in einer Verbindung dazu stehen. Beim Anklicken mit der Maus rückt der jeweils gewählte Begriff in das Zentrum des oberen Fensters und beim Überfahren mit der Maus werden Netzlinien zu verwandten und übergeordneten Begriffen sichtbar.
Jeweils zum im Zentrum befindlichen Begriff werden inhaltliche Informationen über die entsprechende Seite vermittelt. Dies wird solange fortgeführt, bis das interessante Dokument gefunden ist. Während die Dateien z. B. in vielen Windows-Anwendungen in hierarchischer Struktur dargestellt werden, werden sie bei *TheBrain* assoziativ dargestellt. Die Dateien werden nach ihrer Zugehörigkeit zu einem bestimmten Thema gespeichert und dargestellt. *TheBrian* ist für kommerzielle Anwendungen zur Organisation von Informationen entwickelt worden. Es dient als Dateimanagement-System, wobei die Informationen in einem konzeptuellen Raum und nach individuellen Arbeitsgewohnheiten organisiert werden. Es ermöglicht Files, Dokumente und Webseiten über Anwendungen und das Netzwerk zu verbinden. Die Benutzung ist sowohl auf Desktop-Rechnern als auch in Netzwerken, Web-Sites und Handheld-Computern möglich. Die Abbildung 3.13 zeigt das System.

[16] www.thebrain.com/

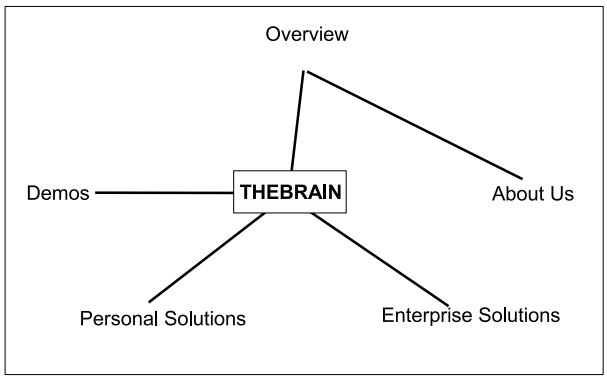



Abbildung 3.13: Informationsdarstellung mit Thebrain, entnommen von www.thebrain.com/

### 3.3.2 VR-Vibe

*VR-VIBE* (Visual Interface Browsing Environment) wurde an der Universität von Pittsburgh entwickelt [Ben95]. VR-VIBE verwendet statistische Techniken, um eine Dokumentbibliographie sichtbar zu machen und erlaubt dem Benutzer auf die Darstellung einzuwirken und den Raum zu manipulieren [Wea96]. Die Suchbegriffe werden Points of Interest (POIs) genannt. Die räumliche Position eines Dokument-Icons zeigt die relative Anziehungskraft eines Dokuments zu den unterschiedlichen POIs an, wobei die Anziehungskraft in thematischen Ähnlichkeiten ausgedrückt geschätzt wird. So ist ein Icon, das zwischen zwei POIs äquidistant ist, zu beiden gleichmäßig relevant, wärend ein Icon nah an einem bestimmten POI nur zu diesem POI relevant ist.
Die absolute Relevanz wird durch die Größe und die Helligkeit der Darstellung des Dokuments ersichtlich. Das Dokument ist stärker relevant, wenn die Icons größer und die Farbe heller ist.
Die Abbildung 3.14 stellt eine Visualisierung mittels VR-Vibe dar. Sie enthält 1581 Eintragungen und fünf Suchbegriffe. In der Abbildung sind drei Benutzer sichtbar, die mit einem Block in "T"-Form dargestellt werden. Die Positionen der Suchbegriffe werden durch grüne Oktaeder mit einem nebenstehenden Text gekennzeichnet, die Dokumente werden durch rosafarbige Blöcke repräsentiert. Ein 3D-Scrollbar an der linken Seite erlaubt die Veränderung der Schwellwerte. Nur die Dokumente in dem definierten Wellenbereich werden angezeigt. Die Benutzer können in dem 3D-Raum navigieren und einzelne Dokumente ansteuern. Durch Verschiebung des Icons kann man Informationen über die relevante Anziehungskraft zu den unterschiedlichen POIs gewinnen. Mit Hilfe von VR-VIBE werden Bibliographien visualisiert.

### 3.3.3 NIRVE-Prototypen

Die NIRVE-Prototypen (NIST Information Retrieval Visualization Engine) [Cug00] umfassen drei spezielle Visualisierungsparadigmen, die am NIST (National Institute of Standards and Technology) für die visuelle Darstellung von Ergebnissen einer modifizierten Version des statistischen Text-Informations-Retrievalsystems *PRISE*[17], entwickelt wurden. Diese sind:

**Dokumentenspirale:** wertet den Relevanzgrad aus, der den Dokumenten durch die Suchmaschine zugeordenet wird,

**Three-keyword axes display:** wertet die Häufigkeit eines speziellen Suchausdruckes in den Dokumenten aus,

[17]NIST PRISE Search Engine: http://www.itl.nist.gov/div894/894.02/works/papers/zp2/main.html

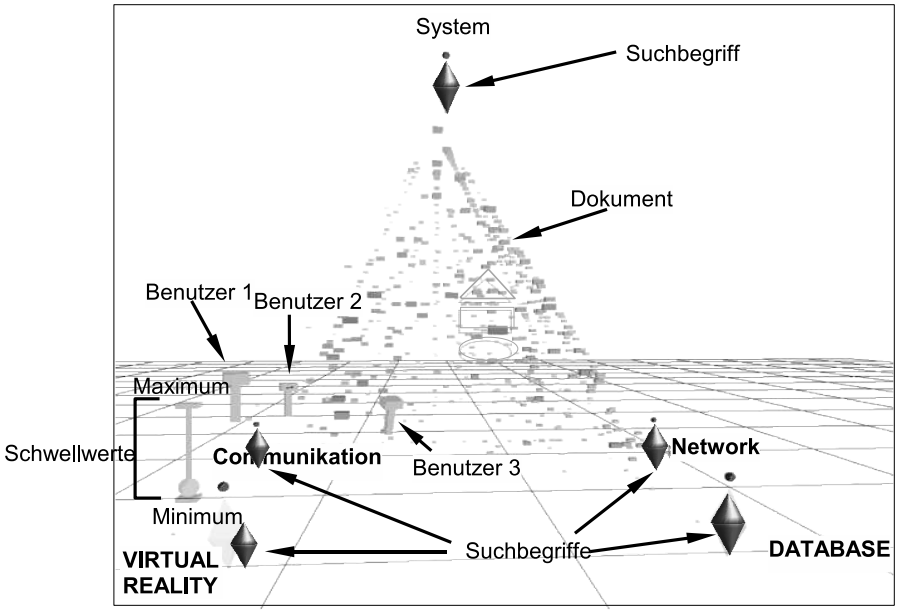



Abbildung 3.14: Informationsdarstellung mit VR-Vibe [Ben95]

**Nearest neighbor clustering:** zeigt welche Beziehung die Dokumente hinsichtlich der Häufigkeit der Schlüsselworte untereinander haben.

Jedes Paradigma gestattet es, zu navigieren und die Anzeige dynamisch auf der Grundlage der ursprünglichen Schlüsselworte zu verändern. Diese Ansätze wurden entwickelt, um dem Benutzer einen Überblick über die Struktur des Ergebnisses zu geben, das Finden eines Dokuments in dieser Struktur wird nicht unterstützt [Cug00].

### Dokumentenspirale

Das Hauptorganisationsprinzip der Spirale ist, daß Dokumente mit hohem Relevanzgrad weiter im Zentrum plaziert werden als die mit geringen Relevanzgrad. Alle Dokument-Icons befinden sich zuerst in der XZ-Ebene. Das Icon mit dem höchsten Rang wird im Zentrum der Ebene plaziert. Nachfolgende Icons werden entlang einer Spirale entsprechend ihrer relativen Bewertung angeordnet. Dokumente mit exakt gleicher Bewertung werden nebeneinander plaziert, um Überlappungen zu vermeiden. Die Darstellung ist in der Abbildung 3.15 zu sehen.
Ein *Keyword Slider* erlaubt dem Benutzer die Dokumente hervorzuheben, die für ihn besonders wichtige Stichwörter enthalten. Auf diese Weise wird der Wichtungsfaktor für jedes Stichwort gesteuert. Die Farben der Slider entsprechen denen, die für die Abbildung der Stichwortstärke auf den Icons verwendet wird. Die Stichwortstärke wird in Abhängigkeit von der Stichworthäufigkeit berechnet: Die Stichworthäufigkeit ist die Anzahl des Auftretens des Stichwortes im Dokument dividiert durch die Länge des Dokuments. Diese Häufigkeit wird dann normalisiert, indem sie durch die Maximalhäufigkeit der Dokumentmenge dividiert wird, wobei sich ein Wert zwischen 0 und 1 ergibt. Letztendlich wird die „Stichwortstärke“ ermittelt, indem die Quadratwurzel aus der normierten Häufigkeit gezogen wird, um die Sichtbarkeit kleiner Werte zu verbessern. Dokumente, deren Stichwortstärke zu sehr stark gewichteten Schlüsselworten gehört, werden durch Erhebung des Icons über die gewöhnliche XZ-Ebene der Spirale hinaus gekennzeichnet. Diese Interaktion kann in zwei Modi ausgeführt werden. Im „OR-Modus“ lautet die Gleichung:

$$elevation = \sum (factor(i) * strength(i))$$

Dabei ist $i$ der Index des Wichtungsfaktors $factor$ und der Stichwortstärke $strength$ für jedes Stichwort. Das hat den Effekt, daß wenn ein Dokument ein Stichwort mit positiver Stärke und positivem Faktor hat, dieses Dokument aus der Default-Ebene heraushoben wird. Im „AND-Modus“ kommt die Bedingung dazu, daß wenn ein Dokument ein Stichwort mit $strength = 0$ hat, für das ein positiver Faktor definiert wurde, sein Icon *nicht* erhoben wird. Das heißt, ein Dokument wird dann, und nur dann erhoben, wenn es eine positive Stärke zu jedem positiven Faktor hat.

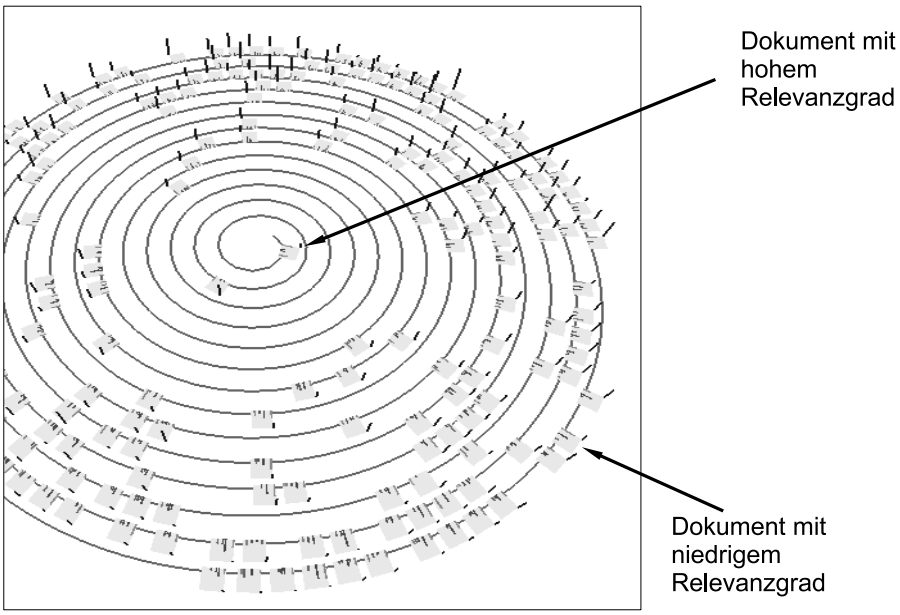



Abbildung 3.15: Document Spiral [Cug00]

**Document Three-Keyword Axes Display**

Die Icons werden hier auf Basis der Stichwortstärke dargestellt. Diese Darstellung wird *three-keyword axes* genannt. Einer Achse kann eine beliebige Untermenge von Stichworten zugeordnet werden. Diese Assoziation wird in einem separaten Fenster mit einer Spalte von Checkboxen für die drei Achsen (X, Y und Z) gesteuert. Jede Achse wird mit den ihr zugewiesenen Stichworten markiert. Die Position des Icons auf einer Achse wird als Durchschnitt der Stichwortstärken berechnet und dynamisch angepaßt, wenn die Achsenauswahl geändert wurde. Die Icons stellen auch weiterhin die Stärke für alle Stichwörter dar. Jedes Icon hat einen Balken, dessen Länge zur Relevanz proportional ist.
Dem Benutzer ist es möglich, interaktiv zu ermitteln in welcher Beziehung die Anfrageausdrücke zu den erhaltenen Dokumenten stehen. Wird zum Beispiel das selbe Stichwort für alle drei Achsen ausgewählt, so wird eine linear angeordnete Liste entlang X=Y=Z angezeigt, die nach der Stichwortstärke geordnet ist. Wenn dem Benutzer ein spezieller Ausdruck als besonders wichtig erscheint, ist die Rangordnung in dieser Dokumentreihe von der Häufigkeit dieses einen Ausdruckes abhängig. Die Auswahl eines zweiten Stichworts für eine der beiden anderen Achsen ergibt eine Ebene der Dokumente, die beide Stichwörter enthalten. Werden drei verschiedene Stichwörter angegeben, resultiert daraus ein sogenannter *Scatterplot*, wobei die Icons scheinbar beliebig im Raum verteilt sind. In dieser Darstellung können interessante Extrempunkte und mögliche Cluster ermittelt werden, wie in der Abbildung 3.16 zu sehen ist.

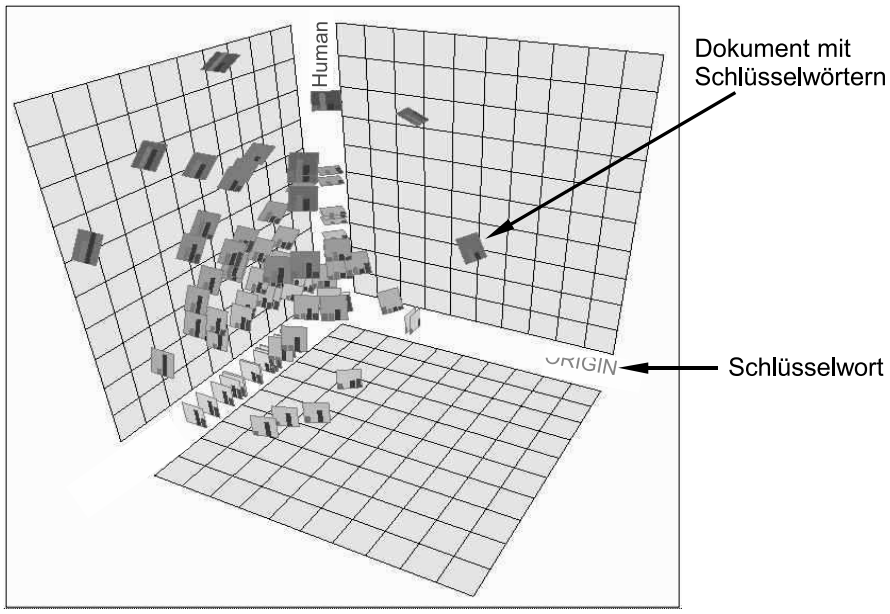



Abbildung 3.16: Document Three-Keyword Axes [Cug00]

**Nearest Neighbor Sequence**

Mit diesem Paradigma werden die semantisch ähnlichen Dokumente gemeinsam in einer Region des Raumes konzentriert (Siehe Abbildung 3.17). Jedes Dokument hat eine Position im semantischen Raum, die als Vektor der Stichwortstärken dargestellt wird. Bei mehr als drei Begriffen wurde eine lineare Ordnung gefunden, die semantisch naheliegende Dokumente dicht beieinander läßt. Es wurden einfache *nearest neighbor*-Algorithmen benutzt, um die Dokumente zu ordnen: von einer ersten Auswahl ausgehend ist jedes Dokument in der Folge der naheliegenste Nachbar seines Vorgängers. Die semantische Distanz zweier beliebiger Dokumente basiert auf dem *keyword strength*-Vektor jedes Dokuments und kann auf zwei Weisen berechnet werden: die einfache euklidische Distanz oder als Winkel zwischen den Vektoren.
Die neugeordneten Dokument-Icons werden dann als Kreis in der X-Z-Ebene angeordnet, wobei jedes Icon senkrecht zu dieser Ebene steht. Die Stichworthäufigkeit angrenzender Dokumente kann visuell verglichen werden, indem durch ein Icon zu dem dahinter befindlichen gesehen wird. Der Abstand zwischen den Icons ist proportional zur semantischen Distanz, die als Teil des *nearest neighbor*-Algorithmus berechnet wird.

## 3.4 Visualisierung der Ergebnisse von Suchdiensten

Wie schon im Kapitel 2 beschrieben wurde, werden Ergebnisse von Suchdiensten gewöhnlich in Textform geliefert. Einige Suchmaschinen haben bereits versucht, diese Form der Darstellung durch eine grafische Repräsentation der Information zu ergänzen. Diese grafischen Darstellungen sind keine Visualisierungen des durch die Suche ermittelten Informationsraumes, daher werden diese Verfahren hier gesondert vorgestellt.

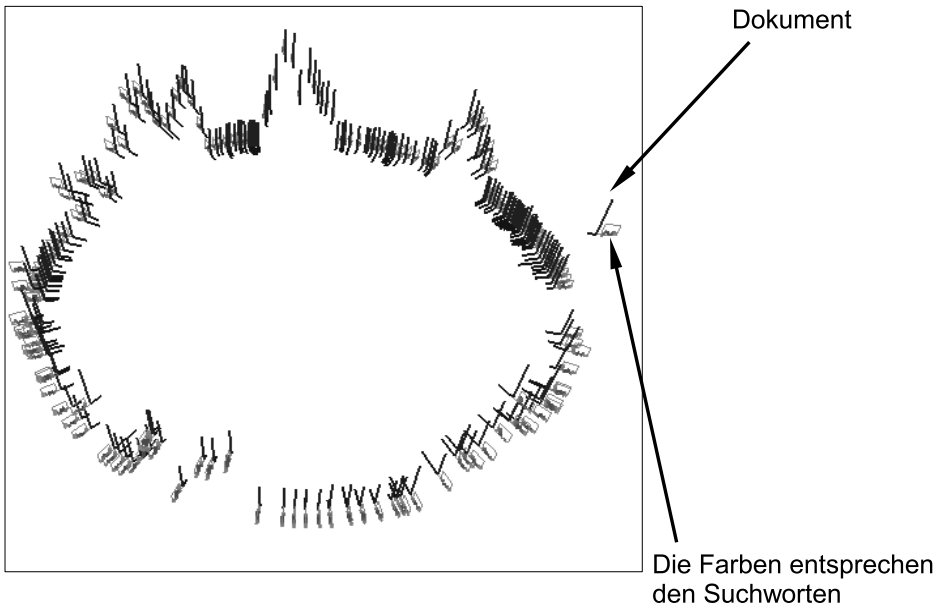



Abbildung 3.17: Nearest Neighbor Sequence [Cug00]

### 3.4.1 Yahoo

Die *Yahoo Corporation*[18] kündete 1996 einen grafischen Zugang zum Katalogsystem an: *Yahoo! 3D* mit einer VRML-basierten virtuellen Welt. Die Entwicklung erfolgte zusammen mit der *Caligari Corporation*[19]. Sowohl die Yahoo-Kategorien als auch die Informationen im Internet sollen im dreidimensionalen Raum dargestellt werden. Nach [Cal96, Yah96] enthält *Yahoo! 3D* über 450 Objekte mit ihren aktuellen Sammlungen wie Kunst, Ausbildung, Regierung, Gesundheit, Nachrichten, Unterhaltung und Wissenschaft. Zur Navigation sollen die Operationen wie Kamera-Bewegung, Drehen, Ansehen und Verschieben dienen. In der Abbildung 3.18 ist die Darstellung, wie sie aussehen sollte, repräsentiert. *Yahoo! 3D* war eine experimentelle dreidimensionale Darstellung von hierarchischen Web-Katalogen, die aber nicht mehr im Internet verfügbar ist.

### 3.4.2 Alta Vista

Alta Vista[20] hat im Februar 1997 eine grafische Darstellung des Suchergebnisses vorgestellt. Es handelt sich dabei um ein interaktives Tool, das das Ergebnis einer Suche dynamisch kategorisiert und ein Begriffschema generiert [Dä98]. Das Tool heißt *Live Topics* und stellt gewichtete Relationen dar, die nach der Häufigkeit und dem Abstand der häufigsten Wörter in den Trefferdokumenten aufgestellt werden. Im *Topic Word* werden die relevanten Begriffe in eine Liste eingegeben. Sie werden zeilenweise einem Oberbegriff untergeordnet. Durch Anklicken der Oberbegriffe werden die untergeordneten Begriffe in einem Fenster angezeigt. Mit „Topic Relationships“ wird eine Baumstruktur mit den Oberbegriffen von „Topic Words“ dargestellt, die wiederum andere gewichtete untergeordnete Begriffe enthalten [Bec97]. In der Abbidlung 3.19 ist die Struktur der Begriffe grafisch dargestellt. Live Topics war für Alta Vista ein neuer Versuch, um den Ausbruch aus der Textretrievalumgebung zu schaffen. Leider ist die Weiterentwicklung dieses Verfahrens eingestellt worden und auf der Webseite von *Alta Vista* ist nichts mehr darüber zu finden.

[18] www.yahoo.com
[19] www.caligari.com
[20] www.altavista.com

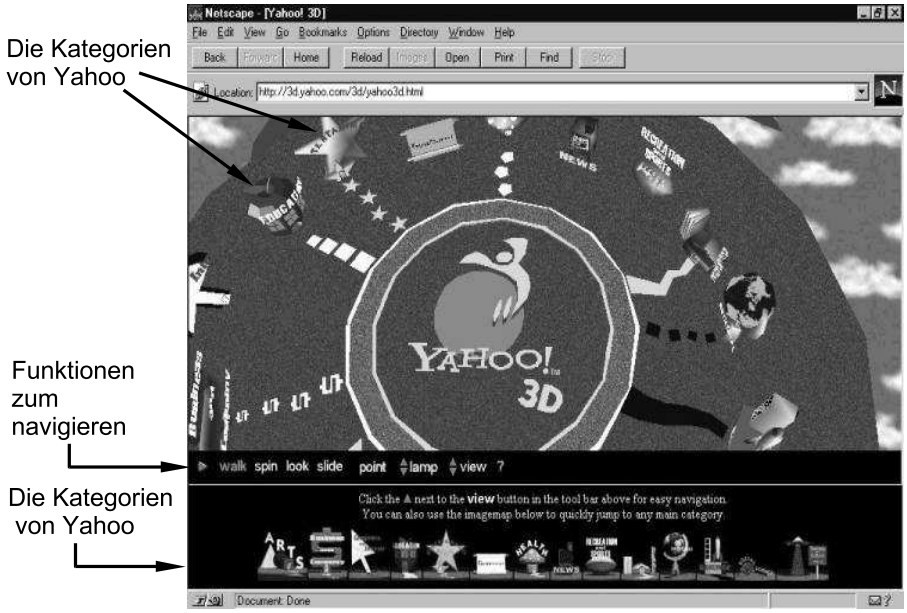



Abbildung 3.18: Die Suchergebnisdarstellung von Yahoo! 3D [Neu01]

### 3.4.3 Vivisimo

*Vivisimo Clustering*[21] ist ein Interface für Meta-Suchmaschinen, das im Juni 2000 von Computerwissenschaftlern der Carnegie Mellon Universität in Pittsburg entwickelt wurde. Sie clustert Such- oder Datenbankanfrageergebnisse in sinnvollen hierarchischen Verzeichnissen vollständig und schnell. Die Verzeichnisse sind nach Wörtern oder Phrasen geordnet, aufgelistet und textuell hervorgehoben. Die Wortkennzeichnung für die Verzeichnisse wird aus der Überschrift und der Beschreibung übernommen. Grafisch dargestellt werden diese Kategorien in einem Verzeichnisbaum auf der linken Seite eines zweigeteilten Fensters. Am Anfang des Verzeichnisbaums steht der gesuchte Suchbegriff mit der gesamten Anzahl der Treffer. Das Ergebnis resultiert aus einer oder mehreren Suchmaschinen oder anderen Informationsquellen. Auf der rechten Seite wird entweder das gesamte Ergebnis (wie von anderen Suchmaschinen bekannt ist, als Trefferliste) angezeigt. oder das Ergebnis bezüglich einer ausgewählten Kategorie.
*Vivisimo* ermöglicht, Inhalte schnell und übersichtlich zu gliedern. Diese Gliederung erleichtert dem Benutzer die Suche zu einem bestimmten Themengebiet. Der Benutzer kann leider die Einordnung und Eingruppierung des Ergebnisses nicht beeinflussen. Innerhalb des Verzeichnisbaumes kann eine Webseite mehrfach unter verschiedenen Begriffen erscheinen, oder auch die Begriffe selbst.

### 3.4.4 VisIT

*VisIT* (**Vis**ualization of **I**nformation **T**ool) ist eine grafische Benutzer-Schnittstelle zu Suchmaschinen und Datenbanken. Sie wurde am Beckmann Institute for Advanced Science and Technology[22] der University of Illinois[23] entwickelt. Die Anfrage wird in der Grundeinstellung an *Google* gestellt, es können auch andere Dienste (z. B. *Teoma, Fast, Yahoo, Open Directory*) in die Suche eingebunden werden. Das Suchergebnis wird als Link-Struktur angezeigt. Die einzelne Webseiten werden als farbige Rechtecke (Icons) symbolisiert. Die Icons werden nach dazugehöriger Website in Kategorien eingruppiert. In jeder Kategorie sind rote und graue Icons zu sehen.

[21] www.vivisimo.com
[22] http://www.beckman.uiuc.edu/
[23] http://www.uiuc.edu/

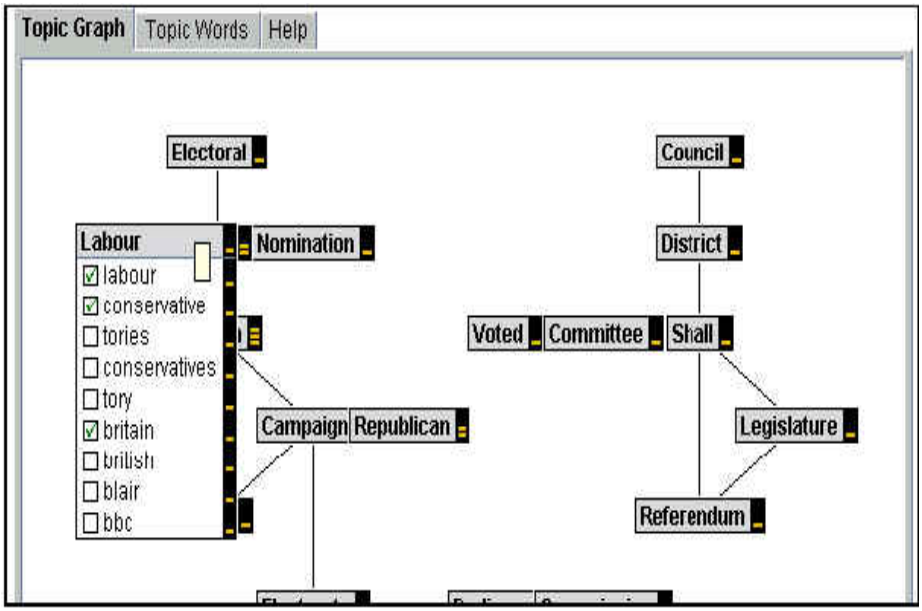



Abbildung 3.19: Die Suchergebnisdarstellung mit Livetopics [Tra97]

Die roten Icons stellen die zutreffendsten Webseiten für die Anfrage in der entsprechenden Kategorie dar, während die grauen Icons interessant sein können. Die Farbe für jedes Element in der grafischen Darstellung kann umgestellt werden. Die inhaltlichen Zusammenhänge werden durch Pfeile angezeigt. Zuerst wird ein Teil des gesamten Ergebnisses dargestellt. Das ganze Ergebnis ist nicht auf einen Blick sichtbar. Durch Scrollen können alle Teile erreicht werden. Der Informationsraum kann als Grafik gespeichert und später wieder geöffnet werden. Es stehen umfangreiche Funktionen zur Interaktion mit den Kategorien zur Verfügung. Diese können gelöscht, mehrfach kopiert und nach Wichtigkeit angeordnet werden. Um die Kategorien besser zu kennzeichnen, können sie entweder umbenannt oder mit Beschriftungen versehen werden. Geht der Benutzer mit dem Mauszeiger auf ein Icon, erhält er in einem sich öffnenden Fenster kurze Informationen zum Inhalt der Webseite. So kann er bestimmen, ob diese Webseite interessant ist oder nicht.

### 3.4.5 Kartoo

*Kartoo*[24] ist eine französische Meta-Suchmaschine, die im Jahr 2002 von Laurent Baleydier und der Firma Kartoo.SA. entwickelt wurde. *Kartoo* ist mit Flash programmiert worden, obgleich sie eine wahlweise freigestellte HTML-Schnittstelle hat. Die Suchmaschine bestimmt mit speziellen Textanalyseverfahren Assoziationen zwischen den Dokumenten. Jede gefundene Webseite wird als Kugel, deren Größe proportional zur Relevanz ist, dargestellt. Die Knoten sind durch dynamische, semantische Links miteinander verbunden und farbig kodiert. Es wird eine Anzahl von zehn Kugeln pro Seite dargestellt. Wird ein Assoziationswort beim Anklicken der Symbole „+“ oder „−“ ausgewählt, so wird interaktiv das assoziierte Themengebiet zur ursprünglichen Suche ein- oder ausgeschlossen. Damit wird eine neue Anfrage gestartet, bis die Suche verfeinert oder in einen anderen Themenbereich verzweigt ist. Bei Berührung einer Kugel mit dem Mauszeiger wird eine kurze Beschreibung des Seiteninhaltes erscheinen. Fährt man mit dem Mauszeiger über eine Kugel, so werden alle möglichen Beziehungen zu anderen Kugeln hervorgehoben. Man muß umblättern, um alle Teile der grafischen Darstellung betrachten zu können.

[24] http://www.kartoo.com

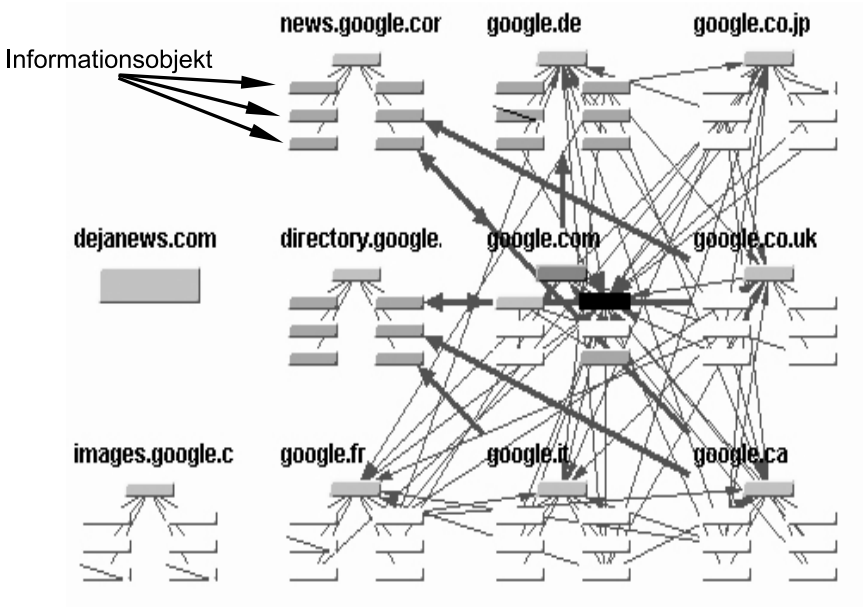



Abbildung 3.20: Suchdarstellung mit VisIT, entnommen von http://www.visit.uiuc.edu/

### 3.4.6 Fazit

Die Suchdienste haben sich seit ihrer Entwicklung von der in Textform gewöhnlichen Ergebnisdarstellung nicht getrennt. Einige Suchmaschinen sind durch eine visuelle Darstellung der Ergebnisse erweitert worden. Alle genannten Erweiterungen haben sich nicht durchgesetzt und wurden meistens nach kurzer Zeit wieder aus den Suchdiensten entfernt. Gründe dafür sind von den Betreibern nicht genannt worden, es kommen folgende Vermutungen dazu in Betracht.

- Die Suchdienste sind kommerziell orientiert und finanzieren sich zum großen Teil durch Werbung. Experimente, die zu weit vom gewohnten Nutzerverhalten abweichen, könnten zur Abwanderung von Benutzern führen, was einen geringeren Marktwert als Werbeträger zur Folge haben würde.

- Die Prioritäten bei der Weiterentwicklung der Suchdienste liegen bei der Effizienz der Suchalgorithmen, guten Ranking-Verfahren und der Größe der Indexe. Die grafische Darstellung der Suchergenisse wurde bisher offensichtlich lediglich als „verzichtbare Zugabe“ betrachtet.

- Es existiert bisher kein anwendbares Konzept für die Integration einer grafischen Darstellung mit intuitiver Interaktion in den Suchprozeß. Die gezeigten Ansätze sind auf die pure grafische Darstellung der Suchergebnisse beschränkt.

- Der Anwender hat durch die gezeigten Ansätze bisher keinen wirklichen Mehrnutzen oder Effizienzgewinn bei der Suche nach interessanten Dokumenten. Hinzu kommt, daß die verwendeten Darstellungsformen sehr gewöhnungsbedürftig und nur für einen geringen Teil der Nutzer geeignet sind.

Zusammenfassend kann gesagt werden, daß es sich bei diesen Ansätzen lediglich um eine „schönere“ Darstellung der Textliste handelt. Die Handhabung ist – mit allen ihren im Abschnitt 2.4 beschriebenen Nachteilen – erhalten geblieben.

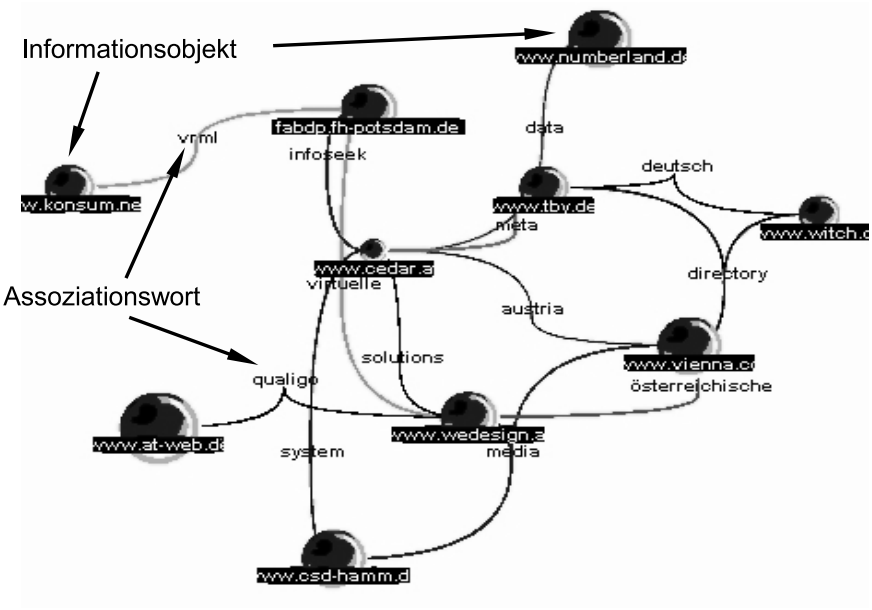



Abbildung 3.21: Suchdarstellung mit Kartoo.com

## 3.5 Der neue Ansatz

In den vorhergehenden Abschnitten wurden Verfahren und Ansätze zur Visualisierung von Informationsräumen vorgestellt, die teilweise ein sehr hohes technisches Niveau aufweisen. In der Praxis ist jedoch keine dieser Methoden zur breiten Anwendung in Suchmaschinen gekommen. Auch viele Versuche von relativ einfachen grafischen Hilfsmitteln wie im Abschnitt 3.4 haben sich nicht durchgesetzt. Mögliche Ursachen dafür sind:

- Es handelt sich um komplizierte grafische Darstellungen mit hoher Komplexität.
- Es existiert eine „Hemmschwelle“ bei Benutzern, die schnell zu ihrem Ergebnis kommen möchten, ohne vorher den Umgang mit einem grafischen System aufwendig erlernen zu müssen (wenn sich Sinn und Aufbau einer grafischen Darstellung nicht sofort dem Benutzer erschließt, wird er weiter die textuelle Liste benutzen),
- Alle vorgestellten Verfahren kennen genau eine Art der grafischen Darstellung, demgegenüber sind sowohl die Preferezen als auch die Fähigkeiten der Benutzer höchst unterschiedlich.
- Die Verfahren sind nicht universell für verschiedene Dimensionen von Informationsräumen einsetzbar.
- Ein Suchlauf mit einer Internet-Suchmaschine bringt häufig mehrere Tausend Dokumente als Ergebnis. Bei jedem der vorgestellten Verfahren würde eine solche Menge von Objekten zu einer überladenen Darstellung führen, die vom Benutzer nicht analysiert werden kann.
- Eine dreidimensionale Darstellung erfordert eine gewisse Performance der verwendeten Hardware. Diese steht erst jetzt auch auf gewöhnlichen Arbeitsplatzrechnern zur Verfügung.

Abhilfe kann ein Ansatz schaffen, der nicht ein monolithisches System als Ziel hat, sondern ein Modul mit folgenden Eigenschaften:

- Es stehen mehrere Möglichkeiten der grafischen Darstellung zur Verfügung, die je nach Dimension des Informationsraumes, den Interessen des Benutzers und den Fähigkeiten der verwendeten Hardware ausgewählt werden können.
- Die Menge der grafischen Objekte kann bereits vor der grafischen Darstellung reduziert werden. Die dazu benötigten Filter können kombiniert und erweitert werden.
- Das Modul verfügt über eine Schnittstelle, die die Anbindung an beliebige Suchmaschinen erlaubt.
- Der Benutzer kann Dokumente als „interessant“ oder „nicht interessant“ kennzeichnen.

## 3.6 Fazit

In diesem Kapitel wurde eine Bestandsaufnahme von existierenden Verfahren zur Darstellung von Informationen durchgeführt. Die Untersuchung umfaßte allgemeine Darstellungsverfahren und Verfahren, die speziell für die Darstellung von Suchergebnissen entwickelt wurden. Dabei hat sich gezeigt, daß derzeit keine Visualisierungsmethode im Einsatz ist, die als effiziente Hilfe bei der Extraktion der „interessanten“ Information aus einem Suchergebnis geeignet ist. Es wird eine Methode benötigt, die die verfügbaren Informationen auf eine übersichtliche Weise für den Benutzer aufbereitet, und anschließend eine schnelle zielgerichtete Suche im Informationsraum unterstützt. Der Versuch eines Entwurfes einer solchen Methode wird in den folgenden Abschnitten unternommen.

# Kapitel 4

# Anforderungspezifikation zur Visualisierung von Informationsräumen

In diesem Kapitel wird dargelegt, wie das Szenario der Visualisierung von Informationsdokumenten, das im Kapitel 2 beschrieben ist, umgesetzt wird. Das Verhalten und die Eigenschaften des zu entwickelnden Systems sollen mit methodischen Hilfsmitteln erfaßt werden. Dies sind geeignete Modelle, mit denen die statische Struktur und das dynamische Verhalten des Systems sichtbar gemacht werden. Dafür wird die Unified Modeling Language (UML) zum Einsatz kommen, die einen Satz von Modellelementen mit zugeordneten grafischen Symbolen definiert. Mit diesen Elementen können Modelle konstruiert werden, die verschiedene Sichten des zu entwickelnden Systems repräsentieren, wie z. B. einen Überblick über das Gesamtsystem, das von außen erkennbare Verhalten des Systems sowie die logische Struktur.

## 4.1 Einführung

Die objektorientierte Entwicklung des Visualisierungssystems wird durch die iterative Abfolge der Phasen *Objektorientierte Analyse* (OOA), *Objektorientiertes Design* (OOD) und *Implementierung* gekennzeichnet [Neu98]. Dies ist unter Umständen ein sehr komplexer und langwieriger Prozeß, an dem eine Vielzahl von Personen beteiligt sein kann. Daher ist es notwendig, dafür gut definierte, abgegrenzte und übersichtliche Methoden für die einzelnen Arbeitsabschnitte zu verwenden.

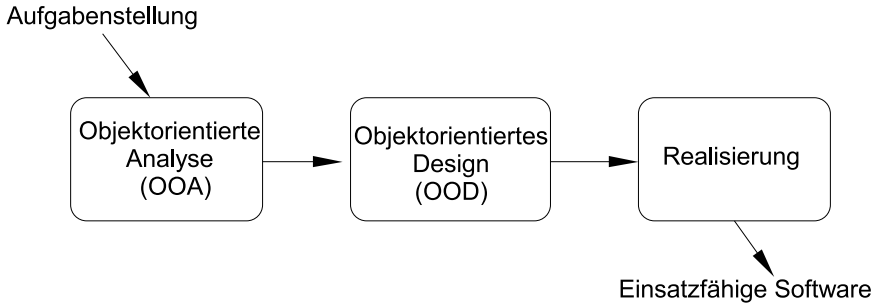



Abbildung 4.1: Der Objektorientierte Entwicklungsprozeß [Oes98]

In den 80er Jahren wurden zahlreiche objektorientierte Methoden entwickelt und in verschiedenen Bereichen eingesetzt. Sie sind ein Hilfsmittel für die Unterstützung der iterativen Phasen des objektorientierten Entwicklungsprozesses, durch welches die Vorgehensweise zur Erzielung bestimmter Ergebnisse genau definiert wird [Neu95]. In [Neu98] ist ein tabellarischer Überblick der verschiedenen Methoden zu finden. Einige der wichtigsten Methoden waren: Boochs Methode [Boo94, Boo96], Methode von Rumbaugh - OMT (Object Modelling Technique) [RJ94, J.95, J.96], Methode von Jacobson - OOSE (Object-Oriented Software Engineering) [Jac92, Jac98], Methode von Booch - OOAD (Object Orientied Analysis and Design) [Boo94] und Shlaer/Mellor (OOSA) [Shl88].
Jede der Methoden ist auf bestimmte Anwendungsbereiche spezialisiert und begrenzt. Obwohl für jede Methode eine eigene grafische Notation und spezielle Konzepte für Modelle definiert sind, basieren sie auf denselben Konzepten. Es wurde versucht, eine vereinheitlichte objektorientierte Methode zu entwerfen, die als Standard in der Software-Industrie eingeführt werden kann.
1995 haben sich Booch und Rumbaugh und etwas später Jacobson entschieden, ihre Methoden gemeinsam zusammenzuführen. Die Zusammenführung der drei Methoden bildete die Unified Modeling Language (UML). Die UML ist erstmals eine international standardisierte Notation, die in diesem Abschnitt einleitend vorgestellt wird.

## 4.2 Die Unified Modeling Language (UML)

### 4.2.1 Entwicklung der UML

Als erstes Ergebnis dieser Zusammenarbeit wurde 1995 die Version 0.8 von UML veröffentlicht. Mitte 1996 folgte die überarbeitete und erweiterte Version 0.9, im September 1996 die revidierte Version 0.91. Im Jahr 1997 wurde die Version 1.0 bei der Object Management Group (OMG) als Standardisierungsvorschlag eingereicht, und eine Beschreibung der Version 1.1 veröffentlicht [Oes98, Neu98]. Im September 1997 ist die überarbeitete Version 1.1 der Spezifikation erschienen. Die Versionen[1] 1.2 bis 1.5 enthalten jeweils einige Korrekturen.

### 4.2.2 Die Notation der UML

Die UML [Oes98] ist eine standardisierte Sprache und Notation zur:

- *Spezifikation:* UML-Elemente sind mit eindeutiger Semantik versehen,
- *Konstruktion:* die Abbildung von Modellen auf verschiedene Programmiersprachen ist möglich,
- *Visualisierung:* UML bietet Diagramme und eine grafische Notation zur Darstellung von Softwaresystemen,
- *Dokumentation:* UML bietet Konstrukte zur Verwaltung und Erstellung einer Dokumentation (Architektur und Design, Anforderungen an die Software, Quell-Code, Projektpläne, Ablaufpläne, Testfälle, Prototypen und verschiedene Versionen eines Softwaresystems.)

Allerdings fehlen eine Notation und Beschreibung für Softwareentwicklungsprozesse. Die UML ist keine Methode, sie ist lediglich ein Satz von Notationen zur Formung einer allgemeinen Sprache zur Softwareentwicklung. Eine Methode beinhaltet Empfehlungen zur Vorgehensweise bei Entwicklungsprozessen. Die Notation der UML ist eine Verschmelzung der grafischen Syntax der verschiedenen Methoden der Entwickler, die UML erschaffen haben. Sie wird heute in vielen Einsatzbereichen angewendet.

[1] Aktuelle Informationen zu UML sind unter http://www.omg.org/uml/ zu finden.

UML wurde für den Software-Entwurf in dieser Arbeit ausgewählt, da sie:

- eine universelle Beschreibungssprache für alle Arten objektorientierter Softwaresysteme bereitstellt,
- den Benutzer mit einer ausdrucksvollen visuellen Modellierungssprache bei der Entwicklung von sinnvollen Modellen und deren Austausch unterstützt,
- eine formelle Basis für das Verstehen der Modellierungssprache liefert und
- Spezifikationen unterstützt, die unabhängig von Programmiersprachen und Entwicklungsprozessen sind.

Weiterhin enthält UML verschiedene Diagramme, die wiederum verschiedene grafische Elemente besitzen. Das bedeutet, daß es mit UML möglich ist, für ein und denselben Sachverhalt mehrere Darstellungsarten zu verwenden. Das hat den Vorteil, daß viel ausgedrückt werden kann, allerdings auch den Nachteil, daß eine längere Einarbeitungszeit notwendig ist.
UML definiert einen Satz von Modellelementen mit zugeordneten grafischen Symbolen, aus denen Modelle konstruiert werden können. Diese Symbole werden hier kurz erläutert [Neu98, For01, Erl00].

### Basiselemente der UML

**Klassen und Objekte**

Klassen und Ojekte werden in UML durch Rechtecke repräsentiert. Eine Klasse hat einen Namen, besitzt eine Anzahl von Attributen und stellt Methoden (Operationen) bereit, mit denen die Attribute verändert werden können. Die Objekte unterscheiden sich von den Klas-

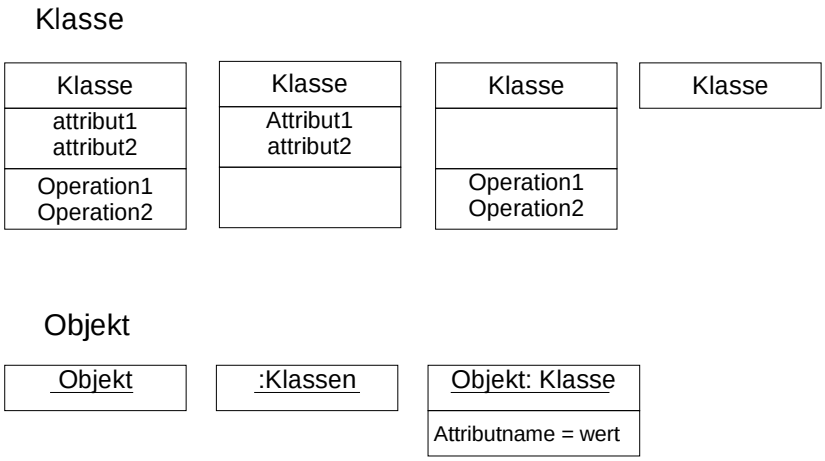



Abbildung 4.2: Darstellung von Klassen und Objekten [Oes98]

sen dadurch, daß ihr Name unterstrichen ist. Durch Doppelpunkt vom Namen getrennt kann auch die Klasse angegeben werden, zu der das Objekt gehört. Attribute sind bei Objekten bereits mit Werten belegt (Siehe Abbildung 4.2).

**Schnittstelle, Schnittstellenklasse**

Die Schnittstellen beschreiben das externe Verhalten von Modellelementen (von Klassen und Komponenten). Schnittstellenklassen sind abstrakte Klassen, die ausschließlich abstrakte Operationen definieren [Oes98]. Eine Schnittstellenklasse wird durch den Stereotyp <<interface>> gekennzeichnet.

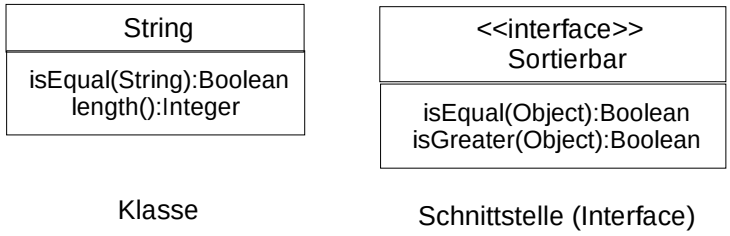



Abbildung 4.3: Schnittstelle [Oes98]

Eine Klasse implementiert eine Schnittstelle, wenn alle definierten Operationen in der Schnittstelle bereitgestellt werden. Eine Klasse kann auch mehrere Schnittstellen implementieren. Wenn das so ist, besteht eine Realisierungsbeziehung zwischen implementierender Klasse und Schnittstellenklassen. Es wird durch den Stereotyp <<implement>> gekennzeichnet. Schnittstellenklassen können andere Schnittstellen erweitern (vererben). Diese Beziehung hat den Stereotyp <<extend>>, (siehe Abbildung 4.3).

**Pakete**

Pakete sind eine Gruppierung von Modellelementen beliebigen Typs [Oes98], die zu einer Einheit gesammelt wurden. Mit Paketen wird das Gesamtmodell in kleinere überschaubare Einheiten gegliedert. Sie helfen, einen Überblick über ein großes Modell zu verschaffen. Innerhalb eines Paketes sind Namen eindeutig vergeben. Ein Paket wird als Form eines Aktenregisters dargestellt (Siehe Abbildung 4.4). Pakete können wiederum Pakete beinhalten.

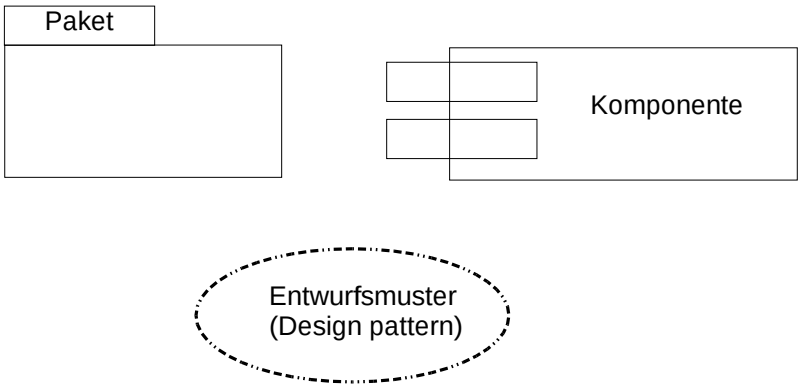



Abbildung 4.4: Paket und Komponenten [Oes98]

**Komponente**

Komponenten sind Softwaremodule mit ausführbaren Bestandteilen. Sie definieren Grenzen, sie gruppieren und gliedern eine Menge einzelner Elemente. Sie können über definierte Schnittstellen verfügen [Oes98]. Eine Komponente wird als Rechteck dargestellt, das am linken Rand zwei kleine Rechtecke trägt (Siehe Abbildung 4.4). Innerhalb der Komponente wird der Name der Komponente beschrieben. Sie können andere Elemente (Objekte, Komponenten, Knoten) enthalten.

**Entwurfsmuster (Design Pattern)**

Entwurfsmuster beschreiben generalisierte Lösungsideen zu immer wiederkehrenden Entwurfsproblemen [For01]. Sie beschreiben die Problemstruktur und den Lösungsansatz. Sie

werden als gestrichelte Ellipse dargestellt, die den Namen des Musters enthält. (Siehe Abbildung 4.4). Zu den Klassen, die vom Entwurfsmuster betroffen sind, werden gestrichelte Pfeile gezeichnet.

### Beziehungenselemente der UML

Die möglichen Beziehungen zwischen Klassen, bzw. deren Instanzen(Objekte) sind in der Abbildung 4.5 dargestellt.

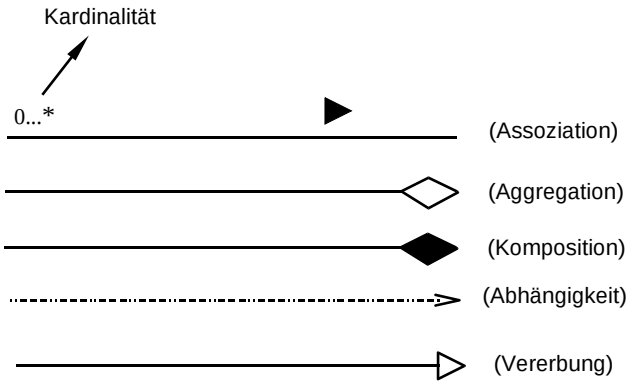



Abbildung 4.5: Beziehungen [Oes98]

**Assoziation**
Eine Assoziation beschreibt eine Relation zwischen Objekten einer oder mehrerer Klassen. Sie modelliert stets Beziehungen zwischen Objekten, nicht zwischen Klassen [Bal99]. Es werden unidirektionale Assoziationen (nur einseitig direkt navigierbar) und bidirektionale Assoziationen (beidseitig direkt navigierbar) unterschieden. Diese Beziehungen werden durch direkte Linien oder durch rechtwinklicke Linien zwischen den Klassen dargestellt. An den Enden kann die Kardinalität der Beziehung angegeben werden. Die Kardinalität gibt an, mit wievielen Objekten der gegenüberliegenden Klasse ein Objekt assoziiert sein kann. Jede Assoziation kann mit einem Namen versehen werden, um zu zeigen, warum oder worin diese Beziehung besteht. Außerdem können zusätzlich Rollennamen angegeben werden. Damit wird beschrieben, wie das Objekt durch das in der Assoziation gegenüberliegende Objekt gesehen wird.

**Aggregation**
Die Aggregation ist eine Assoziation, deren beteiligte Klassen eine Gesamt-Teil-Struktur darstellen [Bal99]. Das bedeutet, daß zwischen den Objekten der beteiligten Klassen eine Rangordnung gilt. Diese läßt sich durch „ist Teil von“ oder „besteht aus“ beschreiben. Die Objekte der Aggregation bilden einen gerichteten azyklischen Graphen. Das heißt, daß eine Klasse B Teil von einer Klasse A ist, dann darf die Klasse A nicht Teil von B sein. Die Aggregation wird als Linie zwischen zwei Klassen dargestellt und zusätzlich mit einer Raute versehen. Die Raute steht auf der Seite des Aggregats (oder des Ganzen).

**Komposition**
Eine Komposition beschreibt eine starke Form der Aggregation, bei der die Teile vom Ganzen existenzabhängig sind [Neu98]. Hier gilt folgendes [Bal99]:

- Jedes Objekt der Teilklasse kann nur Komponente eines einzigen Objektes der Aggregatklasse sein. Die Kardinalität auf der Seite der Aggregatklasse darf nur 1 sein.

- Wird das Ganze kopiert, so werden auch seine Teile kopiert.
- Wird das Ganze gelöscht, so werden auch seine Teile automatisch gelöscht.

Die Komposition wird auch als Linie zwischen Klassen gekennzeichnet und mit einer ausgefüllten Route auf der Seite des Ganzen gekennzeichnet.

**Vererbung (Generalisierung)**
Vererbung beschreibt eine Beziehung zwischen einer allgemeinen Klasse (Oberklasse) und einer spezialisierten Klasse (Unterklasse). Die Eigenschaften der Oberklasse werden an die enstprechenden Unterklassen weitergegeben oder vererbt [Neu98]. Die Vererbung wird mit einem großen nicht ausgefüllten Pfeil dargestellt. Der Pfeil zeigt von der Unterklasse zur Oberklasse.

**Abhängigkeit**
Eine Abhängigkeit ist eine Beziehung zwischen zwei oder mehr Modellelementen, die ausdrückt, daß eine Änderung des unabhängigen Elementes eine Veränderung des anderen, von diesem abhängigen Elementes zur Folge hat. Die Kennzeichnung erfolgt mittels eines gestrichelten Pfeiles, der vom abhängigen zum unabhängigen Element führt (siehe Abbildung 4.5). Die durch eine Abhängigkeit verbundenen Modellelemente können verschiedensten Typs sein (Klassen, Pakete, Methoden, Schnittstellen).

## 4.3 Die Objektorientierte Analyse (OOA)

Die objektorientierte Analyse dient der Spezifikation der Anforderungen an ein zu entwickelndes Softwaresystem. Dies geschieht völlig unabhängig von der später durchzuführenden Implementierung und dem Zielsystem. Das Ausgangsmaterial sind die realen Objekte, ihre Beziehungen und die durchzuführenden Vorgänge, die auf ein Modell abgebildet werden müssen. Aus diesem (objektorientierten) Modell werden Klassen abgeleitet, es beschreibt die Struktur und Semantik des analysierten Problemes.
Eine Methode zum Erarbeiten des OOA-Modells ist die Aufteilung der bestehenden Anforderungen in Anwendungsfälle und deren Analyse.

### 4.3.1 Die Use-Case Analyse (Analyse der Anwendungsfälle)

Die *Use Case Analyse* ist ein aussagekräftiger Ereignisfluß, der eine solide Methode bietet, die die Abgrenzung des zu entwickelnden Systems zu seiner Systemumgebung, die Analyse der operationellen Anforderungen und die Spezifikation des Systemverhaltens untersützt. Die Methode ist von Ivar Jacobson entwickelt und wurde 1992 in [Jac92] beschrieben.
Die zugrundeliegenden Konzepte, die Notation und die methodische Vorgehensweise für die Methode Use-Case-Analyse werden hier erläutert.

#### Aktor

Als Aktor wird eine Rolle, die meistens vom Anwender übernommen wird, bezeichnet [For01]. Aktoren sind nicht Bestandteil des Systems, sie können ein externes Gerät oder ein externes System sein. Der Aktor kann aktiv mit dem System interagieren und kann ebenso ein passiver Informationsempfänger sein. In der UML gibt es kein spezielles grafisches Symbol für einen Aktor. Dafür wird das normale Klassensymbol in Verbindung mit dem Stereotyp $<< actor >>$ angewendet oder die Benutzung eines Ikons in Form eines Strichmännchens vordefiniert (Siehe Abbildung 4.6). Der Name des Benutzers wird unterhalb des Ikons geschrieben.

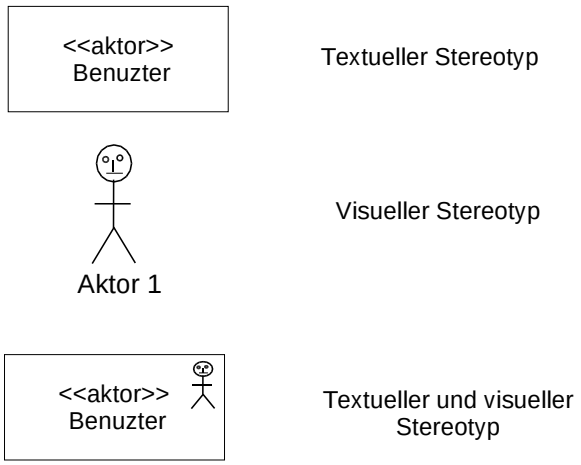



Abbildung 4.6: Aktorenstruktur [Oes98]

**Anwendungsfall (Use Case, Nutzungsfall)**

Ein Anwendungsfall beschreibt eine Reihe von Aktivitäten im System aus der Sicht des Aktors, die ein konkretes faßbares Ergebnis liefern [Pau]. Durch die Anwendungsfälle werden die Anforderungen an das System beschrieben. Die Anwendungsfälle beschreiben typische Interaktionen zwischen Benutzer und dem System. Sie stellen die externe Schnittstelle dar und spezifieren damit die Anforderungen, was das System zu tun hat. Die Anwendungsfälle werden durch Ellipsen grafisch dargestellt. Zur jeder Ellipse existiert ein Text, der den Anwendungsfall genauer beschreibt.
Die folgenden Regeln sollen bei der Arbeit mit Anwendungsfällen beachtet werden [Pau].

- jeder Anwendungsfall ist mindestens mit einem Aktor verbunden,
- jeder Anwendungsfall hat einen Auslöser (z.B. einen Aktor),
- jeder Anwendungsfall führt zu einem relevanten Ergebnis

Eine Spezifikation eines Anwendungsfalls enthält folgende Informationen [Neu98]:

- eine kurze Überschrift,
- eine übersichtliche Beschreibung des Zweckes,
- die Namen der an dem Anwendungsfall beteiligten Aktoren,
- Ereignisse, die einen konkreten Anwendungsfall auslösen,
- die Bedingungen, die zur Auslösung des Anwendungsfalls erfüllt sein müssen (Vorbedingungen),
- kritische Ausnahmesituationen und Fehlerfälle,
- die relevanten Ereignisse und Nachbedingungen.

Damit wird das externe Systemverhalten beschrieben. Da das System unbeschränkte Anwendungsmöglichkeiten haben kann, ist die Beschreibung der Anwendungsfälle durch die grafische und textuelle Notation begrenzt. Dafür werden zusätzlich Diagramme angeboten, die eine Beziehung zwischen Aktoren und Anwendungsfällen zeigen. Die Abbildung 4.7 stellt ein Anwendungsfalldiagramm (Use Case Diagramm, Nutzungsdiagramm) dar. Im Anwendungsdiagramm ist eine Menge

von Anwendungsfällen und eine Menge von Aktoren und Ereignissen, die daran beteiligt sind, dargestellt. Alle Anwendungsfälle zusammen bilden ein Modell, das die Anforderungen an das externe Verhalten des Gesamtsystems beschreibt. Die Anwendungsfälle sind durch Linien mit den beteiligten Klassen verbunden. Die Anwendungsfälle sind eingerahmt. Der Rahmen entspricht der Systemgrenze.

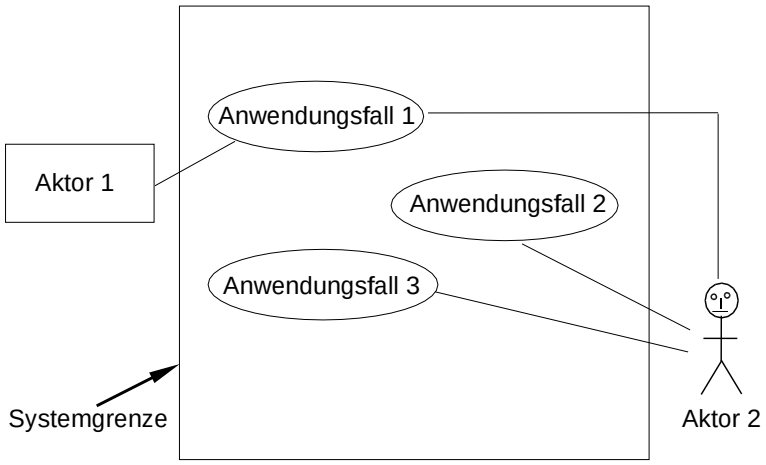



Abbildung 4.7: Anwendungsfalldiagramm [Oes98]

Die Anwendungsfälle können im Anwendungsdiagramm untereinander in Verbindung stehen. In der UML sind drei Arten von Beziehungen zwischen den Anwendungsfällen definiert [Pau, Neu98].

- mit *<<include>>* (ersetzt die *<<uses>>*-Beziehung aus UML 1.1) wird gesagt, daß ein Anwendungsfall in einem anderen Anwendungsfall stattfindet.
- mit *<<extends>>* wird ausgesagt, daß in einer bestimmten Situation der Anwendungsfall durch einen anderen erweitert wird.
- mit *Generalisierung* wird ausgesagt, daß ein Anwendungsfall die Eigenschaften eines übergeordneten Anwendungsfalls erbt und diese überschreiben oder erweitern kann.

**Szenarien**

Die Anwendungsfälle werden zusammen erfaßt und dann schrittweise in nachgeordnete Anwendungsfälle gegliedert und zerlegt. Die weitere Zerlegung der Anwendungsfälle wird durch Szenarien beschrieben. Die Szenarien sind eine spezifische Folge von Aktionen bzw. Reaktionen von Vorgängen im Ablauf des Anwendungsfalls in zeitlicher Ordnung. Szenarien sind Beschreibungen des Geschehens an der Schnittstelle vom Benutzer (Aktor) zum System. Ein Anwendungsfall besteht aus zwei Teilen [Neu98]:

- *einer übersichtlichen Definition*, wobei eine Übersicht über die wesentlichen Aspekte der Anwendungsfälle bereitgestellt wird.
- *den zugeordenten Szenarien*, in denen der detaillierte Ablauf der Anwendungsfälle beschrieben wird.

Die Strukturierung von Szenarien wird durch verschiedene Arten von Szenarien unterstützt. Das sind [Neu98]:

- *die primären Szenarien*, die die Abläufe zur Erfüllung der operationalen Anforderungen spezifizieren,

- *die sekundären Szenarien*, die die Reaktion auf definierte Fehlerfälle (Gerätefehler oder Eingabefehler) und Ausnahmesituationen (z.B. Überlauf eines Eingabepuffers) beschreiben.

## 4.4 Anforderungsspezifikation des Visualisierungssystems

### 4.4.1 Problembeschreibung

Das zu entwerfende Visualisierungstool ist ein System, das die infolge einer Suchanfrage ermittelte Informationsmenge grafisch (vorzugsweise dreidimensional) darstellt. Der Benutzer stellt der Suchmaschine seine Anfrage, auf deren Basis die Suchmaschine eine Liste von Dokumenten als Ergebnis zusammenstellt. Diese Form der Bereitstellung hat sich bis heute nicht wesentlich verbessert. Die Ausgabe erfolgt immer noch in der Listenform, die es nicht gestattet, Zusammenhänge zwischen den Dokumenten und die Nähe der Dokumente zu den einzelnen Suchbegriffen zu zeigen. Es wird angenommen, daß diese Informationsmenge ein Informationsraum ist, der durch die Suchbegriffe aufgespannt wird. Viele der genannten Probleme können gelöst werden, wenn es gelingt, diesen Informationsraum zu visualisieren und „begehbar“ zu machen. Es wird ein dreidimensionales Visualisierungssystem entwickelt, um einen guten Überblick der Dokumentmenge zu vermitteln. Diese Art der Informationssuche kann – wie im Abschnitt 2.7.2 beschrieben – als Zyklus dargestellt werden. Nach den durch den Benutzer angegebenen Suchbegriffen wird die Suchmaschine eine Liste von Dokumenten erzeugen. An dieser Stelle ist wichtig, daß die Suchmaschine zu jedem Dokument auch die berechneten Relevanzen pro Suchbegriff bereitstellt. Diese Dokumentenmenge soll auf ein sinnvolles Maß reduziert werden, da sie unter Umständen dreidimensional dargestellt werden soll und diese Darstellung eine hohe Performance erfordert. Die Reduzierung der Dokumentenmenge findet in der Filterstufe statt, die im Abschnitt 2.6.3 erläutert wurde. Sie kann verschiedene Filter enthalten, die bei folgenden Sachverhalten zum Einsatz kommen können:

**Filterung nach dem Autor:**
Der Ausgangspunkt für diese Filterung ist folgende Situation: Der Nutzer hat die Menge der interessanten Dokumente bereits eingegrenzt, indem er in den entsprechenden Bereich des Informationsraumes navigiert ist. Mit Hilfe der vom System zur Verfügung gestellten Interaktionsmechanismen hat er den Inhalt einiger Dokumente näher untersucht. Ist ein relevantes Dokument gefunden worden, können nun alle vorhandenen Dokumente desselben Autors herausgefiltert werden. Diese Möglichkeit ist besonders dann nützlich, wenn das Ziel der Informationssuche unscharf ist.

**Filterung nach dem Erscheinungsdatum:**
Ein großes Problem des World Wide Web ist, daß ein Teil der Informationsbestände nicht gepflegt wird, was dazu führt, daß eine Reihe der abrufbaren Informationen bereits überholt ist. Einige Dokumente sind auch in verschiedenen Versionen an verschiedenen Orten des Netzes vorhanden.

**Filterung nach der Dokumentengröße:**
Die von einer Suchmaschine gefundenen Dokumente können verschiedenster Art sein: Kurzmeldungen im Umfang weniger Zeilen, Berichte, Aufzählungen, Werbung bis hin zu kompletten Büchern. Die Größe der Datei kann als Hinweis auf die Art des Dokumentes dienen. Es ist jedoch anzumerken, daß dies auf keinen Fall als scharfes Kriterium angesehen werden darf. Bei großen Datenmengen sollten dennoch alle Möglichkeiten zur Reduktion der Datenmenge ausgeschöpft werden.

**Filterung nach dem Ort der Veröffentlichung:**
Dieser Filter kann benutzt werden, um in gewissem Maße Beziehungen zwischen Dokumenten zu visualisieren. Häufig sind wissenschaftliche Veröffentlichungen in Sammlungen eingebunden, z. B. in Büchern, Proceedings von Workshops u.ä. Die auf diese Weise zusammengefaßten Dokumente haben natürlicherweise eine enge thematische Beziehung, die sehr einfach visualisert und nutzbar gemacht werden kann. Hat der Anwender einmal ein solches, für ihn interessantes Dokument gefunden, können alle zur betreffenden Sammlung gehörigen Dokumente herausgefiltert werden. Damit steht dem Nutzer sofort ohne weitere Suche eine Menge von Dokumenten zur Verfügung, die für ihn mit hoher Wahrscheinlichkeit ebenfalls interessant sind.

**Filterung nach Relevanz-Schwellwerten:**
Zu jedem Dokument sind zu jedem Suchbegriff die entsprechenden Relevanzen bekannt. Um die Menge der zu visualiserenden Dokumente zu reduzieren, kann festgelegt werden, daß ein Dokument dessen Relevanz bezüglich eines bestimmten Schlüsselwortes unterhalb eines spezifizierten Schwellwertes liegt, von der Weiterverarbeitung ausgeschlossen wird. Auf diese Weise kann außerdem eine Wichtung der Suchergebnisse vorgenommen werden.

**Verhindern von Redundanzen:**
Redundanzen in der Dokumentmenge (und damit auch im Informationsraum) können auf zwei Arten zustande kommen:

- Dokumente werden auf verschiedenen Servern gefunden und mehrfach erfaßt,
- eine Meta-Suchmaschine wurde verwendet, wobei mehrere beteiligte Suchwerkzeuge Einträge derselben Dokumente in der Ergebnisliste vorgenommen haben.

Mit Hilfe dieses Filters ist es möglich, redundante Dokumente von der Weiterverarbeitung auszuschließen.

Voraussetzung für den Einsatz dieser Filter ist, daß die Parameter, nach denen gefiltert wird, von der Suchmaschine bereitgestellt werden.
Der Informationsraum besitzt keine natürliche Struktur. Daher ist es notwendig, darstellbare Strukturen und Objekte zu generieren. Der Informationsraum mit seinen Dokumenten soll in einer dreidimensionalen Struktur abgebildet werden. Das ist die Funktion der Mappingstufe, die im Abschnitt 2.6.3 beschrieben worden ist.
Das von der Mappingstufe erzeugte dreidimensionale Modell wird nun in ein zweidimensionales Modell umgerechnet. Diese Aufgabe übernimmt die Renderingstufe, die gleichzeitig für die Interaktion mit den Objekten, die Kameraeinstellung und die Navigation im Raum verantwortlich ist.
Das erzeugte Bild wird im den seltensten Fällen sofort das gewünschte Ergebnis präsentieren. Daher muß auf die verschiedenen Etappen des Zyklusses Einfluß genommen werden.
So kann in der Renderingstufe in die interessanten Bereiche des Informationsraumes navigiert werden. In der Mappingstufe können andere Abbildungsvarianten eingestellt werden. Die Filtermechanismen können variiert werden.
Falls es sich heraustellt, daß die Suche kein sinnvolles Ergebnis geliefert hat, sollte die Suchanfrage neu formuliert werden. Dieser Zyklus ist dann beendet, wenn die für den Benutzer interessante Untermenge der gefundenen Dokumente ermittelt ist.

### 4.4.2 Die Analyse des Systemverhaltens

Die Anforderungsspezifikation umfaßt die Definition, die Identifikation und die Darstellung der Nutzungsfälle, sowie die Spezifikation der Szenarien. Der Dialog mit dem Benutzer wird in Form von Anwendungsfällen (Use-Cases) aus der Problembeschreibung extrahiert und beschrieben.
Das System ist für eine Vielzahl von Anwendungsfällen vorgesehen, hier sollen exemplarisch zwei Extremsituationen beschrieben werden. Im ersten Fall wird von einem Benutzer ausgegangen, der auf der Suche nach ganz bestimmten Informationen bzw. Dokumenten ist, also eine *zielgerichtete Suche* durchführt. Das bedeutet, er weiß, mit welchen Suchbegriffen er zum Ziel kommt und findet relativ schnell eine Menge von Dokumenten und kann mit Hilfe einiger weniger Filteroperationen die für ihn relevanten Dokumente finden.
Im zweiten Fall wird ein Benutzer angenommen, der Informationen zu einem Themenkomplex sucht, ohne eine konkrete Vorstellung von den interessanten Dokumenten zu haben. Dieser Nutzer führt also eine *unscharfe Suche* aus. Bei der Eingrenzung der Dokumentenmenge wird er mehr die Visualisierung benutzen, um in den für ihn interessanten Teil des Informationsraumes zu navigieren.

#### Die zielgerichtete Informationssuche

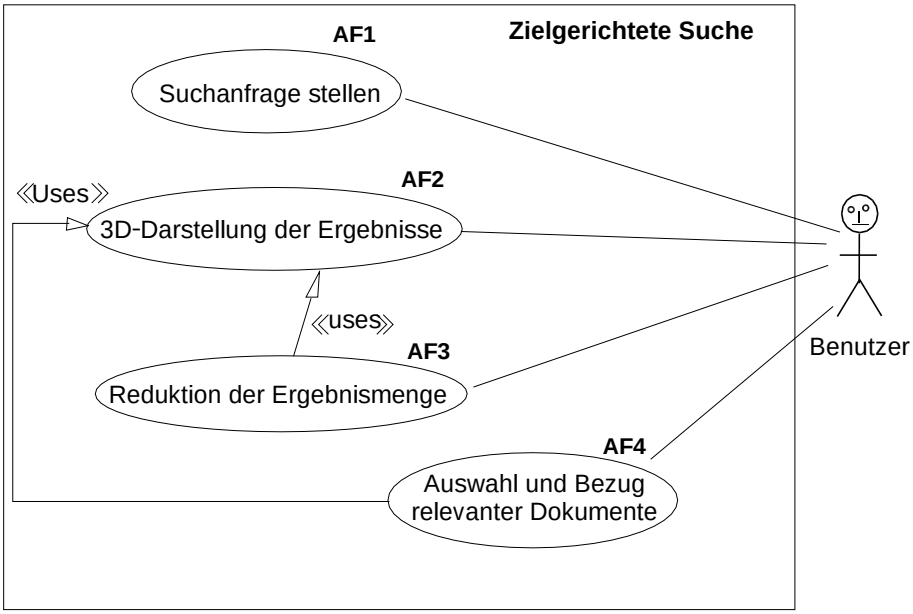



Abbildung 4.8: Zielgerichtete Suche

Das oben beschriebene Szenario einer zielgerichteten Suche kann in vier Anwendungsfälle aufgeteilt werden, die im Abbildung 4.8 beschrieben sind. Dies sind:

1. das Stellen der Suchanfrage,
2. die Auswahl des Mapping-Verfahrens und die Ergebnisdarstellung,
3. die Reduktion der Ergebnismenge und
4. die Auswahl der interessanten Dokumente aus der Ergebnismenge.

Es folgt eine Auflistung dieser vier Anwendungsfälle und ihre Beschreibung. Bei einigen Anwendungsfällen ist eine weitere hierarchische Untergliederung sinnvoll. Diese ist in der Tabelle 4.1, sowie in den Abbildungen 4.9, 4.10 und 4.11 gezeigt. Aus Gründen der Übersichtlichkeit sind die zahlreichen Tabellen der verfeinerten Anwendungsfälle im Anhang A.1 aufgeführt

| **Anwendungsfall** | **elementare Anwendungsfälle** |
|---|---|
| AF1: Suchanfrage stellen | AF11: Initiieren der Suche |
| AF2: 3D-Darstellung der Ergebnisse | AF21: Auswahl des Mapping-Verfahrens<br>AF22: Relevanz-Mapping<br>AF23: Einschalten zusätzlicher Hilfsmittel<br>AF24: Einstellung der Ansichts-Parameter |
| AF3: Reduktion der Ergebnismenge | AF31: Einschalten von Dokument-Filtern<br>AF32: Markieren von uninteresanten Dokumenten<br>AF33: Einstellen von Relevanz-Schwellen |
| AF4: Auswahl und Bezug relevanter Dokumente | AF41: Betrachten von Detailinformationen<br>AF42: Markieren interessanter Dokumente<br>AF43: Download der markierten Dokumente<br>AF44: Speichern der Dokumente<br>AF45: Drucken der Dokumente |

Tabelle 4.1: Relevante Anwendungsfälle für die zielgerichtete Suche

**Anwendungsfall „Stellen der Suchanfrage“**
Aus Sicht des Benutzers ist das Stellen der Suchanfrage identisch zur Arbeitsweise mit konventionellen, textbasierten Suchmaschinen. Der Grundgedanke des Visualisierungsmoduls ist, daß es an eine beliebige Suchmaschine gekoppelt oder in diese integriert werden kann. Entscheidend dafür ist, daß die Suchmaschine ihre berechneten Relevanzen mit den gefundenen Dokumenten über die im Abschnitt 5.9 spezifizierte Schnittstelle zur Verfügung stellt. Mit Hilfe dieser Informationen ist das Visualisierungsmodul in der Lage, entsprechend der Voreinstellungen eine dreidimensionale Darstellung der Dokumentenmenge im aufgespannten Informationsraum zu generieren. Dieser Vorgang ist in der Tabelle A.2 zusammengefaßt. Als „System“ wird in den folgenden Anwendungsfallbeschreibungen stets die Kombination aus Suchmaschine und Visualisierungsmodul betrachtet.

**Anwendungsfall „Ergebnisdarstellung“**
Eine wichtige Eigenschaft des Systems soll in seiner Erweiterbarkeit bestehen. Daher können im Visualisierungsmodul mehrere Arten des Dokument-Mappings implementiert sein, die vom Benutzer ausgewählt und voreingestellt werden können. Die Mapping-Verfahren unterscheiden sich stark, daher kann in Abhängigkeit davon die Einstellung weiterer Parameter nötig sein.
Die grafische Repräsentation der Dokumente erfolgt mittels sogenannter *Glyphs*. Als Glyphs, oder auch Ikonen, werden geometrische Objekte bezeichnet, deren Erscheinungsbild von bestimmten Parametern oder Zuständen abhängig ist [Bar96a]. Diese können sehr einfach definert sein, z. B. als Quader, Kugeln, o. ä., oder auch als komplexere Formen, die zur Abbildung mehrerer Parameter geeignet sind. Geeignete Attribute zur Abbildung von Parametern sind z. B. Position, Größe, Farbe, Transparenz. Im Anwendungsfall aus Tabelle A.5 wird eine Auswahl aus dem Vorrat des Visualiserungsmoduls getroffen.
Um eine übersichtliche Darstellung des Informationsraumes und der eingeschlossenen Dokumente zu erreichen, sind unter Umständen Hilfsmittel notwendig. Bei einer dreidimensionalen Darstellung der Dokumente in einem kartesischen Koordinatensystem kann z. B. die Darstellung der Koordinatenachsen die Orientierung im Raum wesentlich erleichtern. Der Einsatz der Hilfsmittel ist in starkem Maße von der Auswahl des Mappingverfahrens abhängig.

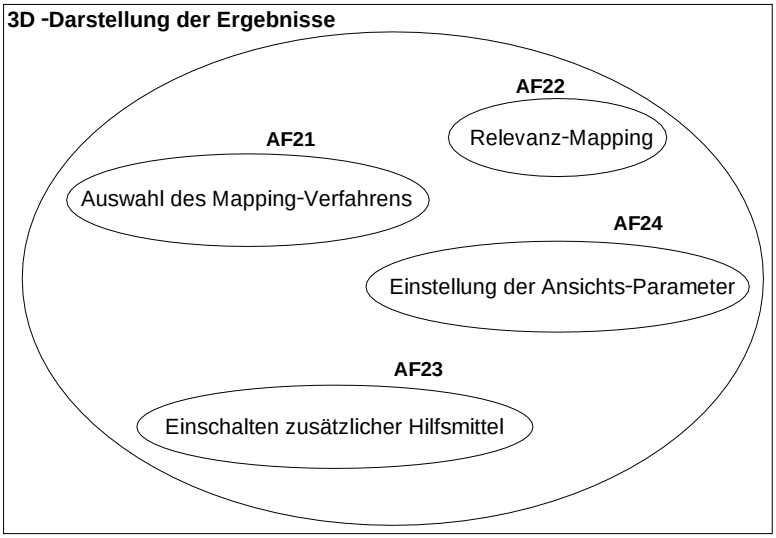



Abbildung 4.9: Anwendungsfall „Ergebnisdarstellung“

Das System muß eine Reihe von Interaktionsmöglichkeiten zur Verfügung stellen, die es erlauben, den Standpunkt der Kamera, den Zoomwinkel und die Blickrichtung der Kamera einzustellen. Auf diese Weise ist es dem Nutzer möglich, im Informationsraum zu navigieren und den für ihn interessanten Bereich zu finden.

**Anwendungsfall „Reduktion der Ergebnismenge“**
Ziel des Zyklusses der Informationssuche mit dem hier zu entwickelnden System ist es, aus der

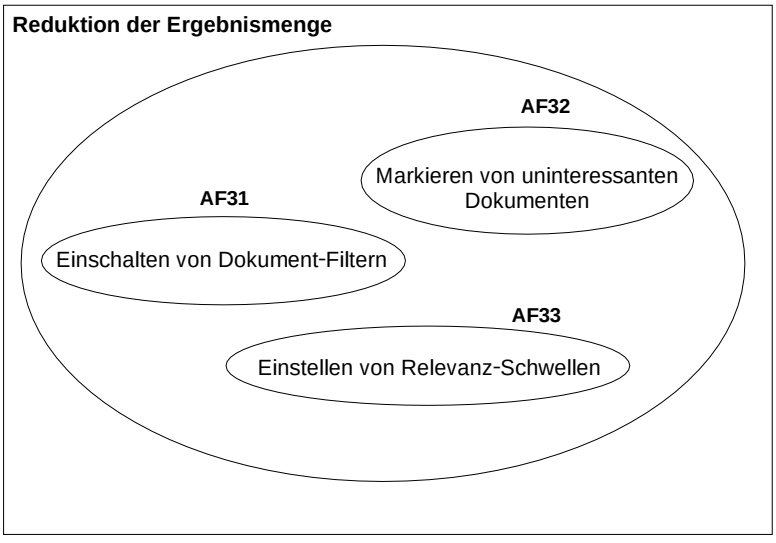



Abbildung 4.10: Anwendungsfall „Dokument-Filter“

von der Suchmaschine gelieferten Dokumentenmenge die „interessanten“ Dokumente zu finden. Daher muß die gegebene Menge durch geeignete Verfahren reduziert wird. Dies geschieht im Anwendungsfall in Tabelle A.8 und Abbildung 4.10. Das System sieht eine Reihe von Filtern vor, die im Abschnitt 5.4.4 beschrieben sind. Jeder Filter kann einzeln zugeschaltet werden, wobei auch Kombinationen von Filtern möglich sind. Die entsprechenden Dialoge gestatten die Versorgung der Filter mit den notwendigen Parametern.

Nach Betrachtung der Detailinformationen zu den Dokumenten kann der Nutzer bestimmte Dokumente als „uninteressant“ klassifizieren und von der weiteren Verarbeitung und Darstellung ausschließen. Dazu stellt das System Interaktionen zur Verfügung, die z. B. durch Mausklick das Markieren oder Löschen gestatten. Dieser Vorgang ist in Tabelle A.10 aufgeführt.
Ein großer Teil der von der Suchmaschine gefundenen Dokumente hat nur einen geringen Bezug zu den Schlüsselworten. Durch die Einführung von Relevanzschwellen kann die Ergebnismenge deutlich reduziert werden. Der Benutzer hat die Möglichkeit, für jedes Schlüsselwort einen Schwellwert zu spezifizieren. Das kann durch die direkte Eingabe von Real-Zahlen oder visuelle Interaktionstechniken, z.B. Slider, erfolgen.

**Anwendungsfall „Auswahl der Ergebnismenge“**
Am Ende des Zyklusses der Informationssuche steht die endgültige Auswahl der interessanten

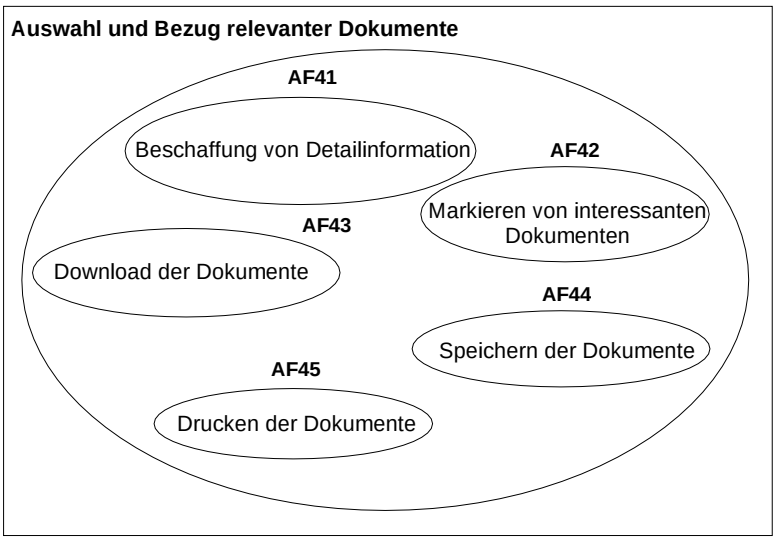



Abbildung 4.11: Anwendungsfall „Dokument-Auswahl“

Dokumente, die im Abbildung 4.11 und der Tabelle A.12 dargestellt ist. Um die entsprechenden Dokumente zu finden, ist es notwendig, detaillierte Informationen zu den Dokumenten zu erhalten. Das Visualisierungstool kann diese Informationen bei Selektion eines Dokumentes z. B. in einem separaten Fenster anzeigen. Voraussetzung dafür ist, daß die Suchmaschine diese Informationen während der Suche aus den Dokumenten extrahiert (z. B. durch die Auswertung von Meta-Tags) und dem Visualisierungstool über die Schnittstelle zur Verfügung stellt. Das System bietet Techniken an, mit denen die Dokumente markiert, geladen, gespeichert oder gedruckt werden können. Diese Schritte sind in den Tabellen A.14 bis A.17 beschrieben.

**Die unscharfe Informationssuche**

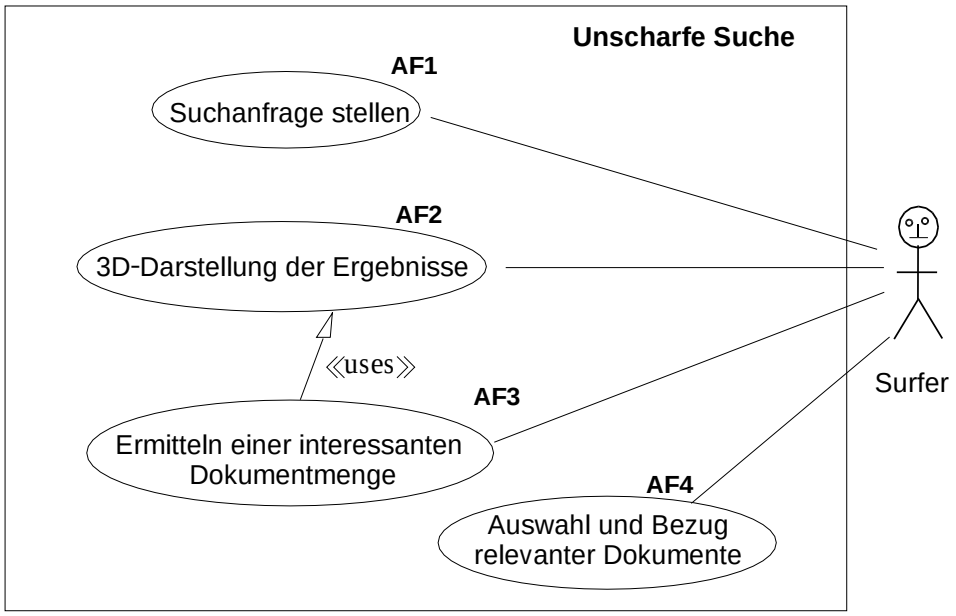



Abbildung 4.12: Unscharfe Suche

Das Szenario einer unscharfen Suche, das am Anfang dieses Kapitels beschrieben ist, kann ebenfalls in vier Anwendungsfälle aufgeteilt werden. Dies sind im Einzelnen:

1. Stellen der Suchanfrage,
2. die Ergebnisdarstellung,
3. das Ermitteln der interessanten Ergebnismenge und
4. die Auswahl der Ergebnismenge.

Sie sind in der Abbildung 4.12 dargestellt und werden im Folgenden genauer beschrieben. Einige Anwendungsfälle werden auch hier einer hierarchischen Gliederung unterzogen, wobei die entsprechenden Anwendungsfalltabellen im Anhang A.2 verzeichnet sind. Verweise auf diese Tabellen befinden sich in der Tabelle 4.2, sowie in den Bildern 4.13, 4.14 und 4.15.

**Anwendungsfall „Stellen der Suchanfrage“**
Das Initiieren der Suche ist in diesem Anwendungsfall identisch zu dem oben beschriebenen Fall der zielgerichteten Suche. Der Benutzer stellt eine Suchanfrage nach herkömmlichem Muster an eine Suchmaschine. Diese ermittelt die Ergebnismenge der Dokumente und berechnet die entsprechenden Relevanzwerte. Diese Daten werden über eine Schnittstelle dem Visualiserungsmodul zur Verfügung gestellt. Zu diesem Vorgang sei auf die Tabellen A.1 und A.2 verwiesen.

**Anwendungsfall „Ergebnisdarstellung“**
Auch dieser Anwendungsfall ist dem zweiten Schritt der zielgerichteten Suche ähnlich. Der Unterschied besteht darin, daß hier der Informationsraum mit einer großen Menge von Dokumenten gefüllt ist, von denen der größte Teil uninteressant ist. Der Nutzer muß eine Untermenge der vorhandenen Dokumente bilden. Zunächst wird wiederum das Mapping-Verfahren eingestellt. Dabei sollte

| **Anwendungsfall** | **elementare Anwendungsfälle** |
|---|---|
| AF1: Suchanfrage stellen | AF11: Initiieren der Suche |
| AF2: Grafische Darstellung der Ergebnisse | AF21: Auswahl des Mapping-Verfahren<br>AF22: Relevanz-Mapping<br>AF23: Einschalten zusätzlicher Hilfsmittel |
| AF3: Ermitteln einer interessanten Dokumentmenge | AF31: Einstellung der Ansichts-Parameter<br>AF32: Relevanz-Schwellen<br>AF33: Betrachten von Detailinformationen<br>AF34: Markieren interessanter und uninteressanter Dokumente |
| AF4: Auswahl und Bezug von Dokumenten | AF41: Download der markierten Dokumente<br>AF42: Speichern der Dokumente<br>AF43: Drucken der Dokumente |

Tabelle 4.2: Relevante Anwendungsfälle für die unscharfe Suche

ein Verfahren gewählt werden, das eine gute räumliche Vorstellung des gegebenen Informationsraumes ermöglicht, da später darin navigiert werden muß. Die Einstellung der verfahrensabhängigen Mapping-Parameter erfolgt ebenfalls in diesem Anwendungsfall, sowie das Zuschalten von Hilfsmitteln zur besseren Orientierung im Informationsraum.

**Anwendungsfall „Ermitteln der interessanten Ergebnismenge“**
Dieser Anwendungsfall ist der wesentliche Teil der unscharfen Suche. Bei der zielgerichteten Suche erhält der Anwender aufgrund seiner gewählten Schlüsselworte a priori eine Menge von für ihn interessanten Dokumenten. Im Falle der unscharfen Suche dagegen muß er diese Menge im Informationsraum zunächst finden und eingrenzen. Dies kann er mit den Mitteln der 3D-Darstellung und entsprechenden Interaktionen tun. Unter Verwendung der im vorigen Anwendungsfall zugeschalteten grafischen Hilfsmittel kann die interessante Region des Informationsraumes lokalisiert werden. Anschließend bewegt sich der Nutzer virtuell in diese Region oder in deren Nähe, um einen Überblick über die darin befindlichen Dokumente zu erhalten. Zu dieser Navigation sind im wesentlichen zwei Schritte notwendig: die Einstellung des Kamerastandpunktes und der Blickrichtung sowie des Öffnungswinkels der Kamera (Zoom). Dazu sind vom User-Interface geeignete Interaktionstechniken vorzusehen. Denkbar ist die direkte Bewegung der Kamera durch den Raum mittels Maus oder Tastatur. Diese Technik ist aus Computerspielen bekannt und hinreichend erprobt. Möglich ist auch die indirekte Einstellung der Ansichtsparameter durch GUI-Elemente wie Buttons, Slider oder die Eingabe von Zahlenwerten.
Die gebildete Untermenge kann nun weiter reduziert werden. Dies kann unter Verwendung der schon beschriebenen Relevanz-Schwellen geschehen. In Abhängigkeit vom Mapping-Verfahren kann der durch diese Schwellen eingegrenzte Informationsraum gegebenenfalls auch grafisch repräsentiert, bzw. hervorgehoben werden.
Die bis hier erläuterten Teilschritte dieses Anwendungsfalles müssen unter Umständen mehrmals wiederholt werden, bis eine Dokumentenmenge gebildet ist, die nur noch Dokumente enthält, die mit hoher Wahrscheinlichkeit interessant sind. Daraus wird nun nochmals eine Auslese getroffen, indem die Detailinformationen betrachtet werden. Dokumente, die sich als irrelevant erweisen, werden entfernt, interessante Dokumente werden markiert.

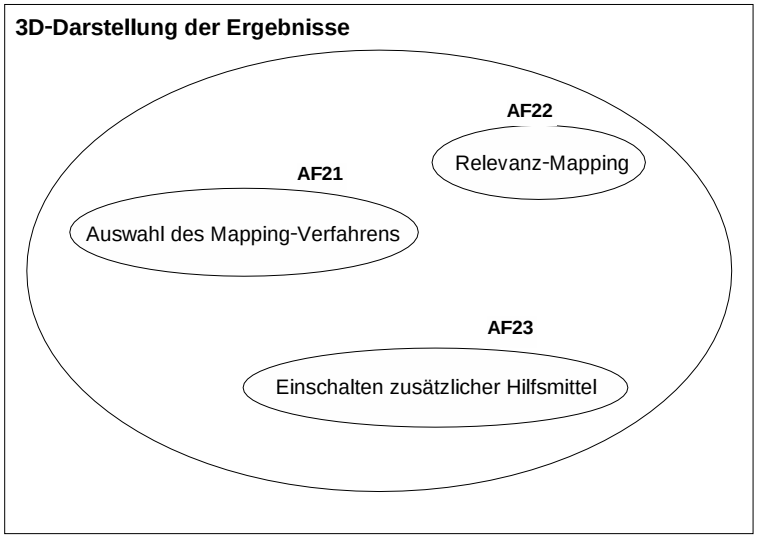



Abbildung 4.13: Anwendungsfall „Ergebnisdarstellung“

**Anwendungsfall „Auswahl der Ergebnismenge“**
Die in diesem Anwendungsfall enthaltenen Szenarien sind bei der Behandlung des gleichnamigen Anwendungsfalles der zielgerichteten Suche (Abschnitt 4.4.2) bereits besprochen worden und sollen hier noch einmal kurz zusammengefaßt werden.
Die zur endgültigen Auswahl der interessanten Dokumente notwendigen Vorgänge sind in der Tabelle A.22 und der Abbildung 4.15 aufgeführt. Eine Auswahl der Dokumente ohne den vorhergehenden Download ist nur möglich, wenn die verwendete Suchmaschine dem Visualisierungsmodul bestimmte Detailinformationen zur Verfügung stellt. Diese Informationen können dann bei Selektion eines Dokuments in der grafischen Darstellung angezeigt werden. Der Benutzer kann dann entscheiden, ob das Dokument für ihn interessant ist und angefordert werden soll, oder ob es eher unwichtig ist und aus der Ergebnismenge entfernt werden kann.

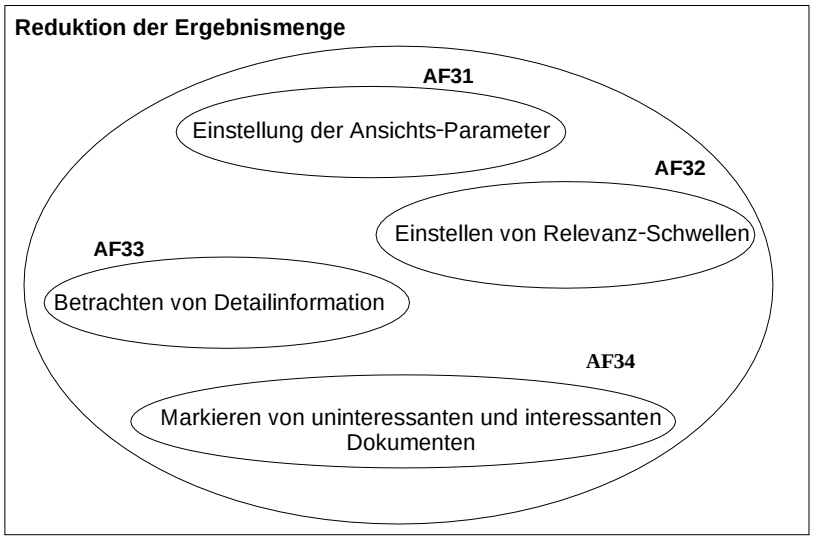



Abbildung 4.14: Anwendungsfall „Dokument-Ermittlung“

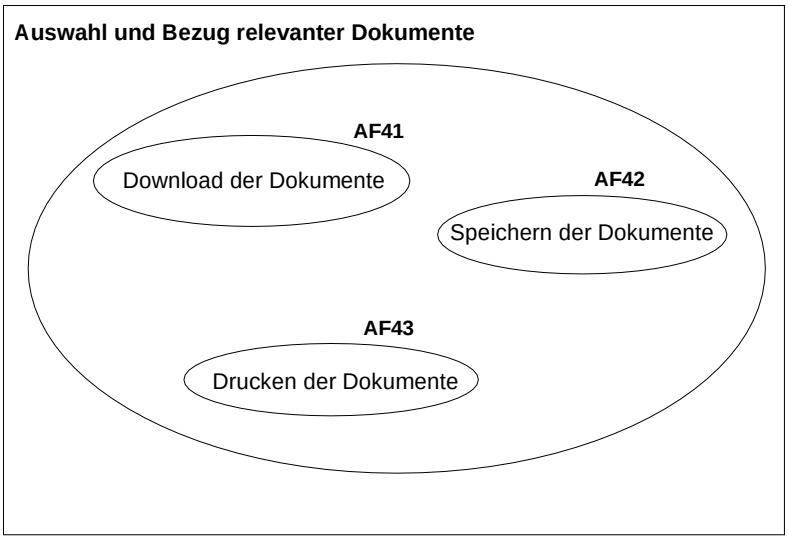



Abbildung 4.15: Anwendungsfall „Dokument-Auswahl“

## 4.5 Fazit

Im Kapitel 3 wurde dargelegt, daß bisher eine Reihe von Visualisierungsverfahren existiert. Bei diesen Verfahren treten Probleme bei der Handhabung großer Dokumentenmengen auf. Daher wurde der in diesem Abschnitt vorgestellte Entwurf so gestaltet, daß die Dokumentenmenge bereits *vor* der grafischen Darstellung reduziert werden kann. Dazu wurden Filtermechanismen und Filtertypen entworfen, die insbesondere auf die Suche nach Dokumenten aus dem wissenschaftlichen Bereich abgestimmt sind. Das System ist so flexibel zu entwerfen, daß sowohl Filter als auch die Methoden der grafischen Darstellung den jeweiligen Bedürfnissen entsprechend kombiniert und erweitert werden können. Mit diesem Ansatz ist es möglich, den in den vergangenen Abschnitten erläuterten Problemen bei der Informationssuche wirksam zu begegnen.
Als Ergebnis der Anwendungsfallanalyse stehen konkrete Aufgaben, die beim Entwurf eines Systems zur Visualisierung von Suchergebnissen zu lösen sind. Es existiert ein breites Spektrum möglicher Anwendungsfälle. Daher sind zwei Extremsituationen behandelt worden. Eine Aufgabe des im nächsten Kapitel folgenden Softwaredesigns ist es, das System so modular zu entwerfen, daß der Benutzer durch Kombination von Komponenten nahezu das gesamte Spektrum der Anwendungsmöglichkeiten abdecken kann.

# Kapitel 5

# Entwurf eines Visualisierungssystems für Informationsräume

Ziel dieses Kapitel ist, das Konzept eines Systems zur Informations-Visualisierung darzustellen. Das grundlegende Ziel eines Sytems zur Informations-Visualisierung ist es, relevante, interessante Informationen aus einer großen Menge zur Verfügung stehender Daten zu selektieren und zu visualisieren. So wird der Benutzer bei der Suche unterstützt und ihm dabei möglichst viel Arbeit und zeitlicher Aufwand abgenommen. Zum Verständnis dieses Visualisierungsprozesses ist es notwendig, den Ablauf der einzelnen Phasen für die Erzeugung einer visuellen Repräsentation von Informationen zu erklären. Aus diesem Grunde ist die Visualisierungs-Pipeline im Kapitel 2.6.3 als Basis für das hier vorgestellte Konzept zu sehen. Die vorgestellte modulare Aufbauweise der Visualisierungs-Pipeline wird adaptiert und angewendet, um Informationsmengen darstellen zu können. Dies ist im Übersichtsbild in der Abbildung 5.1 gezeigt, das auch die im Kapitel 4 beschriebenen Anwendungsfälle widerspiegelt.
Der Prozeß der Informationsfindung beginnt mit dem Stellen einer Suchanfrage an eine Suchmaschine. Als Resultat entsteht eine Liste von Dokumenten, die im Weiteren als Suchergebnis bezeichnet wird. Das zu entwerfende Visualisierungsmodul soll sowohl online (in direkter Verbindung mit einer Suchmaschine) als auch offline (über den Austausch von Dateien) betrieben werden können. Demzufolge muß ein Datenformat entworfen werden, das die Übergabe der Informationen als Parameter an das Visualisierungsmodul gestattet und für den Datei-Austausch in eine persistente Form gebracht werden kann. Der Entwurf dieses Formates wird im Abschnitt 5.9 beschrieben.
An dieser Stelle ist das Information Retrieval abgeschlossen und der Prozeß der Extraktion der wichtigen Informationen beginnt mit der Arbeit des Visualisierungsmodules. Es soll möglich sein, die Sitzung jederzeit zu unterbrechen und zu einem späteren Zeitpunkt fortzusetzen. Aus diesem Grund müssen die wichtigsten Parameter des Systemzustandes gespeichert werden. Die Übernahme der Suchergebnisse und das Speichern und Laden von Zwischenergebnissen sind die Aufgabe der Schnittstelle im oberen Teil des Bildes 5.1.
Alle verfügbaren Daten laufen in einer zentralen Schaltstelle (in der Abbildung als *Control* bezeichnet) zusammen und werden dort in Strukturen aufbereitet, verwaltet und verteilt. Diese Schaltstelle steuert auch alle weiteren Bestandteile des Visualisierungsmoduls und organisiert die Interaktion mit dem Benutzer.
Im rechten unteren Teil der Abbildung 5.1 ist die Visualisierungspipeline mit ihren Stufen Filter, Mapping und Renderer erkennbar. Der Datenfluß von der Filter- zur Mappingstufe verläuft nicht direkt, sondern wird zentral gesteuert. Mappingstufe und Renderer haben dagegen eine direkte und feste Verbindung, die Gründe für diesen Aufbau werden im Abschnitt 5.5 erläutert. Das System soll so gestaltet werden, daß der Benutzer zwischen mehreren Mappingverfahren und Renderern auswählen kann und das System auch um weitere Visualisierungsverfahren erweitert werden kann.

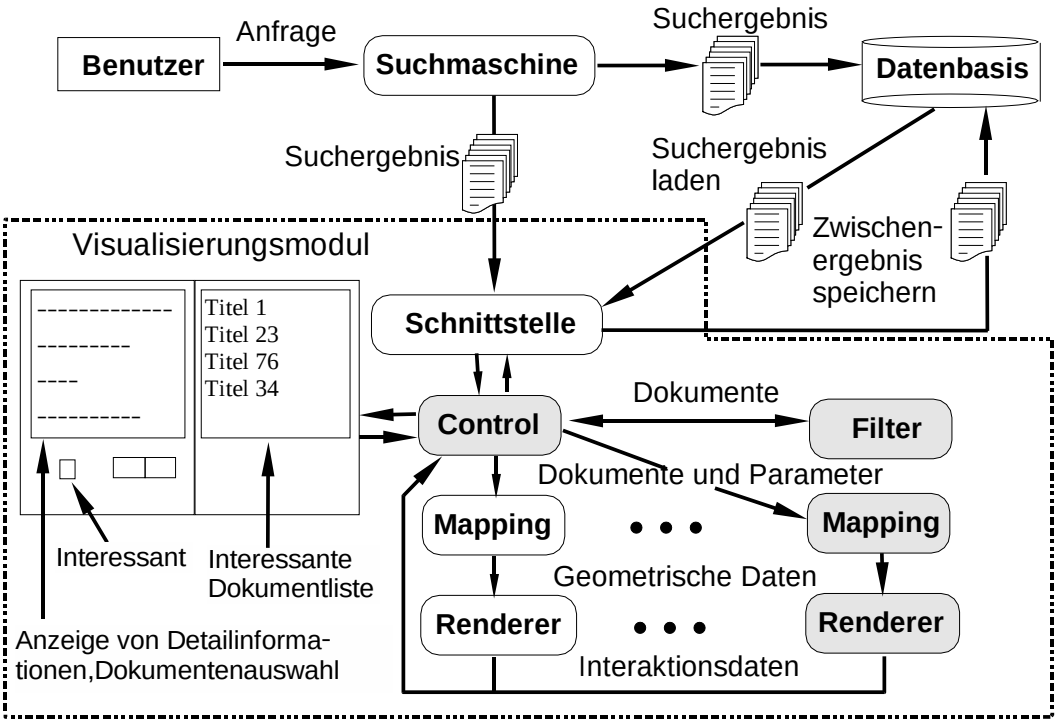



Abbildung 5.1: Das Visualisierungssystem

Dies wird in der Abbildung durch eine weitere Mapping- und Renderstufe verdeutlicht.
Im Renderer hat der Benutzer die Möglichkeit, direkt mit den Dokumenten (bzw. ihren grafischen Darstellungen) zu interagieren. Dazu gehören die Navigation und die Selektion von Dokumenten. Die Resultate dieser Interaktionen werden zentral ausgewertet und die entsprechenden Reaktionen ausgelöst. Bei der Selektion eines Dokuments werden alle verfügbaren Detailinformationen angezeigt. Der Benutzer hat die Möglichkeit, ein Dokument als „interessant“ oder „irrelevant“ einzustufen. Das entsprechende Programmelement befindet sich auf der linken Seite der Abbildung.
Auf diese Weise wird durch wiederholte Anwendung von Filtern, grafischer Darstellung und die Betrachtung detaillierter Informationen die Zahl der Dokumente eingegrenzt. Die Dokumente, die durch den Benutzer als „interessant“ klassifiziert wurden, werden in einer speziellen Datenstruktur gesammelt. Diese Liste von Dokumenten (Abbildung 5.1) ist das Ergebnis des hier beschriebenen Prozesses.
Der Entwurf des oben im Überblick vorgestellten Systems wird in diesem Kapitel mit der UML-Notation beschrieben und ausgeführt. Anknüpfend an die Ausführungen im Kapitel 4 werden zunächst allgemeine Betrachtungen zum objektorientierten Design durchgeführt. Im Abschnitt 3.2.9 wurden vorhandene Visualisierungsmethoden bewertet und derzeit existierende Probleme aufgezeigt. Im Abschnitt 5.2 wird nun ein eigener Ansatz vorgestellt, der dem Benutzer eine übersichtliche Visualisierung von Suchergebnissen ermöglichen soll. Der Entwurf der notwendigen Schnittstelle zwischen Suchmaschine und Visualisierungsmodul wird im Abschnitt 5.9 erläutert. Die danach folgenden Abschnitte legen den objektorientierten Entwurf des Visualisierungsmodules dar.

## 5.1 Objektorientiertes Design (OOD)

Diese Phase schließt sich an die oben beschriebene objektorientierte Analyse an. In der Designphase sind die statischen und die dynamischen Sichten des Systems zu untersuchen [Erl00, Neu98]. Der Entwurf ist zusammen mit der Analyse als iterativer Prozeß zu verstehen. So kann sich z. B. herausstellen, daß Anforderungen nicht vollständig formuliert oder ausreichend verstanden worden sind. Also muß gegebenenfalls die Analyse verbessert werden. Dieser Prozeß hat zur Folge, daß *vor* der Implementierung ein Modell existiert, das den realen Anforderungen möglichst nahe kommt. Die OOA und das OOD stehen zwischen dem Benutzer der Software, bzw. dem Auftraggeber und dem Implementierer. Der Auftraggeber ist kaum in der Lage, seine Anforderungen in einer Form darzulegen, die eine unmittelbare Implementierung erlaubt. Andersherum ist der Implementierer nicht in der Lage, sich genügend tief in verschiedenste Spezialgebiete einzuarbeiten, um dortige komplexe Prozesse und Anforderungen erfassen und in Software umsetzen zu können. Die Analyse- und Designphase dient zur Vermittlung zwischen den beteiligten Seiten. Die Hauptziele des Objektorientierten Designs liegen in erster Linie in der Verbesserung der [Coa94]:

**Produktivität:**
Der Entwurf und die Implementierung sind die kostenintensivsten Teile bei der Softwareentwicklung. Falls erst nach der Inbetriebnahme der Software festgestellt wird, daß Funktionalitäten fehlen oder Anforderungen mißverstanden worden sind, hat das (unter Umständen tiefgreifende) Nachbesserungen am Programmcode zur Folge. Durch gründliche Anwendung von OOA und OOD können solche Kosten eingeschränkt werden.
Eine wohldefinierte und gut dokumentierte Klassenbibliothek kann außerdem in möglichen Folgeprojekten wiederverwendet werden.

**Qualität:**
Durch das Erzeugen von „Klarheit“ hinsichtlich der Funktionalität von Programmelementen während der Designphase ergeben sich Möglichkeiten zur separaten Testung der einzelnen Elemente. Dies ist wesentlich sicherer als der Test des Gesamtsystems, da sich aufgrund der oft hohen Komplexität eine extrem hohe Zahl von möglichen Zuständen ergeben kann.
Ein Bestandteil des Designs ist die Konzipierung der Benutzeroberflache, welche heute entscheidend für die Qualität einer Software ist. Durch Umsetzung der fein strukturierten Anwendungsfälle in der Benutzerführung kann darauf positiv Einfluß genommen werden. Weiterhin wird relativ früh erkannt, wenn sich Teile der vom Auftraggeber formulierten Anforderungen gegenseitig ausschließen.

**Wartbarkeit:**
Eine Software muß in der Regel nach einiger Zeit der Benutzung Veränderungen unterzogen werden. Die Gründe dafür können vielfältig sein, als Beispiel seien hier der technologische Fortschritt und die Veränderung der Anforderungen genannt. Im Design kann dies in gewissem Maße bereits berücksichtigt werden. Dazu muß abgeschätzt werden, welche Funktionalitäten mit hoher Wahrscheinlichkeit in Zukunft geändert werden. Diese sollten möglichst in Programmelementen zusammengefaßt werden, die später ersetzt werden können.

Als Ergebnis der Phase des objektorientierten Designs stehen zwei Modelle, mit denen das zu entwickelnde System mit allen seinen Bestandteilen beschrieben ist [Bal99]:

**Statisches Modell:**
Das statische Modell repräsentiert eine logische Abstraktion des Software-Systems, die auf die statischen Aspekte beschränkt ist. Hier werden die funktionalen Anforderungen analysiert. Dafür werden die Klassen identifiziert und dann beschrieben. Die Vererbungsstrukturen sind festzulegen und die Assoziationen zwischen den Klassen eines Paketes zu berücksichtigen.

**Dynamisches Modell:**
Das dynamische Modell ist die logische Abstraktion des Sofware-Systems, welche die dynamischen Aspekte berücksichtigt, das heißt das Verhalten von Objekten. Eine Identifikation von Interaktionen, Ereignissen und Aktivitäten und die Definition von Zuständen soll für die Erstellung des dynamischen Modell durchgeführt werden. Dafür wird vom statischem Modell ausgegangen.

## 5.2 Entwurf eines Mappingverfahrens

### 5.2.1 Die grafische Darstellung der Dokumente

Das Mapping der gefilterten Daten (Informationen) in einen virtuellen Informationsraum kann wesentlich komplizierter sein als bei der wissenschaftlichen Visualisierung (Siehe Punkt 2.6). Im technischen Bereich wird die grafische Darstellung von einem physikalischen Sachverhalt abgeleitet. Bei der Visualisierung von Informationen haben die darzustellenden Elemente typischerweise keine physikalischen Eigenschaften, die die Grundlage für die Visualisierung bilden könnten.
Die Elemente haben nur semantische Eigenschaften. Weiterhin hängt die Informationsrepräsentation vom Datenmodell und von spezifischen Benutzeranforderungen ab. Die im Visualisierungssystem existierenden Dokumente besitzen nur eine mathematisch erfaßbare Größe, die zur Abbildung herangezogen werden kann - die Relevanz bezüglich der Schlüsselwörter. Ein anderes Problem besteht darin, daß Informationsräume multidimensional sind. Das bedeutet, sie besitzen abstrakte Objekte mit oft mehr als drei Objektdimensionen. Da maximal drei der Dimensionen auf die Raumachsen abgebildet werden können, müssen zusätzlich grafische Attribute (Form, Farbe, usw.) benutzt werden.

### 5.2.2 Mapping mit Glyphs

Als *Glyphs* [Bar96a], oder auch Ikonen, werden geometrische Objekte bezeichnet, deren Erscheinungsbild von bestimmten Parametern oder Zuständen abhängig ist. Parameter können auf die Eigenschaften von Glyphs abgebildet werden. Derartige Objekte werden in der wissenschaftlich-technischen Visualisierung erfolgreich angewendet, was besonders in der hohen Flexibilität dieser Methode begründet ist. Glyphs können einen einzigen skalaren Parameter in einem Knotengitter darstellen, sie können aber auch komplexe Formen annehmen und zur Visualisierung von Tensoren oder Multiparameterfeldern (wie sie z. B. bei Strömungssimulationen anfallen) eingesetzt werden.
Die Anzahl der Informationen, die der Mensch in der Praxis einem Glyph visuell entnehmen kann, ist allerdings begrenzt. Das gilt besonders dann, wenn die Objekte dicht im Raum angeordnet sind. Die Anzahl der abzubildenden Parameter sollte nicht auf das theoretische Maximum gebracht werden. Diese Gefahr besteht jedoch bei der Visualisierung von Suchergebnissen eher nicht, da in der Praxis die Anzahl der bei einer Suche verwendeten Schlüsselwörter begrenzt ist. Suchergebnisse, die auf dem Einsatz von zwei bis fünf Schlüsselwörtern basieren, lassen sich mit dieser Methode problemlos visualisieren, wie die folgenden Ausführungen zeigen werden.

#### Abbildung von Parametern auf die Position

Grundlage dieser Form der Abbildung ist ein rechtwinkliges, kartesisches Koordinatensystem. Prinzipiell wird jedem Schlüsselwort eine Achse zugeordnet. Jeder Glyph wird initial mit der Position $P(x, y, z) = (0.0, 0.0, 0.0)$ erzeugt. Die abzubildenden Relevanzen sind zu diesem Zeitpunkt bereits normiert. Das System verfügt über die Information, welche zu einem Suchbegriff gehörende Relevanz $R$ auf welche Raumachse abzubilden ist. Die Abbildung selbst erfolgt über eine Translation des

Glyphs. Hier müssen - je nach Anzahl der abzubildenden Schlüsselworte - drei Fälle unterschieden werden:

1. Ein Schlüsselwort:
   Es wird ein Relevanz-Parameter auf eine Raumachse abgebildet, so daß der Glyph auf einer Achse verschoben wird. Die neue Position ergibt sich aus:

$$[P'] = \begin{bmatrix} x_0 + R_1 \\ y_0 \\ z_0 \end{bmatrix} \tag{5.1}$$

   Diese Gleichung beschreibt ein Mapping auf die X-Achse, das natürlich auch auf die anderen Achsen anwendbar ist.

2. Zwei Schlüsselworte:
   Das Abbildungsverfahren des ersten Falles wird um eine Raumdimension erweitert, die neue Position des Glyphs ergibt sich aus:

$$[P'] = \begin{bmatrix} x_0 + R_1 \\ y_0 + R_2 \\ z_0 \end{bmatrix} \tag{5.2}$$

   Das Erscheinungsbild ist das eines gebräuchlichen X-Y-Diagramms.

3. Drei Schlüsselworte:
   Es werden alle drei Raumachsen für die Abbildung der Relevanzen verwendet. Die neue Position des Glyphs wird berechnet durch:

$$[P'] = \begin{bmatrix} x_0 + R_1 \\ y_0 + R_2 \\ z_0 + R_3 \end{bmatrix} \tag{5.3}$$

**Abbildung von Parametern auf die Größe**

Häufig ist es sinnvoll, einen Parameter auf die Größe der grafischen Erscheinungsform der Dokumente abzubilden. Dieses Mapping ist stark von der Art der ausgewählten Repräsentationsform abhängig. Für drei geometrische Formen soll dieses Verfahren hier beschrieben werden.

**Bei Verwendung eines Kreises / einer Kugel:**
Die Kugel hat einen geometrischen Parameter, der zur Abbildung einer Größe herangezogen werden kann - den Radius $r$. Jede Kugel benötigt einen Initial-Radius $r_i > 0.0$. Ein Radius größer Null ist notwendig, weil zum gleichen Zeitpunkt andere Relevanzen auf die Position der Kugel abgebildet sein können und diese daher in jedem Fall sichtbar sein muß. Es wird ebenfalls ein Maximal-Radius $r_{max}$ benötigt, der verhindert, daß viele Dokumente durch ein „großes“ verdeckt werden können. Die Größe des Minimal- und Maximal-Radius kann im System festgelegt werden. Der Radius nach der Relevanzabbildung wird durch die Gleichung

$$r' = r_0 * \frac{(r_{max} - r_0)}{r_{max}} * R \tag{5.4}$$

berechnet. Diese Berechnung ist auch für den zweidimensionalen Fall - den Kreis - gültig.

**Bei Verwendung eines Viereckes / eines Quaders:**

Ein Quader hat drei geometrische Größen, die zur Abbildung von Relevanzen dienen können: die Breite $b$, die Höhe $h$ und die Tiefe $t$. Wie bereits bei der Kugel beschrieben ist es auch hier sinnvoll, für die Dimensionen $D(b,h,t)$ des Quaders die Minima $b_0, h_0$ und $t_0$ sowie die Maxima $b_{max}, h_{max}$ und $t_{max}$ zu spezifizieren. Die Berechnungsvorschriften für die neuen Dimensionen lauten:

$$D' = \begin{bmatrix} b' \\ h' \\ t' \end{bmatrix} = \begin{bmatrix} b_0 * \frac{b_{max}-b_0}{b_{max}} * R_1 \\ h_0 * \frac{h_{max}-h_0}{h_{max}} * R_2 \\ t_0 * \frac{t_{max}-t_0}{t_{max}} * R_3 \end{bmatrix} \tag{5.5}$$

Beim zweidimensionalen Anwendungsfall entfällt die Berechnung der Tiefe $t'$.

Bei dieser Form der Abbildung ist allerdings zu bedenken, daß die Dimensionen der Quader in stark belegten Regionen des Informationsraumes optisch nur schwer zu differenzieren sind. Daher ist es einfacher, nur eine Relevanz auf einen Würfel abzubilden:

$$D' = \begin{bmatrix} b' \\ h' \\ t' \end{bmatrix} = \begin{bmatrix} b_0 * \frac{b_{max}-b_0}{b_{max}} * R \\ b_0 * \frac{b_{max}-b_0}{b_{max}} * R \\ b_0 * \frac{b_{max}-b_0}{b_{max}} * R \end{bmatrix} \tag{5.6}$$

**Bei Verwendung eines Dreieckes:**

Aus Gründen der Übersichtlichkeit wird von einem gleichseitigen Dreieck ausgegangen. Die geometrische Größe, die sich zur Abbildung eines Parameters eignet, ist die Kantenlänge $l$. Die Abbildung erfolgt analog zum Kreis, es wird eine initiale Länge $l_0$ und eine maximale Kantenlänge $l_{max}$ benötigt. Die Berechnung der resultierenden Kantenlänge unter Einbeziehung der Relevanz $R$ des dazugehörigen Dokuments erfolgt nach folgender Gleichung:

$$l' = l_0 * \frac{(l_{max} - l_0)}{l_{max}} * R \tag{5.7}$$

### Abbildung auf die Farbe

Ein wichtiges Mittel zur Visualisierung von Parametern stellt die Farbe dar. Ein Parameter kann die Farbe eines Objekts beeinflussen, mit der das Objekt wahrgenommen wird. Durch diesen Effekt wird dem roten Objekt mehr Bedeutung als dem grünen Objekt zugeordnet. Um die Farbe hier einsetzen zu können, muß jeder Farbe ein eindeutiger Wert zugeordnet werden, um die Farben klar identifizieren und voneinander unterscheiden zu können.

In der Computergrafik finden mehrere Farbmodelle Verwendung. Das bekannteste ist das physikalisch begründete RGB-Modell (**R**ot – **G**rün – **B**lau), das die additive Farbmischung der Monitortechnologie berücksichtigt. Der resultierende Farbraum kann als Würfel veranschaulicht werden (siehe Abbildung 5.2a). Üblicherweise wird in der wissenschaftlich-technischen Visualisierung bei den sogenannten „Falschfarbendarstellungen“ das Farbspektrum verwendet (von Blau nach Rot). Die Abbildung eines Parameters auf ein Spektrum ist im RGB-Farbraum nicht ohne Weiteres möglich. Wesentlich besser geeignet ist das benutzerorientierte HSV-Modell (**H**ue – Farbe, **S**aturation – Sättigung, **V**alue – Helligkeit). Der entsprechende Farbraum kann als Kegel, oder vereinfacht pyramidenförmig veranschaulicht werden (Abbildung 5.2b). Das Basissechseck dieser Pyramide ist das Ergebnis der Projektion des RGB-Einheitswürfels entlang seiner Diagonalen von Weiß (1,1,1) nach Schwarz (0,0,0). Die Farbe wird als Winkel $H$ in diesem Sechseck angegeben, beginnend mit Rot (Winkel 0 Grad). Die Sättigung $S$ ist der Abstand von der V-Achse und beschreibt den Grauanteil der Farbe, d.h. je weiter man sich von dieser senkrechten Achse entfernt, desto kräftiger wird die Farbe. Daraus ergibt sich auch, daß auf dieser V-Achse die einzelnen Grauwerte liegen. Der Wert $V$ beschreibt die Helligkeit einer Farbe (bzw. eines Grauwertes). Bei diesem Modell liegt das gesamte

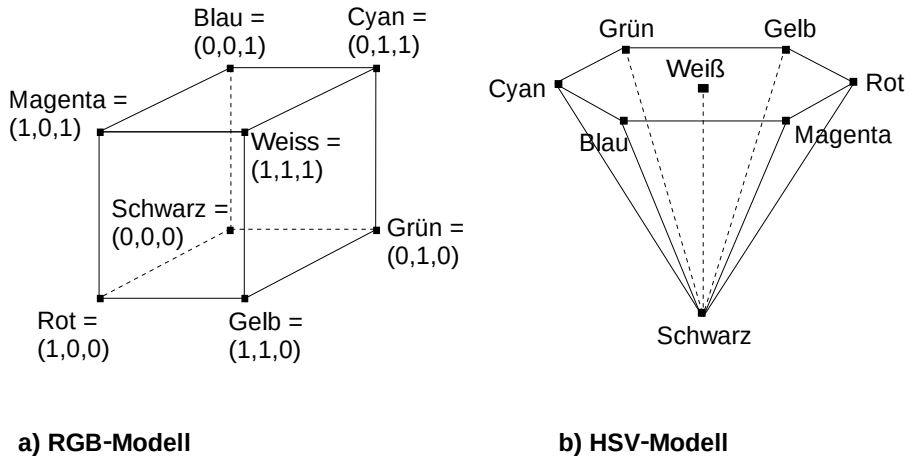



Abbildung 5.2: Farbmodelle

Farbspektrum auf einem Kreis. Damit kann ein gegebener Parameter direkt auf einen Winkel $\alpha$ im Bereich $[0.0 \leq \alpha \leq= 2\pi]$ abgebildet werden. Beide Modelle können ineinander umgerechnet werden. Auf die entsprechenden Berechnungsvorschriften soll an dieser Stelle nicht eingegangen werden. Sie sind in den einschlägigen Werken zur Computergrafik ([Fra97, Sow98, Enc88, Fol90]) dargelegt und alle modernen Grafiksysteme können mit beiden Modellen arbeiten. Mit der Gleichung

$$H = R * 2\pi \tag{5.8}$$

kann der Farbwert zu einer gegebenen Relevanz bestimmt werden.

#### Abbildung auf die Transparenz

Das Abbilden einer Relevanz auf die Transparenz der Dokumentrepräsentation ist besonders bei Informationsräumen mit hoher Dokumentendichte von Bedeutung. Dabei sollte so vorgegangen werden, daß eine niedrige Relevanz eine hohe "Durchsichtigkeit" der entsprechenden grafischen Objekte bedeutet. Dadurch wird erreicht, daß Dokumente, die eventuell verdeckt sind, aufgrund der Transparenz anderer Dokumente dennoch sichtbar bleiben.
Alle modernen Grafikschnittstellen (siehe auch Kapitel 6.1) bieten die Möglichkeit, bei einer Farbdefinition zusätzlich zu den Farbanteilen (Rot, Grün, Blau) einen Wert für die Transparenz, den sogenannten Alpha-Wert mit anzugeben. Unter der Annahme, daß $\alpha = 0.0$ vollständige Transparenz und $\alpha = 1.0$ vollständige Opazität bedeutet, ergibt sich die „Durchsichtigkeit" eines Glyphs wie folgt:

$$\alpha = R$$

Dabei ist $R$ die Relevanz eines Dokuments, die per Definition im Intervall $[0.0 \leq R \leq 1.0]$ liegt. Diese Art der Abbildung ist für Informationsräume mit hoher Dokumentendichte nicht geeignet, da das menschliche Auge verschiedene Abstufungen der Transparenz (anders als Farbabstufungen) schlecht auflösen kann. Die Unterscheidung von sich gegenseitig verdeckenden, verschiedenfarbigen Objekten ist schwierig.

### 5.2.3 Zusammenfassung

Die hier vorgestellten Verfahren der Abbildung von Relevanz-Parametern auf Glyphs können entsprechend der Anzahl der verwendeten Schlüsselwörter und Preferenzen des Benutzers beliebig miteinander kombiniert werden. Ein Beispiel dafür ist in der Abbildung 5.3 gezeigt. Darin wird

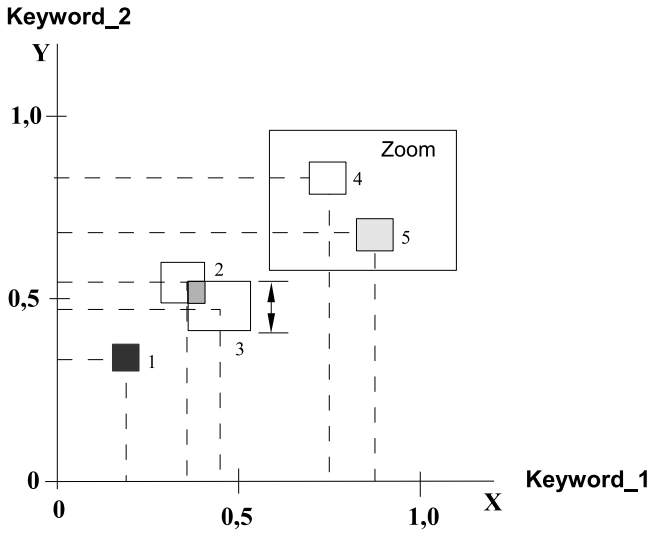



Abbildung 5.3: Beispiel für das Relevanz-Mapping

ein aus fünf Schlüsselwörtern resultierendes Suchergebnis grafisch dargestellt. Die Relevanzen der Suchbegriffe werden wie folgt abgebildet:

keyword_1: auf die Position bezüglich der X-Achse
keyword_2: auf die Position bezüglich der Y-Achse
keyword_3: auf die Größe der Glyphs
keyword_4: auf die Farbe der Glyphs
keyword_5: auf die Transparenz der Glyphs

Die Glyphs repräsentieren die folgenden Relevanzwerte (in Reihenfolge der oben aufgelisteten Schlüsselwörter):

| Grafisches Objekt | Rlv. 1 | Rlv. 2 | Rlv. 3 | Rlv. 4 | Rlv. 5 |
|---|---|---|---|---|---|
| Glyph Nr.1 | 0.2 | 0.3 | 0.2 | 0.6 | 1.0 |
| Glyph Nr.2 | 0.35 | 0.55 | 0.7 | 0.5 | 0.2 |
| Glyph Nr.3 | 0.45 | 0.48 | 1.0 | 0.6 | 0.2 |
| Glyph Nr.4 | 0.75 | 0.8 | 0.5 | 0.4 | 0.1 |
| Glyph Nr.5 | 0.85 | 0.7 | 0.5 | 0.5 | 0.6 |

Wie in der Abbildung 5.3 ersichtlich wird, sind nicht alle Mappings gleich gut ablesbar. Die beste Erkennbarkeit weisen die Mappings auf die Position und Größe des Glyphs auf. Für den Benutzer sind jedoch auch nicht alle verwendeten Suchbegriffe gleich wichtig. Daher sollte bei der Anwendung des Mappings auf Glyphs sichergestellt sein, daß der Benutzer sein Werkzeug so konfigurieren kann, daß die wichtigsten Suchbegriffe auf die am besten erkennbaren grafischen Attribute ( die geometrischen Attribute) abgebildet werden. Die Abbildung auf die Position hat weiterhin den Vorteil, daß die Dokumentenmenge auf einfache Weise reduziert werden kann – durch „Zoomen“ in den interessanten Teil des Informationsraumes. Dies kann sogar grafisch unterstützt werden, wie es durch das Rechteck in der Abbildung 5.3 angedeutet wird. Aus diesen Gründen bietet es sich an, dieses Verfahren auch im dreidimensionalen Raum zu nutzen, da so drei Relevanz-Parameter auf die Position der Glyphs abgebildet werden können. Durch die dritte Dimension lassen sich auch wesentlich mehr Objekte übersichtlich darstellen und durch interaktives „Begehen“ des Raumes untersuchen.

Mit den hier beschriebenen Abbildungsverfahren lassen sich bei Verwendung einer dreidimensionalen Darstellung sechs Relevanz-Werte gleichzeitig auf einen Glyph abbilden. Diese Zahl läßt sich theoretisch noch weiter erhöhen (z. B. indem verschiedene Formen für die Glyphs verwendet werden). Wie bereits gesagt, strapazieren solche Darstellungen jedoch die Auffassungsgabe des Benutzers in unzumutbarer Weise. Praktisch gesehen haben diese Betrachtungen eher geringe Relevanz, da die Mehrheit der Suchprozesse mit weniger als fünf Suchbegriffen durchgeführt wird.
Zusammenfassend kann gesagt werden, daß dieses Verfahren flexibel bezüglich der verwendeten Suchbegriffe und Darstellungsmöglichkeiten einsetzbar ist. Weiterhin hat es gegenüber vielen der im Kapitel 3 vorgestellten Mappingverfahren den Vorteil, daß es mit geläufigen Darstellungsformen (Koordinatensystemen) arbeitet und somit geringe Anforderungen an das räumliche Vorstellungsvermögen des Benutzers stellt und eine einfache Orientierung ermöglicht.

## 5.3 Die Architektur des Visualisierungsmodules

In diesem Abschnitt werden die bei der Analyse des Systemverhaltens gewonnenen Erkenntnisse umgesetzt. Zunächst werden die Strukturen der Anwendungsfälle auf Klassen und Komponenten abgebildet und diese zueinander in Beziehungen gesetzt.
Die Abbildung 5.4 zeigt einen Überblick der Architektur in UML-Notation. Im Zentrum befindet

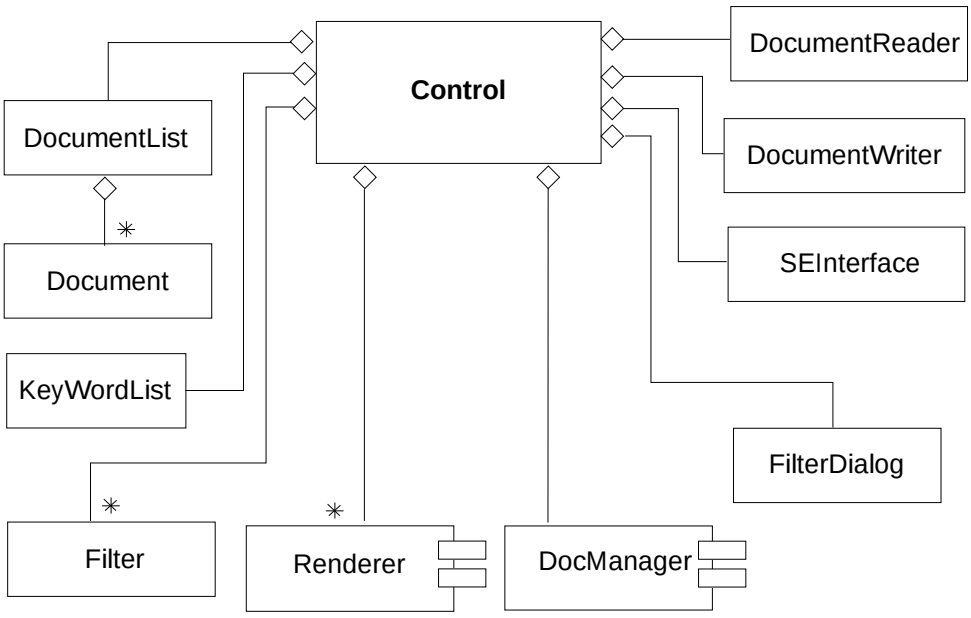



Abbildung 5.4: Architektur des Visualisierungsmodules

sich die Klasse *Control*, die für die Steuerung des Systems verantwortlich ist. Das System benötigt eine Reihe von internen Datenstrukturen. Diese werden von den Klassen *Document, DocumentList* und *KeyWordList* zur Verfügung gestellt. Die Versorgung des Visualisierungsmodules mit Daten (Suchergebnissen) sowie das Speichern und Laden von Arbeitsständen ist die Aufgabe der Klassen *DocumentReader, DocumentWriter* und *SEInterface*. Eine zentrale Aufgabe des Moduls ist das Herausfiltern von Dokumenten, die nicht genügend den Interessen des Benutzers entsprechen. Dazu ist die Einstellung einer Reihe von Optionen und Parametern notwendig, die in der Klasse *FilterDialog* implementiert werden. Die Filterung selbst erfolgt durch die Klasse *Filter*. Im Visualisierungsmodul sind im Allgemeinen mehrere Filter-Klassen implementiert, die ein gemeinsames Interface verwenden. Für die eigentliche Arbeit des Benutzers mit dem System wird die Kompo-

nente *DocManager* entworfen, die eine eigene Benutzeroberfläche aufbaut. Diese Komponente wird im Abschnitt 5.6 ausführlich beschrieben.
Die wichtigste Aufgabe des Visualisierungsmoduls ist die grafische Darstellung und die Sicherstellung der Interaktion. Die dazu notwendige Funktionalität ist in der Komponente *Renderer* gekapselt, die im Abschnitt 5.5 erläutert wird. Das Modul kann mehrere Komponenten dieses Types enthalten, die alternativ verwendet werden können.

### 5.3.1 Die Elemente des Visualisierungsmoduls

Die im Entwurf des Visualisierungsmoduls definierten Elemente sollen kurz beschrieben und ihre Beziehungen zueinander erläutert werden. Auf einige Elemente von hoher Wichtigkeit wird in weiteren Abschnitten noch detailliert eingegangen.

**Control**
Diese Klasse stellt insgesamt die Lauffähigkeit des Visualisierungsmodules sicher und initialisiert und steuert alle anderen Klassen und Komponenten. Die Aufgaben dieser Klasse sind:

- Aufbau und Verwaltung der internen Datenstrukturen
- Initialisierung und Aufbau der grafischen Benutzeroberfläche und Bereitstellung der erforderlichen Elemente
- die Behandlung von Ereignissen.

Die Benutzeroberfläche, die von dieser Klasse bereitgestellt wird, bezieht sich ausschließlich auf die Funktionalität dieser Klasse. Die Interaktionen bezüglich grafischer Darstellungen und Dokumentenauswahl werden von den entsprechenden Klassen bzw. Dokumenten sichergestellt.

**Document**
Dies ist die Klasse, die ein Dokument (wie es von einer Suchmaschine ermittelt wird) als Datensatz systemintern repräsentiert. Eine detaillierte Beschreibung erfolgt im Abschnitt 5.4.1.

**DocumentList**
Diese Klasse stellt eine grundlegende Datenstruktur für das Visualisierungssystem bereit. Sie besteht aus einer Menge von Dokumenten sowie einigen zusätzlichen Informationen (siehe Abschnitt 5.4.2).

**KeyWordList**
In dieser Datenstruktur sind die Suchbegriffe, mit denen die Dokumentenmenge ermittelt wurde, enthalten.

**SEInterface**
Im Falle der Einbindung des Visualisierungsmodules in eine Suchmaschine erfolgt durch diese Klasse die Übernahme des Ergebnisses des Suchprozesses. Das Ergebnis muß von der Suchmaschine in der im Abschnitt 5.9 beschriebenen XML-Datenstruktur bereitstehen und als Parameter an das Modul übergeben werden. Die Klasse *SEInterface* liest die Struktur und stellt die entsprechenden Daten für die *DocumentList* und die *KeyWordList* bereit.

**DocumentWriter**
Eine der gestellten Anforderungen ist die Möglichkeit, während der Evaluierung eines Suchergebnisses einen Zwischenstand zu speichern und die Arbeit zu einem späteren Zeitpunkt fortsetzen zu können. Die Klasse *DocumentWriter* sichert die aktuelle Dokumentenliste sowie

alle Informationen, die zur Restaurierung des aktuellen Bearbeitungsstandes notwendig sind. Die Sicherung erfolgt ebenfalls in einer XML-Struktur.

**DocumentReader**
Das Visualisierungsmodul soll sowohl direkt an eine Suchmaschine koppelbar sein, als auch offline Suchergebnisse einlesen können. Für das Einlesen von gespeicherten Suchergebnissen wird die Klasse *DocumentReader* entworfen. Analog zur Klasse *SEInterface* verwendet sie die XML-Ergebnis-Struktur und stellt die Parameter für die internen Datenstrukturen bereit. Sie ist ebenso in der Lage, die gespeicherten Zwischenergebnisse zu lesen und die zur Restaurierung des Arbeitsstandes notwendigen Informationen bereitzustellen.

**Filter**
Filter dienen dazu, Dokumente von der weiteren Verarbeitung, insbesondere der grafischen Anzeige, auszuschließen. Die verschiedenen Arten der Filter werden im Abschnitt 5.4.4 beschrieben. Der Entwurf sieht das Vorhandensein mehrerer Filter-Klassen vor, die sequentiell arbeiten. Ein Filter wird von der Klasse *Control* getriggert und mit den Optionen der Klasse *FilterDialog* gespeist. Er führt Operationen mit den Elementen vom Typ *Document* durch, die Bestandteil der Klasse *DocumentList* sind.

**FilterDialog**
Diese Klasse stellt einen Dialog bereit, mit dem die zu einem Zeitpunkt auszuführenden Filteroperationen definiert werden können. Folgende Parameter können eingestellt werden:

- das Zu- und Abschalten jedes einzelnen Filters,
- Parameter für die Filter (z. B. Daten, Namen, etc.),
- die Negation von Filtern (nicht bei allen Typen).

Die zu den einzelnen Filtertypen verfügbaren Parameter werden im Abschnitt 5.4.4 beschrieben. Die Parameter werden über die Klasse *Control* den Elementen vom Typ *Filter* zur Verfügung gestellt.

**Renderer**
Diese Komponente enthält die Funktionalität der grafischen Darstellung von Suchergebnissen. Dazu gehört:

- ein Mapping-Verfahren zur Abbildung des abstrakten Informationsraumes auf grafische Elemente mit maximal drei Dimensionen (siehe Kapitel 2),
- die Initialisierung eines Grafiksystems,
- die Bereitstellung eines Ausgabe- und Interaktionsmediums (i. A. ein Fenster),
- der Aufbau einer eigenen Benutzeroberfläche,
- eine eigene Ereignisbehandlung,
- die ordnungsgemäße Terminierung und Freigabe der benötigten Ressourcen.

Dem Design der Renderer-Komponente ist der Abschnitt 5.5 gewidmet. Die Komponente wird von der Klasse *Control* mit den erforderlichen Daten (*DocumentList, KeyWordList*) gespeist.

**DocManager**
Diese Komponente dient dem Auffinden und der Auswahl der interessanten Dokumente durch den Benutzer. Sie erfüllt folgende Teilaufgaben:

- Aufbau einer eigenen Benutzeroberfläche,

- Aufbau und Verwaltung eigener Datenstrukturen,
- Anzeige von allgemeinen Informationen über das Suchergebnis,
- Auswahl, Starten und Stoppen der zu verwendenden Renderer-Komponente,
- Anzeige von Informationen zu ausgewählten Dokumenten,
- Markieren von interessanten Dokumenten,
- Löschen von uninteressanten Dokumenten,
- Download von Dokumenten.

Die Komponente *DocManager* wird von der Klasse *Control* initialisiert und mit den erforderlichen Daten versorgt. Sie kommuniziert mit der Klasse *Control* und kann die Initialisierung oder Aktualisierung anderer Klassen oder Komponenenten anfordern. Die Einzelheiten zum Entwurf dieser Komponente sind im Abschnitt 5.6 beschrieben.

### 5.3.2 Die Benutzeroberfläche des Visualisierungsmoduls

Zwei der Anforderungen, die an das zu entwickelnde System gestellt wurden, sind Erweiterbarkeit und Systemunabhängigkeit. Daher wird die Zusammenstellung der Benutzeroberfläche auf mehrere Elemente des Entwurfes verteilt. Grundsätzlich ist ein Menü vorhanden, das die Funktionalitäten der Klasse *Control* abbildet. Die grafische Struktur ist in der Abbildung B.1 im Anhang B dargestellt.
Die Menüeinträge lösen die folgenden Aktionen aus:

| | |
|---|---|
| File/Load | Laden von Suchergebnissen oder gespeicherten Zwischenergebnissen in Form einer XML-Datei |
| File/Save | Sichern des aktuellen Arbeitsstandes |
| File/Exit | Beenden des Programmes |
| Filter/Redundancy | Ein-/Ausschalten des Redundanz-Filters |
| Filter/Thresholds | Aufruf des Dialoges zur Einstellung der Relevanz-Schwellwerte |
| Filter/Options | Aufruf des Dialoges der Filter-Einstellungen |
| Extra/Browser | Einstellung des Pfades und der Parameter des zur Anzeige von Dokumenten zu verwendenden Browsers |
| Help/Index | Aufruf der Hilfe |
| Help/About | Anzeige von Programm-Informationen |

Ein Teil der Benutzeroberfläche wird von der Komponente *DocManager* generiert, dadurch ist die Rücksichtnahme auf verschiedene Systemumgebungen und unterschiedliche Verhaltensweisen in Browsern gegeben (siehe Abschnitt 5.6). Die Benutzeroberfläche dieser Komponente kann sowohl in ein schon vorhandenes Fenster eingebunden, als in einem externen Fenster verwendet werden.
Aus Gründen der Erweiterbarkeit bringt auch die Komponente *Renderer* ihre eigene Benutzeroberfläche mit (siehe Abschnitt 5.5). Ein Renderer wird grundsätzlich in einem eigenen Fenster mit eigenen Ressourcen initialisiert.

## 5.4 Klassenbeschreibungen

In diesem Abschnitt werden die Definitionen der wichtigsten Klassen des Systemes besprochen. Weitere Klassenbeschreibungen befinden sich im Anhang C

### 5.4.1 Die Dokument-Klasse

Innerhalb des Systems zur Informations-Visualisierung wird unter dem Begriff *Dokument* ein Datensatz verstanden, der folgende Informationen beinhaltet:

**URL:** Ort des real im Internet oder in einer Datenbank existierenden Dokuments.

**Titel:** Titel, unter dem das Dokument verzeichnet ist, diese Information muß ebenfalls verfügbar sein.

**Autor:** Name des Autors des Dokuments sollte verfügbar sein, es ist jedoch nicht zwingend erforderlich.

**Datum:** Erscheinungsdatum des Dokuments kann verarbeitet werden.

**Größe:** Die Angabe der Größe des Dokuments (z.B. in Bytes) ist optional.

**Relevanzen:** Eine Relevanz ist das von der Suchmaschine berechnete Maß für die „Wichtigkeit“ des gefundenen Dokuments hinsichtlich eines Suchbegriffs. Die Relevanzwerte müssen zwingend vorhanden sein, um eine grafische Darstellung zu ermöglichen.

Alle genannten Informationen werden von der Suchmaschine ermittelt und über die Klassen *SEInterface* und *DocumentReader* (siehe Seite 94) an die einzelnen Dokument-Objekte übermittelt.
Dies führt zum Entwurf der Klasse *Document*, die im Anhang C beschrieben ist. Die set- und get-Methoden zum Eingeben und Abfragen der obengenannten Informationen sind selbsterklärend. Von besonderer Bedeutung sind die Methoden *setValid(pValid)* und *isValid()*. Die erste wird von den Filter-Klassen benutzt. Falls ein Dokument die in einem Filter definierten Bedingungen nicht erfüllt, wird sie mit dem Parameter *false* aufgerufen und das Dokument wird von der weiteren Verarbeitung ausgeschlossen. Mit der Methode *isValid()* wird geprüft, ob das Dokument-Objekt verarbeitet werden soll, oder nicht (z. B. während der grafischen Darstellung).

### 5.4.2 Die Dokument-Datenstruktur

Alle Dokument-Objekte sind in einer internen Datenstruktur angeordnet, die einige zusätzliche Informationen enthält. Dies ist die zentrale Datenstruktur, die von einer Reihe von Klassen und Komponenten verwendet wird. Das entsprechende Klassendiagramm ist im Anhang C beschrieben. Die Methoden dieser Klasse erfüllen den folgenden Zweck:

**DocumentList()** Der Konstruktor dient der Initialisierung einer Liste von Objekten vom Typ *Document*.

**getAllAuthors()** liefert alle Autoren, die in mindestens einem Dokument der Liste angegeben sind. Diese Funktion wird zum Aufbau einer Autorenliste verwendet, die in den Optionen zum Autoren-Filter angeboten wird.

**getAllPublications()** Alle Publikationen, die in mindestens einem Dokument der Liste verzeichnet sind. Die Liste der Veröffentlichungen wird im Filterdialog zur Auswahl durch den Benutzer benötigt.

**getNDocs()** Der Rückgabewert dieser Methode entspricht der Anzahl der in der Dokument-Liste enthaltenen Dokumente.

**getNRelDocs()** gibt die Anzahl der aktuell als „relevant“ eingestuften Dokumente zurück. Das betrifft die Dokument-Objekte, deren Methode *isValid()* den Wert *true* liefert.

**getCreationDate()** Datum, an dem das Suchergebnis erzeugt worden ist, kann in der Dokument-Liste verzeichnet werden. Die Übergabe kann über die Online-Schnittstelle erfolgen oder aus der XML-Ergebnis-Struktur ausgelesen werden. Diese Methode dient der Abfrage des Erzeugungsdatums.

**getModificationDate()** Falls ein Zwischenergebnis weiter bearbeitet wird, ist unter Umständen das Datum der letzten Bearbeitung von Interesse. Die Abfrage erfolgt mit dieser Methode.

### 5.4.3 Lesen und Schreiben von Suchergebnissen

Das System muß auf verschiedene Arten ein Suchergebnis übernehmen und speichern können:

- Lesen des von einer Suchmaschine beim Start als Parameter übergebenen Suchergebnisses (bei einer Online-Sitzung),
- Einlesen eines von einer Suchmaschine in eine Datei geschriebenen Suchergebnisses,
- Schreiben des aktuellen Bearbeitungsstandes in eine Datei,
- Lesen eines Bearbeitungsstandes und Restaurierung der internen Datenstrukturen.

Alle genannten Aufgaben werden einer speziellen Klasse zugeordnet. Die Basis der Kommunikation ist die im Abschnitt 5.9 beschriebene XML-Struktur. Pakete zum Parsen von XML-Strukturen sind für die wichtigsten Programmiersprachen verfügbar, die entsprechenden Klassen werden daher lediglich im Anhang C aufgeführt. Das Klassendiagramm der Klasse *DocumentWriter* ist im Anhang C beschrieben. Die Methoden erfüllen den folgenden Zweck:

**DocumentWriter(FileName, pDocList, KeyWordList)**
Dies ist der Konstruktor, der alle auszugebenden Daten in der Parameterliste enthält:

- den Namen der Datei, in der der Arbeitsstand gespeichert werden soll,
- die aktuelle Menge der Dokumente,
- die Schlüsselwörter

Bei Aufruf des Konstruktors wird zunächst die Datei geöffnet, dann über die weiteren Methoden der Schreibvorgang durchgeführt und anschließend die Datei geschlossen.

**dumpKeyWordList(KeyWordList)**
schreibt die Schlüsselwörter mit den entsprechenden XML-Tags in die angegebene Datei.

**dumpDocumentList(pDocList)**
Die Menge der verfügbaren Dokumente wird mit allen dazugehörigen Informationen in die XML-Datei geschrieben.

**dumpSingleDocument(Document)**
wird iterativ von *dumpDocumentList* verwendet.

**dumpRelevance(pRelevance)**
Die Relevanzen werden als Vektor von Real-Zahlen übergeben und, der beschriebenen XML-Konvention entsprechend, in die Datei geschrieben. Die Methode wird von *dumpSingleDocument* verwendet.

**WriteErrorMsg()**
Bei Auftreten eines Fehlerzustandes (z. B. der angegebene Pfad existiert nicht) wird über diese Methode eine entsprechende Fehlermeldung ausgegeben.

### 5.4.4 Filter

Für die Darstellung der Informationen liegen die dafür benötigten Daten in einer unaufbereiteten, abstrakten Form vor. Während die Filterstufe (siehe Punkt 2.6.3) in der Pipeline der wissenschaftlich-technischen Visualisierung Aufgaben wie Glättung, Normierung oder Vervollständigung der Daten übernimmt, ist in der Informationsvisualisierung ausschließlich die Reduktion der Datenmenge interessant. Die folgenden Varianten der Datenfilterung sollen genauer betrachtet werden. Filterung nach:

- dem Namen des Dokumentautors,
- dem eingetragenen Datum,
- der Größe des Dokuments,
- der Erscheinungsart,
- einer Relevanz-Schwelle,
- Redundanzen.

Voraussetzung für das Funktionieren der einzelnen Filter ist das korrekte Vorhandensein der entsprechenden Parameter (Datum, Autor, etc.) zu den jeweiligen Dokumenten. Die Abbildung 5.5

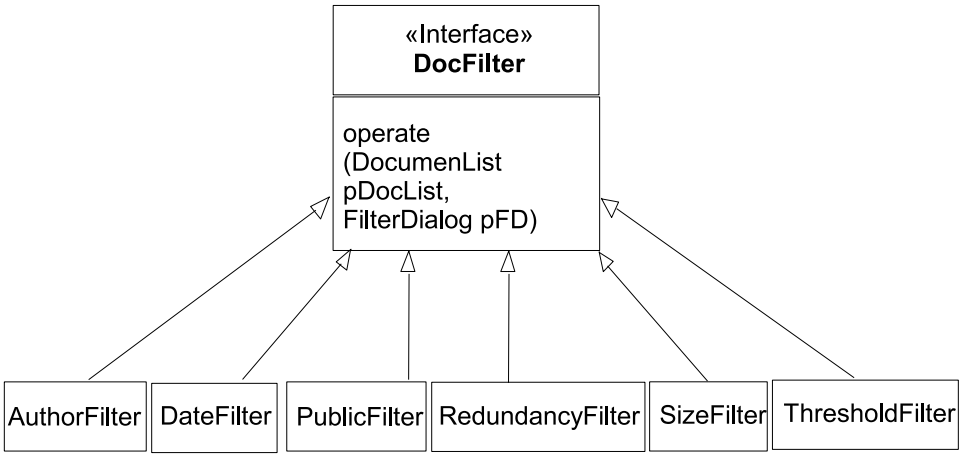



Abbildung 5.5: Vererbung der Filterklassen

zeigt die Vererbungshierarchie der im Entwurf berücksichtigten Filterklassen. Soll das System um weitere Filter erweitert werden, müssen diese das Interface *DocFilter* implementieren. In dieser Interface-Klasse ist lediglich die Methode *operate* vorgeschrieben, die als Parameter die Liste der

Dokumente *DocList* und die aktuellen Einstellungen des Filterdialoges benötigt. Die Aufgabe eines solchen Filters ist das Bilden der Teilmenge $M_2$ aus der Menge der Dokumente $M_1$.

$$M_2 \subset M_1 \tag{5.9}$$

Dabei ist zu beachten, daß mehrere Filter sequentiell arbeiten können. Die Menge $M_1$ bezeichnet die Dokumente, die zum aktuellen Zeitpunkt als „gültig“ gekennzeichnet sind. Nach der Operation eines Filters wird die gebildete Menge $M_2$ automatisch zur Menge $M_1$ für eine gegebenenfalls anstehende neue Filter-Operation.
Die Filter-Operationen, die bisher im Entwurf des Visualisierungsmodules vorgesehen sind, werden nun im Einzelnen betrachtet.

**AuthorFilter**

Der Ausgangspunkt für diese Filterung ist folgende Situation: Der Nutzer hat die Menge der interessanten Dokumente bereits eingegrenzt, indem er in den entsprechenden Bereich des Informationsraumes navigiert ist. Mit Hilfe der im Abschnitt 2.6 beschriebenen Mechanismen hat er den Inhalt einiger Dokumente näher untersucht. Ist ein relevantes Dokument gefunden worden, können nun alle vorhandenen Dokumente desselben Autors herausgefiltert werden. Diese Möglichkeit ist besonders dann nützlich, wenn das Ziel der Informationssuche unscharf ist.
Dieser Filter benötigt als Parameter eine Liste von Namen $L_{Author}$, nach denen die Dokumente durchsucht werden. Dokumente $D$, für die die Bedingung

$$\forall L_{Author}[i] \neq D_{Author}$$

erfüllt ist, werden als „ungültig“ gekennzeichnet. Dieser Filter kann in den Dialogeinstellungen negiert werden, sodaß die Autorenliste als Ausschlußliste interpretiert wird.

**DateFilter**

Ein Problem des WWW ist, daß ein Teil der Informationsbestände nicht gepflegt wird, was dazu führt, daß eine Reihe der abrufbaren Informationen bereits überholt ist. Oft ist bekannt, in welchem Zeitraum eine Veröffentlichung erschienen ist.
Dieser Filter benötigt als Parameter zwei Daten, $t_{min}$ und $t_{max}$. Die Dokumente $D$, für die die Bedingung

$$t_{min} < D.Date \parallel D.Date > t_{max}$$

nicht erfüllt ist werden als „ungültig“ erklärt.

**SizeFilter**

Die von einer Suchmaschine gefundenen Dokumente können verschiedenster Art sein: Kurzmeldungen im Umfang weniger Zeilen, Berichte, Aufzählungen, Werbung bis hin zu kompletten Büchern. Die Größe der Datei kann als Hinweis auf die Art des Dokuments dienen. Es ist jedoch anzumerken, daß dies auf keinen Fall ein scharfes Kriterium ist. Bei großen Datenmengen sollten dennoch alle Möglichkeiten zur Reduktion der Datenmenge ausgeschöpft werden.

$$D.valid = D.size \leq S$$

**PublicFilter**

Dieser Filter kann benutzt werden, um in gewissem Maße Beziehungen zwischen Dokumenten zu visualisieren. Häufig sind wissenschaftliche Veröffentlichungen in „Sammlungen“ wie Bücher, Proceedings von Workshops u.ä., enthalten. Die auf diese Weise zusammengefaßten

Dokumente haben eine enge thematische Beziehung, die einfach visualisiert und nutzbar gemacht werden kann. Hat der Anwender ein solches, für ihn interessantes Dokument gefunden, können alle zur betreffenden „Sammlung“ gehörigen Dokumente herausgefiltert werden. Damit steht dem Nutzer ohne weitere Suche eine Menge von Dokumenten zur Verfügung, die für ihn mit hoher Wahrscheinlichkeit ebenfalls interessant sind.

$$\forall L_{Publicaton}[i] \neq D_{Publicaton}$$

**ThresholdFilter**

Zu jedem Dokument sind zu jedem Suchbegriff die entsprechenden Relevanzen bekannt. Um die Menge der zu visualiserenden Dokumente zu reduzieren, kann festgelegt werden, daß ein Dokument dessen Relevanz bezüglich eines bestimmten Schlüsselwortes unterhalb eines spezifizierten Schwellwertes liegt, von der Weiterverarbeitung ausgeschlossen wird. Auf diese Weise kann außerdem eine Wichtung der Suchergebnisse vorgenommen werden.

$$D_{valid} = \forall R \geq T$$

**RedundancyFilter**

Wie im Abschnitt 2.3.3 beschrieben wurde, erscheinen besonders bei der Verwendung von Meta-Suchmaschinen Dokumente mehrfach in der Ergebnisliste und sind damit auch mehrfach im Informationsraum vorhanden. Mit Hilfe dieses Filters ist es möglich, redundante Dokumente von der Weiterverarbeitung auszuschließen. Die Menge der „gültigen“ Dokumente ergibt sich aus:

$$D_{valid} = D \notin M_2,$$

wobei $M_2$ die in Gleichung 5.9 beschriebene Menge der bereits von den Filtern ermittelten Menge ist.

Die vorgestellten Filter können beliebig miteinander kombiniert werden, um möglichst schnell eine kleine Menge interessanter Dokumente zu finden. Wenn z. B. mit zwei Filtern gearbeitet wird, liefern diese die beiden Mengen $F_1$ und $F_2$. Beide sind nach Gleichung 5.9 echte Teilmengen der Dokumentenmenge $M_1$. Die resultierende Dokumentenmenge $M_2$ ergibt sich somit aus

$$M_2 = F_1 \cap F_2 \tag{5.10}$$

Aufgrund Gleichung 5.10 kann der Fall $M_2 = \oslash$ auftreten. In dieser Situation soll das System an Stelle einer grafischen Darstellung ohne Elemente einen Hinweis auf die leere Dokumentenmenge erzeugen.

## 5.5 Die Renderer-Komponente

Die Bildung von Komponenten bietet sich an, wenn Teilaufgaben des Projektes zu einer selbständig lauffähigen funktionalen Einheit mit einer Schnittstelle zusammengefaßt werden können. Dies trifft bei dem hier entworfenen Sytem auf den Renderer und das Dokumenten-Management zu. Die Renderer-Komponente entspricht dem Anwendungsfall „Darstellung der Ergebnisse“ aus dem Kapitel 4, der sowohl bei der zielgerichteten als auch bei der unscharfen Suche nach Dokumenten auftritt. Der Komponentenbildung kommt in diesem Fall besondere Bedeutung zu, da der Renderer austauschbar sein soll und die Erweiterbarkeit um weitere Renderer gefordert ist. Um deren Einbindung zu erleichtern, enthält diese Komponente eine *interface*-Definition (siehe Abbildung 5.6). Jeder Renderer, der in das System eingebunden werden soll, muß zumindest dieses Interface implementieren. Dabei spielt es keine Rolle, was für eine Grafikschnittstelle der Renderer verwendet – damit wird eine gewisse Beständigkeit hinsichtlich der rasanten Entwicklung in diesem Segment erreicht. Das Interface enthält die folgenden Methoden:

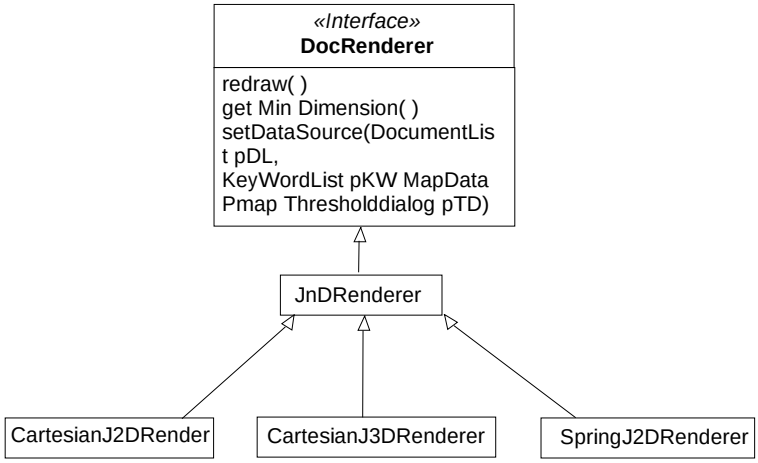



Abbildung 5.6: Vererbung der Renderer-Klassen

**setDataSource(DocumentList, KeyWordList, ThresholdData)**
: vermittelt dem Renderer die Referenzen auf die internen Datenstrukturen (Liste der Dokumente, Liste der Schlüsselwörter, Relevanz-Schwellwerte) der Applikation. Aus diesen Datenstrukturen übernimmt der Renderer bei jeder Aktualisierung der Darstellung die notwendigen Informationen. Die Auswertung der Relevanz-Schwellwerte ist zwar prinzipiell eine Filteroperation, der Renderer soll jedoch die Grenzen des Informationsraumes anzeigen können, die durch die Schwellwerte bedingt sind. Daher ist die separate Übergabe dieser Daten notwendig.

**getRendererInfo()**
: Da mehrere Renderer im System integriert sein können, müssen unter Umständen Informationen über den ausgewählten Renderer bekannt sein. Diese Methode muß daher mindestens die geringstmögliche Dimension des Renderers zurückgeben. Somit kann das System z. B. verhindern, daß ein dreidimensionaler Renderer gestartet wird, wenn nur zwei Schlüsselwörter vorhanden sind.

**update()**
: Beim Aufruf dieser Methode beschafft sich der Renderer alle aktuellen Parameter aus den mit setDataSource angegebenen Quellen und generiert eine neue grafische Darstellung.

In der Visualisierungspipeline wurden die Stufen *Filter, Mapping* und *Renderer* definiert. Im hier entworfenen System enthält die Renderer-Komponente sowohl die Mapping- als auch die Render-Stufe. Die Gründe dafür liegen in dem engen Zusammenhang zwischen dem ausgewählten Mappingverfahren und dem Typ des Renderers:

- Es ist z. B. wenig sinnvoll, bei einem Suchergebnis, das aus lediglich zwei Schlüsselworten resultiert, einen ressourcenintensiven Java3D-Renderer zu initialisieren. Andersseits läßt sich ein Suchergebnis auf Basis vieler Suchbegriffe mit einer 2D-Darstellung (die eine bessere Performance bietet) kaum überschauen.

- Eine Trennung von Mappingverfahren und der zu verwendenden Rendererstufe würde als Konsequenz die Auswahl von beiden Elementen in der Oberfläche bedeuten. Eine solche Auswahl würde aber vom Benutzer Kenntnisse verlangen, die laut Aufgabenstellung nicht vorausgesetzt werden dürfen.

Es können beliebige, austauschbare Renderer-Komponenten entwickelt werden, die das oben beschriebene Interface implementieren. In den folgenden Abschnitten werden zwei Renderer beschrieben, die das im Abschnitt 5.2 entwickelte Abbildungsverfahren verwenden. In der Abbildung 5.6 ist

zusätzlich ein weiterer Renderer verzeichnet, der auf der Basis des Mapping mit Gravitationsquellen (SQWID, siehe Abschnitt 3.2.6) arbeitet.

### 5.5.1 Entwurf eines 2D-Renderers (kartesische Abbildung)

Hierbei handelt es sich um die zweidimensionale Variante des im Abschnitt 5.2 vorgestellten Mapping-Verfahrens (Mapping mit Glyphs). Die Dokumente werden in einem zweidimensionalen, rechtwinkligen Koordinatensystem angeordnet, wobei auf den beiden Achsen Relevanzen abgetragen werden. Die Elemente dieses Renderers werden im Folgenden erläutert:

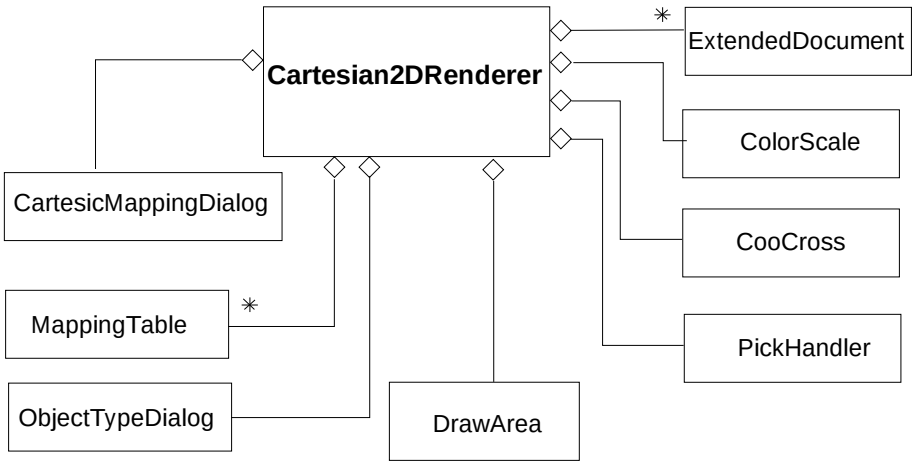



Abbildung 5.7: Aufbau der 2D-Renderer-Komponente

**Klassen:**

**Cartesian2DRenderer**
Hauptklasse des Renderers. Sie initialisiert das verwendete Grafiksystem, baut die zur Kommunikation mit der Grafikschnittstelle notwendigen internen Datenstrukturen auf und verwaltet sie.

**MappingTable**
beinhaltet eine interne Datenstruktur, die beschreibt, auf welche grafische Option die Relevanz welchen Schlüsselwortes abgebildet wird. Vorgesehene grafische Optionen sind derzeit: Position (X- und Y-Achse), Größe, Farbe und Transparenz der Glyphs. Daraus ergibt sich die folgende Tabelle, wobei *SW 1 – SW 5* die Schlüsselwörter bezeichnen:

| Grafische Option | SW 1 | SW 3 | SW 3 | SW 4 | SW 5 |
|---|---|---|---|---|---|
| X-Position | | | | | |
| Y-Position | | | | | |
| Größe | | | | | |
| Farbe | | | | | |
| Transparenz | | | | | |

Zu beachten ist, daß die Zuordnung eineindeutig sein muß, jedem Schlüsselwort darf genau ein grafischer Parameter zugewiesen werden und jedem Parameter genau ein Schlüsselwort.

**CartesicMappingDialog**
Das Ausfüllen der Mapping-Tabelle aus der Klasse *MappingTable* ist die Aufgabe des Benutzers. Selbstverständlich wird während der Initialisierung von der Klasse *Cartesian2DRenderer* eine sinnvolle Vorbelegung entsprechend der Anzahl der verwendeten Suchbegriffe getroffen. Der Benutzer soll jedoch die Möglichkeit haben, Anpassungen gemäß seinen Bedürfnissen vornehmen zu können. Diese Klasse stellt den Dialog dafür zur Verfügung und stellt die Konsistenz der Tabelle sicher, indem die Eineindeutigkeit während der Manipulation geprüft wird.

**ObjectTypeDialog**
Die Darstellung der Dokumente erfolgt mittels Glyphs, die verschiedene Formen haben können. Im zweidimensionalen Fall sind dafür z. B. Kreise, Rechtecke, Dreiecke (jeweils leer oder gefüllt) denkbar. Die Einstellung der Form erfolgt mit dem Dialog dieser Klasse.

**DrawArea**
stellt die Kopplung zur verwendeten 2D-Grafikschnittstelle sicher. Die grafische Ausgabe der Zeichenelemente erfolgt im Allgemeinen über einen *grafischen Kontext*, der vom Grafiksystem zur Verfügung gestellt wird. Dieser Kontext ist eine Datenstruktur, die mit einem Ausgabefenster und dessen Ressourcen verbunden ist. In die Datenstruktur werden die grafischen Elemente und ihre Attribute eingetragen und vom Grafiksystem angezeigt. Die Klasse kapselt den grafischen Kontext und stellt außerdem die Interaktion zur Selektion von Dokumenten sicher. Dazu ist unter anderem die Verbindung zur Klasse *PickHandler* notwendig.

**PickHandler**
Zweidimensionale Grafiksysteme arbeiten im sogenannten *immediate-mode*. Das bedeutet, daß ein Ausgabeelement sofort nach dem Zeichenbefehl dargestellt wird. Eine Speicherung in Datenstrukturen (wie z. B. in einer Szene) erfolgt nicht, das macht diesen Ausgabemodus schmal und effizient. Somit können einmal gezeichnete Objekte jedoch nicht „wiedererkannt“ werden. Im Falle einer Selektion mit der Maus muß die Erkennung auf der Grundlage der Position erfolgen, diese Funktionalität stellt diese Klasse zur Verfügung.

**ExtendedDocument**
enthält alle Informationen der Klasse *Document* und weitere zusätzliche Informationen, wie z. B. die Position des entsprechenden Glyphs auf der Zeichenfläche. Diese sind für ein effizientes Auffinden der Dokumente bei Interaktionen nützlich und können in verschiedenen Grafiksystemen unterschiedlich sein.

**ColorScale**
Falls ein Schlüsselwort auf das grafische Attribut *Farbe* abgebildet wird, ist es nützlich eine Farbskala zur besseren Einschätzung der Relevanz einzublenden. Diese Klasse stellt die Farbskala zur Verfügung.

**CooCross**
erzeugt ein zweidimensionales Koordinatensystem, wobei die Achsen mit den entsprechenden Schlüsselwörtern beschriftet sind. Die Übersichtlichkeit der Darstellung wird dadurch erhöht.

Die Menüstruktur des kartesischen zweidimensionalen Renderers ist in der Abbildung B.2 im Anhang B dargestellt. Die Einträge lösen die folgenden Aktionen aus:

| | |
|---|---|
| Options/Glyph Type | Auswahl der grafischen Repräsentation der Dokumente |
| Mapping/Parameters | Einstellung der Mapping-Tabelle, welches Schlüsselwort wird auf |

| | |
|---|---|
| | welches grafische Attribut abgebildet |
| Show/Coordinate Axis | Einblenden eines Koordinatensystemes mit Achsenbeschriftungen (Schlüsselwörter) zur besseren Orientierung |
| Show/Color Scaling | Einblenden einer Farbskala |
| Show/Document Titles | Einblenden der Dokumenten-Titel |

### 5.5.2 Entwurf eines 3D-Renderers (kartesische Abbildung)

In diesem Abschnitt wird die Ausführung des kartesischen Mappings im dreisimensionalen Raum beschrieben. Die Arbeitsweise ist prinzipiell analog zum äquivalenten Verfahren im zweidimensionalen Fall. Die Elemente dieses Renderers werden im Folgenden erläutert:

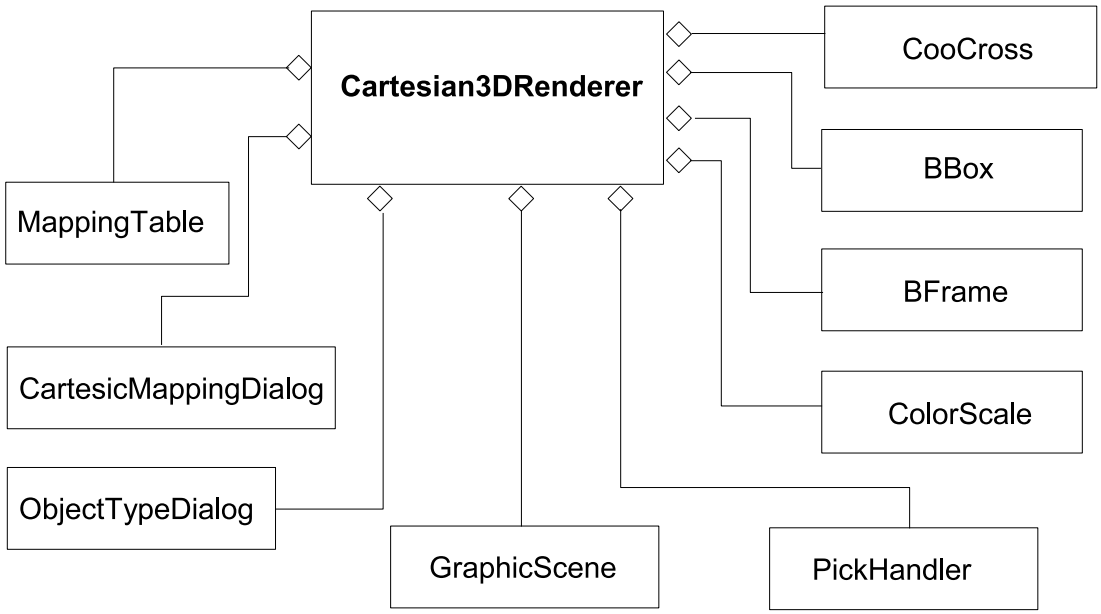



Abbildung 5.8: Aufbau der 3D-Renderer-Komponente

**Cartesian3DRenderer**
Aufbau und Verwaltung von internen Datenstrukturen zur Kommunikation mit dem verwendeten Grafikksystem sowie die Initialisierung des Grafiksystems. Sie steuert alle anderen Klassen der Renderer-Komponente.

**MappingTable**
Diese Klasse beinhaltet eine interne Datenstruktur, die beschreibt, auf welche grafische Option die Relevanz welchen Schlüsselwortes abgebildet wird. Vorgesehene grafische Optionen sind derzeit: Position (X-, Y- und Z-Achse), Größe, Farbe und Transparenz der Glyphs. Daraus ergibt sich die folgende Tabelle, wobei *SW 1 – SW 6* die Schlüsselwörter bezeichnen:

| Grafische Option | SW 1 | SW 3 | SW 3 | SW 4 | SW 5 | SW 6 |
|---|---|---|---|---|---|---|
| X-Position | | | | | | |
| Y-Position | | | | | | |
| Z-Position | | | | | | |
| Größe | | | | | | |
| Farbe | | | | | | |
| Transparenz | | | | | | |

Auch hier ist zu beachten, daß die Zuordnung eineindeutig erfolgen muß.

**CartesicMappingDialog**
Dialog, mit dem der Benutzer die interne Mapping-Tabelle der Klasse *MappingTable* füllen, bzw. manipulieren kann. Eine sinnvolle Vorbelegung erzeugt die Klasse *Cartesian3DRenderer* während der Initialisierung. Die geforderte Eineindeutigkeit der Zuordnungen von Schlüsselwörtern und grafischen Attributen wird von der Dialogklasse sichergestellt.

**ObjectTypeDialog**
Der Typ des Glyphs zur Darstellung der Dokumente kann vom Benutzer mit Hilfe des Dialoges dieser Klasse eingestellt werden. Für die dreidimensionale Darstellung sind z. B. Kugeln oder Würfel geeignet.

**GraphicScene**
Moderne 3D-Grafiksysteme verfügen über einen sogenannten *retained mode*. Das bedeutet, daß die Prozesse der Geometriegenerierung und der Anzeige getrennt ablaufen. Die Schnittstelle beider Prozesse ist ein sogenannter *Szenengraph*. Während der Geometriegenerierung werden die erzeugten grafischen Objekte mit ihren Attributen in den Szenengraph eingetragen. Der Anzeigeprozeß traversiert gleichzeitig den Szenengraph fortlaufend und generiert ein zweidimensionales Bild in dem zugeordneten Fenster. Diese Funktionalität sowie die Verwaltung der notwendigen Ressourcen und das Interaktionsverhalten werden von dieser Klasse gekapselt. Die Grafiksysteme stellen selbst Methoden zur Interaktion zur Verfügung (siehe Kapitel 6.1). Dazu gehören sowohl grafische Manipulationen wie Drehung und Verschiebung der Kamera oder Zoom als auch das Selektieren und Erkennen von grafischen Objekten (*pick*)

**PickHandler**
Die Aufgabe dieser Klasse besteht in der Bereitstellung der Informationen zu dem Dokument, dessen Repräsentation in der grafischen Darstellung über eine Interaktion ausgewählt wurde. Die Informationen werden zur Evaluierung des Dokuments durch den Benutzer benötigt, die in der Komponenente *DocumentManager* realisiert wird.

**CooCross**
Zur besseren Orientierung in der grafischen Darstellung kann ein dreidimensionales Koordinatensystem eingeblendet werden, dessen Achsen mit den entsprechenden Schlüsselwörtern beschriftet sind. Das Koordinatensystem wird von dieser Klasse bereitgestellt.

**BBox**
Aufgrund der vielfältigen Einstellungsmöglichkeiten der Kamera ist unter Umständen eine Unterstützung des räumlichen Vorstellungsvermögens des Benutzers sinnvoll. Eine solche Unterstützung bietet diese Klasse, indem sie einen halbdurchlässigen Würfel um den Informationsraum erzeugt. Für den Benutzer wird der Informationsraum somit visuell zu einem grafischen Objekt mit einem Bezugspunkt, wodurch die Orientierung im Raum wesentlich verbessert wird.

**BFrame**
erfüllt dieselbe Funktion wie die Klasse *BBox*. Hierbei wird der Würfel jedoch nicht mittels Flächen erzeugt, sondern es werden lediglich die Kanten gezeichnet.

**ColorScale**
stellt eine Farbskala zur Verfgung, mit deren Hilfe der Benutzer die Relevanz von Dokumenten besser einschätzen kann. Diese Skala kann bei Bedarf in das Bild eingeblendet werden.

Die Menüstruktur des kartesischen dreidimensionalen Renderers ist in der Abbidlung B.3 im Anhang B dargestellt. Die Einträge lösen die folgenden Aktionen aus:

| | |
|---|---|
| Options/Glyph Type | Auswahl der grafischen Repräsentation der Dokumente |
| Mapping/Parameters | Einstellung der Mapping-Tabelle, welches Schlüsselwort wird auf welches grafische Attribut abgebildet |
| Show/Coordinate Axis | Einblenden eines Koordinatensystems mit Achsenbeschriftungen (Schlüsselwörter) zur besseren Orientierung |
| Show/Bounding Box | Einblenden eines halbdurchlässigen Würfels mit den Dimensionen des Koordinatensystemes $[1, 1, 1]$ |
| Show/Bounding Frames | Einblenden eines Rahmens mit den Dimensionen des Koordinatensystemes |
| Show/Color Scaling | Einblenden einer Farbskala |
| Show/Document Titles | Einblenden der Titel der Dokumente |

## 5.6 Die Komponente „Dokument-Manager“

Diese Komponente deckt im Wesentlichen die geforderte Funktionalität des Anwendungsfalles „Auswahl und Bezug relevanter Dokumente“ ab. Sie erhält eine eigene Benutzeroberfläche und ist die zentrale Einheit zur Steuerung des gesamten Systems durch den Benutzer. Die Komponente erfüllt die folgenden Teilaufgaben:

- Anzeige von allgemeinen Informationen zum Suchergebnis,
- Auswahl einer Kombination aus Mapping-Verfahren und Renderer,
- Starten und Anhalten der entsprechenden Renderer,
- Anzeige von Detail-Informationen zu selektierten Dokumenten,
- Markieren von Dokumenten als „interessant/uninteressant“,
- Verwaltung von ausgewählten Dokumenten in einer speziellen Liste,
- Entfernen von Dokumenten aus der Datenstruktur,
- Anzeige und Download von markierten Dokumenten

Der Aufbau des Dokument-Managers ist in der Abbildung 5.9 dargestellt. Diese Komponente übernimmt die programminterne Verwaltung von Objekten vom Typ *Document* und steht in keiner Beziehung zum „Dokument-Management“ mit realen Dokumenten.

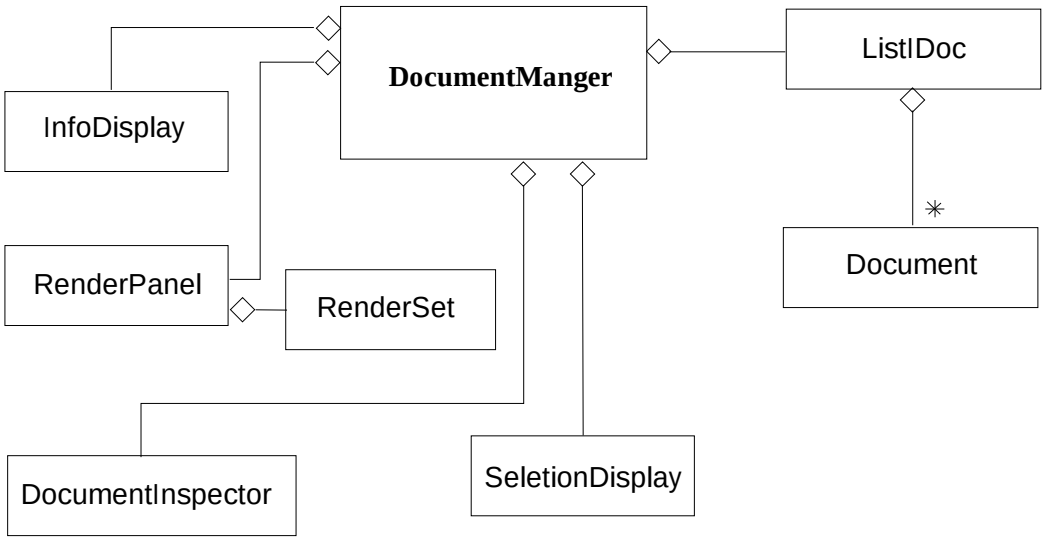



Abbildung 5.9: Die Komponente „Dokument-Manager“

### 5.6.1 Die Elemente

Die Komponente wird zum Programmstart von der Klasse *Control* initialisiert. Von dieser erhält sie alle notwendigen Informationen zur Weitergabe an den Benutzer und übermittelt die Ergebnisse der Interaktionsschritte.

**Klassen:**

**DocumentManager**
Dies ist die Hauptklasse der Komponente. Sie initialisiert alle Interaktionselemente und baut die Benutzeroberfläche und interne Datenstrukturen auf.

**InfoDisplay**
erhält alle Informationen über das zu bearbeitende Suchergebnis sowie den Status der Bearbeitung und zeigt diese an.

**RenderPanel**
bietet die Liste der implementierten Renderer mit ihren Mappingverfahren in einem Dialog an.

**RenderSet**
Die Renderer, die vom System angeboten werden, müssen in einer internen Liste verzeichnet sein, die von dieser Klasse bereitgestellt wird. Dies ist wichtig für die geforderte einfache Erweiterbarkeit des Systems.

**DocumentInspector**
Die Anzeige der zu einem Dokument verfügbaren Informationen und das Markieren eines Dokuments werden in dieser Klasse realisiert.

**SelectionDisplay**
Die Liste der interessanten Dokumente wird durch diese Klasse zur Anzeige gebracht. Weiterhin wird die Möglichkeit zur Selektion von Dokumenten in dieser Liste realisiert.

**ListIDoc**
Dokumente, die als „interessant“ markiert sind, werden in einer internen Datenstruktur verwaltet, die von dieser Klasse bereitgestellt wird.

### 5.6.2 Die Benutzeroberfläche

Über die Benutzeroberfläche dieser Komponente finden bis auf wenige Ausnahmen (Laden/Speichern von Suchergebnissen, Filtereinstellungen) alle Interaktionen statt. Daher ist eine übersichtliche Anordnung der Interaktionselemente von enormer Bedeutung. Die Interaktionen können in Gruppen eingeteilt werden. In der oben gezeigten Anlage der Klassen wurde dies bereits berücksichtigt. Diese Gruppen sind:

**Informationsanzeige**
Innerhalb dieses Interaktionselements werden die Informationen zum Suchergebnis und der aktuelle Bearbeitungsstand angezeigt. Dies sind im Einzelnen:

- der Dateiname (falls es sich um ein von einer Suchmaschine gespeichertes Ergebnis oder das Zwischenergebnis einer früheren Sitzung handelt),
- das Erzeugungsdatum des Suchergebnisses,
- das Datum der letzten Änderung bei einem Zwischenergebnis,
- die verwendeten Schlüsselwörter,
- die Anzahl der gefundenen Dokumente,
- die Anzahl der relevanten Dokumente,
- die Anzahl der interessanten Dokumente.

Unter der Anzahl der relevanten Dokumente wird die Zahl der Dokumente verstanden, die durch den Filterprozeß nicht von der weiteren Verarbeitung ausgeschlossen wurden. Nur diese Dokumente werden vom Renderer angezeigt.
Die Anzahl der interessanten Dokumente ist die Zahl der Dokumente, die vom Benutzer explizit markiert worden sind. Diese und die Zahl der relevanten Dokumente werden in der Anzeige ständig aktualisiert.
Die genannten Anzeigen sind in der oben beschriebenen Klasse *InfoDisplay* realisiert. Eingaben durch den Benutzer sind nicht vorgesehen.

**Grafische Darstellung**
Diese Gruppe von Elementen dient zur Initialisierung und Steuerung der Renderer (siehe Klasse *RenderPanel*). Folgende Interaktionen werden unterstützt:

- Auswahl des Renderers/Mappingverfahrens
- Starten des Renderers
- Aktualisieren des Renderers
- Anhalten des Renderers

Wie oben beschrieben, ist ein Mappingverfahren in einem bestimmten Renderer implementiert und das System kann mehrere Renderer enthalten. Der Benutzer hat die Möglichkeit, aus der Liste der verfügbaren Renderer einen ihm als geeignet erscheinenden auszuwählen. Dies kann z. B. mit einem Interaktionselement *Auswahlbox* erfolgen.
Das Starten, Anhalten und Aktualisieren der Renderer kann mittels *Buttons* realisiert werden, wobei darauf zu achten ist, daß ein Renderer erst dann gestartet werden kann, wenn

bereits ein Suchergebnis geladen ist.
Auf die Interaktion „Aktualisieren“ könnte prinzipiell verzichtet werden, da die Architektur eine automatische Aktualisierung zuläßt. Diese müßte nach allen Änderungen der Filtereinstellungen ausgelöst werden. Besonders bei den Einstellungen der Relevanz-Schwellwerte würde dieser Vorgang jedoch häufig durchgeführt, was bei großen Datenmengen in einer dreidimensionalen Darstellung zu Performance-Problemen führen kann. Daher kann der Benutzer die von ihm gewünschten Einstellungen im Komplex vornehmen und anschließend die Aktualisierung der grafischen Darstellung manuell auslösen.

**Dokumentenauswahl**
Der Benutzer kann innerhalb dieser Interaktionsgruppe die Entscheidung über die Weiterverarbeitung jedes einzelnen Dokuments treffen. Die folgenden Interaktionen werden zur Verfügung gestellt (siehe Klasse *DocumentInspector*):

- Anzeige detaillierter Informationen zu einem Dokument,
- markieren eines Dokuments als „interessant“,
- Download eines Dokuments und Anzeige in einem Browser,
- Löschen eines Dokuments

In jedem Renderer hat der Benutzer die Möglichkeit, die grafischen Repräsentationen der Dokumente mit der Maus zu selektieren. Der Renderer ermittelt das dazugehörige Dokument-Objekt, welches vom System zu dieser Anzeige übermittelt wird. Hier werden alle zu diesem Dokument verfügbaren Informationen angezeigt, dazu kann ein *Textausgabefeld* verwendet werden.
Falls das System online betrieben wird, kann der Benutzer ein angewähltes Dokument sofort downloaden und in einem Browser darstellen lassen. Besteht diese Möglichkeit nicht, oder möchte der Benutzer das Dokument erst vormerken, kann es als „interessant“ markiert werden (z. B. mit einer *CheckBox*).
Wenn ein Dokument vom Benutzer mit Sicherheit als „nicht interessant“ eingestuft wird, kann es von der weiteren Verarbeitung ausgeschlossen werden, indem es permanent aus der Datenstruktur entfernt wird.

**Anzeige der „interessanten“ Dokumente**
Die als „interessant“ gekennzeichneten Dokumente werden in einer speziellen Datenstruktur (siehe Klasse *ListIDoc*) gehalten. In dieser Interaktionsgruppe sind sie in einem Listenformular sichtbar, wobei jeweils der Titel des Dokuments angezeigt wird. Der Benutzer kann ein Dokument aus der Liste mit der Maus selektieren. Diese Aktion verläuft analog zur Selektion im Renderer – die verfügbaren Informationen werden in der Dokumentauswahl angezeigt. Dieses Dokument befindet sich dann im Zugriff des Systems, der Benutzer kann die oben beschriebenen Aktionen mit dem Dokument ausführen. Diese Funktionalität wird von der Klasse *SelectionDisplay* zur Verfügung gestellt.
Hat der Benutzer die Liste der für ihn wichtigen Dokumente zusammengestellt, besteht die Möglichkeit, diese über einen einzigen *Button* zu laden.

Eine Übersicht der Interaktion des Benutzers mit dieser Komponente ist in der Abbildung 5.10 dargestellt. Andere Aktionen, z. B. das Einschalten von Filtern, sind hier nicht berücksichtigt. Eine detaillierte Beschreibung erfolgt später in Form von Sequenzdiagrammen (siehe Abschnitt 5.8.1).
Zunächst erfolgt der Zugriff auf das Suchergebnis durch den Start des Systems aus einer Web-Applikation mit Parameterübergabe oder durch den expliziten manuellen Start und das Laden eines Such- oder Zwischenergebnisses über das Menü. Diese Aktionen werden nicht von der Komponente *Dokument-Manager* sichergestellt, sondern von der Klasse *Control* (Siehe Abschnitt 5.3.1).

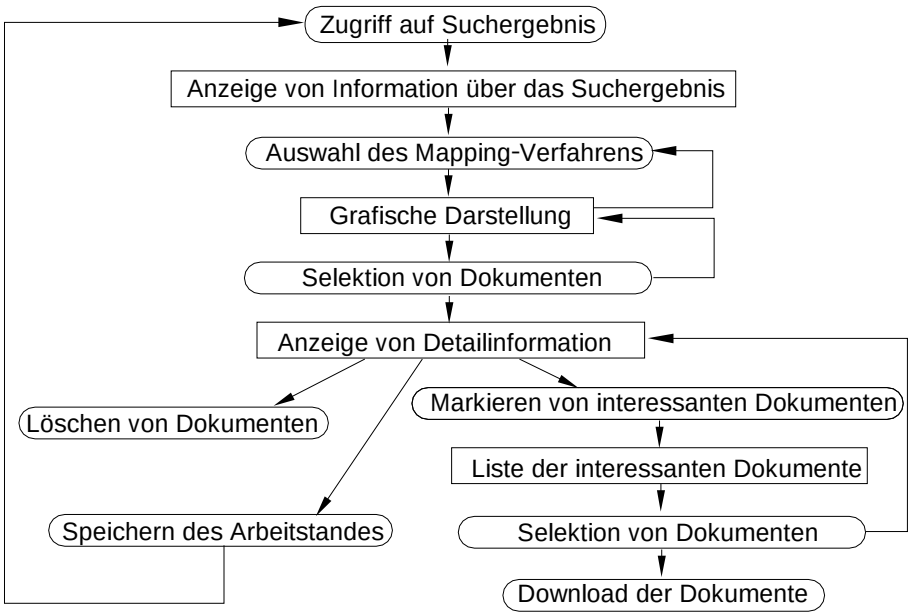



Abbildung 5.10: Interaktionen im „Dokument-Manager“

Im nächsten Schritt wird sich der Benutzer über das Suchergebnis im Allgemeinen informieren. Die meisten Anzeigen haben informativen Charakter, eine wichtige Kenngröße ist jedoch die Anzahl der Schlüsselwörter. Diese ist entscheidend für die Auswahl des Mappingverfahrens/Renderers. Das System kann zwar Fälle verhindern, die technisch nicht machbar sind (z. B. Verwendung eines Renderers, der drei Dimensionen voraussetzt, wenn nur zwei Schlüsselwörter vorhanden sind), der Benutzer muß jedoch anhand der Schlüsselwörter ein geeignetes Verfahren auswählen.
Anschließend wird der Renderer ausgewählt, gestartet und die grafische Darstellung generiert. Falls diese nicht den Vorstellungen des Benutzers entspricht, kann der Renderer angehalten und ausgetauscht werden.
Innerhalb der nun erzeugten grafischen Darstellung können die angezeigten Dokumente mit der Maus selektiert werden. Alle Informationen, die zu dem momentan selektierten Dokument verfügbar sind, werden angezeigt. Anhand dieser Informationen kann der Benutzer entscheiden, ob das Dokument gelöscht, sofort angefordert (nur während einer Online-Sitzung möglich), oder vorgemerkt werden soll. Falls der Benutzer das Dokument markiert, wird es in eine Liste übernommen, die ebenfalls angezeigt wird.
Innerhalb dieser Liste kann ein Dokument wie im Renderer selektiert werden und befindet sich dann wieder im Zugriff der Detailanzeige, wobei auch wieder die Entscheidung zum Download, Löschen oder Vormerken möglich ist. Auf diese Weise kann die Liste der interessanten Dokumente immer weiter eingegrenzt werden. Besonders effizient ist dieser Vorgang in Kombination mit den Filtereinstellungen.
Der Benutzer kann den Vorgang jederzeit unterbrechen, den aktuellen Bearbeitungsstand speichern und später mit dem Laden des Zwischenergebnisses wieder fortsetzen. Am Ende steht eine Liste mit ausgewählten Dokumenten, die auf Wunsch vom System angefordert werden können.

## 5.7 Packages

Es empfiehlt sich, die Elemente des Entwurfes ihren Funktionalitäten entsprechend in Pakete einzuordnen. Dieses Vorgehen hat aufgrund der vorgesehenen Erweiterbarkeit besondere Bedeutung. Bei der Implementation und Integration zusätzlicher Renderer sollten diese in neuen Packages (mit eigenem Namensraum) definiert werden, um die Gefahr von Seiteneffekten möglichst gering zu halten.
Die Pakete werden wie folgt bezeichnet, eine genaue Zuordnung aller definierten Klassen kann dem Anhang D entnommen werden.

**docvis.control**
: Dieses Paket enthält alle Elemente, die für die Lauffähigkeit des Systems notwendig sind. Dazu gehören die Initialisierung der Oberfläche, der Zugriff auf die Datenbasis und die Bedienung der externen Schnittstellen.

**docvis.filter**
: In diesem Paket befindet sich die Interface-Definition zur Implementation weiterer Filter, der Dialog zur Einstellung der Filter-Parameter sowie die Filterklassen selbst.

**docvis.renderer**
: Das Paket enthält die Interface-Definition, nach der Renderer implementiert werden müssen. Jede Renderer-Implementation soll in einem eigenen Subpackage erfolgen. Für die in diesem Entwurf beschriebenen Renderer sind dies:

  **docvis.renderer.cartesian2**
  : zweidimensionaler Renderer, der die Dokumente als Icons in einem gewöhnlichen Koordinatensystem präsentiert,

  **docvis.renderer.cartesian3**
  : dreidimensionaler Renderer, bei dem die Dokumente an berechneten Positionen im Raum in Form von Glyphs dargestellt werden,

  **docvis.renderer.gravity2**
  : zweidimensionaler Renderer, der Schlüsselworte als Gravitationsquellen interpretiert, wobei sich die Dokumente in Abhängigkeit dieser Quellen positionieren.

**docvis.docmanager**
: In diesem Paket sind alle Elemente zusammengefaßt, mit denen der Benutzer die Dokumente evaluiert und letztendlich auswählt oder verwirft. Dazu gehören eine Reihe von Dialogelementen und Datenstrukturen.

Die Import-Beziehungen der Pakete sind in der Abbildung 5.11 dargestellt.

## 5.8 Dynamische Betrachtungen

Der Prozeß der Visualisierung von Suchergebnissen ist durch diverse Interaktionen zwischen dem Benutzer und dem System gekennzeichnet. Daher ist es wichtig, neben dem statischen Aufbau des Systemes auch die Vorgänge zu betrachten, die bis zum Erreichen des Zieles des Prozesses durchzuführen sind.

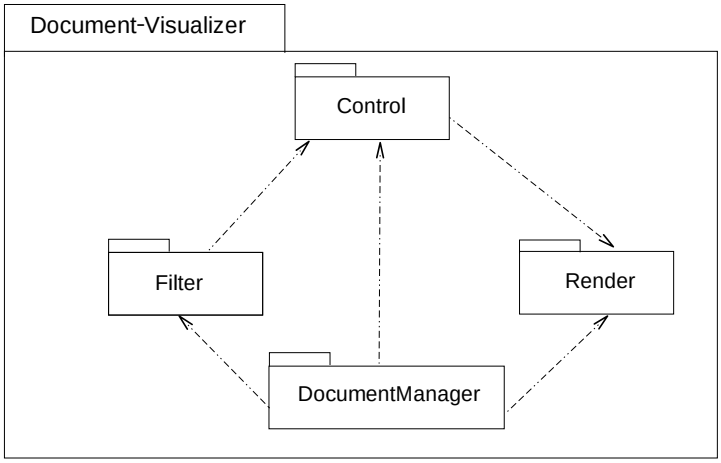



Abbildung 5.11: Import-Beziehungen der Pakete

### 5.8.1 Dynamische Modelle im Objektorientierten Entwurf

Das dynamische Modell soll das Verhalten zwischen Benutzer und System in Form von Diagrammen wiedergeben. Um das gesamte Verhalten abdecken zu können, müssen die im Abschnitt 4.4.2 erläuterten Anwendungsfälle analysiert und die darin enthaltene Dynamik beschrieben werden. Zu diesem Zweck werden hier zwei Darstellungsdiagramme vorgestellt, die von UML als dynamisches Modell angeboten werden. Dies sind das *Sequenzdiagramm* und das *Kollaborationsdiagramm* [Erl00, Bal99, Oes98], die in den folgenden Abschnitten beschrieben werden.

#### Sequenzdiagramme

Die Sequenzdiagramme (auch Interaktionsdiagramme genannt) verdeutlichen eine Menge von Interaktionen zwischen Objekten, die Nachrichten austauschen (interagieren), um eine Aufgabe zu erfüllen. Sequenzdiagramme sind die Darstellung eines Ablaufes (Szenario). Die Darstellung erfolgt zweidimensional, wobei die vertikale Richtung einer Zeitachse entspricht. Auf der horizontalen Achse werden die betrachteten Objekte aufgetragen. Jedes Objekt wird durch eine vertikale gestrichelte Linie (Lebenslinie) repräsentiert. Sie zeigt die Lebenszeit eines Objektes an. An dem oberen Ende der Linie wird ein Objektsymbol eingetragen, das Name und Klasse des Objekts enthält. Zur Ausführung einer Objektoperation wird für die Dauer dieser Operation die Lebenslinie zu einem Aktivierungsbalken verbreitert. Durch die Beendigung der Objektexistenz werden die Lebenslinie bzw. Aktivierungslinie verkürzt und durch ein Kreuz abgeschlossen. Die Botschaften werden durch horizontale Pfeile notiert, die sich von der Lebenslinie des Senderobjekts zu einem Empfängerobjekt erstrecken. Die Reihenfolge der Nachrichten wird durch die vertikalen Zeitachsen von oben nach unten definiert.

#### Kollaborationsdiagramme

Das Kollaborationsdiagramm ist eine Darstellungsform, welche die Zusammenarbeit und Interaktion der Objekte unterstreicht. Es zeigt die gleichen Sachverhalte wie ein Sequenzdiagramm, jedoch aus einer anderen Perspektive. Hier stehen die Objekte und deren Zusammenarbeit untereinander im Vordergrund [Oes98].
Die einzelnen Objekte werden als Rechtecke dargestellt. Diese werden durch Assoziationslinien miteinander verbunden. Die Nachrichten werden in Form von Pfeilen gekennzeichnet und durch eine Nummerierung gekennzeichnet.

### 5.8.2 Sequenzdiagramme zum Visualisierungssystem

Mit den beiden oben beschriebenen Diagrammen können unterschiedliche Aspekte des Systemverhaltens modelliert werden. Das Sequenzdiagramm ist für diese Darlegungen ausgewählt worden, da es als Mittel zur Beschreibung der dynamischen Struktur des zu modellierenden System gewisse Vorteile gegenüber dem Kollaborationsdiagramm bietet. Ein Sequenzdiagramm beschreibt das Interobjektverhalten, d.h. wie einzelne Objekte miteinander Nachrichten austauschen (interagieren) in einem abgegrenzten Kontext. Dies geschieht in der Regel für einen Anwendungsfall, unter Betonung des zeitlichen Aspekts, so daß die Aufeinanderfolge der Nachrichten deutlicher und transparenter wird als in Kollaborationsdiagrammen. Da genügend Platz für Erläuterungen zu den Nachrichten bzw. Abfolgen von Nachrichten in Form von Texten vorhanden ist, kann der Informationsgehalt verbessert werden. Dabei bleibt die grafische Darstellung übersichtlicher als bei Kollaborationsdiagrammen.

#### Dokumente suchen und darstellen

In der Abbildung 5.12 ist der genannte Anwendungsfall aus dem Kapitel 4 aufbereitet:

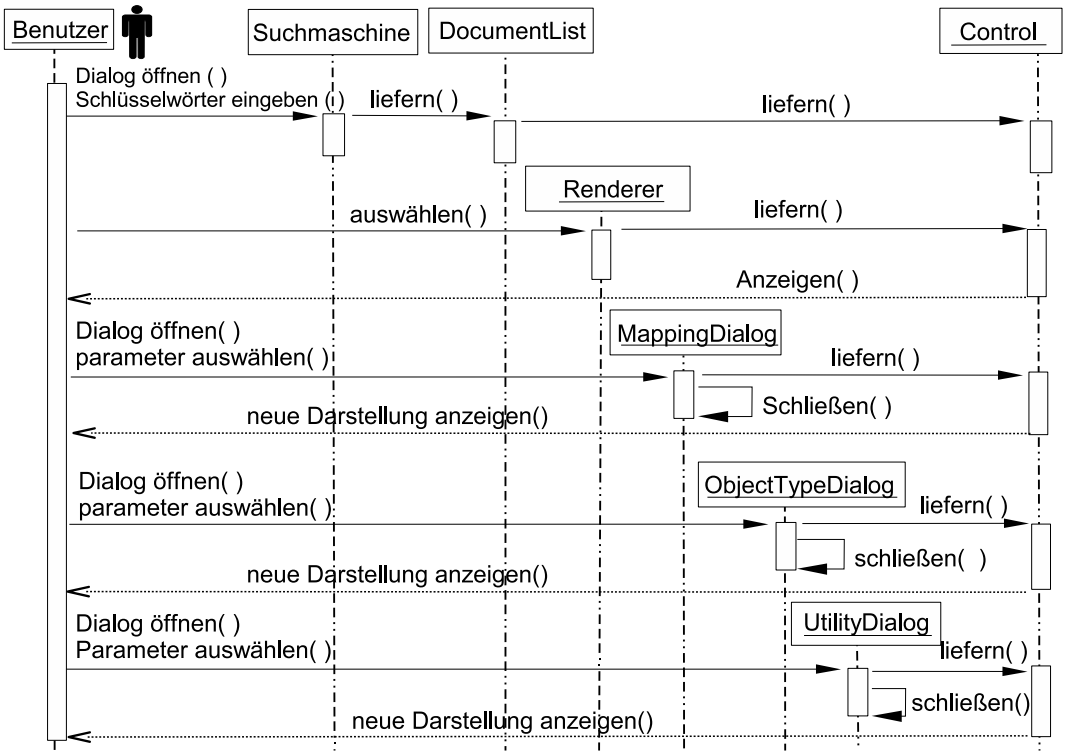



Abbildung 5.12: Sequenzdiagramm: Dokumente suchen und darstellen

1. Das Objekt *Benutzer* sendet über das Anfrageformular eine Nachricht in Form von Schlüsselwörtern an das Objekt *Suchmaschine*.

2. Diese führt die Suche aus und berechnet durch ein sogenanntes Ranking die Relevanz der Dokumente anhand der gegebenen Schlüsselwörter.

3. Dann erzeugt das Objekt *Suchmaschine* eine Liste der gefundenen Dokumente mit den berechneten Relevanzen, die im Objekt *DocumentList* enthalten ist. Diese wird an das Objekt *Control* gesendet.

4. Der Benutzer wählt die gewünschte Art der grafischen Darstellung der Dokumente aus.

5. Das Objekt *Renderer* generiert die grafische Darstellung mit den vordefinierten Parametern und zeigt diese an.

6. Der Benutzer kann die Art und Weise der Abbildung der Relevanzen auf grafische Eigenschaften entsprechend seinen speziellen Erfordernissen über das Objekt *MappingDialog* anpassen (siehe Abschnitt 5.5).

7. Über den Dialog *ObjectTypeDialog* kann die Form der für die Darstellung der Dokumente zu verwendenden Symbole eingestellt werden.

8. Mit Hilfe des Objektes *UtilityDialog* werden Hilfsmittel zur Verbesserung der grafischen Darstellung gesteuert.

Nach diesen Einstellungen verfügt das Visualisierungssystem über alle notwendigen Daten und generiert eine grafische Darstellung des Informationsraumes und der darin enthaltenen Dokumente entsprechend den aktuellen Bedürfnissen des Benutzers. Die Schritte 6 bis 8 sind nicht an die hier beschriebene Reihenfolge gebunden und können vom Benutzer beliebig oft wiederholt werden.

### Reduzierung der Dokumentenmenge

Der Benutzer betrachtet die Darstellung und kann diese seinen Wünschen entsprechend verändern. Ebenso kann er die Menge der angezeigten Dokumente regulieren. Dabei ist er an keine bestimmte Reihenfolge seiner Aktionen gebunden. Gegebenenfalls werden die folgenden Aktionen, die in der Abbildung 5.13 dargestellt sind, auch mehrfach ausgeführt, bis ein optimales Ergebnis erreicht ist.

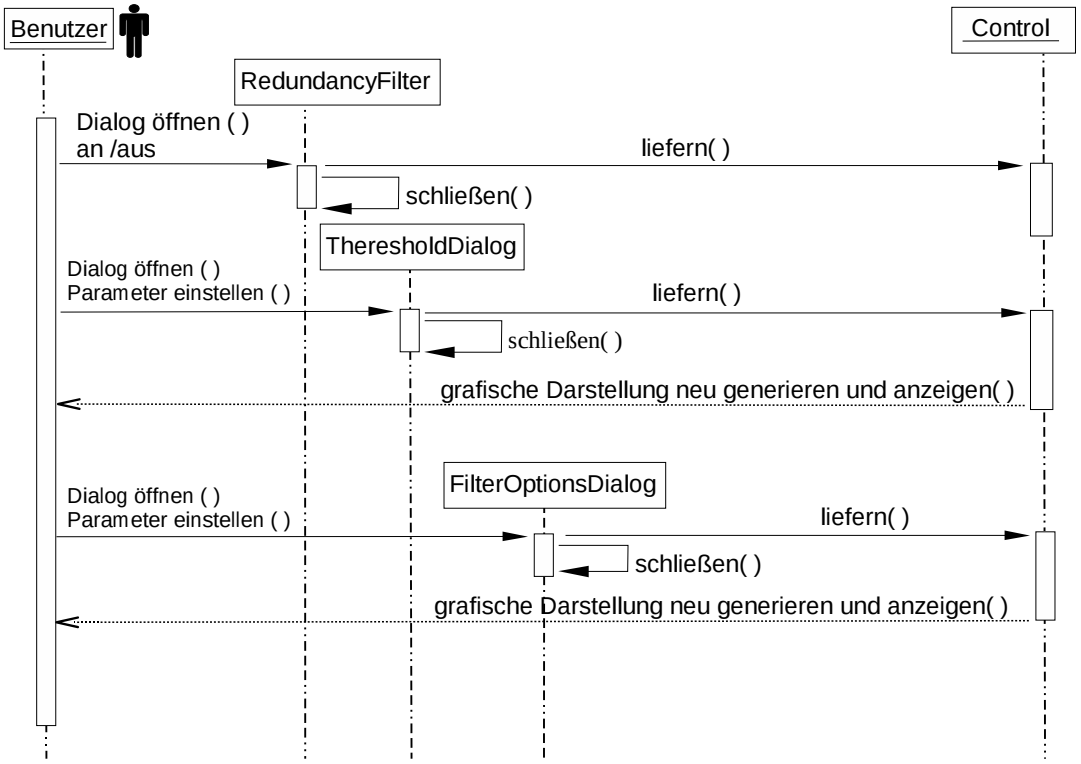



Abbildung 5.13: Sequenzdiagramm: Aktionen für die Reduzierung der Dokumentenmenge

1. Mit der Aktivierung des Parameters im Dialog *RedundancyFilter* ist es möglich, redundante Dokumente von der Weiterverarbeitung auszuschließen.

2. Es wird der Dialog *ThresholdDialog* geöffnet, der die Eingabe von Relevanz-Schwellwerten gestattet. Die Dokumente, bei denen mindestens eine Relevanz unter dem entsprechenden eingestellten Schwellwert liegt, werden zukünftig von der Darstellung ausgeschlossen. Es können nur Dokumente angezeigt werden, deren Relevanzwerte sich in einem bestimmten Wertebereich befinden.

3. Der Dialog *FilterOptionsDialog* wird geöffnet. Hier wird vom Benutzer die Aktivierung oder Deaktivierung angebotener Filter oder die Eingabe der zu dem Filter gehörenden Parameter gefordert. Nach dem Beenden dieser Dialoge wird jeweils die Menge der anzuzeigenden Dokumente neu ermittelt und die grafische Repräsentation aktualisiert.

### Interaktion mit Dokumenten

Hier wird der Anwendungsfall „Auswahl der Ergebnismenge“ aus dem Abschnitt 4.4.2 abgebildet. Die Abbildung 5.14 zeigt den entsprechenden Ablauf.

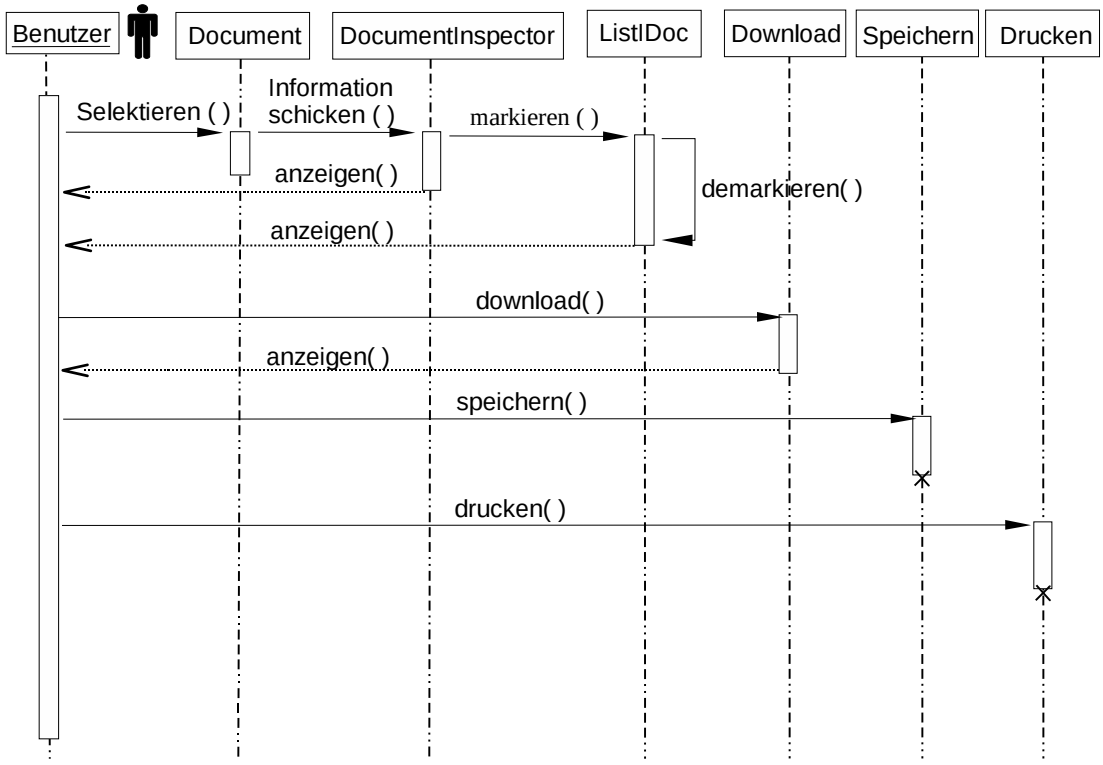



Abbildung 5.14: Sequenzdiagramm: Interaktion mit Dokumenten

1. Wenn ein Dokument durch Mausklick vom Benutzer selektiert wird, werden die zum Dokument verfügbaren Detailinformationen in einem integrierten Fenster *DocumentInspector* angezeigt.

2. Der Benutzer hat die Möglichkeit, das selektierte Dokument als „interessant“ oder „uninteressant“ zu kennzeichnen.

3. So werden die Dokumente, die als „interessant“ markiert sind, von der Klasse *ListIDoc* archiviert und in einem weiteren integrierten Fenster angezeigt. Der Benutzer kann auch bei einem bereits als „interessant“ markierten Dokument, das in der Liste eingetragen ist, diese Kennzeichnung wieder aufheben. Nach der Demarkierung wird das Dokument aus dieser Liste wieder entfernt.

4. Mit dem Aufruf vom *Download* kann der Benutzer ein Dokument anfordern, daraufhin wird es im voreingestellten Browser angezeigt.

5. Das Dokument kann dann gespeichert oder gedruckt werden und unterliegt den Funktionalitäten, die der jeweils verwendete Browser bietet.

## 5.9 Die Schnittstelle zu Suchmaschinen

Das System soll so konzipiert werden, daß es sowohl direkt an eine Suchmaschine gekoppelt werden kann (online) als auch von einer Suchmaschine gespeicherte Ergebnisse verarbeiten kann (offline). Im ersten Fall würde das Visualisierungsmodul über eine Option der Suchmaschine direkt aus dem Web-Browser gestartet. Das Suchergebnis muß in einer strukturierten Form an das Modul übergeben werden.
Im Offline-Betrieb speichert die Suchmaschine das Ergebnis zunächst in eine Datei. Diese kann zu einem beliebigen späteren Zeitpunkt eingelesen und zur Darstellung des Informationsraumes genutzt werden. Eine Verbindung zum Internet ist zu diesem Zeitpunkt nicht mehr notwendig.
Bei beiden Varianten werden vom Visualisierungsmodul dieselben Informationen über das Suchergebnis benötigt:

- die verwendeten Schlüsselwörter,
- die gefundenen Dokumente,
- die Relevanzwerte jedes Dokuments bezüglich der Schlüsselwörter.

Nicht zwingend notwendig, aber aus Gründen der Nutzerfreundlichkeit sinnvoll können weitere Informationen, wie z.B. das Erzeugungsdatum, der Benutzername oder die Bezeichnung der verwendeten Suchmaschine sein.
Für beide beschriebenen Arten der Anbindung müssen die Informationen in einem bestimmten Datenformat übergeben werden, das sowohl der Suchmaschine als auch dem Visualisierungsmodul bekannt sein muß. Dieses Datenformat soll wohlstrukturiert und erweiterbar sein. Für diese Zwecke geeignet ist die *Extensible Markup Language (XML)* [Ray01, Ste01, Beh98]. Dabei werden die Informationen in selbst definierten Elementen strukturiert. Die für die Übermittlung eines Suchergebnisses von der Suchmaschine an das Visualisierungsmodul notwendigen Informationen werden in den folgenden XML-Tags codiert:

| | |
|---|---|
| $< search\_results >$ | Beginn/Ende des Suchergebnisses |
| $< creation\_date >$ | Datum der Erzeugung des Suchergebnisses |
| $< keyword\_list >$ | Beginn/Ende der Liste der Schlüsselwörter |
| $< keyword >$ | Eintrag eines Schlüsselwortes |
| $< document\_list >$ | Beginn/Ende der Dokumentenliste |
| $< document >$ | Beginn/Ende einer Dokumentenbeschreibung |
| | *title* – der Titel des Dokuments |
| | *author* – der Autor des Dokuments |
| | *published* – der Ort der Veröffentlichung |

| | |
|---|---|
| | *url* – die Adresse des Dokuments |
| | *date* – das Erzeugungsdatum des Dokuments |
| | *size* – die Größe des Dokuments |
| $< relevances >$ | Beginn/Ende der Relevanzen zu den Schlüsselwörtern |
| $< rvalue >$ | Relevanzwert |

Der XML-Standard bietet die Möglichkeit, die Struktur eines XML-Dokuments zu definieren. Dies kann innerhalb oder außerhalb der XML-Datei vorgenommen werden. Externe Deklarationen sind vor allem dann sinnvoll, wenn mehrere XML-Dateien mit ein und derselben Struktur erzeugt werden sollen. Eine solche Strukturierungsdefinition heißt *Document Type Definition (DTD)*. Die Festlegung der Struktur für die oben aufgeführten Informationen ist im Listing 1 im Anhang E gezeigt. Die Syntax einer DTD ist in den Werken zu XML beschrieben. Zu jedem Dokument *müssen* zumindest der Titel und die URL verfügbar sein, dies ist mit dem Parameter $\#REQUIRED$ definiert. Das Attribut *valid* beschreibt die Gültigkeit eines Dokuments nach der Filterung. Dies ist zwar für das Einlesen eines Suchergebnisses nicht relevant (daher wird dieses Attribut mit „yes“ initialisiert), jedoch für das Einlesen von gespeicherten Arbeitsständen. Die anderen Attribute sind zwar für die gegebenenfalls durchzuführenden Filteroperationen notwendig, jedoch nicht zur Identifikation eines Dokuments. Ein von der Suchmaschine erzeugtes und in der oben beschriebenen XML-Struktur formatiertes Ergebnis besitzt das im Beispiel in Listing 2 (Anhang E) gezeigte Erscheinungsbild. Dieses Beispiel zeigt das Ergebnis einer Suche mit vier Schlüsselwörtern. In der Dokumentenliste sind zwei Dokumente enthalten. Das Attribut *valid* ist – wie oben beschrieben – nicht ausgeführt, daher werden die Dokumente beim Einlesen als „gültig“ gekennzeichnet.
Eine weitere Anforderung an das Visualisierungsmodul ist die Möglichkeit für den Benutzer, die Aufarbeitung des Suchergebnisses unterbrechen und zu einem anderen Zeitpunkt fortsetzen zu können. Dazu ist es notwendig, neben den Schlüsselwörtern, den Dokumenten und den Relevanzen auch den gesamten Systemzustand zu speichern. Dazu gehören folgende Informationen:

- Informationen über den Zustand aller vorhandenen Filter (aktiv/inaktiv),
- die aktuellen Parameter und Optionen zu jedem Filter,
- die eingestellten Relevanz-Schwellwerte,
- die Liste der vom Benutzer als „interessant“ eingestuften Dokumente,
- der aktuell ausgewählte Renderer,
- das Datum der Erzeugung des Zwischenergebnisses, bzw. der letzten Modifikation.

Diese Informationen werden auf die folgenden XML-Tags und Attribute abgebildet:

| | |
|---|---|
| $< modification\_date >$ | Datum der letzten Modifikation |
| $< filter >$ | Beginn/Ende der aktuellen Filtereinstellungen |
| $< author\_filter >$ | Einstellungen für den Autoren-Filter |
| | *active*– Filter aktiv/inaktiv |
| | *type*– Interpretation als Inclusions- oder Exclusionsliste |
| $< author\_list >$ | Beginn/Ende der Liste der ein- oder auszuschließenden Autoren |
| $< author\_name >$ | Eintrag eines Autors |
| $< public\_filter >$ | Einstellungen für die Filterung nach Veröffentlichungen |
| $< publication\_list >$ | Liste der ein- oder auszuschließenden Veröffentlichungen |
| | *active*– Filter aktiv/inaktiv |

| | |
|---|---|
| | *type*– Interpretation als Inclusions- oder Exclusionsliste |
| $< location >$ | Name einer Veröffentlichung |
| $< date\_filter >$ | Einstellungen für den Datumsfilter |
| | *active*– Filter aktiv/inaktiv |
| | *mindate*– untere Datumsgrenze |
| | *maxdate*– obere Datumsgrenze |
| $< size\_filter >$ | Einstellungen für den Größenfilter |
| | *active*– Filter aktiv/inaktiv |
| | *size*– Größe in Bytes |
| | *type*– Interpretation als Untergrenze/Obergrenze |
| $< redundancy\_filter >$ | Zustand des Relevanz-Filters |
| | *active*– Filter aktiv/inaktiv |
| $< threshold\_filter >$ | Einstellungen der Relevanz-Schwellwerte |
| | *active*– Filter aktiv/inaktiv |
| $< threshold\_list >$ | Liste der Schwellwerte |
| $< thvalue >$ | Schwellwert |
| $< renderer >$ | Name des eingestellten Renderers |
| $< idoclist >$ | Liste der interessanten Dokumente |

Die DTD im Listing 3 im Anhang E zeigt die Strukturierung der Informationen für das XML-Format. Der erste Teil entspricht der DTD für Suchergebnisse, zusätzlich sind die Filter- und Renderereinstellungen sowie die Liste der vom Benutzer zum Zeitpunkt des Speicherns als „interessant“ eingestuften Dokumente aufgenommen. Die ergänzenden Elemente werden ebenfalls im Anhang E in einem Beispiel für eine XML-Datei gezeigt (Listing 4), die als Zwischenergebnis während einer Sitzung des Visualisierungsmodules erzeugt wird.

## 5.10 Fazit

Der Entwurf eines Moduls zur Visualisierung von durch Suchmaschinen ermittelten Informationsräumen ist somit von der Struktur, über Klassendefinitionen, Interaktionen bis hin zu externen Schnittstellen statisch und dynamisch vollständig beschrieben. Wie im Kapitel 4 erläutert, umfaßt die Objektorientierte Softwareentwicklung die Analyse, den Entwurf und die Verifizierung und ist ein zyklischer Prozeß. So wurde auch der hier dargelegte Entwurf durch eine Probe-Implementierung unterstützt, verbessert und bestätigt. Diese Implementierung wird im Kapitel 6 vorgestellt.

# Kapitel 6

# Prototyp

Dem Schema des objektorientierten Entwicklungsprozesses entsprechend wurde der im Kapitel 5 vorgestellte Entwurf fortlaufend durch eine Probeimplementierung verifiziert und verbessert. Dieses Kapitel beschreibt die Implementierung und enthält Betrachtungen zur Auswahl der Software-Umgebungen.

## 6.1 Auswahl des Grafiksystems

Bei der Entwicklung eines Systems zur Visualisierung spielt die Auswahl des zu verwendenden Grafiksystems eine zentrale Rolle. Die Anforderungen an ein Grafiksystem in Bezug auf die Ausgabeelemente, die technischen Voraussetzungen und die Benutzerinteraktionen sollen in diesem Kapitel untersucht werden. Von den derzeit verfügbaren Systemen kommen die beiden Grafikschnittstellen *OpenGL* und *Java3D* sowie die Beschreibungssprache *VRML* für den Einsatz in einem Modul zur Visualisierung von Informationsräumen in Betracht.

### 6.1.1 Anforderungen an das Grafiksystem

Das Grafiksystem soll dem Benutzer erlauben, sofort einen Überblick über die Struktur seines Informationsraumes in Form einer *3D-Szene* zu erhalten und somit schnell Zusammenhänge zu erkennen. Darüberhinaus soll er sich durch den Raum bewegen und lokal detaillierte Informationen erhalten können. Zum Erreichen dieser Ziele sind folgende Kriterien von Bedeutung:

- Da das System für die Zusammenarbeit mit dem Internet vorgesehen ist, ist die Plattformunabhängigkeit notwendig.
- Die darzustellende Informationsmenge - und damit die Anzahl der generierten grafischen Objekte – kann erhebliche Ausmaße annehmen. Die Rendering-Stufe muß daher eine gewisse Performance bieten.
- Das System soll Möglichkeiten bereitstellen, um Benutzereingaben bearbeiten zu können.
- Die Manipulation der grafischen Szene durch den Benutzer muß unmittelbar visuell erkennbar sein.
- Das Sytem muß stabil und robust gegenüber Benutzerfehlern sein.

### Anforderung an die Ergebnisdarstellung

Jedes Grafiksystem verfügt über eine definierte Menge von Ausgabeelementen (auch *Primitiven* genannt), mit deren Hilfe geometrische Objekte für die grafische Darstellung spezifiziert werden können. Jedes Ausgabeelement kann wiederum mit Attributen versehen werden, die das Erscheinungsbild beeinflussen. Dazu können die folgenden Betrachtungen gemacht werden:

- Hinsichtlich der Ausgabeelemente bestehen keine außergewöhnlichen Anforderungen, die Präsentationen der Dokumente müssen vom Grafiksystem effizient und in guter Qualität dargestellt werden. Der Vorrat üblicher Primitiven wie Punkte, Polylines und Polygone – evtl. ergänzt durch einige high-level-Primitiven wie Quader oder Kugel ist ausreichend.
- Wichtig ist, daß den Elementen Eigenschaften zugeordnet werden können. Dazu gehören z.B. Farbe und Transparenz. Diese Attribute werden unter Umständen auch von der Mapping-Stufe verwendet, um bestimmte Parameter auf die Objekte abzubilden.
- Es ist zu bedenken, daß die Zahl der darzustellenden Objekte groß sein kann, was in einer großen Anzahl zu zeichnender Polygone resultiert. Trotzdem soll Interaktion möglich sein. Das bedeutet, daß die gesamte grafische Szene ständig neu gezeichnet werden muß.
- Die Darstellung beleuchteter Polygone benötigt viel Zeit. Daher sollte das Grafiksystem die Möglichkeit bieten, zwischen verschiedenen Darstellungsmodi umzuschalten, z.B. von der Polygondarstellung in die schnellere Darstellung mittels Drahtgitter.

### Anforderungen an die Interaktion

Geht man davon aus, daß die Dokumente als grafische Objekte in einem dreidimensionalen kartesischen Koordinatensystem dargestellt werden, und ihre Position Aufschluß über ihre Relevanz geben kann, kann man sich folgende Interaktionsmöglichkeiten vorstellen:

**Grafische Manipulation des gesamten Informationsraumes:**
Mit den folgenden Manipulationen ist es möglich, die Szene aus jeder gewünschten Richtung zu betrachten und sogenannte „Kamerafahrten“ durchzuführen. Jeder Teil des Raumes kann also durch den Benutzer erreicht werden. Zu diesen Manipulationen zählen:

- die Translation und Rotation der grafischen Objekte,
- die Festlegung des Kamerastandpunktes (view point),
- die Festlegung der Blickrichtung mittels eines Referenzpunktes,
- die Definition des Öffnungswinkels der Kamera,
- Bestimmung der Position im Raum (Locator).

**Erzeugen und Manipulieren einer Teilmenge:**
Es sollte möglich sein, eine Teilmenge von grafischen Objekten zu selektieren und anschließend grafisch zu manipulieren (z.B. Ausblenden).

**Manipulation von einzelnen Objekten(Dokumenten):**
Das System muß die Selektion einzelner Objekte („Picken“/Selektieren) zulassen. Mit der Repräsentation eines Objekts sollten eine Reihe von Interaktionen möglich sein:

- Anzeige detaillierter Informationen zum Dokument (Titel, Autor, ...),
- evtl. Download des Dokuments und Anzeige des Inhalts (zusätzlich Möglichkeit zum Speichern der URL oder des gesamten Dokuments),

- Markieren des Dokuments (als interessant oder uninteressant),
- Löschen oder Ausblenden eines Dokuments.

Bei der Festlegung der Interaktionsparadigmen ist wiederum auf Plattformunabhängigkeit zu achten, es ist z.B. wenig sinnvoll, alle Manipulationen auf eine Drei-Tasten-Maus abzustimmen, obwohl andere Plattformen nur Ein-Tasten-Geräte kennen.

### 6.1.2 Die Grafik-Schnittstelle *OpenGL*

Die Grafik-Bibliothek OpenGL(*open graphics library*) ist aus der IRIS GL der Firma Silicon Graphics (SGI) entstanden. Diese war seit Mitte der 80er Jahre die Programmierschnittstelle für die Hardware von Silicon Graphics unter deren Betriebssystem IRIX [Bar96b]. Später wurde beschlossen, einen plattformunabhängigen Standard für ein 2D/3D-Application Programming Interface (API) zu entwerfen und darin die mit der IRIS GL gemachten Erfahrungen einfließen zu lassen. Mitte 1992 wurde dann das Architecture Review Board gegründet, ein Zusammenschluß der Vertreter nahmhafter Firmen wie SGI, IBM, DEC, Intel und Microsoft, HP und anderer, Die Aufgabe des ARB ist die Kontrolle und Verwaltung des OpenGL-Standards. Mit seiner Gründung wurde auch die Spezifikation von OpenGL 1.0 offiziell verabschiedet. Mitte 1995 wurde OpenGL 1.1 spezifiziert, zur Zeit ist die Version 1.4 aktuell, die als Vorgänger zur Version 2.0 gilt. Die Kompatibilität mit den Vorgängerversionen blieb gewährleistet, es wurden Fehler und Widersprüche beseitigt sowie neue Fähigkeiten (besonders im Bereich der Texturverarbeitung) hinzugefügt.
OpenGL versteht sich als eine Software-zu-Hardware-Schnittstelle. Alle Funktionen dieser Schnittstelle lassen sich sowohl durch Software implementieren, als auch durch geeignete Hardware – falls verfügbar – beschleunigen. Der erste Absatz von OpenGL in [Fra97] beschreibt das grundlegende Prinzip von OpenGL wie folgt:

> OpenGL ist ein Software-Interface zur Hardware. Zweck ist die Darstellung von zwei-und dreidimensionalen Objekten mittels eines Bildspeichers. Diese Objekte sind entweder Images, die aus Pixeln zusammengesetzt sind, oder geometrische Objekte, die durch Vertices (Raumpunkte) beschrieben sind.

Insgesamt umfaßt das OpenGL-Interface ca. 120 Basisfunktionen, die zur Spezifikation von Objekten und Operationen benötigt werden. OpenGL ist client- serverfähig. Die Protokolle, welche OpenGL-Befehle übermitteln, sind identisch. Dadurch können OpenGL-Programme im Netzwerk ablaufen, in denen Client und Server Rechner verschiedenen Typs sind. Dies erreicht man, indem die Befehle zum Ablauf von Window-Tasks oder zur Interaktion mit dem Benutzer nicht in dem OpenGL-Programm implementiert werden, sondern diese Aufgaben direkt von der Fensteroberflächenprogrammierung der entsprechenden Hardware übernommen werden. Da OpenGL unabhängig von Window-Systemen und von Betriebssystemen spezifiziert wurde, ist diese Bibliothek *nur* für das Rendering einer Szene mit zwei oder dreidimensionalen Objekten ausgelegt [Bar96b]. Die Spezifikation kann aufgrund dieser Unabhängigkeit keinerlei Festlegungen zur Anbindung an Fenstersysteme oder zur Ereignisbehandlung treffen. Zu diesem Zweck existieren Zusatzbibliotheken mit folgender Funktionalität:

- Öffnen und Schließen von Fenstern,
- Bereitstellung der benötigten Ressourcen eines Fensters (Anzahl der Buffer für das Fenster, Zahl der Bit pro Pixel, Ausgabemodus),
- Bestimmung der Größe des Fensters und Festlegung seiner Position auf der zentral vom Window-Manager verwalteten Zeichenfläche,

- Behandlung von Ereignissen des Window-Managers (z. B. Veränderung der Größe, Maus-Interaktionen, Verlagerung in den Vorder- oder Hintergrund).

Diese Aufgaben müssen auf jeweils systemspezifische Art und Weise erledigt werden. Daher existieren Bibliotheken für die verschiedenen Fenstersysteme, z. B. GLX (OpenGL Extension to the X Window System) für X11 oder WGL (windows graphics library) für MS-Windows. Der OpenGL-Standard ist sprachunabhängig definiert, es wird ausschließlich die Funktionalität und das API beschrieben. Entsprechende Sprachbindungen sind in gesonderten Dokumenten standardisiert.
Die Arbeitsweise ist im OpenGL-Ausführungsmodell beschrieben. Der diesem Modell folgende Rendering-Prozeß beginnt mit mathematisch beschriebenen Objekten und endet mit deren zweidimensionaler Darstellung in einem Bildspeicher. Die grafischen Objekte werden mit Hilfe einfacher Ausgabeprimitiven (z. B. Punkt, Linie, Polygon, Polygonfläche) beschrieben. Dreidimensionale Körper wie Kugel oder Zylinder gehören nicht zum OpenGL-Funktionsumfang. OpenGL ist streng prozedural. Das heißt, daß nicht das Aussehen eines Objekts beschrieben wird, sondern wie die aktuell zu zeichnenden Primitive darzustellen sind. Die Beschreibung der OpenGL-Ausgabeelemente erfolgt mittels zwei- oder dreidimensionaler *Vertices*. Diese repräsentieren einen Punkt im Raum, z.B. den Endpunkt einer Linie oder die Ecke eines Polygons. Sie müssen entsprechende Koordinaten enthalten, und können darüberhinaus eine Farbdefinition, einen Normalenvektor, Texturkoordinaten usw. besitzen [Bar96b]. Diese Vertices durchlaufen die *Rendering-Pipeline*. Wie Vertices zu Primitiven zusammengesetzt werden und in den Frame Buffer gezeichnet werden, wird durch eine Vielzahl von Einstellmöglichkeiten kontrolliert. An dieser Stelle soll ein Überblick über den Ablauf bzw. die Steuerung der Modellierung der darzustellenden Objekte gegeben und mit Hilfe der Abbildung 6.1 erläutert werden [Cla97, Bar96b].

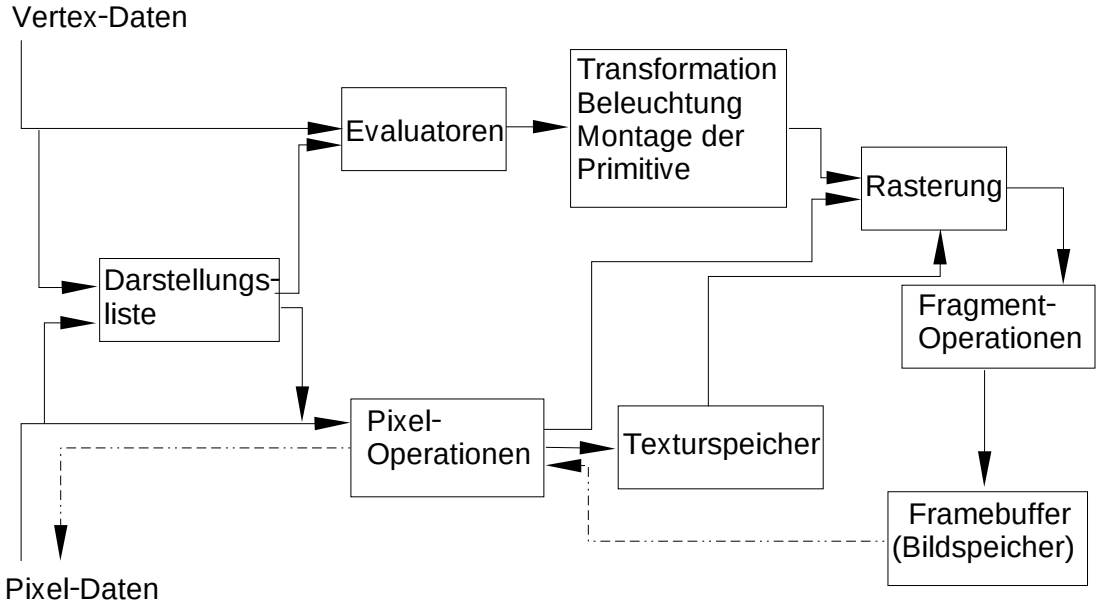



Abbildung 6.1: Schema der Umsetzung von Geometrie und Bilddaten [Cla97]

- Die Eckpunktinformationen durchlaufen eine Pipeline, in der sie transformiert, beleuchtet und sonstigen Operationen unterzogen werden.

- Die *Darstellungsliste* ist eine Zusammenfassung von grafischen Befehlen zu einer Einheit mit eigenem Namen. Somit ist es beispielsweise möglich, komplexere grafische Objekte aus den Primitiven der OpenGL zusammenzubauen.

- Freiformkurven oder -flächen werden analytisch spezifiziert. In diesem Fall ist die Kurven- und Flächengeometrie auf normale Vertex-Daten zu approximieren. Sogenannte *Evaluatoren* übernehmen diese Umwandlung.
- Alle Arten von Transformationen, wie zum Beispiel Modelltransformation oder Beleuchtungsrechnung, werden in der nächsten Stufe durchgeführt.
- Die Primitiven wie z.B. Liniendicke oder Linientyp werden dann in der Stufe der *Rasterisierung*, auf ein Raster abgebildet, dessen Ausmaße dem Fenster entsprechen.
- Das Ergebnis dieses Vorganges heißt *Fragment*, das als Rasterquadrat angesehen werden kann.
- Nach der Rasterisierung in Fragmente finden weitere Operationen statt, die sogenannten *Fragmentoperationen*. Hier wird z. B. die Berechnung der Textur durchgeführt.
- Am Ende der Pipeline der Fragmentoperationen werden die Fragmente in Form von Pixeln repräsentiert.

Neben Eckpunkten oder Funktionen können auch Pixeldaten bzw. Bitmaps direkt in die Darstellung eingebracht werden. Transformierte Pixeldaten fließen dann entweder direkt in die Rasterisierung ein oder werden im Texturspeicher für die Verwendung in anderen Fragmenten zwischengespeichert. Die OpenGL verfügt über eine Reihe von Features, die es erlauben, hochqualitative Bilder und Animationen mit guter Performance zu erzeugen. Dies ist im Kontext der Visualisierung jedoch nicht relevant und soll daher hier nicht untersucht werden.

### 6.1.3 Die Grafik-Schnittstelle *Java3D*

Das Java3D-API ist eine plattformunabhängige Anwendungsprogrammier-Schnittstelle, die ein Teil des Java Media API ist. Java3D kann zur Programmierung dreidimensionaler grafischer standalone-Anwendungen oder web-basierter 3D-Applets eingesetzt werden, die auf jedem unterstützten Java-2-fähigen Betriebssystem arbeiten [Sun97b]. Mit Java3D kann der Entwickler auf eine einfache Weise dreidimensionale Körper erstellen und manipulieren.
Das SUN-Projekt Java3D ist das jüngste aller 3D-API's und stammt vom Dezember 1998. Das Ziel dieses Projektes war, eine Lösung zu entwerfen, die auf den Erfahrungen der Entwicklung anderer API's (z.B. Direct3D, OpenGL, QuickDraw3D) beruht, damit Daten, bzw. virtuelle Welten im Internet visualisiert werden können.
Die Applikationen erzeugen separate grafische Elemente als Objekte und verbinden diese in einer Baumstruktur – dem *Szenengraph*. Die Applikation manipuliert diese Objekte, indem sie deren vordefinierte Methoden benutzt. Das szenengraph-basierte Programmierungsmodell von Java3D bietet einen flexiblen und einfachen Mechanismus, um komplexe 3D-Umgebungen zu rendern. Der Szenengraph besteht aus einer vollständigen Beschreibung der Szene. Dazu gehören geometrische Daten, die Attributinformation und die Viewing-Information, die erforderlich sind, um die Szene von einem bestimmten Blickpunkt zu rendern [Sun97a]. Das Konzept von Java3D beinhaltet[Sun97a]:

- eine Erweiterung der Programmiersprache Java, Nutzung des objektorientierten Programmier-Modells,
- ein „high level - Grafik-Interface“, der Benutzer muß sich nicht um den Ablauf des Rendering-Prozesses kümmern, sondern kann sich auf die Modellierung konzentrieren,
- Anordnung der Objekte in einem Szenengraph
- über die eingebauten Primitive hinaus werden sogenannte *Loader* angeboten, die externe Dateiformate einlesen und darstellbar machen können (z. B. Loader für VRML-Dateien).

Heute verfügbare Java3D - Implementationen benutzen intern andere Low-Level-API's (häufig OpenGL) und deren Rendering-Fähigkeiten. Die Einordnung von Java3D in eine Programmierumgebung ist in der Abbildung 6.2 dargestellt.
Ein Szenengraph organisiert und steuert das Rendering seiner Objekte. Der Java3D-Renderer zeich-

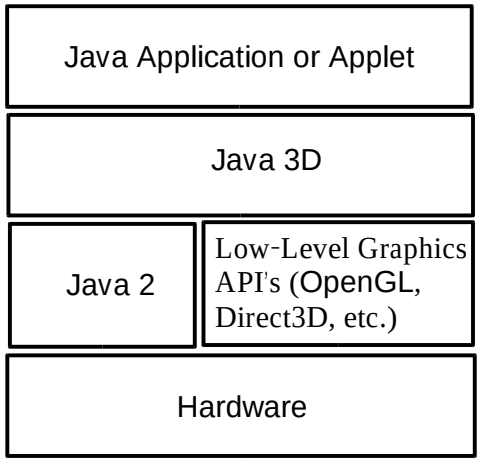



Abbildung 6.2: Java3D-Umgebung [Sun97a]

net einen Szenengraph in einer festgelegten Weise. Er kann ein Objekt unabhängig von anderen Objekten zeichnen. Java3D kann eine solche Unabhängigkeit gestatten, da seine Szenengraphen eine besondere Form haben und die Zustände nicht unter den Zweigen eines Baumes geteilt werden [Sun97a]. Die Hierarchie des Szenengraphs unterstützt eine natürliche räumliche Gruppierung der geometrischen Objekte, die an den Blättern des Graphen gefunden werden. Interne Knoten wirken als Gruppe ihrer Kinder. Ein Gruppen-Knoten definiert ebenso einen abgegrenzten Raum, der alle Geometrien enthält, die durch seine Nachkommen definiert sind. Das Prinzip der räumlichen Gruppierung erlaubt eine effiziente Implementation vieler Verfahren wie zum Beispiel Kollisionserkennung, *view frustum culling* und *occlusion culling*. Die Verbindung zwischen den Knoten ist

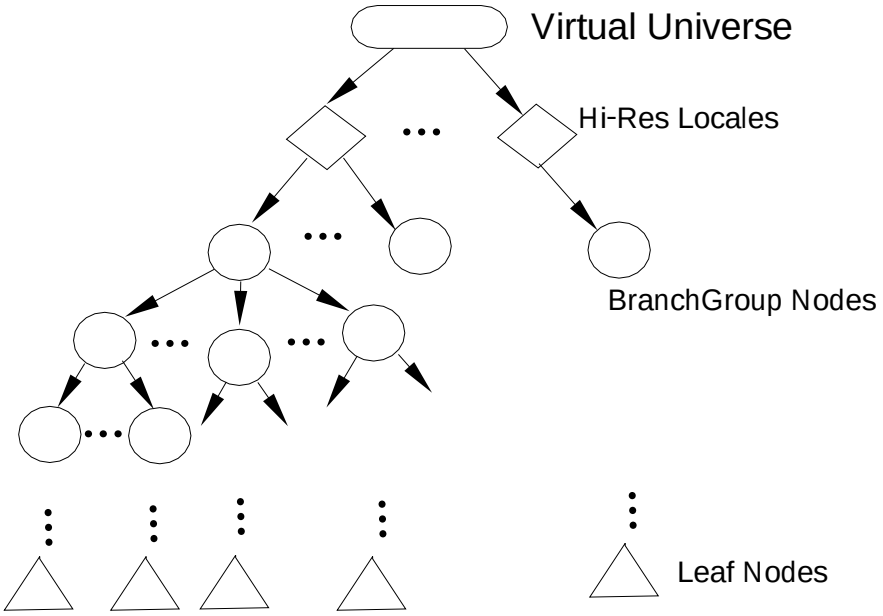



Abbildung 6.3: Der Szenengraph als Directed Acyclic Graph [Sun97a]

eine gerichtete Verwandschaft (Eltern-Kind-Beziehung). Java3D unterscheidet sich von anderen szenengraph-basierten Systemen dadurch, daß der Szenengraph keine Zyklen enthalten darf. Daher ist ein Java3D-Szenengraph ein gerichteter aperiodischer Graph (DAG – **D**irected **A**cyclic **G**raph), wie die Abbildung 6.3 zeigt.
Java3D kennt drei unterschiedliche Ausführungsmodi (*Rendering Modi*) um dreidimensionale Objekte, bzw. Szenen darzustellen. Dies sind der *Immediate Mode*, der *Retained Mode* und der *Compiled-Retained Mode* [SUN99]. Die Unterschiede dieser Modi liegen in der Flexibilität bei der Darstellung, der Manipulation der Szene und der Ausführungsgeschwindigkeit.

**Der Immediate-Modus**

Der *immediate-mode* bietet die Möglichkeit, den Szenengraphen fast vollständig zu ignorieren, wodurch die größtmögliche Freiheit bei der Gestaltung einer dreidimensionalen Szene geboten wird. Für die Benutzung dieses Modus ist nur ein minimaler Szenengraph zum Betrachten der Szene nötig. Der Entwickler muß sich um keinerlei hierarchische Strukturierung bemühen und kann seine geometrischen Objekte beliebig im Raum platzieren.

**Der Retained-Modus**

Der *Retained-Modus* bietet sowohl eine große Flexibilität, als auch eine gesteigerte Darstellungsgeschwindigkeit im Vergleich zum Immediate-Modus. In diesem Modus wird ausschließlich der Szenengraph verwendet. Der Szenengraph enthält sämtliche für die Szene relevanten Objekte und Eigenschaften. Seine baumartige Struktur ermöglicht es, zur Laufzeit Veränderungen an den Objekten vorzunehmen oder interaktiv ihre Verhaltensweisen zu verändern. In diesem Modus sind dem Java3D-System die Objekte und deren Kombination zu zusammengesetzten Objekten oder Szenengraphen bekannt, womit übergreifende Optimierungen möglich werden.

**Der Compiled-Retained-Modus**

Im *Compiled-Retained Modus* sind weitergehende Optimierungen am Szenengraphen möglich. Der Entwickler kann ein Objekt von Java3D kompilieren lassen, wodurch er nur noch minimale Zugriffsmöglichkeiten auf die interne Strukur dieses Objekts oder Szenengraph hat. Über sogenannte *Capability-Flags* muß der Entwickler exakt angeben, welche Attribute und Funktionalitäten eines Objekts zur Laufzeit noch veränderbar sein sollen [Wir01]. Die anderen Objekte sind nicht mehr zugänglich, und der Compiler versucht, möglichst umfassende Optimierungen auszuführen, um die Darstellungsperformance zu steigern. So können zum Beispiel mehrere geometrische Objekte zu einer Geometrie zusammengefaßt werden. Es könnte ebenso effizienter sein, ein Objekt in kleinere Teile aufzubrechen und diese in neuer Konstellation zu einem neuen Objekt zusammenzufügen [Ste98].

Der Java3D-Renderer enthält alle grafischen Zustandsänderungen, die in einem direkten Pfad von der Wurzel zu einem Blatt-Objekt während des Zeichnens des Blatt-Objekts durchgeführt wurden. Java3D stellt diese Semantik für den *retained-mode* und den *compiled-retained-mode* bereit. Der Zustand eines Blatt-Knotens ist durch die Knoten auf einem direkten Pfad zwischen der Wurzel des Szenengraphen und dem Blatt definiert. Da der Grafik-Kontext eines Blattes nur von einem linearen Pfad zwischen der Wurzel und diesem Knoten abhängt, kann der Java3D-Renderer entscheiden, den Szenengraph in der jeweils gewünschten Richtung zu durchlaufen. Der Renderer kann den Szenengraph von links nach rechts und von oben nach unten oder auch parallel durchlaufen. Die einzige Ausnahme für diese Regel sind räumliche beschränkte Attribute wie zum Beispiel Lichtquellen und Nebel. Diese Eigenschaft steht im Gegensatz zu vielen älteren szenengraph-basierten APIs (einschließlich *PHIGS* und *SGI-Inventor*), bei denen - wenn sich ein Knoten über oder auf der linken Seite eines Knotens seinen Zustand ändert, das einen Einfluß auf alle Knoten rechts

oder darunter hat. Durch die Speicherung der meisten Zustands-Attribute in speziellen Knoten ist paralleles Rendering möglich.

### 6.1.4 Die Beschreibungsprache *VRML*

Der Grundstein zur Entwicklung der *Virtual Reality Modeling Language (VRML)* wurde auf der ersten internationalen Konferenz zum WWW (World Wide Web) 1994 in Genf gelegt. Dort fand eine Sitzung mit dem Thema „Virtual Reality Markup Language and the World Wide Web“ statt. Auf dieser Sitzung wurde die Vorstellung von einem plattformunabhängigen Standard für 3D-Anwendungen entwickelt und der Begriff *VRML* geprägt [Car97a]. Das Wort „Markup“ wurde später durch „Modeling“ ersetzt. Die VRML-Gruppe wählte das Open Inventor- [1] ASCII Format von Silicon Graphics als die Grundlage für VRML. Das Inventor-File-Format unterstützt eine komplette Beschreibung von dreidimensionalen Szenen mit Geometrie, Beleuchtung, Materialien, User-Interface-Widgets und Viewer [R.98].
Am Ende des Jahres 1995 wurde die erste Spezifikation VRML 1.0 als internationaler Standard für die Darstellung von dreidimensionalen Welten präsentiert, welcher alle Elemente enthielt, die benötigt werden, um dreidimensionale Objekte in statischen Welten zu definieren. Die Interaktion mit diesen Welten war nur möglich, indem die Links, die wiederum auf andere Welten oder gewöhnliche HTML-Seiten verweisen, verfolgt wurden. Dies war jedoch nicht ausreichend, denn Elemente für eine Interaktion in einer Szene in dieser statischen Welt waren nicht vorhanden. Demzufolge wurde weltweit in Diskussiongruppen an einer neuen Spezifikation gearbeitet.
Um die Weiterentwicklung des Standards voranzutreiben, wurde die VRML Architektur-Gruppe (VAG) [2] gegründet. Deren Aufgabe ist es, die Übereinstimmung in der VRML-Gesellschaft zu fördern, um einen völlig neuen, interaktiven Standard für 3D-Welten zu entwickeln. Im August 1996 wurde die neue Version VRML 2.0 der VRML-Architecture Group zum internationalen Standard verabschiedet. Der Entwurf wurde mit Unterstützung der Firmen Silicon Graphics Inc., Sony Research und Mitra entwickelt [Mü98, Dä98]. Er basiert auf der Sprache *Moving Worlds* von Silicon Graphics. Diese ermöglicht Animationen und sich selbständig bewegende Objekte. VRML 2.0 wurde als ISO Standard (ISO/IEC 14772-1:1997) ratifiziert. Die Kurzschreibweise dieses Standard lautet VRML97.
Zur Darstellung von Szenen, die mittels VRML beschrieben sind, ist ein VRML-Viewer notwendig. SUN arbeitet in der Arbeitsgruppe „VRML-Java3D“ eng mit dem Web3D-Konsortium zusammen, um unter anderem einen effizienten VRML Browser zu entwickeln, der komplett auf Java3D basiert [IX 98]. Aus dieser Zusammenarbeit könnte eine enge Verbindung zwischen Java3D und VRML entstehen.
Eine VRML-Datei ist eine aus reinem ASCII-Text bestehende Beschreibung der zu generierenden dreidimensionalen Welt. VRML-Dateien können wie HTML-Dateien sowohl im Online-Bereich des WWW als auch in Offline-Bereich zur Unterstützung von Anwendungen genutzt werden, die dreidimensionale Visualisierungen integrieren [Car97b].
Die Abbildung 6.4 beschreibt die Textform des VRML-Szenengraphes. Der Kopf dieser Datei besteht aus einer verbindlichen Informationszeile mit dem Text #VRML V2.0 <utf8> [comment] <line terminator> . Darin wird die Information vermittelt, in welcher Version des Standards der Programmtext abgefaßt worden ist. In VRML 2.0 sind internationale Zeichensätze gemäß der ASCII-Erweiterung UTF-8 erlaubt. In der Abbildung 6.4 wird eine rote Kugel mit dem Radius 2.5 erzeugt, die um drei Einheiten in X-Richtung verschoben wird. Die Knoten sind fettgedruckt dargestellt.

[1] Open Inventor ist ein objektorientiertes Werkzeug, um interaktive 3D grafische Applikationen zu entwickeln.
[2] http://vag.vrml.org/

```
#VRML V2.0 utf8
Transform {
        Translation 3 0 0
        Children [
            DEF Kugel Shape {
                                    Appearance Appearance {
                                            Material Material {
                                                                    diffuseColor 1 0 0
                                    }
                                     geometry Sphere {
                                             radius 2.5}
                                     }
             ]
}
```

Abbildung 6.4: Eine VRML-Datei

Eine dreidimensionale Szene wird in VRML durch einen Szenengraph beschrieben. Der Szenengraph ist eine Ansammlung von Knoten (Nodes), die hierarchisch aufgebaut sind. Es ist ein gerichteter azyklischer Graph. In der Hierarchie sind sogenannte Gruppen-Knoten vorhanden, die eine beliebige Anzahl von Kind-Knoten enthalten. Diese Kind-Knoten können Gruppenknoten oder Blattknoten sein. Blattknoten enthalten keine Kind-Knoten, aber oft untergeordnete Knoten. Diese treten ausschließlich in Feldern der Blattknoten auf und nie als eigenständige Knoten. Die Unterknoten können weitere Unterknoten enthalten. Der aus der oben beschriebenen VRML-Datei erzeugte Szenengraph wird in der Abbildung 6.5 gezeigt.
Der Szenengraph beginnt mit einem Gruppenknoten, in diesem Fall mit einem Transform-Knoten, der in seinen Feldern die Zuweisung von Rotationen oder Translationen ermöglicht. Als Kindknoten ist ein Shape-Blattknoten eingetragen, der wiederum Felder mit Unterknoten enthält. Das Feld *appearance* ermöglicht den Aufruf eines Appearance-Knotens. Ein Unterknoten des Appearance-Knotens ist der Material-Knoten, der in seinen Feldern u.a. Farbzuordnungen enthält. Das *geometry*-Feld beinhaltet einen geometrischen Knoten wie einen Sphere-Knoten.

### 6.1.5 Vergleich der Grafik-Schnittstellen / Beschreibungsprache

In diesem Abschnitt wird ein Vergleich zwischen den Grafik-Schnittstellen OpenGL und Java3D und der Beschreibungssprache VRML bezüglich der Grundeingeschaften und Leistungsfähigkeit vorgenommen. Als Ergebnis wird entschieden, welches 3D-System für die Entwicklung des Prototypen in dieser Arbeit ausgewählt werden kann.
In der Tabelle 6.1 werden einige Eigenschaften von OpenGL, Java3D und VRML dargestellt und erläutert. Einige Kriterien werden hier noch genauer betrachtet.

**Leistungsfähigkeit:**

*OpenGL* ist als Low-Level-API mit direktem Zugriff auf die Grafik-Hardware konzipiert. Da OpenGL als plattformunabhängige Schnittstelle spezifiziert ist, sind Anwendungen mit optimaler Leistungsfähigkeit portierbar. OpenGL arbeitet wie eine Zustandmaschine, das heißt das System befindet sich zu jedem Zeitpunkt in einem definierten Zustand, der durch Aufruf von Funktionen manipuliert werden kann. OpenGL bildet beim Rendern ein abgeschlossenes System, so daß nur von außen, über die Programmierschnittstelle, in das Rendering eingegriffen werden kann. Die Visualisierung von Informationsräumen könnte auf Basis von OpenGL als Stand-alone-Applikation realisiert werden.
*Java3D* ist eine Erweiterung der Programmiersprache Java und bietet eine hochqualifizierte

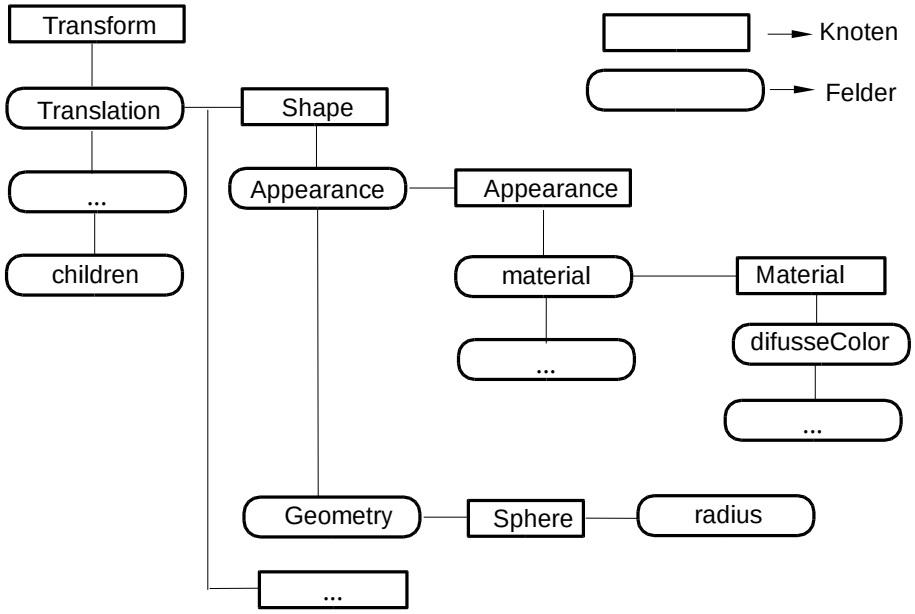



Abbildung 6.5: Darstellung eines VRML-Szenengraphen

Anwendungsprogramm-Schnittstelle für die Beschreibung von dreidimensionalen Szenen und deren Steuerung. Dabei benuzt Java3D bestehende Low-Level-Grafiksysteme, wie zum Beipiel OpenGL oder DirectX und „bedient“ sich deren Leistungsfähigkeit. Java3D ist plattformunabhängig und objektorientiert.
*VRML* ist weder ein API, noch eine Programmiersprache. Sie stellt keine Erweiterung bzw. Bibliothek für eine bestehende Programmiersprache dar. Es handelt sich um eine Beschreibungsprache, in der die Szenen mit Texteditoren oder grafischen Entwicklungsumgebungen erstellt werden. Für die Betrachtung einer VRML-Szene werden VRML-Viewer benötigt, die VRML-Quellen parsen und daraus eine dreidimensionale grafische Darstellung erzeugen können. Dadurch bleibt sie plattformunabhängig und eignet sich hervorrangend zur Gestaltung von virtuellen 3D-Welten im Internet.

**Benutzerinteraktion und Animation:**
*OpenGL* besteht nicht aus speziellen Konstrukten für das Erstellen von Animationen und verfügt über keine Interaktionsmechanismen.
In *Java3D* werden Animation, Interaktion und Kollisionserkennung durch Behavior-Objekte spezifiziert. Der Entwickler implementiert darin entsprechende Bedingungen für Kollisionen, Maus-Events, usw. Auf diese Weise kann der Szenengraph im Sinne von Interaktionen und Animationen manipuliert werden.
In *VRML* sind vordefinierte Konstrukte wie Sensoren, Routes und Interpolatoren vorhanden, womit Animationen und Interaktionen realisiert werden können. Die Sensoren erzeugen Ereignisse, sobald der Benutzer in einen bestimmten Bereich der Szene verstößt oder ein Objekt mit der Maus selektiert. Routes dienen der Erzeugung von Ereignisketten. Mit Interpolatoren können in VRML Animationen erzeugt werden, indem Start- Zwischen- und Endwerte festgelegt werden. Der VRML-Viewer berechnet die entsprechenden Zwischenschritte für eine Animation.

**Erweiterungsmöglichkeiten der Funktionalität:**
*OpenGL* bietet als Funktionsschnittstelle keinerlei Erweiterungsmöglichkeiten. Das API wird effizient benutzt, aber nicht erweitert.

| Eigenschaften | OpenGL | Java3D | VRML |
|---|---|---|---|
| Primitive (geometrische Objekte wie Punkte, Linien, Flächen, ...) | ja | ja | ja |
| Komplexe geometrische Objekte (Sphere, Quader, ...) | ja, in Zusatzbibliotheken | Utils | ja |
| Benuzerinteraktion und Animation | nein | ja | ja |
| Dynamische Objektgenerierung | ja | ja | nein, nur über Anbindung von Java/JavaScript |
| Erweiterungsmöglichkeit | nein | ja | ja |
| Plattformverfügbarkeit | alle Plattformen | alle Plattformen | VRML-Browser für alle Plattformen |

Tabelle 6.1: Ein Vergleich zwischen OpenGL, Java3D und VRML

*Java3D* ist eine Klassenbibliothek. Durch die Objektorientierung mit ihren Vererbungsmechanismen ist die Erweiterungsmöglichkeit gegeben.
Bei *VRML* stehen verschiedene Möglichkeiten zur Verfügung, die Funktionalität von VRML zu erweitern. Diese beschränken sich aber auf die mehrfache Verwendung von Teilszenen und die Erstellung benutzerdefinierter Knoten (*Prototypen*). Eine Vererbung im objektorientierten Sinne ist nicht möglich.
Eine andere Möglichkeit zur Erweiterung liegt in den sogenannten Skriptknoten, die Java- und Java Script-Code enthalten können. Auf diese Weise können weitere Elemente integriert und Ereignisse behandelt werden.

**Einsatzbereich:**

*OpenGL* bietet die größtmögliche Flexibilität in Bezug auf die Darstellung von 3D-Szenen, da sie einen niedrigen Abstraktionsgrad hat. Aus diesem Grund wird sie von vielen Programmen (auch von Computerspielen) als Rendering-Maschine verwendet.
*Java3D* wird beispielsweise in verschiedensten Arten der Visualisierung, geografischen Informationssystemen (GIS) und Computer Aided Design (CAD) eingesetzt. Java3D ist durch die Plattformunabhängigkeit von Java gut für die Entwicklung von 3D-Anwendungen für das Internet geeignet. Java3D-Programme arbeiten auf allen Plattformen, für die eine Java-2-Umgebung vorhanden ist.
*VRML* besitzt einen hohen Abstraktionsgrad und wird in verschiedenen Bereichen eingesetzt. Als Beispiele seien hier die Visualisierung von Häusern (Architektur), und die Darstellung von chemischen und biologischen Abläufen genannt. Weiterhin wurden bereits Versuche zur Veranstaltung von virtuellen Messen durchgeführt, in denen Firmen ihre Produkte vorstellen. In der Mathematik wird VRML zur Anzeige von dreidimensionalen mathematischen Objekten eingesetzt.

### 6.1.6 Fazit

*VRML* ist eine eigenständige Beschreibungsprache, die leicht zu erlernen ist, da keine programmiertechnischen Fähigkeiten vorausgesetzt werden. Der Funktionsumfang ist eingeschränkt. Wie sich beim Entwurf des Systems zur Visualisierung von Informationsräumen (siehe Kapitel 5) herausgestellt hat, werden hohe Anforderungen an Flexibilität und Interaktivität gestellt. Das Konzept einer Beschreibungssprache ist dazu nicht geeignet.
Die Fähigkeiten der *OpenGL* werden den im Abschnitt 6.1.1 genannten Anforderungen gerecht. Mit diesem Low-Level-API ist eine effiziente Umsetzung aller Anwendungsfälle möglich. Dazu sind tiefere Kenntnisse der Computergrafik Voraussetzung. OpenGL ist hauptsächlich auf Leistung optimiert, was gerade bei großen Datenmengen vorteilhaft ist.
Die Programmiersprache *Java3D* bietet die Möglichkeit, Stand-alone-Applikationen *und* Applets für das WWW zu generieren. Das bedeutet, daß der gesamte in Abbildung 2.2 dargestellte Zyklus in einer Applikation realisiert werden kann, die ohne Anpassungen auf allen Plattformen lauffähig ist.
Momentan bietet eine Applikation auf OpenGL-Basis eine bessere Performance als eine Java3D-Anwendung. Entscheidend ist der Vorteil, mit einer plattformunabhängigen Sprache arbeiten zu können, die Netzwerkfähigkeit, Integration des WWW und ein ausgereiftes 3D-Grafik-API bietet. Java3D bietet die Möglichkeit, durch Plattformunabhängigkeit und die bestehende Akzeptanz von Java als „Websprache“ einige Standards im Bereich 3D-orientierter Internetanwendungen zu setzen. Vor allem die übersichtliche Struktur des Java3D-Universums mit dem Szenengraphen-Modell und der Möglichkeit verschiedener ViewPlatforms ermöglicht eine einfache Handhabung von komplexen Welten.
Die Perfomanceprobleme, denen sich Java3D im Moment noch stellen muß, könnten sich in Zukunft durch die rapide Entwicklung der Hardware von selbst lösen. Die Koexistenz und Verflechtung mit etablierten Standards wie OpenGL und VRML können dazu beitragen, daß sich in Zukunft ein erheblicher Teil von webbasierten dreidimensionalen Anwendungen auf Java3D stützen wird.

## 6.2 Umsetzung des Entwurfes

Eine der Anforderungen an das Visualisierungsmodul besagt, daß es sowohl online als auch offline einsetzbar sein soll. Diese Anforderung läßt sich mit der Programmiersprache Java gut umsetzen, da sie auf einfache Weise das Erstellen von *Applets*[3] und stand-alone-Applikationen gestattet. Aus diesem Grund und aufgrund der Existenz eines geeigneten 3D-Grafik-APIs wurde die Probe-Implementation in Java vorgenommen.
Die Implementation folgt dem im Kapitel 5 vorgestellten Entwurf. Die darin und in den Anhängen C und D beschriebenen Programmelemente sind in dem Prototyp umgesetzt worden.

## 6.3 Die Benutzeroberfläche

Der Aufbau der Benutzeroberfläche der Probeimplementation ist sehr eng an den im Kapitel 5 beschriebenen Entwurf angelehnt. Die entworfene Struktur der Menüs wurde exakt umgesetzt. Die Abbidlung 6.6 zeigt die Oberfläche des Moduls *control*. Sie ist so organisiert, daß der Benutzer die auszuführenden Arbeitsschritte „zeilenweise“ von links oben nach rechts unten vorfindet:

1. Informationen zur zu untersuchenden Dokumentenmenge
   Die allgemeinen Informationen zum Suchergebnis werden angezeigt, dazu gehören:

[3] in einem Browser lauffähige Programme

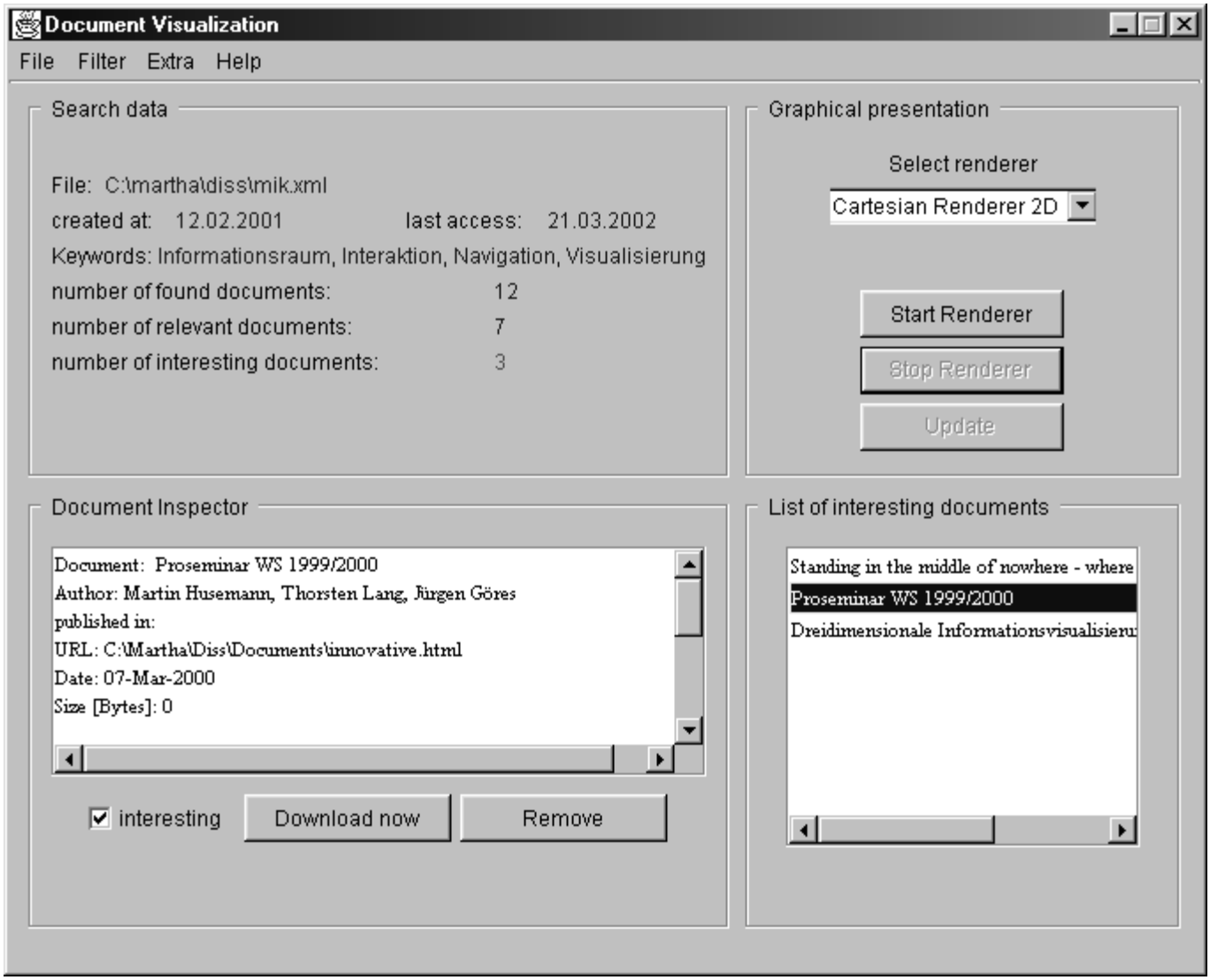



Abbildung 6.6: Benutzeroberfläche

- der Name der Datei, die entweder das Suchergebnis oder ein gespeichertes Zwischenergebnis enthält,
- das Erzeugungsdatum und das Datum des letzten Zugriffs (nur bei Zwischenergebnissen relevant),
- die Schlüsselwörter,
- die Anzahl der Dokumente in der Datenstruktur,
- die Anzahl der mit den aktuellen Filtereinstellungen vom System als „relevant“ erkannten Dokumente,
- die Anzahl der vom Benutzer als „interessant“ eingestuften Dokumente

Besonders die letzten drei der genannten Anzeigen sind für den Benutzer besonders wertvoll, da sie während der Arbeit ständig aktualisiert werden und so zu jedem Zeitpunkt Auskunft darüber geben, wie weit die Dokumentenmenge bereits eingegrenzt ist.

2. grafische Darstellung
   In einer Auswahlbox stellt der Benutzer den Renderer ein, den er gern benutzen möchte. Es ist denkbar, daß das System nur die Renderer anbietet, die für die Analyse der geladenen Datenmenge geeignet sind. Diese Entscheidungen können z. B. anhand der Dimension des Informationsraumes und der Anzahl der Dokumente getroffen werden.
   Der ausgewählte Renderer wird in dieser Sektion der Benutzeroberfläche gestartet und beendet. Die Aktualisierung der grafischen Darstellung erfolgt bewußt manuell durch den Benutzer. Eine automatische Aktualisierung müßte bei jeder einzelnen Änderung eines Parameters (z. B. Filtereinstellung) erfolgen, was insbesondere bei großen Datenmengen zu Performanz-Problemen führen kann.

3. Klassifizierung von ausgewählten Dokumenten anhand detaillierter Informationen
   Jeder der implementierten Renderer bietet die Möglichkeit, Objekte (Dokumente) zu selektieren. In dieser Sektion der Benutzeroberfläche werden alle zum aktuell selektierten Dokument verfügbaren Informationen angezeigt. Der Benutzer kann das Dokument sofort herunterladen oder vollständig aus der Datenstruktur löschen. Falls er dieses aufgrund der angezeigten Informationen als „interessant“ einstuft, kann es entsprechend markiert werden. Ein auf diese Weise klassifiziertes Dokument wird in die Liste der interessanten Dokumente aufgenommen, die in der rechten unteren Sektion angezeigt wird.

4. Ergebnisliste
   Diese Sektion enthält das Ergebnis des Aufbereitungsprozesses – die Liste der interessanten Dokumente. Jedes Dokument kann selektiert werden, worauf es wieder in der Detailansicht („Document Inspector“) verfügbar wird.

## 6.4 Implementierte Renderer

In der Testimplementation des Visualisierungsmoduls sind drei Renderer exemplarisch implementiert worden. Die Sichten dieser drei Renderer auf einen Informationsraum, der aus vier Suchbegriffen aufgespannt wird, sind in der Abbildung 6.7 gezeigt. Zwei dieser Renderer sind Umsetzungen des kartesischen Visualisierungsverfahrens mit Glyphs, das im Abschnitt 5.2.2 ausführlich beschrieben wurde. Abbildung 6.7a zeigt die zweidimensionale Darstellungsform, die hauptsächlich für zweidimensionale Informationsräume geeignet ist. Wie im Abschnitt 5.2.1 erläutert wurde, sind aber auch Informationsräume mit drei und vier Dimensionen noch übersichtlich darstellbar. Einstellbare Parameter dieses Renderers sind:

- die Form der Glyphs,
- die Koordinatenachsen (an/aus),
- die Farbskala (an/aus).

Die Schlüsselworte können den grafischen Darstellungsparametern frei zugeordnet werden: X- und Y-Achse (erforderlich), Größe, Farbe.

Die Abbildung 6.7b zeigt die dreidimensionale Variante der kartesischen Visualisierungsmethode. Voraussetzung für dieses Verfahren ist ein Informationsraum mit mindestens drei Dimensionen. Unter Zuhilfenahme von grafischen Attributen wie Größe und Farbe der Objekte sind auch bei diesem Verfahren noch höherdimensionale Informationsräume darstellbar. Suchergebnisse, die auf fünf Schlüsselworten basieren, lassen sich noch relativ übersichtlich visualisieren, was in der Praxis durchaus als hinreichend betrachtet werden kann.

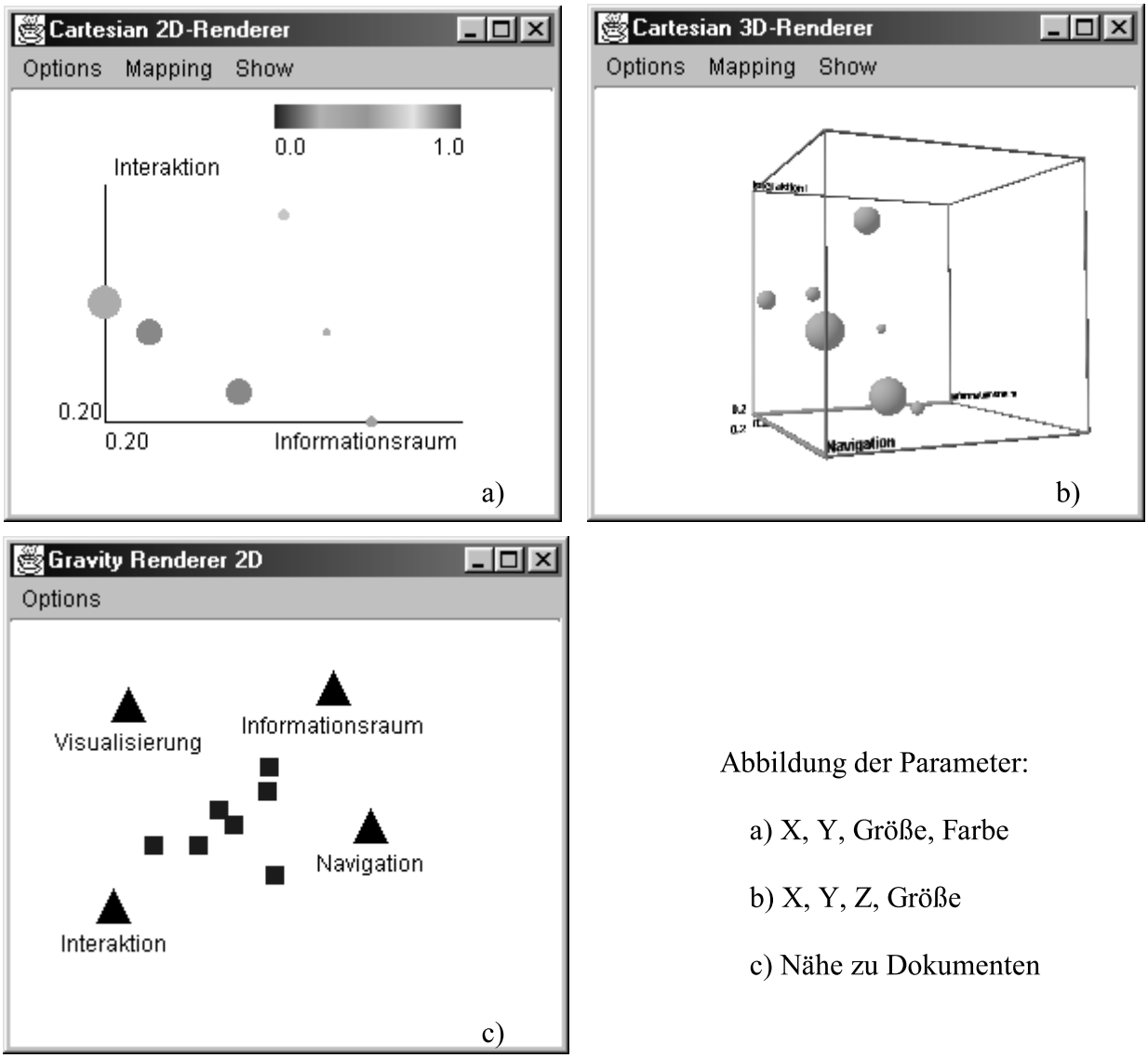



Abbildung 6.7: Implementierte Renderer

Einstellbare Parameter dieses Renderers sind:

- die Form der Glyphs (Kugel, Würfel, ...),
- die Anzeige der Dokumenttitel (sinnvoll, wenn die Zahl der angezeigten Dokumente bereits reduziert worden ist),
- die Koordinatenachsen (an/aus),
- eine Bounding-Box (Rahmen aus Linien um den Informationsraum),
- einen flächenhaften Rahmen um das Volumen des Informationsraumes zur besseren Orientierung (Darstellung als Würfel),
- die Farbskala (an/aus).

Der dritte Renderer in der Abbildung 6.7c ist an das SQWID-Verfahren angelehnt, das im Abschnitt 3.2.6 beschrieben worden ist. Die Schlüsselworte werden als Gravitationsquellen betrachtet,

die Dokumente werden entsprechend ihrer Relevanz zu den einzelnen Suchbegriffen in der Fläche angeordnet. Einstellbare Parameter dieses Renderers sind:

- die grafische Darstellungsform der Schlüsselwort-Objekte,
- die Form der Dokument-Objekte.

Die Schlüsselwort-Objekte können interaktiv in der Fläche bewegt werden. Die Reaktion auf solch eine Bewegung ist die unmittelbar folgende Neuanordnung der Dokument-Objekte. Mit diesem Renderer können Informationsräume mit beliebiger Anzahl von Dimensionen visualisiert werden.

## 6.5 Filtereinstellungen

Es sind alle im Abschnitt 5.4.4 aufgeführten Filter implementiert worden, um Erfahrungen bezüglich der Effizienz von Filterkombinationen zu sammeln. Auf die Funktionsweise der einzelnen Filter soll hier nicht noch einmal eingegangen werden, sie wurde im Abschnitt 4.4.1 ausführlich beschrieben. Die Filter sind nach ihrer Bedeutung in drei Gruppen eingeteilt, was auch im Menü „Filter“ der Applikation deutlich wird:

- der Redundanzfilter (standardmäßig eingeschaltet),
- die Schwellwertfilter,
- Filter für die Feinauswahl

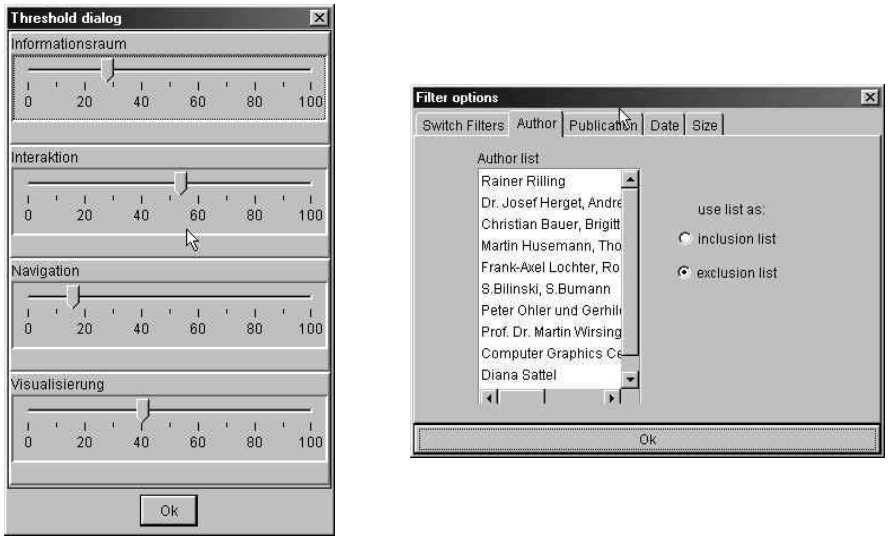



Abbildung 6.8: Filter-Dialoge

Die Schwellwertfilter haben sich als effizientes Mittel zur schnellen Eingrenzung der Dokumentenmenge herausgestellt. Die entsprechende Dialogkomponente ist in der Abbildung 6.8 dargestellt. Jedes Schlüsselwort erhält ein Dialogelement vom Typ „Slider“, die Anzahl dieser Regler wird automatisch an die Dimension des zu visualisierenden Informationsraumes angepaßt.
Der Schwellwert für die Relevanz eines Schlüsselwortes kann als Wert im Intervall [1..100] eingestellt werden. Dokumente, deren Relevanz bezüglich mindestens eines Schlüsselwortes unter dem für dieses Schlüsselwort eingestellten Schwellwertes liegt, werden nach Aktivierung dieser Werte durch den Button „Ok“ und der nächsten Aktualisierung des Renderers von der Darstellung ausgeschlossen.

Die Filter-Optionen für die Feinauswahl der Dokumente sind in einer separaten Dialogkomponente zusammengefaßt (siehe Abbildung 6.8). Auf der ersten Registerkarte sind alle Filter verzeichnet, diese können mittels Checkboxen einzeln zugeschaltet werden. Zu jedem Filter gehört eine Reihe von Parametern, die jeweils auf einer eigenen Registerkarte einstellbar sind:

**Autoren-Filter:** In einer Textliste sind alle Autoren der aktuell gültigen Dokumente aufgeführt. Eine beliebige Anzahl von Autoren kann vom Benutzer ausgewählt werden. Über eine Checkbox wählt der Anwender aus, ob nur die Dokumente der ausgewählten Autoren angezeigt werden sollen, oder die betreffenden Dokumente herausgefiltert werden sollen (siehe Abbildung 6.8).

**Publikations-Filter:** Die Einstellung erfolgt analog zum Autoren-Filter, wobei die Auswahl aber nach den Publikationen vorgenommen wird, in die die Dokumente eingebettet sind.

**Datum-Filter:** In dieser Maske werden zwei Daten angegeben – ein Start- und ein Enddatum. Ist dieser Filter aktiviert, werden nur noch Dokumente angezeigt, deren Erscheinungsdatum zwischen diesen beiden Daten liegt.

**Größen-Filter:** Auf dieser Registerkarte wird ein Wert in Byte definiert. Mittels Chekboxen wird festgelegt, ob nur Dokumente angezeigt werden sollen, deren Größe über oder unter diesem Wert liegt.

Die vorgenommenen Einstellungen werden aktiv, nachdem die Dialog-Komponente mit dem „Ok-Button“ beendet und die Aktualisierung des Renderer initiiert wurde.

## 6.6 Tests mit Suchergebnissen

### 6.6.1 Der Simulationsmodus

In die Testimplementation wurde ein Simulationsmodus integriert, der schnell und einfach Informationsräume mit einer beliebigen Anzahl von Dokumenten generieren kann. Die möglichen Dimensionen dieser Informationsräume liegen zwischen Eins und Vier. Die Titel der Dokumente, ihre URLs, die Erstellungsdaten und die Größe werden mit Hilfe eines Zufallsgenerators erzeugt. Die Anzahl der Autoren kann vorgegeben werden. Die Relevanzen hinsichtlich der verwendeten Schlüsselworte werden ebenfalls auf Zufallsbasis ermittelt. Ein auf diese Weise generiertes Dokument kann beispielsweise mit folgenden Informationen bestückt sein:

```
<document>
    <title>Title 3</title>
    <author>Author 12</author>
    <url>http://www.url2</url>
    <date>17-6-1992</date>
    <size>59663</size>
    <relevances>
        <rvalue>0.8667968</rvalue>
        <rvalue>0.49134594</rvalue>
        <rvalue>0.3243801</rvalue>
    </relevances>
</document>
```

Dieses fiktive Dokument ist Bestandteil eines dreidimensionalen Informationsraumes. Die Schlüsselworte der generierten Suchergebnisse tragen die Bezeichnungen *Keyword_1 . . . Keyword_n*, wobei $n$

der Dimension des Informationsraumes entspricht.
Um möglichst realistische Bedingungen zu simulieren, wurde der Datumsbereich auf 1990 bis 2002 eingegrenzt.

### 6.6.2 Das Auffinden relevanter Dokumente

Mit Hilfe der simulierten Informationsräume ist es möglich, die Effizienz der Verwendung einer grafischen Darstellung in Kombination mit verschiedenen Filtern zu prüfen. Dabei sind Informationsräume mit verschiedenen Dimensionen zu untersuchen.
Der Fall des eindimensionalen Informationsraumes soll hier nicht betrachtet werden, da eine grafische Darstellung bei einem Suchbegriff kaum Vorteile gegenüber der Verwendung einer Textliste bringt. Daher wurde im Prototyp auch kein geeigneter Renderer für diesen Fall implementiert.
Experimentell werden jeweils 10000 Dokumente in Informationsräumen verschiedener Dimensionen simuliert. Im Prototyp sind die Schwellwertfilter auf den Wert 0.2 voreingestellt, dies ist auch gleichzeitig der einzige aktive Filter. Bei der Simulation eines zweidimensionalen Informationsraumes werden nach dem Durchlaufen der Filterstufe noch 6394 Dokumente vom System als relevant eingestuft. Nach einer Einstellung der Schwellwerte auf 0.8 existieren noch 372 relevante Dokumente. Wird die Schwelle für eines der beiden Schlüsselworte auf 0.9 erhöht, reduziert sich die Zahl der relevanten Dokumente auf 173. Liegt der Schwellwert bei beiden Suchbegriffen bei 0.9, werden nur noch 87 Dokumente als relevant klassifiziert. Diese Menge kann in der grafischen Darstellung gut übersehen und mit den weiteren zur Verfügung stehenden Filtern und Werkzeugen analysiert werden. Die Tabelle 6.2 zeigt eine Meßreihe, bei der der Schwellwert schrittweise von 0.5 auf 0.9 erhöht wurde. Dabei ist zu beachten, daß der Wert jeweils für *alle* vorhandenen Suchbegriffe eingestellt wurde. Eine solche Einstellung wird in der Praxis kaum vorgenommen, anhand der Zahlen der verbliebenen Objekte wird aber deutlich, wie fein die Menge der relevanten Dokumente durch gutes Dosieren der Schwellwerte eingegrenzt werden kann.

| Schlüsselworte | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|
| 2 | 2490 | 1534 | 895 | 372 | 87 |
| 3 | 1219 | 604 | 233 | 67 | 11 |
| 4 | 622 | 240 | 70 | 10 | 0 |
| 5 | 331 | 110 | 30 | 2 | 0 |

Tabelle 6.2: Simulation verschiedener Informationsräume

Aus diesem Experiment wird ersichtlich, daß die Stärke der Filtermechanismen genau dort zum Tragen kommt, wo im Umgang mit der Textliste die größten Schwierigkeiten auftreten – bei der Auswertung von Ergebnissen mehrerer Suchbegriffe.
In der Praxis dürfte der Effekt noch stärker sein. Im Experiment sind durch die Verwendung von Zufallszahlen die Relevanzwerte gleichmäßig über die Dokumentenmenge verteilt. Bei realistischen Suchergebnissen steht eine große Menge von Dokumenten mit geringer Relevanz gegen eine sehr geringe Menge mit hoher Relevanz. Daher wird die Reduktion der reellen Dokumentenmenge noch effizienter sein.

## 6.7 Fazit

Die Implementation der grundlegenden Funktionalität eines Systems zur Visualisierung von Informationsräumen in diesem Prototyp zeigt, daß die vorgeschlagene Lösung generell tragbar ist. Es

ist deutlich geworden, daß die Suche nach relevanten Informationen mit einer Kombination aus grafischer Darstellung und Filtermechanismen erheblich vereinfacht und beschleunigt werden kann. Der Benutzer wird in die Lage versetzt, binnen kürzester Zeit mit einigen wenigen unkomplizierten Einstellungen sein Suchergebnis auf eine Menge von Dokumenten zu reduzieren, die mit hoher Wahrscheinlichkeit seinen Erwartungen entsprechen. Diese kleine Menge kann er komfortabel mit mehreren Werkzeugen analysieren und beliebig zwischenspeichern. Solche Möglichkeiten bieten die im Kapitel 3 vorgestellten Verfahren nicht.
Der Prototyp stellt keine Applikation für einen Endanwender dar. Er zeigt, daß es durch den modularen Entwurf des Systems möglich ist, die Komponenten in benutzerfreundliche Oberflächen für verschiedenste Anwendungsanforderungen einzubetten.

# Kapitel 7

# Diskussion und Ausblick

In diesem Kapitel sollen die Ergebnisse dieser Arbeit noch einmal zusammengefaßt und abschließend diskutiert werden. Weiterhin werden Vorschläge für mögliche weiterführende Arbeiten und Erweiterungen des vorgestellten Modells unterbreitet.

**Die Aufgabenstellung**
Es ist gezeigt worden, daß herkömmliche Retrievalmethoden wie die Suche mit Schlüsselworten und Browsen in bibliographischen Datenbasen aufgrund des schnelleren steigenden Informationswachstums keine ausreichend effiziente Unterstützung des Benutzers bei der Informationssuche ermöglichen. Ziel dieser Arbeit ist es, bessere Methoden zur Untersuchung der aus den Suchergebnissen resultierenden Informationsräume vorzuschlagen.
Ein geeignetes und in vielen Bereichen erprobtes Mittel zur Analyse umfangreicher Datenmengen ist die Visualisierung. Es existieren bereits einige Ansätze zur Visualisierung von Dokumentenmengen, die sich aber nicht durchsetzen konnten. Zu diesen Ansätzen wurde eine umfassende Recherche durchgeführt, wobei die Verfahren nach ausgewählten Kriterien untersucht und gegenübergestellt wurden. Das Ergebnis dieser Recherche begründet die Notwendigkeit eines neuen Ansatzes, der sich nicht ausschließlich auf eine grafische Darstellung des Informationsraumes beschränkt.

**Der Lösungsweg**
Es wurde untersucht, wie ein Benutzer von einem sehr umfangreichen und unstrukturierten Suchergebnis zu der Untermenge von Dokumenten gelangen kann, die wirklich seinen Anforderungen entspricht. Dabei hat sich herausgestellt, daß ein sehr breites Spektrum von Anwendungsfällen existiert, von denen zwei Extremsituationen genau untersucht wurden – die zielgerichtete Informationssuche und die unscharfe Informationssuche. Ausgehend von diesen Anwendungsfallanalysen und den Ergebnissen der Recherche wurden folgende Anforderungen für einen neuartigen Ansatz definiert:

- Es wird ein modulares, erweiterbares Visualisierungsmodul benötigt, das über eine definierte Schnittstelle an Suchmaschinen angekoppelt werden kann.
- Das Modul soll über eine Anzahl von intuitiv benutzbaren Darstellungsverfahren verfügen, die je nach Anwendungssituation ausgewählt und angepaßt werden können.
- Es werden Filtermechanismen benötigt, mit deren Hilfe die Menge der gefundenen Dokumente effektiv reduziert werden kann.
- Der Benutzer soll die Möglichkeit haben, einen Zwischenstand seiner Analyse des Suchergebnisses speichern und die Arbeit später fortsetzen zu können.

- Das Analysewerkzeug soll nach dem Suchprozeß auch offline, ohne Verbindung zum Internet einsetzbar sein.

Suchmaschinen arbeiten mit Ranking-Algorithmen, um die Reihenfolge der Dokumente in den textuellen Ergebnislisten zu berechnen. Dazu werden pro Suchbegriff für jedes Dokument Relevanzwerte berechnet, die auch für die Nachbearbeitung des Suchergebnisses nutzbar sind. Diese Relevanzen könnten von Suchmaschinen mit wenig Aufwand bereitgestellt werden. Der in dieser Arbeit vorgestellte Lösungsansatz beruht darauf, die Relevanzwerte als Parameter in Visualisierungsmodellen und zum Filteren nicht relevanter Dokumente zu verwenden. Durch zusätzliche Angaben (z. B. Autor, Datum, ...) zu den Dokumenten ergeben sich weitere Möglichkeiten zur Steigerung der Effizienz bei der Analyse von Suchergebnissen, die in diesem Ansatz ebenfalls berücksichtigt sind.

**Ergebnisse**
Die konkreten Ergebnisse dieser Arbeit sind:

- eine Analyse von bereits vorhandenen Systemen zur grafischen Darstellung von Informationsräumen,
- die Architektur eines offenen, erweiterbaren Systems zur Visualisierung und Aufbereitung von Informationsräumen,
- die Definition einer Schnittstelle zu Suchmaschinen, über die alle zur Analyse der Dokumentenmenge benötigten Informationen bereitgestellt werden,
- der Entwurf einer Klassenhierarchie, die ein komplettes Visualisierungs- und Analysesystem mit Schnittstellen für spätere Erweiterungen beschreibt,
- die Definition eines XML-basierten und erweiterbaren Datenaustauschformates für Suchergebnisse,
- die Beschreibung von Interaktionsabläufen, die in Abhängigkeit vom konkreten Anwendungsfall eine effektive Eingrenzung der Dokumentenmenge gestatten,
- ein Prototyp, der zur Verifizierung des Entwurfes und zu Untersuchungen bezüglich der Effizienz der vorgeschlagenen Arbeitsweise diente.

Insbesondere die Experimente mit dem Prototyp haben gezeigt, daß es möglich ist, mit einem solchen System sehr schnell zu einer Untermenge von Dokumenten zu gelangen, die tatsächlich dem Interessengebiet des Benutzers entspricht. Das trifft vor allem dann zu, wenn die Suche mit mehreren Schlüsselwörtern durchgeführt wurde.

**Ausblick**
Ausgehend von dem vorgeschlagenen Ansatz sind eine Reihe von Weiterentwicklungen denkbar:

- die Implementierung weiterer effizienter Renderer,
- die Weiterentwicklung von Filtermechanismen,
- die Ankopplung an konkrete Suchmaschinen

Es wäre wünschenswert, daß sich im Bereich der wissenschaftlichen Publikationen im Internet gewisse Regeln durchsetzen. Diese könnten unter anderem beinhalten, daß Dokumente bei ihrer Veröffentlichung ähnlich zum Bibliothekswesen auf eine festgelegte Weise beschrieben werden. Diese Beschreibungen können während der Analyse der Suchergebnisse zur Auswertung genutzt werden und so die Zeit zum Auffinden der interessanten Dokumente verkürzen. Daß dies technisch möglich ist, hat diese Arbeit gezeigt.

# Anhang A

# Anforderungsfall-Tabellen

## A.1 Zielgerichtete Informationssuche

### Spezifikation der Szenarien

In der Spezifikation der Szenarien werden die primären Szenarien der elementaren Nutzungsfälle erklärt.

#### Stellen der Suchanfrage

| **AF1** | **Suchanfrage stellen** |
|---|---|
| Zweck | Die Schlüsselwörter bzw. die Themen werden erfaßt, eine Suche wird durchgeführt und das Ergebnis ermittelt. |
| Kontext | Suche von Informationen |
| beteiligte Aktoren | Benutzer |
| auslösende Ereignisse | Ein Benutzer stellt seine Anfrage durch mit Hilfe von Schlüsselwörtern oder Themenbezeichnungen |
| Vorbedingugen | Das Anfrageformular ist auf dem Bildschirm dargestellt. |
| Ausnahme | Keine Dokumente mit den entsprechenden Schlüsselwörtern vorhanden, Fehlbedienungen, Ausfall des Systems |
| relevante Ergebnisse | eine Menge von Dokumenten und deren Relevanzwerten wird ermittelt. |

Tabelle A.1: Definition des Anwendungsfalls: *Suchanfrage stellen*

| **AF11** | **Primäres Szenario: Initiieren der Suche** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Die Suchmaschine stellt eine Bildschirmmaske zur Eingabe der Schlüsselwörter zur Verfügung. |
| 2 | Der Benutzer spezifiziert die Suchbegriffe. |
| 3 | Die Suchmaschine stellt eine Liste gefundener Dokumente zusammen und berechnet für jedes Dokument die Relevanzen hinsichtlich jedes Schlüsselwortes. |
| 4 | Die Schlüsselwörter und die Liste der Dokumente mit allen Informationen und Relevanzen werden an das Visualisierungsmodul übergeben. |
| 5 | Das Visualisierungsmodul generiert eine dreidimensionale Ansicht des Informationsraums und der darin befindlichen Dokumente. |

Tabelle A.2: Spezifikation des Szenarios: *Initiieren der Suche*

**Ergebnisdarstellung**

| **AF2** | **Grafische Darstellung der Ergebnisse** |
|---|---|
| Zweck | Die Schlüsselwörter, die Dokumente und die Relevanzen werden übernommen. Das Ergebnis wird in Form grafischer Elemente dargestellt. |
| Kontext | Darstellung von Informationen |
| beteiligte Aktoren | Benutzer |
| auslösende Ereignisse | Der Informationsraum wurde durch die Suchmaschine ermittelt und es ist ein Mapping-Verfahren spezifiziert worden, es wurden neue Filter-Einstellungen vorgenommen. |
| Vorbedingungen | Es sind alle darstellungsrelevanten Daten von der Suchmaschine übermittelt worden. |
| Ausnahme | Der Informationsraum enthält keine Dokumente, kein Mapping-Verfahren spezifiziert, Fehler im Grafiksystem, Fehlbedienung. |
| relevante Ergebnisse | Der Informationsraum mit allen Dokumenten, deren Relevanzen oberhalb der eingestellten Schwellwerte liegen wird grafisch dargestellt. Die zugeschalteten optischen Hilfsmittel erscheinen in der Anzeige. |
| Verweis auf Szenario | AF21, AF22, AF23, AF24 |

Tabelle A.3: Definition des Anwendungsfalls: *Grafische Darstellung der Ergebnisse*

| **AF21** | **Primäres Szenario: Auswahl des Mapping-Verfahrens** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Aufrufen des Dialoges zur Auswahl des Verfahrens |
| 2 | Einstellen des gewünschten Verfahrens |
| 3 | Einstellung weiterer Systemparameter in Abhängigkeit vom gewählten Mapping-Verfahren |
| 4 | Aktualisieren der Ansicht |

Tabelle A.4: Spezifikation des Szenarions: *Auswahl des Mapping-Verfahrens*

| **AF22** | **Primäres Szenario: Relevanz-Mapping** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Zuordnung der Relevanzen zu den Eigenschaften des eingestellten Mapping-Verfahrens. |
| 2 | Aktualisieren der Ansicht |

Tabelle A.5: Spezifikation des Szenarios: *Relevanz-Mapping*

| **AF23** | **Primäres Szenario: Einschalten zusätzlicher Hilfsmittel** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Aufruf des Hilfsmittel-Dialogs |
| 2 | Auswahl des/der Hilfsmittel(s) |
| 3 | Aktualisieren der Ansicht |

Tabelle A.6: Spezifikation des Szenarios: *Einschalten zusätzlicher Hilfsmittel*

| **AF24** | **Primäres Szenario: Einstellung der Ansichts-Parameter** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Das System bietet geeignete Interaktionsmittel in Form von Buttons, Slidern, Maus-Aktionen, Key-Aktionen bereit. |
| 2 | Der Benutzer stellt mit den genannten Mitteln die gewünschte Ansicht ein. |
| 3 | Das System aktualisiert die Ansicht parallel zu den vom Benutzer eingestellten Parametern. |

Tabelle A.7: Spezifikation des Szenarios: *Einstellung der Ansichts-Parameter*

**Reduktion der Ergebnismenge**

| **AF3** | **Reduktion der Ergebnismenge** |
|---|---|
| Zweck | Der Filter-Dialog wird aufgerufen, damit die Parameter der Dokument-Filter definiert werden können. |
| Kontext | Filterung von Dokumenten aus der Dokumentenmenge |
| beteiligte Aktoren | Benutzer |
| auslösende Ereignisse | Der Benutzer möchte die Menge der dargestellten Objekte manipulieren (erhöhen, reduzieren oder Beziehungen anzeigen) und ruft den Filter-Dialog explizit auf. |
| Vorbedingungen | Das Ergebnis ist grafisch dargestellt. |
| Ausnahme | Keine Dokumente mit den entsprechenden Parametern dargestellt, Fehlbedienungen, Ausfall des Systems |
| relevante Ergebnisse | Neue Filtereinstellungen zum Bilden der Untermenge der darzustellenden Elemente. |
| Verweis auf Szenario | AF31, AF32, AF33 |

Tabelle A.8: Definition des Anwendungsfalls: *Reduktion der Ergebnismenge*

| **AF31** | **Primäres Szenario: Einschalten von Dokument-Filtern** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Aufruf des Filter-Dialogs |
| 2 | Auswahl des/der Filter |
| 3 | Einstellung evtl. notwendiger Parameter der einzelnen Filter |
| 4 | Aktualisieren der Ansicht |

Tabelle A.9: Spezifikation des Szenarios: *Einschalten von Dokument-Filtern*

| **AF32** | **Primäres Szenario: Markieren von uninteressanten Dokumenten** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Als „uninteressant“ eingestufte Dokumente werden mittels geeigneter Interaktion (z. B. Mausklick) markiert. |
| 2 | Aktualiseren der Ansicht, wobei die markierten Objekte von der Ansicht ausgeschlossen werden. |

Tabelle A.10: Spezifikation des Szenarios: *Markieren von uninteressanten Dokumenten*

| **AF33** | **Primäres Szenario: Einschalten von Relevanz-Schwellen** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Aufruf des Relevanz-Dialogs |
| 2 | Das System stellt Interaktionsmöglichkeiten zum Einstellen eines Relevanz-Schwellwertes für jedes Schlüsselwort bereit. |
| 3 | Das System aktualisiert die Ansicht, wobei Dokumente, die für mindestens ein Schlüsselwort den eingestellten Schwellwert unterschreiten, von der Darstellung ausgeschlossen werden. |

Tabelle A.11: Spezifikation des Szenarios: *Einschalten von Relevanz-Schwellen*

**Dokument-Auswahl**

| **AF4** | **Auswahl und Bezug relevanter Dokumente** |
|---|---|
| Zweck | Auswahl der (subjektiv) interessanten Dokumente aus der gesamten Menge |
| Kontext | Der Benutzer kann vorhandene Detailinformationen zu den Dokumenten betrachten und entscheiden, ob er ein Dokument als für ihn „interessant“ oder „uninteressant“ einstuft. |
| beteiligte Aktoren | Benutzer |
| auslösende Ereignisse | Selektion eines Dokuments durch den Benutzer. |
| Vorbedingungen | Die vorhandenen Dokumente sind grafisch dargestellt. |
| Ausnahme | es sind keine Dokumente vorhanden, es sind keine Detailinformationen verfügbar, Fehlbedienungen, Ausfall des Systems |
| relevante Ergebnisse | Die Menge der für den Benutzer interessanten Dokumente. |
| Verweis auf Szenario | AF41, AF42, AF43, AF44, AF45 |

Tabelle A.12: Definition des Anwendungsfalls: *Auswahl und Bezug relevanter Dokumente*

| **AF41** | **Primäres Szenario: Betrachten von Detailinformationen** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Der Benutzer selektiert ein bestimmtes Dokument (z. B. per Mausklick) |
| 2 | Das System stellt eine Anzeigemöglichkeit für zusätzliche Informationen zur Verfügung (z. B. ein separates Fenster) und listet alle zum selektierten Dokument verfügbaren Informationen auf. |

Tabelle A.13: Spezifikation des Szenarios: *Betrachten von Detailinformation*

| **AF42** | **Primäres Szenario: Markieren interessanter Dokumente** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Als „interessant“ eingestufte Dokumente werden mittels geeigneter Interaktion (z. B. Mausclick) markiert. |
| 2 | Aktualiseren der Ansicht, wobei die markierten Objekte auf besondere Weise (z. B. spezielle Farb- oder Formgebung, Blinken, etc.) hervorgehoben werden. |

Tabelle A.14: Spezifikation des Szenarios: *Markieren interessanter Dokumente*

| **AF43** | **Primäres Szenario: Download der markierten Dokumente** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Aufruf des Download-Dialogs |
| 2 | Feinauswahl der zu beschaffenden Dokumente durch den Benutzer |
| 3 | Download und Pufferung der in der Feinauswahl spezifizierten Dokumente |
| 4 | Abschließende Begutachtung der Dokumente durch den Benutzer |
| 5 | Entfernen weiterer, evtl. vorhandener uninteressanter Dokumente |

Tabelle A.15: Spezifikation des Szenarios: *Download der markierten Dokumente*

| **AF44** | **Primäres Szenario: Speichern der Dokumente** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Aufruf des Speichern-Dialogs |
| 2 | Speichern der verbliebenen Dokumente |

Tabelle A.16: Spezifikation des Szenarios: *Speichern der Dokumente*

| **AF45** | **Primäres Szenario: Drucken der Dokumente** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Aufruf des Druck-Dialogs |
| 2 | Drucken der verbliebenen Dokumente |

Tabelle A.17: Spezifikation des Szenarios: *Drucken der Dokumente*

## A.2 Unscharfe Informationssuche

### Spezifikation der Szenarien

In der Spezifikation der Szenarien werden die primären Szenarien der elementaren Nutzungsfälle erklärt.

### Ergebnisdarstellung

| **AF2** | **Grafische Darstellung der Ergebnisse** |
|---|---|
| Zweck | Die Suchergebnisse werden in Form von grafischen Objekten dargestellt. |
| Kontext | Darstellung des ermittelten Informationsraums |
| beteiligte Aktoren | Benutzer (Surfer) |
| auslösende Ereignisse | Übergabe der Dokumente und deren Relevanzen an das Visualisierungsmodul, Interaktion des Benutzers |
| Vorbedingungen | Der Informationsraum ist vollständig von der Suchmaschine ermittelt |
| Ausnahme | es sind keine Dokumente gefunden worden, die Relevanzwerte sind nicht vorhanden, Fehler im Grafik-System |
| relevante Ergebnisse | die Menge der gefundenen Dokumente wird dargestellt. |
| Verweis auf Szenario | AF21, AF22, AF23 |

Tabelle A.18: Definition des Anwendungsfalls: *Grafische Darstellung der Ergebnisse*

AF21: **Primäres Szenario: Auswahl des Mapping-Verfahrens**
Dieser Anwendungsfall ist identisch mit AF21 im Anwendungsfall „zielgerichtete Suche“

AF22: **Primäres Szenario: Relevanz-Mapping**
Dieser Anwendungsfall ist identisch mit AF22 im Anwendungsfall „zielgerichtete Suche“

AF23: **Primäres Szenario: Einschalten zusätzlicher Hilfsmittel**
Dieser Anwendungsfall ist identisch mit AF23 im Anwendungsfall „zielgerichtete Suche“

### Ermitteln der relevanten Dokumentenmenge

| **AF3** | **Reduktion der Ergebnismenge** |
|---|---|
| Zweck | Die Ergebnismenge wird reduziert. |
| Kontext | Reduktion der Ergebnismenge durch die Verwendung von Filtern, die in einem speziellen Dialog zu definieren sind |
| beteiligte Aktoren | Benutzer(Surfer) |
| auslösende Eregnisse | expliziter Aufruf des Filterdialoges durch den Benutzer |
| Vorbedingungen | es ist eine Dokumentenmenge vorhanden; die Parameter, nach denen gefiltert werden soll, sind vorhanden |
| Ausnahme | es sind keine Dokumente vorhanden, Fehlbedienungen, Ausfall des Systems |
| relevante Ergebnisse | Neue Filtereinstellungen zum Bilden der Untermenge der darzustellenden Elemente. |
| Verweis auf Szenario | AF31, AF32, AF33, AF34 |

Tabelle A.19: Definition des Anwendungsfalls: *Reduktion der Ergebnismenge*

| **AF31** | **Primäres Szenario: Einstellung der Ansichts-Parameter** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Das System bietet geeignete Interaktionsmittel in Form von Buttons, Slidern, Maus-Aktionen, Key-Aktionen bereit. |
| 2 | Der Benutzer stellt mit den genannten Mitteln die gewünschte Ansicht ein. |
| 3 | Das System aktualisiert die Ansicht parallel zu den vom Benutzer eingestellten Parametern. |

Tabelle A.20: Spezifikation des Szenarios: *Einstellung der Ansichts-Parameter*

AF32: **Primäres Szenario: Einschalten von Relevanz-Schwellen**
Dieser Anwendungsfall ist identisch mit AF33 im Anwendungsfall „zielgerichtete Suche“

AF34: **Primäres Szenario: Markieren von uninteressanten Dokumenten**
Dieser Anwendungsfall ist identisch mit AF32 im Anwendungsfall „zielgerichtete Suche“

| **AF33** | **Primäres Szenario: Betrachten von Detailinformationen** |
|---|---|
| **Nr.** | **Aktion** |
| 1 | Der Benutzer selektiert ein bestimmtes Dokument (z. B. per Mausklick) |
| 2 | Das System stellt eine Anzeigemöglichkeit für zusätzliche Informationen zur Verfügung (z. B. ein separates Fenster) und listet alle zum selektierten Dokument verfügbaren Informationen auf. |

Tabelle A.21: Spezifikation des Szenarios: *Betrachten von Detailinformation*

### Dokument-Auswahl

| **AF4** | **Auswahl und Bezug relevanter Dokumente** |
|---|---|
| Zweck | Auswahl der (subjektiv) interessanten Dokumente aus der gesamten Menge |
| Kontext | Der Benutzer kann vorhandene Detailinformationen zu den Dokumenten betrachten und entscheiden, ob er ein Dokument als für ihn „interessant“ oder „uninteressant“ einstuft. |
| beteiligte Aktoren | Benutzer (Surfer) |
| auslösende Ereignisse | Selektion eines Dokuments durch den Benutzer. |
| Vorbedingungen | Die vorhandenen und gefilterten Dokumente sind grafisch dargestellt. |
| Ausnahme | es sind keine Dokumente vorhanden, es sind keine Detailinformationen verfügbar, Fehlbedienungen, Ausfall des Systems |
| relevante Ergebnisse | Die Menge der für den Benutzer interessanten Dokumente. |
| Verweis auf Szenario | AF41, AF42, AF43 |

Tabelle A.22: Definition des Anwendungsfalls: *Auswahl und Bezug relevanter Dokumente*

AF41: **Primäres Szenario: Download der markierten Dokumente**
Dieser Anwendungsfall ist identisch mit AF43 im Anwendungsfall „zielgerichtete Suche“

AF42: **Primäres Szenario: Speichern der Dokumente**
Dieser Anwendungsfall ist identisch mit AF44 im Anwendungsfall „zielgerichtete Suche“

AF43: **Primäres Szenario: Drucken der Dokumente**
Dieser Anwendungsfall ist identisch mit AF45 im Anwendungsfall „zielgerichtete Suche“

# Anhang B

# Benutzeroberfläche

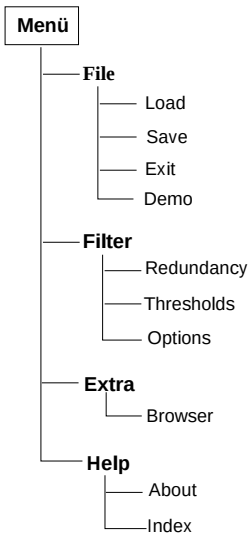



Abbildung B.1: Menüstruktur des Visualisierunsmoduls

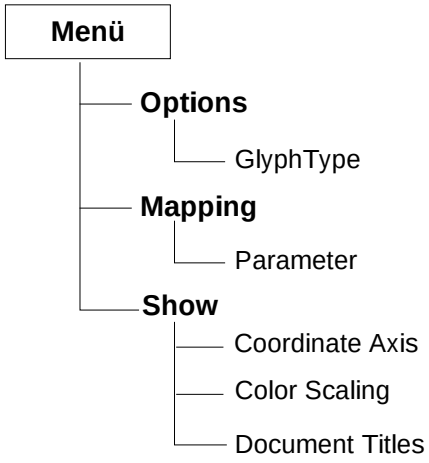



Abbildung B.2: Menüstruktur des 2D-Renderers

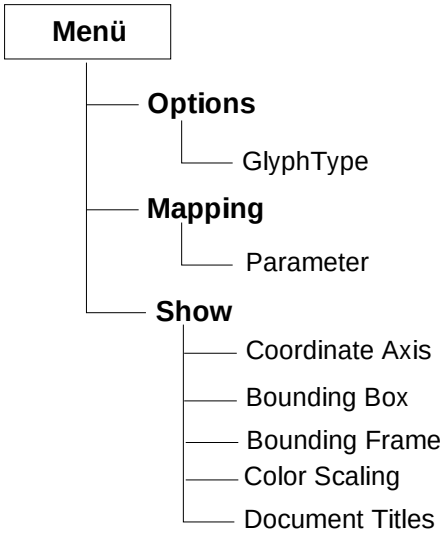
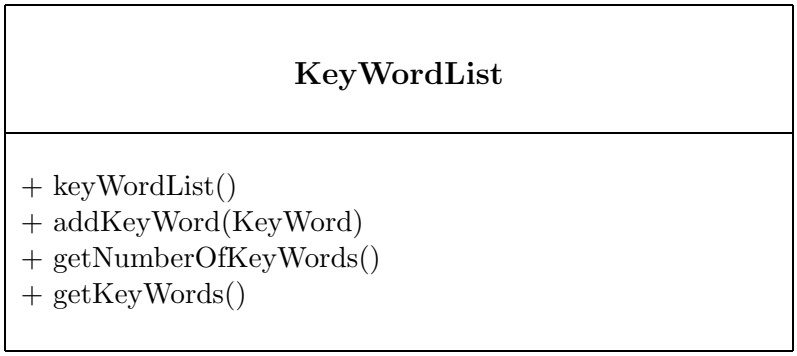



Abbildung B.3: Menüstruktur des 3D-Renderers

# Anhang C

# Klassenbeschreibungen

Zu dieser Kategorie gehören die global verwendeten Klassen, auf die von nahezu allen Komponenten des Systems zugegriffen wird.




Die Klasse *KeyWordList* enthält die Liste der von der Suchmaschine übermittelten Schlüsselwörter des zu analysierenden Suchergebnisses.

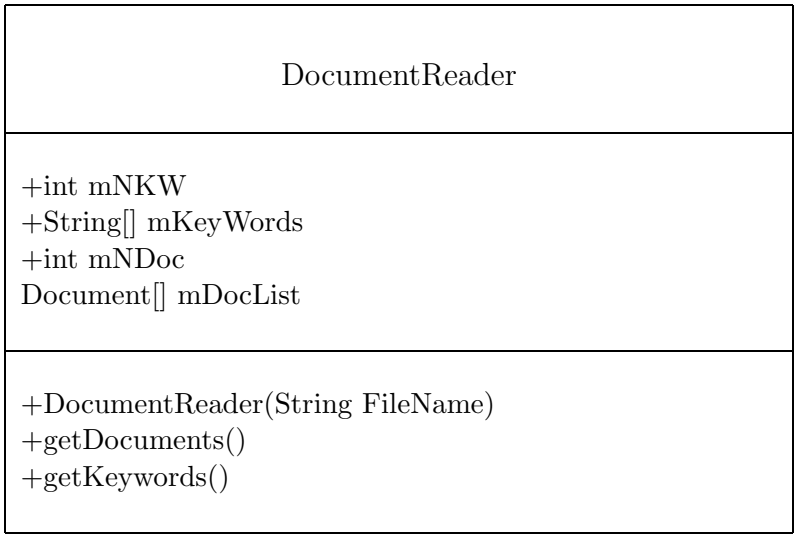
**Methoden**

- *KeyWordList*
  die Liste der bei der Suche verwendeten Suchbegriffe.

- *addKeyWord(KeyWord)*
  fügt einen Suchbegriff in die Liste ein. Der Suchbegriff wird an das Ende der Suchbegriffsliste angehängt.

- *getNumberOfKeyWords()*
  liefert die Anzahl der gegebenen Suchbegriffe.

- *getKeyWords()*
  liefert die Suchbegriffe.



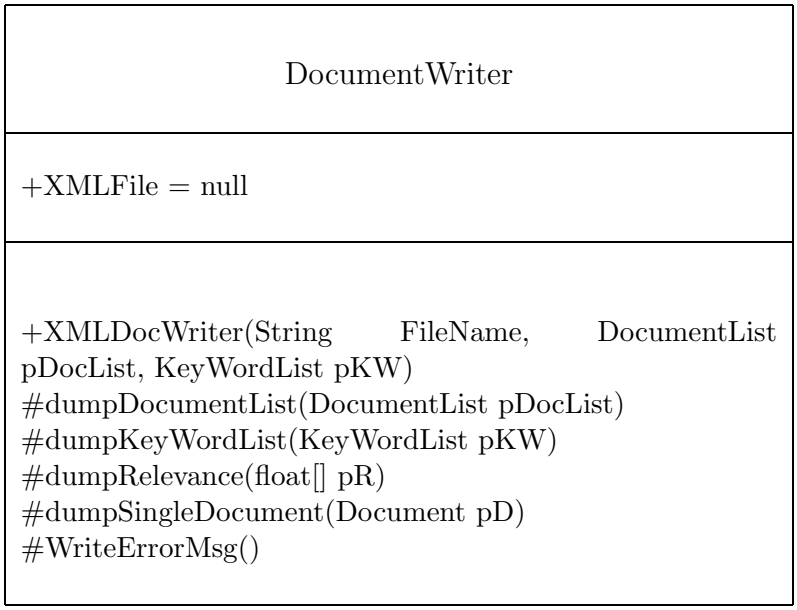

Die Klasse *DocumentReader* wird dazu verwendet, um die Ergebnisdatei von einer Suchmaschine einzulesen. Diese enthält die Schlüsselwörter, sowie alle gefundenen Dokumente.

**Variablen**

- *mNKW*
  beinhaltet die Anzahl der Schüsselwörter,
- *mKeyWords*
  beinhaltet alle Schüsselwörter,
- *mNDoc*
  beinhaltet die Anzahl der Dokumente,
- *mDocList*
  beinhaltet alle Dokumente mit ihren Parametern.

**Methoden**

- *DocumentReader*
  Die angegebene Datei wird geöffnet und ausgelesen. Dabei werden eine Liste der Schlüsselwörter und eine Liste aller Dokumente erstellt.
- *getDocuments*
  gibt die Liste der Dokumente zurück.
- *getKeyWords*
  gibt die Liste der Schlüsselwörter zurück.




Die Klasse *DocumentWriter* dient zum Herausschreiben der aktuellen Dokumentliste im XML-Format.

**Variablen**

- XMLFile = null

**Methoden**

- *DocumentWriter(String FileName, DocumentList pDocList, KeyWordList pKW)*
  Konstruktor, schreibt die übergebene Liste der Schlüsselwörter und die komplette Liste der Dokumente mit allen vorhandenen Informationen in eine Datei mit dem spezifizierten Namen.

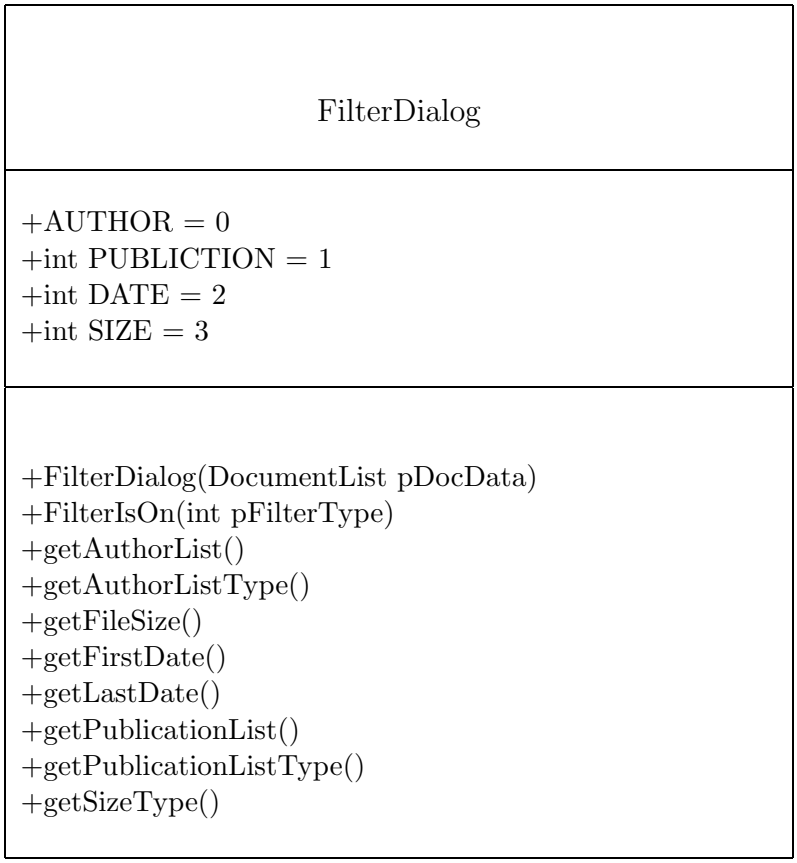
- *dumpKeyWordList(KeyWordList pKW)*
  schreibt die Liste der Schlüsselworte,

- *dumpDocumentList(DocumentList pDocList)*
  schreibt die Liste der Dokumente,

- *dumpSingleDocument(Document pD)*
  schreibt die Informationen eines einzelnen Dokuments,

- *dumpRelevance(float[] pR)*
  schreibt die Relevanzen eines Dokumentes,

- *WriteErrorMsg()*
  gibt Fehlermeldungen bei Ausnahmesituationen (z. B. Schreiben einer Datei nicht möglich)




Die Klasse *FilterDialog* dient zur Einstellung aller Parameter für die Filterung der Dokumente.

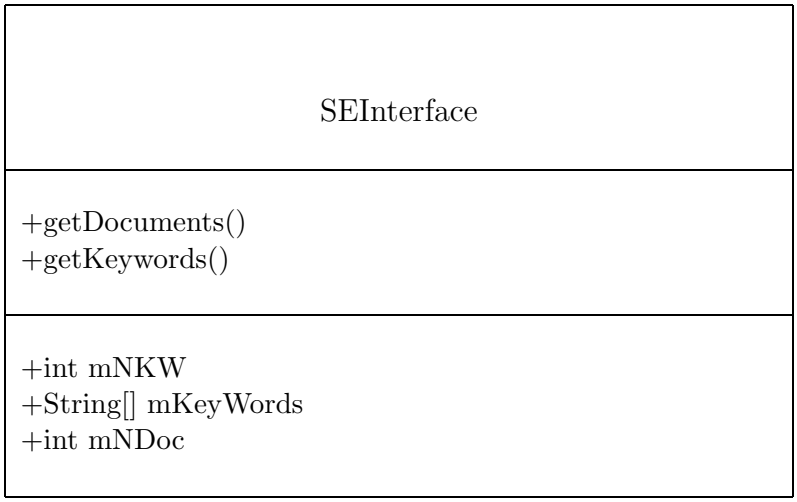
**Variablen**

- *AUTHOR = 0*
- *PUBLICTION = 1*
- *DATE = 2*
- *SIZE = 3*

**Methoden**

- Der Konstruktor *FilterDialog(DocumentList pDocData)* ruft die Methode zum Fensteraufbau und führt sonst keine eigenständigen Aktionen aus.
- *FilterIsOn(int pFilterType)* legt fest, daß die Filterparamenter nur zugänglich sind, wenn eine Dokumentdarstellung aktiv ist.
- *getAuthorList()* gibt die Liste von Authoren züruck.
- *getAuthorListType()* fragt, ob ein bestimmter oder mehrere Autoren aus der Liste ausgeschlossen werden müssen.
- *getFileSize()* gibt die Größe der eingestellten Dateien züruck.
- *getFirstDate()* liefert das erste eingestellte Datum von einem Zeitbereich zurück

- *getLastDate()* liefert das letzte eingestellte Datum von einem Zeitbereich zurück.
- *getPublicationList()* gibt die Liste von Veröffentlichungen zurück.
- *getPublicationListType()* fragt ab, ob bestimmte Veröffentlichungen aus der Liste ausgeschloßen werden sollen oder nicht.
- *getSizeType()* fragt nach der Bestimmung der Dateigröße. Diese kann vom Filterobjekt als Minimum oder Maximum betrachtet werden.



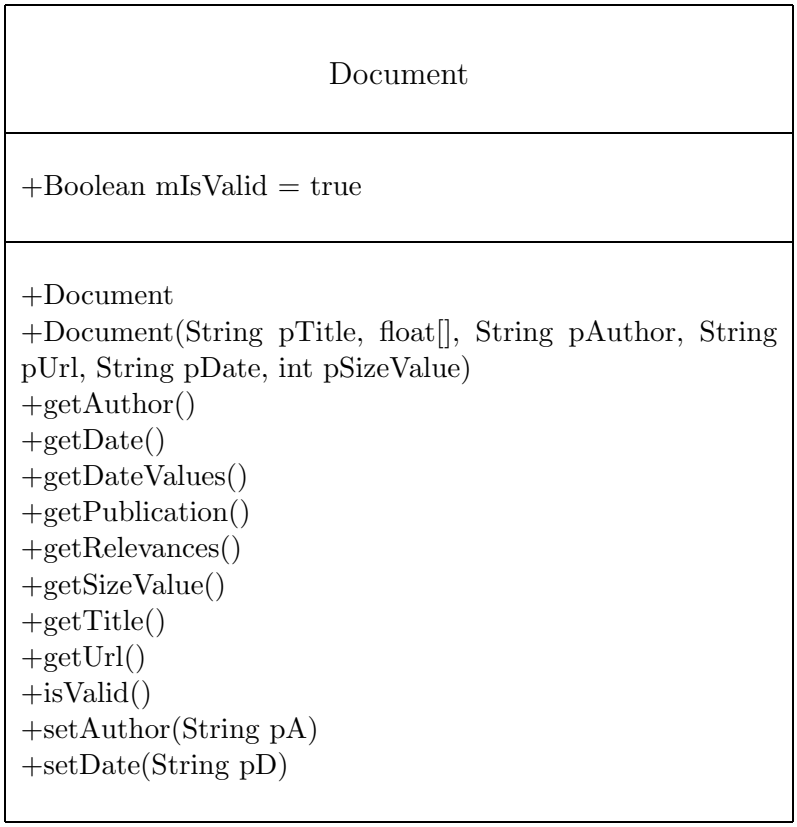

Die Klasse *SEInterface* bildet die Schnittstelle zur Suchmaschine.

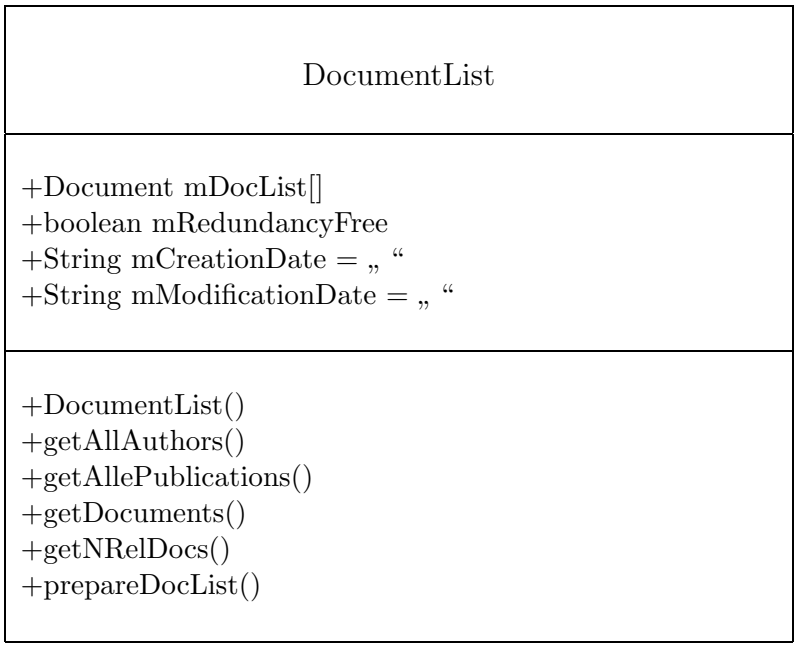
**Variablen**

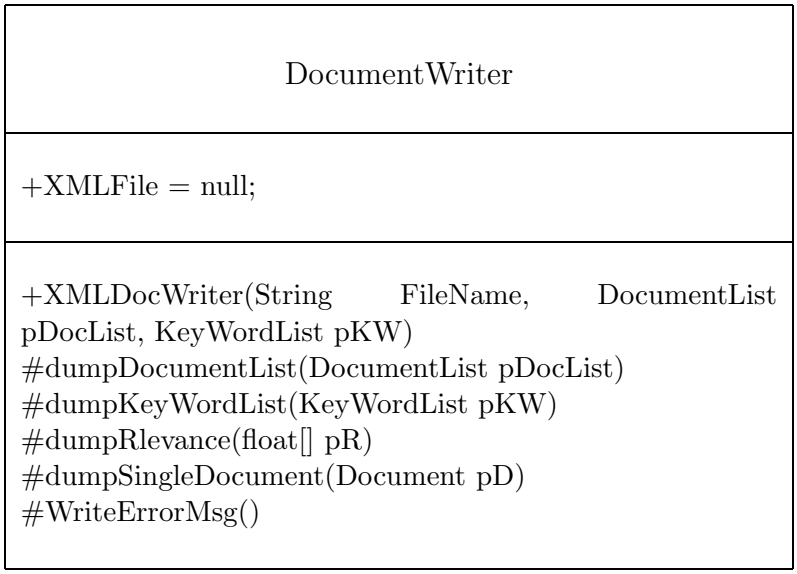
- *int mNKW*
  die Anzahl der verwendeten Suchbegriffe,
- *int mNDoc*
  die Anzahl der gefundenen Dokumente,
- *String[] mKeyWords*
  die Suchbegriffe

**Methoden**

- *getKeyWords()*
  liefert die Liste der Suchbegriffe,
- *getDocuments()*
  liefert die Liste der Dokumente


Document
+Boolean mIsValid = true
+Document
+Document(String pTitle, float[], String pAuthor, String pUrl, String pDate, int pSizeValue)
+getAuthor()
+getDate()
+getDateValues()
+getPublication()
+getRelevances()
+getSizeValue()
+getTitle()
+getUrl()
+isValid()
+setAuthor(String pA)
+setDate(String pD)


DocumentList
+Document mDocList[]
+boolean mRedundancyFree
+String mCreationDate = „ “
+String mModificationDate = „ “
+DocumentList()
+getAllAuthors()
+getAllePublications()
+getDocuments()
+getNRelDocs()
+prepareDocList()


DocumentWriter
+XMLFile = null;
+XMLDocWriter(String FileName, DocumentList pDocList, KeyWordList pKW)
#dumpDocumentList(DocumentList pDocList)
#dumpKeyWordList(KeyWordList pKW)
#dumpRlevance(float[] pR)
#dumpSingleDocument(Document pD)
#WriteErrorMsg()

# Anhang D

# Package-Deklarationen

## D.1 Package: Control

**DocControler:** Hauptklasse des gesamten Visualisierungssystems. Sie enthält alle notwendigen Steuerelemente, Zugriffe auf die Datenbasis, alle globalen Datenstrukturen und erzeugt alle Dialog-Objekte.

**DocumentInspector:** Stellt die Ausgabe aller verfügbaren Informationen zum jeweils selektierten Element sicher.

**DocumentReader:** Liest eine Ergebnisdatei von einer Suchmaschine. Sie enthält die Schlüsselwörter, sowie alle gefundenen Dokumente.

**GoodList:** Die vom Benutzer als „interessant“ gekennzeichneten Dokumente werden durch diese Klasse verwaltet und angezeigt.

**IntroPanel:** Steuerung der Größen und Positionen der einzelnen Dialogelemente.

**RenderPanel:** Dient zur Steuerung der grafischen Ausgabe (Auswahl, starten und beenden des Renderers).

**SearchInfo:** Die Informationen zum Suchergebnis (Datum, Dateiname, Anzahl der Dokumente und Schlüsselwörter, etc.) werden in einem Panel angezeigt, das von dieser Klasse zur Verfügung gestellt wird.

**BrowserDialog:** Stellt einen Dialog zur Einstellung des Pfades zur gewünschten Browser-Applikation bereit.

**KeyWordList:** Interne Datenstruktur, in der die bei der Suche verwendeten Schlüsselwörter gespeichert sind.

## D.2 Package: Docmanager

**DocumentGenerator:** Erzeugt eine Menge von virtuellen Dokumenten und Schlüsselwörtern, die im Simulationsmodus des Prototypen verwendet werden.

**DocLoadDialog:** Initialisiert den Dialog zum Laden eines Suchergebnisses. Es werden alle Dateien vom übergebenen Typ (XML) gescannt.

**DocSaveDialog:** Dialogklasse zum Bestimmen einer Datei zum Speichern der aktuellen internen Datenstruktur.

**Document:** Grundlegende Klasse des Packages. Ein Objekt vom Typ *Document* entspricht einem von der Suchmaschine ermittelten Dokument. Ein solches Dokument wird charakterisiert durch (mindestens):

- Titel
- URL
- Relevanzen bezüglich der verwendeten Suchbegriffe

Weitere Informationen können sein:

- Autor
- Erscheinungsdatum
- Größe
- Erscheinungsort (z.B. in einem Sammelband, Proceedings, o.ä.)

Für alle Charakteristika sind entsprechende set-/get-Methoden implementiert.

**DocumentList:** enthält alle Dokumente, sowie die aktuelle Mapping-Tabelle und die Schlüsselwörter.

**InterestingList:** In dieser Klasse werden alle in der Dokumentenliste vorkommenden Titel abgefragt.

**XMLDocumentHandler:** Parser für die XML-Struktur und Konverter in die interne Dokument-Liste.

**XMLDocumentReader:** Lesen der Dokumentliste im XML-Format.

**XMLDocWriter:** Schreiben der aktuellen internen Dokumentliste in das XML-Format.

## D.3 Package: Filter

**AuthorFilter:** Ermittelt die Liste der zu filternden Autoren und führt die entsprechende Filteroperation aus.

**DateFilter:** Ermittelt den eingestellten Datumsbereich und filtert die entspechenden Dokumente heraus.

**DocFilter:** Interface-Klasse für alle Filter.

**FilterDiaolog:** Stellt den Dialog zur Einstellung aller Parameter für die Filterung der Dokumente zur Verfügung.

**MainFilterPanel:** Dialog zum Ein- und Ausschalten der einzelnen Filter.

**PublicFilter:** Ermittelt die Liste der zu filternden Veröffentlichungen und führt die entsprechende Filteroperation aus.

**RedundancyFilter:** Untersucht die Liste der Dokumente nach mehrfachen Einträgen und entfernt diese gegebenenfalls.

**SizeFilter:** Filtert Dokumente entsprechend des eingestellten Größenbereichs heraus.

**ThresholdDialog:** Dialog zur Einstellung der Relevanz-Schwellwerte mittels Slidern.

**ThresholdFilter:** Filtert Dokumente heraus, bei denen mindestens ein Relevanzwert unter dem eingestellten Schwellwert liegt.

## D.4 Package: Renderer

**DocRenderer:** Interface für alle Renderer

**JnDRenderer:** Oberklasse aller Renderer, stellt die grundlegende Funktionalität und die internen Datenstrukturen zur Verfügung.

### D.4.1 Subpackage: cartesian2

**Cartesian2DRenderer:** Hauptklasse dieses Renderers, organisiert Dialogelemente sowie die grafische Anzeige.

**CartesicMappingDialog:** Dialogklasse zur Spezifizierung der Abbildungseigenschaften. Es wird eine Matrix aus Schlüsselwörtern und darstellbaren Eigenschaften aufgebaut. Der Benutzer legt in dieser Matrix fest, welches Schlüsselwort welcher grafischen Eigenschaft (Position, Größe, Farbe, ...) zugeordnet werden soll.

**DrawArea:** Panel mit Datenstrukturen und Mapping-Algorithmen zur grafischen Ausgabe.

**ExtendedDocument:** Erweiterte Dokument-Klasse, die zusätzlich grafische Informationen enthält, die z. B. bei der Selektion und der damit verbundenen Rückerkennung von Dokumenten benötigt werden.

**ObjectTypeDialog:** Stellt einen Dialog zur Einstellung des grafischen Objekts für die Repräsentation der Dokumente bereit.

**PickMouseAdapter:** Handler zur Ermittlung des entsprechenden Dokuments anhand der Bildschirm-Koordinaten bei Selektion mit der Mouse.

### D.4.2 Subpackage: cartesian3

**BBox:** Berechnet und zeichnet die Kanten eines Quaders um die Dokumente in der Darstellung.

**BFrame:** Berechnet und zeichnet einen semitransparenten Quader zur besseren Orientierung um die Dokumente.

**CartesianJ3DRenderer:** Hauptklasse dieses Renderers, organisiert Dialogelemente sowie die grafische Anzeige.

**CartesicMappingDialog:** Dialogklasse zur Spezifizierung der Abbildungseigenschaften. Es wird eine Matrix aus Schlüsselwörtern und darstellbaren Eigenschaften aufgebaut. Der Benutzer legt in dieser Matrix fest, welches Schlüsselwort welcher grafischen Eigenschaft (Position, Größe, Farbe, ...) zugeordnet werden soll.

**CooCross:** Blendet ein dreidimensionales Koordinatensystem zur besseren räumlichen Orientierung in die Darstellung ein.

**ObjectTypeDialog:** Stellt einen Dialog zur Einstellung des grafischen Objekts für die Repräsentation der Dokumente bereit.

**PickDocumentBehavior:** Handler zur Ermittlung des entsprechenden Dokuments anhand der Bildschirm-Koordinaten bei Selektion mit der Mouse.

### D.4.3 Subpackage: Gravity2

**DocTypeDialog:** In dieser Klasse werden die Typen des grafischen Objekts für die Repräsentation der Dokumente eingestellt.

**DragMouseAdapter:** Handler, der bei Bewegung eines Schlüsselwort-Icons mit der Mouse die Neuordnung der Dokument-Icons auslöst.

**DrawArea:** Panel für die grafische Ausgabe und Mouse-Interaktionen.

**ExtendedDocument:** Erweiterte Dokument-Klasse, die zusätzlich grafische Informationen enthält, die z. B. bei der Selektion und der damit verbundenen Rückerkennung von Dokumenten benötigt werden.

**KeywordTypeDialog:** Mit dieser Klasse wird der Typ des grafischen Objekts für die Repräsentation der Dokumente eingestellt.

**PicMouseAdapter:** Handler zur Ermittlung des entsprechenden Dokuments anhand der Bildschirm-Koordinaten bei Selektion mit der Mouse.

**Gravity2DRenderer:** Hauptklasse dieses Renderers, organisiert Dialogelemente sowie die grafische Anzeige.

# Anhang E

# XML-Strukturen

Dieser Anhang enthält die DTDs (Document Type Descriptions), die zur Definition der Schnittstelle zwischen Visualisierungsmodul und Suchmaschine notwendig sind, sowie Beispiele für Suchergebnisse und Arbeitsstände in XML-Notation. Die Erläuterung zu den Codefragmenten wurde im Abschnitt 5.9 vorgenommen.

```
<!ELEMENT search_results (creation_date, keyword_list, document_list+)>
    <!ELEMENT creation_date (#PCDATA)>
    <!ELEMENT keyword_list (keyword+)>
        <!ELEMENT keyword (#PCDATA)>
    <!ELEMENT document_list (document+)>
        <!ELEMENT document (relevances+)>
            <!ATTLIST document
            title CDATA #REQUIRED
            url CDATA #REQUIRED
            author CDATA #IMPLIED
            published CDATA #IMPLIED
            date CDATA #IMPLIED
            size CDATA #IMPLIED
            valid (yes | no) "yes">
            <!ELEMENT relevances (rvalue+)>
                <!ELEMENT rvalue (#PCDATA)>
```

Listing 1: search.dtd – Dokumentbeschreibung eines Suchergebnisses

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE search_results SYSTEM "search.dtd">
<search_results>
    <creation_date>12.02.2001</creation_date>
    <keyword_list>
        <keyword>Informationsraum</keyword>
        <keyword>Interaktion</keyword>
        <keyword>Navigation</keyword>
        <keyword>Visualisierung</keyword>
    </keyword_list>
    <document_list>
        <document
            title="This is document number one"
            author="First author"
            url="C:\DocVis\Documents\doc1.html"
            published="in proceedings number one"
            date="29-Jun-1998"
            size="768000">
            <relevances>
                <rvalue>0.2</rvalue>
                <rvalue>0.6</rvalue>
                <rvalue>0.4</rvalue>
                <rvalue>0.5</rvalue>
            </relevances>
        </document>
        <document
            title="This is document number two"
            author="Second author"
            url="C:\DocVis\Documents\doc2.html"
            date="12-Dec-1999"
            size="23785">
            <relevances>
                <rvalue>0.8</rvalue>
                <rvalue>0.2</rvalue>
                <rvalue>0.3</rvalue>
                <rvalue>0.4</rvalue>
            </relevances>
        </document>
    </document_list>
</search_results>
```

Listing 2: search.xml – Suchergebnis in XML-Form

```
<!ELEMENT search_results (creation_date, keyword_list, document_list+, modification_date,
            filter+, renderer, idoclist+)>
    <!ELEMENT creation_date (#PCDATA)>
    <!ELEMENT keyword_list (keyword+)>
        <!ELEMENT keyword (#PCDATA)>
    <!ELEMENT document_list (document+)>
        <!ELEMENT document (relevances+)>
            <!ATTLIST document
            title CDATA #REQUIRED
            url CDATA #REQUIRED
            author CDATA #IMPLIED
            published CDATA #IMPLIED
            date CDATA #IMPLIED
            size CDATA #IMPLIED
            valid (yes | no) "yes">
            <!ELEMENT relevances (rvalue+)>
                <!ELEMENT rvalue (#PCDATA)>
    <!ELEMENT modification_date (#PCDATA)>
    <!ELEMENT filter (author_filter+, public_filter+, date_filter, size_filter,
                redundancy_filter, threshold_filter+)>
        <!ELEMENT author_filter (author_list+)>
            <!ATTLIST author_filter
                active (yes | no) #REQUIRED
                type (inclusion | exclusion) #REQUIRED>
            <!ELEMENT author_list (author_name+)>
                <!ELEMENT author_name (#PCDATA)>
        <!ELEMENT public_filter (publication_list)>
            <!ATTLIST public_filter
                active (yes | no) #REQUIRED
                type (inclusion | exclusion) #REQUIRED>
            <!ELEMENT publication_list (location+)>
                <!ELEMENT location (#PCDATA)>
        <!ELEMENT date_filter (#PCDATA)>
            <!ATTLIST date_filter
                active (yes | no) #REQUIRED
                mindate CDATA #IMPLIED
                maxdate CDATA #IMPLIED>
        <!ELEMENT size_filter (#PCDATA)>
            <!ATTLIST size_filter
                active (yes | no) #REQUIRED
                size CDATA #IMPLIED
                type (less | more) #REQUIRED>
        <!ELEMENT redundancy_filter (#PCDATA)>
            <!ATTLIST redundancy_filter
                active (yes | no) #REQUIRED>
        <!ELEMENT threshold_filter (threshold_list)>
            <!ATTLIST threshold_filter
                active (yes | no) #REQUIRED>
            <!ELEMENT threshold_list (thvalue+)>
               <!ELEMENT thvalue (#PCDATA)>
    <!ELEMENT renderer (#PCDATA)>
    <!ELEMENT idoclist (document+)>
```

Listing 3: all.dtd – Dokumentbeschreibung eines Zwischenergebnisses

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE search_results SYSTEM "all.dtd">
<search_results>
    <creation_date>12.02.2001</creation_date>
    <keyword_list>
        <keyword>Informationsraum</keyword>
        <keyword>Interaktion</keyword>
        <keyword>Navigation</keyword>
        <keyword>Visualisierung</keyword>
    </keyword_list>
    <document_list>
        <document
            title="This is document number one"
            author="First author"
            url="C:\DocVis\Documents\doc1.html"
            published="in proceedings number one"
            date="29-Jun-1998"
            size="768000">
            <relevances>
                <rvalue>0.2</rvalue>
                <rvalue>0.6</rvalue>
                <rvalue>0.4</rvalue>
                <rvalue>0.5</rvalue>
            </relevances>
        </document>
        <document
            .....
        </document>
    </document_list>
    <modification_date>21.03.2001</modification_date>
    <filter>
        <author_filter active="no" type="inclusion">
            <author_list>
                <author_name>Albert Einstein</author_name>
                <author_name>Konrad Zuse</author_name>
            </author_list>
        </author_filter>
        <public_filter active="no" type="inclusion">
            <publication_list>
                <location>Proceedings of ...</location>
                <location>Lexikon der ...</location>
            </publication_list>
        </public_filter>
        <date_filter active="no" mindate="1.1.1990" maxdate="31.12.2001">
        </date_filter>
        <size_filter active="no" size="10000" type="more">
        </size_filter>
        <redundancy_filter active="yes">
        </redundancy_filter>
        <threshold_filter active="yes">
            <threshold_list>
                <thvalue>0.2</thvalue>
                <thvalue>0.3</thvalue>
                <thvalue>0.25</thvalue>
                <thvalue>0.4</thvalue>
            </threshold_list>
        </threshold_filter>
    </filter>
    <renderer>cartesian 2D</renderer>
    <idoclist>
        <document
            title="This is document number three"
            author="Third author"
            url="C:\DocVis\Documents\doc3.html"
            date="23-Mar-1999"
            size="23785">
            <relevances>
                <rvalue>0.2</rvalue>
                <rvalue>0.3</rvalue>
                <rvalue>0.7</rvalue>
                <rvalue>0.4</rvalue>
            </relevances>
        </document>
     </idoclist>
</search_results>
```

Listing 4: all.xml – Arbeitsstand einer Session in XML-Form

# Anhang F

# Glossar

**Aktor:** ist ein außerhalb des Systems Agierender, der an der in einem Anwendungsfall beschriebenen Interaktion beteiligt ist. Aktoren nehmen in der Interaktion gewöhnlich eine definierte Rolle ein. Es sind Benutzer des Systems oder andere Systeme. Ein Aktor ist eine stereotypisierte Klasse.

**Analyse:** Die Analysephase dient im Rahmen der objektorientierten Softwareentwicklung der Ermittlung und Definition der Anforderungen an ein Softwaresystem. Ergebnis der Analysephase ist eine Leistungsbeschreibung des zu entwickelnden Systems in Form eines objektorientierten Modells.

**Anwendungsfall:** beschreibt im Klartext eine Menge konsistenter und zielgerichteter Interaktionen von Aktoren mit einem System, an deren Ende ein definiertes Ergebnis entsteht ist. Ein Anwendungsfall wird stets durch einen Aktor initiiert und ist eine komplette, unteilbare Beschreibung. Anwendungsfälle beschreiben das gewünschte externe Systemverhalten aus der Sicht und in der Sprache der Anwender und somit Anforderungen, die das System zu erfüllen hat. Beschrieben wird, was es leisten muß, aber nicht wie es dies leisten soll.

**Anwendungsfalldiagramm:** Ein Diagramm, das die Beziehungen zwischen Aktoren und Anwendungsfällen zeigt.

**Browser:** sind Programme, welche Daten aus dem weltweiten Netz (von HTTP-Servern) abrufen und dann am Computer (Client) verarbeiten und anzeigen. Mit Hilfe der Querverweise im Hypertext-Format (HTML) werden die Dokumente im World Wide Web miteinander verknüpft. Neben Text beherrschen moderne Browser - z.T. mit Hilfe sogenannter PlugIns, AddOns oder Viewern - auch die Anzeige von Grafiken, Videoclips und weiteren Datenformaten. Oftmals unterstützen Browser auch FTP und Gopher, können E-Mails versenden und für Videokonferenzen und als Newsreader eingesetzt werden.

**Design:** Mit (objektorientiertem) Design werden alle Aktivitäten im Rahmen des Softwareentwicklungsprozesses bezeichnet, mit denen ein Modell logisch und physisch strukturiert wird und die dazu dienen zu beschreiben, „wie“ das System die in der Analyse beschriebenen Anforderungen erfüllt.

**Flash:** Ein Flash ist ein Programm der Firma Macromedia zum Erstellen von vektorbasierten Animationen auf Webseiten.

**Fish-Eye:** ist eine Metapher, die auf die optischen Eigenschaften einer Art von extrem weitwinkligen fotografischen Spezialeffekt-Objektiven anspielt, bei denen das Zentrum des Blickfeldes

übertrieben groß und die Randbezirke, im Gegensatz dazu, übertrieben klein abgebildet werden.

**Fokus+Kontext-Verfahren:** Als Fokus+Kontext Verfahren bezeichnet man eine Klasse von Visualisierungskonzepten, die strukturierte Informationen so darstellen, daß eine möglichst große Menge relevanter Informationen gleichzeitig auf dem Bildschirm angezeigt werden kann. Dabei wird die Information im Fokusbereich in höchstmöglichem Detail gezeigt, während im Kontextbereich die restlichen Informationen verkleinert dargestellt werden.

**Glyph:** Ein Objekt oder Symbol, welches Datenwerte repräsentiert. Insbesondere lassen sich dabei simultan mehrere Datenwerte darstellen. Ein einfacher, häufig verwendeter Glyph ist der Pfeil, welcher z. B. zur Visualisierung von Vektorfeldern verwendet werden kann. Mit seiner Hilfe lassen sich Windgeschwindigkeit und -richtung in einem Punkt darstellen.

**HTML:** Abkürzung für HyperText Markup Language. HTML ist eine Beschreibungssprache zur Strukturierung von Dokumenten. Die HTML-Dokumente werden mittels des HTTP-Protokolls transferiert und bilden so die Basis des WWW. Die Sprache definiert sowohl die Gestaltung, den Inhalt und die Grafik der Seite als auch die Links (Hyperlinks, Verbindungen) zu eigenen oder fremden Seiten.

**Internet:** Weltweit größtes Computernetzwerk, das aus miteinander verbundenen Netzwerken besteht. Das Internet unterstützt viele verschiedene Services. Die wichtigsten sind u.a.: Telnet für den Aufruf von Programmen auf anderen Computern, FTP (File Transfer Protocol) für die Übertragung von Files auf andere Computer, Electronic Mail (elektronische Post), Usenet /Newsgroups (Veröffentlichungen in Diskussionsforen) und IRC (Internet Relay Chat) für den Austausch von Informationen mit anderen Computer-Benutzern, WWW für den Zugriff auf Informationssysteme in aller Welt.

**Internetressource:** Mit Ressource werden im Allgemeinen die im Internet verfügbaren Informationen gemeint: Dateien, WWW-Seiten, Nachrichten usw. Auf eine Ressource kann entweder direkt über eine Netzadresse oder über eine Verzweigung von einer anderen Ressource aus zugegriffen werden.

**Klasse, OO-Klasse:** repräsentiert eine generische Beschreibung einer Menge von Objekten, mit gemeinsamen Attributen, Operationen, Relationen und Semantik. Dabei werden Verhalten, Zustand und Aktionen der Instanzen in genereller Art und Weise beschrieben.

**Klassendiagramm:** zeigt eine Menge statischer Modellelemente, vor allem Klassen und ihre Beziehungen.

**Kollaborationsdiagramm:** zeigt eine Menge von Interaktionen zwischen einer Menge ausgewählter Objekte in einer bestimmten begrenzten Situation (Kontext) unter Betonung der Beziehungen zwischen den Objekten und ihrer Topographie.

**Komponente:** ist ein ausführbares Softwaremodul mit eigener Identität und wohldefinierten Schnittstellen (Sourcecode, Binärcode, DLL oder ein ausführbares Programm).

**Meta-Tags:** stehen im Header eines HTML Dokuments und werden vom Browser nicht angezeigt. In Meta-Tags lassen sich zum Beispiel Schüsselwörter und eine kurze Zusammenfassung des Seiteninhalts definieren. Die meisten Suchmaschinen verwenden diese Informationen, um eine Seite in ihren Datenbestand einzuordnen und für die Anzeige der Suchergebnisse.

**Methode:** beschreibt ein Vorgehen bei der Lösung einer bestimmten Aufgabe. In der UML ist eine Methode die Implementierung einer Operation.

**Notation:** ist eine Menge von Symbolen, die zur Darstellung von Konzepten nach bestimmten Regeln dienen.

**Nicht-euklidische Geometrie:** zeichnet sich von der herkömmlichen euklidischen Geometrie durch eine verzerrte Winkel- und Längenmessung aus. Diese verzerrte Maßbestimmung führt unter anderem dazu, daß in hyperbolischer Geometrie die Winkelsumme im Dreieck kleiner als 180 % ist, oder daß es Pflasterungen der Ebene mit regulären 5-Ecken gibt.

**Objektorientierte Analyse:** (siehe Analyse)

**Objektorientiertes Design:** baut auf dem Modell der objektorientierten Analyse auf und verfeinert es aus Sicht der Realisierung. Ergebnis dieser Phase der objektorientierten Softwareentwicklung ist ein Designmodell.

**Szene:** umfaßt sämtliche für das Rendering benötigten Informationen wie z. B. die Positionen der Visualisierungsobjekte (Modelle), Lichtquellen und Kameras.

**Sequenzdiagramm:** zeigt eine Menge von Interaktionen zwischen einer Menge ausgewählter Objekte in einer bestimmten begrenzten Situation (Kontext) unter Betonung der zeitlichen Abfolge. Ähnlich dem Kollaborationsdiagramm. Sequenzdiagramme können in generischer Form (Beschreibung aller möglichen Szenarien) existieren oder in Instanzform (Beschreibung genau eines speziellen Szenarios).

**Rendering:** Im Grafikbereich versteht man unter Rendering die grafische Darstellung eines dreidimensionalen rechnerinternen Modells mittels computerunterstützter Prozesse /Algorithmen. Dazu können beliebige Lichtquellen positioniert sowie Farben bzw. Texturen und weitere Effekte zugeordnet werden.

**WWW:** Das Web World Wide ist ein weltweiter, verteilter und Hypertextbasiertet Informationsdienst im Internet. Im WWW wird das HTTP-Protokoll verwendet, das es ermöglicht, Informationen als HTML-Dokumente zu versenden. Im Rahmen der Gestaltung von HTML-Dokumenten können diese Dokumente besonders übersichtlich und häufig auch grafisch aufbereitet dargestellt werden. WWW kann als Teilnetz des Internet verstanden werden. Der Zugriff auf die Informationen erfolgt über WWW-Browser - Netscape, Mosaic oder Internet Explorer.

# Anhang G

# Abkürzungsverzeichnis

**API:** Application Programming Interface

**ARB:** Architectural Review Board

**API:** Application Programming Interface

**CAD:** Computer Aided Design

**CRT:** Cathode Ray Tube

**DAG:** Directed Acyclic Graph

**DTD:** Document Type Definition

**DBMS:** Datenbankmanagementsystem

**FSN:** File System Navigator

**FTP:** File Transfer Protocol

**GIS:** Geografische Informationssysteme

**GLX:** OpenGL Extension to the X Window System

**GMD:** Gesellschaft für Mathematik und Datenverarbeitung

**GUI:** Graphical User Interface

**HMD:** Head-Mounted Display

**HTML:** HyperText Markup Language

**HSV:** Hue-Saturation-Value

**HTTP:** HyperText Transfer Protocol

**IPSI:** Integrierte Publikations- und Informationssysteme

**KOAN:** Kontext Analysator

**MR:** Minimal Reality

**NIST:** National Institute of Standards and Technology

**OOA:** Objektorientierte Analyse

**OOD:** Objektorientiertes Design

**OOI:** Object of Interest

**OOSA:** Object Oriented Systems Analysis

**OOSE:** Object-Oriented Software Engineering

**OMG:** Object Management Group

**OML:** Object Modeling Technique

**OMT:** Object Modelling Technique

**OpenGL:** Open Graphics Library

**RGB:** Red-Green-Blue

**POI:** Points of Interest

**PARC:** Palo Alto Research Center

**SQWID:** Search Query Weighted Information Display

**UML:** Unified Modeling Language

**UIR:** User Interface Research

**URL:** Uniform Ressource Lokator

**VIBE:** Visual Interface Browsing Environment

**ViSC:** Visualisation in Scientific Computing

**VR:** Virtuelle Realität

**VisIT:** Visualization of Information Tool

**VRML:** Virtual Reality Modeling Language

**XML:** eXtensible Markup Language

**WGL:** Windows Graphics Library

**WML:** Wireless Markup Language

**WWW:** World Wide Web

# Literaturverzeichnis


[Are00] Aregger A., Przygienda A.: *Visualisierung von großen Datenmengen im Finanzbereich.* Internetseminar SS 2000 Institut für Informatik Universität Zürich, Mai 2000.

[Aut94] Authoren-Kollektiv: *Brockhaus-Enzyklopädie.* Verlag F.A. Brockhaus GmbH, Band 23: ISBN 3-7653-1123-5, 1994.

[B.92] B., Shneiderman (Herausgeber): *Tree Visualization with Tree-maps: A 2-d Space-Filling Approach.* Department of Computer Science and HCI Laboratory, University of Maryland, January 1992.

[B.01] B., Shneiderman (Herausgeber): *Treemaps for space-constrained visualization of hierarchies.* Department of Computer Science and HCI Laboratory, University of Maryland, November 2001.

[Bal99] Balzert H.: *Lehrbuch der Objektmodellierung.* Spektrum Akademischer Verlag, 1999.

[Bar96a] Barth R.: *Graphische Datenverarbeitung III, „Spezielle Verfahren der Computer Graphik“.* Vorlesungsunterlagen: Institut für Praktische Informatik und Medieninformatik, Technische Universität Ilmenau, 1996.

[Bar96b] Barth R., Beier E., Pahnke B.: *Grafikprogrammierung mit OpenGL.* Addison–Wesley Verlag GmbH, ISBN:3-89319-975-6, 1996.

[Bau99] Baumert J. Reinbach K.: *Anwendbarkeit der software-ergonomischen Kriterien zur Reduzierung erzwungener Sequentialität für die Bewertung und das Redesign von Dialogen aus dem SAP R/3 System.* Diplomarbeit, Integrierter Studiengang Informatik, Universität Paderborn Fachbereich 17(Mathematik/Informatik), Dezember 1999.

[Bec97] Becker Ferenc: *Internet-Suchmaschinen, Funktionsweise und Beurteilung.* Diplomarbeit, Hochschule für Bibliotheks- und Informationswesen, April 1997.

[Beh98] Behme H., Mintert S.: *Extensible Markup Language (XML) 1.0.* http://www.edition-w3c.de/TR/1998/REC-xml-19980210.html, 1998.

[Bek97] Bekavac B., Rittberger M. (Herausgeber): *Kontextsensitive Visualisierung von Suchergebnissen.* Konferenz: Hypertext - Information Retrieval - Multimedia 1997 (HIM ’97) Dortmund, 29. September - 2. Oktober, 1997.

[Bek01] Bekavac B.: *Tutorial zur Suche im WWW (Version 2.0).* Informationswissenschaft an der Universität Konstanz, Fachbereich Informatik und Informationswissenschaft, Mai 2001.

[Ben95] Benford, S., Snowdon, D., Greenhalgh, C., Inrgam, R., Knox, I., and Brown, C. (Herausgeber): *VR-VIBE: A Virtual Environment for Co-operative Information Retrieval*. Proceedings of Eurographics'95, 30th August - 1st September, Maastricht, The Netherlands, pp 349-360., 1995.

[Boo94] Booch G.: *Objekt-Orientierte Analyse und Design*. Addison-Wesley, Bonn, 1994.

[Boo96] Booch G.: *Best of Booch - Designing Strategies for Object Technology*. Eykholt Ed, SIGS Reference Library, SIGS Books, New York, 1996.

[Cal96] Caligari Corporation: *Yahoo! and Caligari Corporation Announce Yahoo! 3D, A New 3D World Opened for 3D Enthusiasts and Mainstream Web Users*. www.caligari.com, November 1996.

[Car95] Carriere J., Kazman, R. (Herausgeber): *Interacting with Huge Hierarchies: Beyond Cone Trees*. Proceedings of the IEEE Symposium on Information Visualization, Atlanta, pp. 74-81, October 1995.

[Car97a] Carey R., Bell G.: *The Annotated VRML 2.0 Reference Manual*. Addison-Wesley Pub Co; ISBN: 0201419742, 1997.

[Car97b] Carey R., Bell G. und Marrin C.: *ISO/IEC 14772-1:1997 Virtual Reality Modeling Language (VRML97)*. URL: http://www.vrml.org/Specifications/VRML97, 1997.

[Cha94] Charwat, H.J.: *Lexikon der Mensch-Maschine-Kommunikation*. München; Wien : Oldenbourg, ISBN 3-486-22618-5, 1994.

[Cha98] Chang-Sung J. und Pang A. (Herausgeber): *Reconfigurable Disc Trees for Visualizing Large Hierarchical Information Space*. Proceedings of IEEE InfoVis '98, (October 19-20, Research Triangle, North Carolina, USA), Computer Society Press, 19-25., 1998.

[Chi94] Chimera R. and Shneiderman B. (Herausgeber): *An exploratory evaluation of three interfaces for browsing large hierarchical tables of contents*. ACM Transactions on Information Systems, October 1994.

[Cla97] Claussen U.: *Programmieren mit OpenGL: 3D-Grafik und Bildverarbeitung*. Springer-Verlag, ISBN 3-540-57977, Berlin Heidelberg New York, 1997.

[Coa94] Coad P., Yourdan E.: *Objektorientiertes Design*. Prentice Hall: Englewood Cliffs, NJ, 1994.

[Cug00] Cugini J., Laskowski S. (Herausgeber): *Design of 3D Visualization of Search Results: Evolution and Evaluation*. Proceedings of IST/SPIE's 12th Annual International Symposium: Electronic Imaging 2000: Visual Data Exploration and Analysis (SPIE 2000), January 2000.

[Dä98] Dässler R., Palm H.: *Virtuelle Informationsräume mit VRML: Informationen recherchieren und präsentieren in 3D*. dpunkt-Verlag, Heidelberg, ISBN 3-920993-78-0, 1998.

[Dä99] Dässler R. (Herausgeber): *Informationsvisualisierung, Stand, Kritik und Perspektiven*. In Methoden/Strategien der Visualisierung in Medien, Wissenschaft und Kunst; Wissenschaftlicher Verlag Trier (WVT); Fachhochschule Potsdam, 1999.

[Dom00] Domik G.: *Tutorial on Visualization.* Computer-generated Visualization, University of Paderborn, Department of Computer Science, 2000.

[Enc88] Encarnação J, Strasser W.: *Computer Graphics.* Oldenburg Verlag, 1988.

[Eng95] Englberger H.: *Computergestützte Informationsvisualisierung: Eine Klassifikation aktueller Techniken und ihre Einsatzpotentiale für die Unternehmung.* Diplomarbeit, Technische Universität München, Fakultät für Informatik, November 1995.

[Erl00] Erler T.: *Das Einsteigerseminar UML.* Verlag BHV, Kaarst-Bttgen, ISBN 3-8287-1027-2, Dezember 2000.

[Fai93a] Fairchild M. K.: *Information Management Using Virtual Reality-Based Visualizations.* In:„Virtual Reality: Applications and Explorations“ Alan Wexelblat (ed.), Academic Press Professional, Cambridge, MA, ISBN 0-12-745045-9, 1993.

[Fai93b] Fairchild M. K., Serra L., Hern Ng, Hai L. B., Leong T.: *Dynamic fisheye information visualizations.* In Alan Wexelblat, editor, Virtual Reality: Applications and Explorations, Academic Press, pp 45-74, 1993.

[Fer00] FernUniversität: *BibRelEx: Erschließung Bibliographischer Datenbanken durch Visualisierung von inhaltsbasierten Beziehungen.* Technischer Bericht, FernUniversität Hagen, Gesamthochschule in Hagen, Praktische Informatik VI, März 2000.

[Fol90] Foley J., van Dam A, Feiner S. and Hughes J.: *Computer Graphics, Principles and Practice.* Second Edition, Addison-Wesley, 1990.

[For01] Forbrig P.: *Objektorientierte Softwareentwicklung mit UML.* Fachbuchverlag Leipzig im Carl Hanser Verlag, ISBN 3-446-215727, 2001.

[Fra97] Frazier C., Kempf R. : *Opengl Reference Manual: The Official Reference Document to Opengl, Version 1.1.* Addison Wesley Professional, ISBN: 0-201-46140-4, 1997.

[Fur86] Furnas, G. W.: *Generalized Fisheye Views.* Proceedings of CHI 86, 1986.

[Gre92] Green M. und White L.: *Minimal Reality Toolkit Version 1.2 Programmer's Manual.* Department of Computing Science, University of Alberta, 1992.

[H.97] H., Schauer: *Grundlagen und Überblick „Information“.* Skript zur Vorlesung Informatik I: Universität Zürich, Institut für Informatik, 1997.

[Hem93a] Hemmje M. (Herausgeber): *LyberWorld - eine 3D-basierte Benutzerschnittstelle für die computerunterstützte Informationssuche in Dokumentmengen.* In: GMD-Spiegel, pp. 56-63, 1993.

[Hem93b] Hemmje M., Kunkel C. und Willet A. (Herausgeber): *A 3D Based User Interface for Information Retrieval Systems.* In: Proceedings of IEEE Visualization '93, Workshop on Database Issues for Data Visualization, San Jose, California, October 25-29, 1993.

[Hem94] Hemmje M., Kunkel C. und Willet A. (Herausgeber): *LyberWorld - A Visualization User Interface Supporting Fulltext Retrieval.* Proceedings of the 17th annaul international ACM SIGIR conference on Research and development in information retrieval. Dublin, Ireland, 1994.

[Hem97] Hemmje M. Leissler M.: *Portierung und Erweiterung der Relevanzkugelmetapher um interaktive Informationsvisualisierungsmechanismen.* ISBN 3-88457-321-7, September 1997.

[Her00] Herman I, Melanon G, Marshall M. S. (Herausgeber): *Graph Visualisation and Navigation in Information Visualisation: A Survey.* IEEE Transactions on Visualization and Computer Graphics, 6(1), pp. 24-43, 2000.

[Hon97] Honkela, T., Kaski, S., Lagus, K., and Kohonen: *Newsgroup exploration with WEBSOM method and browsing interface.* Technischer Bericht, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1997.

[Inx00] Inxight Software: *Inxight Star Tree StudioTM.* http://www.inxight.com, 2000.

[IX 98] IX Magazin: *Sun will Java 3D und VRML integrieren.* Magazin für professionelle Informationstechnik; News, Februar 1998.

[J.95] J., Rumbaugh (Herausgeber): *OMT: The Development Process.* JOOP (Journal of Object-Oriented Programming), May 1995.

[J.96] J., Rumbaugh (Herausgeber): *OMT: Insights - Perspectives on Modeling from the Journal of Object-Oriented Programming.* Rational Software Corp., SIGS Referenz Library, SIGS Book,New York, 1996.

[Jac92] Jacobson I. et al.: *Object-Oriented Software Engineering: A Use Case Driven Approach (ObjectOry).* ACM Press, Addison-Wesley, 1992.

[Jac98] Jacobson I., Booch G. and Rumbaugh J.: *The Objectory Software Development Process.* Addison-Wesley Object Technology Series, 1998.

[Joh91] Johnson B. und Shneiderman B. (Herausgeber): *Treemaps: A space-filling approach to the visualization of hierarchical information structures.* Proceedings of the 2nd International IEEE Visualization Conference, San Diego, pp. 284-291, October 1991.

[Keh97] Kehoe Colleen: *Visualizing Search Results: Three Aproaches.* College of Computing, Georgia Institute of Technology, Atlanta, September 1997.

[Kei90] Keil-Slawik R.: *Konstruktives Design: Ein ökologischer Ansatz zur Gestaltung interaktiver Systeme.* Doktorarbeit, Forschungsberichte des Fachbereichs Informatik, Bericht Nr. 90-14, TU Berlin, 1990.

[Kei98] Keil-Slawik R.: *Leitprinzipien und Gestaltungsbeispiele für die Entwicklung von Multimediasystemen.* Workshop zur Mediengestaltung, Einführung in die Gestaltung von Multimedia-Anwendungen, Hagen, 30.09.98 - 01.10.98, 1998.

[Koi93] Koike, H. and Yoshihara H. (Herausgeber): *Fractal Approaches for Visualizing Huge Hierarchies.* Proceedings of 1993 IEEE/CS Symposium on Visual Languages, pp. 55-60, 1993.

[Kon96] Konrad–Zuse–Zentrum: *Wissenschafliche Visualisierung.* Technischer Bericht, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1996.

[Lag96a] Lagus K., Honkela T., Kaski S., und Kohonen T. (Herausgeber): *Self-organizing maps of document collections: A new approach to interactive exploration.* In Simoudis, E., Han, J., and Fayyad, U., editors, Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, pp. 238-243. AAAI Press, Menlo Park, California., 1996.

[Lag96b] Lagus K., Honkela T., Kaski S., und Kohonen T. (Herausgeber): *WEBSOM - A Status Report.* In Alander, J., Honkela, T., and Jakobsson, M., editors, Proceedings of STeP'96, Finnish Artificial Intelligence Conference, pp. 73-78; Vaasa, Finland, 1996.

[Lam95] Lamping J., Rao R, Pirolli P. (Herausgeber): *A focus+context technique based on hyperbolic geometry for visualizing large hierarchies.* Proceedings of CHI95 Conference on Human Factors in Computing Systems. Denver, pp. 401-408, 7-11 May, 1995.

[Mar93] Marmann, M.: *Datenmodellierungskonzepte für Hypermedia und ihre Abbildung in Datenbanken.* Doktorarbeit, Fernuniversität in Hagen, 1993.

[McC87] McCormick, B., DeFanti T.A., Brown M.D. (Herausgeber): *Visualization in Scientific Computing.* ACM SIGGRAPH, Computer Graphics Vol.21, No.6, November 1987.

[McC97] McCrickard D. Scott, Kehoe Colleen M. (Herausgeber): *Visualizing Search Results using SQWID.* In: Sixth International World Wide Web Conference, Santa Clara, CA, April 7-11, 1997.

[McG01] McGookin D. Brewster S. (Herausgeber): *FISHEARS-The Design of a Multimodal Focus and Context System.* Science University of Glasgow, Department of Computing Science of Glasgow,G12 8QQ UK, November 2001.

[Mor98] Morse Dr. E.: *Document Visualization.* National Institute of Standards und Technology (NIST), Information Technology Laboratory, 1998.

[Mü98] Müller U., Leissler M., Hemmje M.: *Entwurf und Implementierung eines generischen Mechanismus zur dynamischen Einbettung multimedialer Daten in VRML–Szenen auf der Basis eines objektrelationalen DBMS.* GMD Forschungszentrum Informationstechnik, GmbH. - Sankt Augustin, ISBN 3-88457-347-0, 1998.

[Mul97] Mullet K., Fry C., Schiano D. (Herausgeber): *The Great CHI'97 Browse Off.* Proceedings of the CHI 1997 Conference on Human Factors in Computing Systems (CHI '97), Atlanta, Georgia, USA, 22-27 March, 1997.

[Neu95] Neumann, Horst A.: *Objektorientierte Softwareentwicklung von Software- Systemen.* Addison-Wesley, ISBN 3-89319-452-5, 1995.

[Neu98] Neumann, Horst A.: *Objektorientierte Softwareentwicklung mit der Unified Modeling Language (UML).* Carl Hanser Verlag, ISBN 3-446-18879-7, 1998.

[Neu01] Neumüller, M.: *Hypertext Semiotics in the Commercialized Internet.* Doktorarbeit, Wirtschaftsuniversität Wien, 2001.

[Oes98] Oestereich Bernd: *Objekt-orientierte Sofware-Entwicklung: Analyse und Design mit der Unified Modeling Language.* R. Oldenbourg Verlag, ISBN: 3-486-24787-5, 1998.

[Ort93] Ortmanns S.: *Modelle der Kooperation von Graphik-Workstations und Höchstleistungsrechnern.* Bericht des Forschungszentrums Jülich, ISBN 0366-0885, 1993.

[Pau] Paul Hensgen: *Umbrello UML Modeller.* http://uml.sourceforge.net.

[Phi00] Philippus Thomas: *Informationssuche im Internet: Tips für Profis.* VDE-Verlag GMBH, ISBN 3-8007-2214-3, 2000.

[Pot98] Potempa T., Franke P., Osowski W. Schmidt M.: *Informationen finden im Internet; Leitfaden für die gezielte Online-Recherche.* Carl Hanser Verlag München Wien, ISBN 3-446-19242-5, 1998.

[R.98] R., Schumann: *VRML-Grundlagen.* Seminararbeit, Lehrstuhl für Praktische Informatik IV, Universität Mannheim, 1998.

[Ray01] Ray, E. T.: *Einführung in XML.* O'Reilly Verlag GmbH & Co.KG, 1. Auflage, ISBN 3-89721-286-2, 2001.

[Rek93] Rekimoto J. und Green M. (Herausgeber): *The Information Cube: Using Transparency in 3D Information Visualization.* Proceedings of the Third Annual Workshop on Information Technologies & Systems (WITS-93), pp. 125-132, 1993.

[RJ94] Rumbaugh J., Blaha M., et al: *Objektorientiertes Modellieren und Entwerfen.* Carl Hanser Verlag/Prentice-Hall International, 1994.

[Rö00] Röber N., Möller T., Celler A.M., Strothotte T. (Herausgeber): *Multidimensional Analysis and Visualization Software for dynamic SPECT.* J. Nucl. Med, Otto-von-Guericke Universität, Magdeburg, 2000.

[Rö01] Rösner D.: *Dokumentverarbeitung.* Vorlesungsunterlagen, Otto-von-Guericke- Universität Magdeburg, Fakultät für Informatik, 2001.

[Rob91a] Robertson G., Mackinlay Jock D. und Card Stuart K. (Herausgeber): *The Perspective Wall: Detail und context smoothly integrated.* In: Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems, pp. 173-179, 1991.

[Rob91b] Robertson, G.G., Mackinlay, J.P. und Card, S.K. (Herausgeber): *Cone Trees: Animated 3D Visualization of Hierarchical Information.* In: Proceedings of ACM Conference of Human Factors and Computing Systems (SIGCHI91), New Orleans, Louisiana, April 28-May 2 ACM Press, pp. 189-194, 1991.

[Rob92] Robertson G., Card Stuart K. und Mackinlay J. (Herausgeber): *The Information Visualizer: A 3D User Interface for Information Retrieval.* In: Advanced Visual Interfaces: Proc. of the International Workshop AVI'92, Advanced Visual Interfaces. World Scientific, Singapore, pp. 173-179, 1992.

[Rob93] Robertson G., Card Stuart K. und Mackinlay J. (Herausgeber): *Information Visualization Using 3D Interactive Animation.* Readings in Information Visualization Using Vision To Think, pp. 515-569, April 1993.

[Rom97] Romann M. Hemmje M (Herausgeber): *Erweiterung der Relevanzkugelmetapher um eine interaktive Partitionierungsfunktionalität.* In: GMD-Studien Nr 323, GMD - Forschungszentrum Informationstechnik, Sankt Augustin, September 1997.

[SC92] Shaw Chris, Liang J. Green M. und Sun Y. (Herausgeber): *The decoupled simulation model for virtual reality systems.* In Proceedings of ACM CHI '92 Conference Proceedings, 1992.

[Sch94] Scheller M., Boden K.-P., Geenen A., Kampermann J.: *Internet: Werkzeuge und Dienste.* Springer Verlag Berlin Heidelberg, 1994.

[Sch99] Schwartz D. (Herausgeber): *Mehr Information durch Visualisierung von Daten?: Konventionelle und innovative Visualisieurungstechniken.* Zeitschrift für Bibliothek-Information-Technoligie-Online (B.I.T. Online), Ausgabe, April 1999.

[SH00] Schumann H., Müller W.: *Visualisierung-Grundlagen und allgemeinen Methoden.* Springer-Verlag ISBN:3-540-64944-1, 2000.

[Shl88] Shlaer S. and Mellor St.: *Object-Oriented System Analysis: Modeling the World in Data.* Yourdon Press, Englewood Cliffs, New Jersey, 1988.

[Sil91] Silicon Graphics, Inc.: *Graphic Library Programming Guide.* Technischer Bericht, Silicon Graphics, Inc., Mountain View, California., 1991.

[Sow98] Sowizral H., Rushforth K., Deering-Reading M., Mass. [u.a.]: *Java 3D API Specification.* Addison-Wesley, Version 1.1.2, ISBN 0-201-32576-4, 1998.

[Ste98] Stein C.: *Ein konzeptioneller Vergleich der 3 D-Graphik- Schnittstellen OpenGL, Java 3D und VRML hinsichtlich ihrer Eignung für die Erstellung von Animationen.* Diplomarbeit, Phlipps-Universität Marburg, Fachbereich Mathematik und Informatik, Version 1.1.2, Dezember 1998.

[Ste01] Stein, M.: *Workshop XML.* Addison-Wesley, ISBN: 3-8273-1867X, 2001.

[Sun97a] Sun Microsystems: *The Java 3D–API Technical White Paper.* http://java.sun.com/products/java-media/3d/, 1997.

[Sun97b] Sun Microsystems: *Java 3D API Tutorial.* http://java.sun.com/products/java-media/3d/, 1997.

[SUN99] SUN Microsystems: *Java 3D API Spezifikation, Version 1.2 Alpha1.* Sun Microsystems, Inc. All rights reserved.; http://java.sun.com, August 1999.

[Tra97] Tracey S.: *Search Engines Corner, Alta Vista LiveTopics.* University of Leeds Library, UK, 1997.

[Vic87] Vicente, K. J., Hayes, B. C., und Williges, R. C. (Herausgeber): *Assaying and isolating individual differences in searching a hierarchical file system.* Human Factors, 29(3), 349-359, 1987.

[Wea96] Weal M.: *3D Interfaces to Hypermedia Information Spaces.* Diplomarbeit, University of Southampton, 1996.

[Wie98] Wiechert Holger: *3D-Visualisierungstechniken zur Navigation in Hypertextstrukturen - Eine vergleichende Bewertung nach software-ergonomischen Kriterien.* Diplomarbeit, Universität, Gesamthochschule Paderborn, Dezember 1998.

[Wir01] Wirsam W.: *VisualXXL: eine grafische 3D-Schnittstelle für XXL.* Diplomarbeit, Fachbereich Informatik und Mathematik der Philipps- Universität Marburg, April 2001.

[Wol96] Wolf P: *Three-Dimensional Information Visualisation: The Harmony Information Landscape.* Diplomarbeit, Technische Universität Graz, Institut für Informationsverarbeitung und Computergestützte neue Medien (IICM), 1996.

[Wü97] Wünsche V.: *Visualisierung strukturierter Informationen mit VRML.* Diplomarbeit, Universität Rostock, Fachbereich Informatik, 1997.

[Yah96] Yahoo Company Information: *YAHOO! AND CALIGARI CORPORATION ANNOUNCE YAHOO! 3D.* Yahoo Media Relation, October 1996.

[You96] Young P.: *Three Dimensional Information Visualisation.* Technischer Bericht, Computer Science Technical Report, No. 12/96, Visualisation Research Group, Department of Computer Science University of Durham, 1996.

[Zol96] Zoller B.: *Supertips Internet-Werkzeuge.* Bürohandels- und Verlagsgesellschaft GmbH ISBN: 3-89360-379-4, 1996.

# Tabellenverzeichnis

# Abbildungsverzeichnis

# Index