Werner König
Mat. 01/458695
Universität Konstanz

Konzeption und Implementation eines 3D-Scatterplots zur Visualisierung von Metadaten

Bachelor - Arbeit

Information Engineering
Fachbereich Informatik und Informationswissenschaften

1. Gutachter: Prof. Dr. Harald Reiterer
2. Gutachter: Prof. Dr. Oliver Deussen

Vorgelegt von
Werner A. König

Kurzfassung (Deutsch)

Der Nutzen von dreidimensionalen Visualisierungen wird oft durch eine erhöhte kognitive und mentale Belastung des Anwenders, verursacht durch eine komplizierte räumliche Navigation und missverständliche Darstellung herabgesetzt. Diese Belastung versucht der in dieser Arbeit thematisierte 3D-Scatterplot durch eine zweckgebundene vereinfachte Interaktion und eine unterstützende Visualisierung zu reduzieren, um damit dem Anwender eine effiziente und intuitive Umgebung zur Erfüllung seiner Aufgaben zu bieten. Dieser 3D-Scatterplot ist eine Visualisierung des generischen Visuellen Metadaten Browsers (VisMeB), der unabhängig von der Anwendungsdomäne den Benutzern bei der Suche und Extraktion von relevanten Daten aus einer großen Datenmenge effizient unterstützen soll und im Rahmen der Forschungsprojekte INVISIP und VisMeB entwickelte wurde. In dieser Arbeit werden zusätzlich zur Vorstellung des VisMeB-Projektes und der detaillierten Beschreibung der Konzeption, Implementation und Funktionsweise des 3D-Scatterplots in einer State-of-the-Art-Analyse bestehende Systeme mit 3D-Scatterplots aus dem kommerziellen und wissenschaftlichen Umfeld aufgeführt und ihre Besonderheiten herausgestellt. Basierend auf den Erfahrungen mit der Konzeption und Implementation des hier vorgestellten 3D-Scatterplots und im Vorfeld erstellten Studien, wurden darüber hinaus Richtlinien definiert, welche als Leitfaden für die Konzeption einer derartigen Visualisierung und als Kriterium zur Qualitätsprüfung dienen sollen.

Abstract (English)

The benefit of three-dimensional visualizations is often reduced because of an increased cognitive and mental workload for the user, imposed by a complicated spatial navigation and a mistakable display. The 3D-Scatterplot, which is described in this bachelor thesis, tries to lower this workload by a task orientated, simplified interaction and a supporting visualization. The goal is to offer the user an efficient and intuitive environment, in which he can easily solve his tasks. This 3D-Scatterplot is part of the generic Visual Metadata Browser (VisMeB) which was developed in the context of the research projects INVISIP and VisMeB. It was designed to efficiently support users in a way that they are able to search and extract relevant data from a large amount of data regardless of the context of use. In addition to the presentation of the VisMeB project and the detailed description of the conceptual design, the implementation and the functionality of the 3D-Scatterplot, this thesis also presents existing commercial and scientific applications with 3D-Scatterplots in a state-of-the-art analysis. Based on the experience made during the conception and implementation of this 3D-Scatterplot and further studies, this thesis also defines a style guide which should be used as a guideline for the conception of similar visualizations and as a criteria for quality audits.

Danksagung

Grundlegend für diese Arbeit war die Möglichkeit an den Forschungsprojekten INVISIP und VisMeB aktiv mitarbeiten zu können. Hierfür möchte ich allen an diesen Projekten Beteiligten für ihren Einsatz und die gute Zusammenarbeit danken. Mein herzlichster Dank gilt Herrn Prof. Dr. Harald Reiterer, welcher mir Einblicke in das wissenschaftliche Umfeld ermöglichte und daran teilhaben ließ.

Des Weiteren möchte ich mich auch bei seinen Mitarbeitern für ihre Vorschläge und konstruktive Kritik bedanken. Sie gaben mir die Möglichkeit, selbständig zu arbeiten, eigene Ideen zu entfalten und intensive Projekterfahrungen zu sammeln.

Besonderer Dank gilt meinem Mitstudenten und Teamkollegen Philipp Liebrenz für die sehr produktive und zugleich amüsante Zusammenarbeit während des gesamten Studiums und der verschiedenen Projektphasen. Weiterhin gebührt mein Dank allen Mitstudenten, die mich mit Kollegialität und Freundschaft in meinem Studium begleitet haben.

Nicht zuletzt möchte ich mich bei meiner Familie auf das Herzlichste bedanken, die mich zu jeder Zeit mit Vertrauen und Rückhalt unterstützten und mir die Freiheit gaben, meine Ziele zu verfolgen. Ebenso bin ich Kathrin Schötzer für die moralische Unterstützung und das mir gegenüber aufgebrachte Verständnis sehr dankbar.

Inhalt

1 Motivation

„We need to separate out the features that make 3D useful and understand how they help overcome the challenges of dis-orientation during navigation and distraction from occlusion" [Shne03].

Dieses Zitat von Ben Shneiderman charakterisiert treffend die Problematik beim Einsatz von drei Dimensionen in Visualisierungen. 3D-Visualisierungen bieten dem Anwender ohne Zweifel bei bestimmten Aufgaben gewisse Vorteile gegenüber 2D-Darstellungen, wie beispielsweise die intuitivere Wahrnehmung von räumlichen Daten. Oft werden diese aber von einer verwirrenden Navigation und unklarer Darstellung überschattet und stellen daher den Nutzen von 3D-Visualisierungen infrage.

Auch Jakob Nielsen verdeutlicht die Schwierigkeiten der 3D-Navigation in seinem Artikel "2D is Better Than 3D": *„Users need to pay attention to the navigation of the 3D view in addition to the navigation of the underlying model: the extra controls for flying, zooming, etc. get in the way of the user's primary task" [Niel98].*

Ein Ansatz zur Lösung der Problematik ist, dem Anwender nur die für seine Aufgabe unbedingt notwendigen und sinnvollen Navigationsmöglichkeiten anzubieten und damit die Komplexität der Interaktion zu minimieren. Soll zum Beispiel ein Gegenstand im Raum visualisiert werden, ist anzunehmen, dass es den Benutzer weniger fördert, wenn seine Position im Raum fixiert ist und nur der Gegenstand manipuliert werden kann, als wenn er sich zusätzlich zur Manipulation des Gegenstandes auch noch virtuell völlig frei im Raum bewegen könnte.

In dieser Arbeit wird ein 3D-Scatterplot thematisiert, welcher versucht, die Vorteile, die sich aus der Darstellung mit drei Dimensionen ergeben können, im Umfeld der Ergebnisvisualisierung von Suchanfragen zu nutzen und ihre Nachteile durch geeignete Interaktionsmechanismen und unterstützende Darstellung zu reduzieren.

Ebenso werden Richtlinien zur Konzeption eines „guten" 3D-Scatterplots angeführt, welche aus Recherche, Konzeption, Implementation und Anwendung des VisMeB 3D-Scatterplots resultieren und als Orientierung für Neukonzipierungen und als Qualitätskriterium für Bewertungen eingesetzt werden können.

2 Definitionen

2.1 Scatterplots

Unter einem *Scatterplot* versteht man eine Visualisierung von Beziehungen zwischen bestimmten Variablen eines Datenbestandes.

Jeder Datensatz wird in der Regel durch einen Punkt oder ein anderes visuelles Objekt in einem kartesischen Koordinatensystem, welches durch die entsprechenden Daten-Dimensionen aufgespannt wird, repräsentiert. Die Koordinaten jedes Punktes im Scatterplot korrespondieren zu den jeweiligen Werten der Variablen entsprechend der Achsenbelegungen.

Daraus resultiert eine Visualisierung mit zerstreuten Datenpunkten, auch *Punktwolke* genannt – engl. *Scatterplot*.

Scatterplots unterscheidet man je nach Anzahl der dargestellten Dimensionen bzw. Achsen üblicherweise in zwei Arten: 2D- und 3D-Scatterplots.

Beim 2D-Scatterplot besteht das kartesische Koordinatensystem aus genau zwei Achsen, welche mit den jeweiligen Variablen des Datenbestandes belegt werden. 2D-Scatterplots sind im Allgemeinen als Scatterplots bekannt und finden in den verschiedensten Bereichen häufige Anwendung.

3D-Scatterplots dagegen werden derzeit noch selten und gegebenenfalls meist nur als zusätzliche Darstellung eingesetzt. Sie visualisieren Daten mit Hilfe eines dreiachsigen Koordinatensystems, welches sich in einem virtuellen dreidimensionalen Raum befindet und auf 2D-Anzeigegeräte projiziert wird.

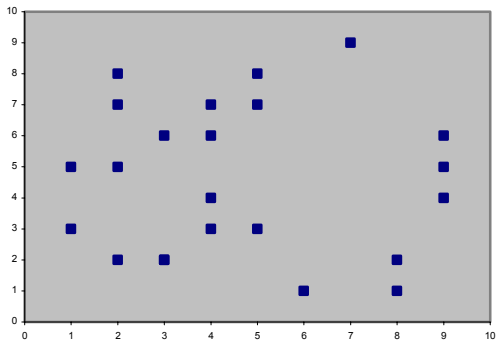


Abbildung 2.1: 2D-Scatterplot, generiert mit MSt Excel.

Der 3D-Raum bietet als Interaktionen zumeist das Zooming sowie das freie Rotieren des Koordinatensystems über die jeweiligen Achsen, so dass der Scatterplot und dessen Daten aus jeglichem Blickwinkel betrachtet werden können.

Die sich aus der erhöhten Dimensionalität beim 3D-Scatterplot ergebende zusätzliche Achse ermöglicht nun drei Ausprägungen auf die verschiedenen Achsen und weitere, wie auch beim 2D-Scatterplot, auf Form, Farbe, Größe etc. zu legen und infolge dessen direkt zu vergleichen und zu bewerten.

Scatterplots sind eine der ältesten und verbreitetsten Methoden Daten zu visualisieren und Cluster, Korrelationen, Ausreißer und Ähnliches zu erkennen.

2.2 MultiDataPoints

Besitzen mehrere der in einem Scatterplot dargestellten Datensätze die gleichen Werte bezüglich der auf die Achsen abgebildeten Variablen, so liegen ihre Repräsentanten auf der gleichen Position und damit übereinander. Das führt dazu, dass die Objekte nicht mehr voneinander unterschieden werden können.

Bei großen Datenmengen erhöht sich die Wahrscheinlichkeit dieses Effektes und kann die Benutzbarkeit der Visualisierung erheblich vermindern. Die MultiDataPoint-Technik verhindert wirksam die genaue Überlagerung von Datenrepräsentanten.

Ein *MultiDataPoint* (MDP) vereint die sich überdeckenden Repräsentanten zu einem neuen Objekt, das geeignete Interaktionsmöglichkeiten zur Verfügung stellt, um die eingeschlossenen Daten bei Bedarf zu explorieren.

Für die Visualisierung der verschiedenen Datenobjekte innerhalb eines MDP gibt es die unterschiedlichsten Ansätze: Sie können von einfachen Listen der überlappenden Elemente bis hin zu animierten Darstellungen reichen.

Bei dem in dieser Arbeit beschriebenen 3D-Scatterplot können die Objekte eines MDP in einer neuen Visualisierung exploriert werden. Dafür bewegen sich die einzelnen Repräsentanten auf einer Ellipsenbahn, wobei zum jeweils vordersten Objekt nähere Informationen angezeigt werden.
Auf diese Visualisierung wird in Kapitel 5.2.7 näher eingegangen.

2.3 Metadaten

Unter *Metadaten* versteht man strukturierte Daten über Daten oder Objekte, mithilfe derer das Beschriebene besser auffindbar und elektronisch verarbeitbar gemacht wird.

Meta kommt aus dem Griechischen und bedeutet „jenseits, über“. Metadaten können also als Daten höherer Ordnung bzw. grundlegenderer Art aufgefasst werden.

Der Begriff Metadaten scheint ein Produkt der heutigen Zeit zu sein, das Konzept dahinter ist aber tatsächlich schon lange bekannt. So konnte und kann man auch zum Teil heute noch in Bibliotheken den Standort und andere Informationen von Büchern anhand von Karteikarten, welche Angaben über Titel, Autor, Erscheinungsjahr, Standort etc. beinhalten, ausfindig machen.

Ein Grund hierfür liegt in der Erleichterung des Suchvorgangs. Für einen Suchenden ist es wesentlich einfacher, kleine Karteikarten zu durchsuchen, als jedes Buch einzeln aus den Regalen zu nehmen und die gewünschten Informationen nachzuschlagen.

Dieser Vorteil liegt auch in der Struktur der Daten. Würden die Informationen auf den Karteikarten als Fließtext vorliegen, wäre der Aufwand, das richtige Buch zu finden, erheblich gesteigert.

Die hohe strukturelle Ordnung ist auch bei der elektronischen Verarbeitung von Metadaten von großer Bedeutung, da ein automatischer Algorithmus immer noch ungenügende Resultate bei der Unterscheidung von wichtigen und unwichtigen Informationen im Fließtext liefert und diese nur schwer in den Kontext setzen kann.

Daher wurden Standards zur Anfertigung von Metadaten festgelegt. Der Bekannteste dürfte der *Dublin Core Metadata Element Set* [DCM03] sein. Dieser definiert 15 Elemente, welche das zu beschreibende Dokument näher spezifizieren und auch durch ihre Einfachheit von Laien angegeben werden können. So soll eine weite Verbreitung ermöglicht werden. Doch bisher werden dieser und auch andere Standards nur selten verwendet.

Eine klare Abgrenzung von Daten und Metadaten ist nicht immer möglich. Im gewissen Kontext können Metadaten wiederum als Daten verstanden werden, welche von anderen Metadaten beschrieben werden. Werden zum Beispiel von verschiedenen Anbietern Filmdaten durch Metadaten, wie Titel und Inhalt beschrieben, können die Metadaten der Filmdaten selber als Daten und die Informationen zu den angebotenen Metadaten wieder als Metadaten verstanden werden.

3 3D-Scatterplots – Stand der Technik

In diesem Kapitel werden beispielhaft kommerzielle und akademische Produkte vorgestellt, in denen 3D-Scatterplots schon zur Anwendung kommen.
Es soll ein Überblick über bisher eingesetzte Visualisierungs- und Interaktionstechniken im Zusammenhang mit 3D-Scatterplots geben und deren Aufgaben und Einsatzgebiete angeführt werden.

3.1 NIRVE

Zwischen 1995 und 2000 entstand *NIRVE* (NIST Information Retrieval Visualization Engine) [CLS00] als Forschungsprojekt von John Cugini und Marc Sebrecht von der „Visualization and Virtual Reality Group“ des „National Institute of Standards and Technology“ (NIST). NIRVE visualisiert Ergebnisse von Suchanfragen auf die PRISE-Suchmaschine [PRIS03] mithilfe von verschiedenen Text-, 2D- und 3D-Darstellungen. Dabei liegt der Schwerpunkt der Visualisierung eher darauf, dem Anwender einen Gesamtüberblick über die Dokumentenmenge zu liefern, als ihn bei der Suche nach einem speziellen Dokument zu unterstützen. Die PRISE-Suchmaschine liefert auf eine Stichwortanfrage auf die zugrunde liegende Datenbank mit 90.000 Dokumenten, Berichte aus der „Associated Press“ bis 1988, pro Dokument den Dokumententitel, Dokumenten-ID, Relevanz, Dokumentenlänge und Vorkommnisse der Stichworte an NIRVE. Typischerweise wird die Ergebnismenge bei NIRVE auf die 100 – 500 treffendsten Dokumente reduziert.

Die Oberfläche von NIRVE ist zweigeteilt: Ein Fenster mit der jeweiligen Visualisierung der Suchergebnisse, auch „Document Space“ genannt und ein kleineres Kontrollfenster mit Dialogelementen zur Manipulation der Suchanfrage und der Visualisierung oder sonstigen Funktionen der Anwendung, die nicht direkt im „Document Space“ verfügbar sind (s. Abbildung 3.1).

Die Dokumente im „Document Space“ können mit verschiedenen Visualisierungsmodellen angezeigt werden:

- „Spiral Design“,
- „3-D Axes Model“,
- „Nearest Neighbor Circle Model“,
- „Spoke and Wheel Design“,
- „Concept Globe Model“ und
- „2.5-D Design“.

Wir beschränken uns hier auf die Beschreibung des „3-D Axes Model“, welches eine sehr interessante Umsetzung eines 3D-Scatterplots darstellt.

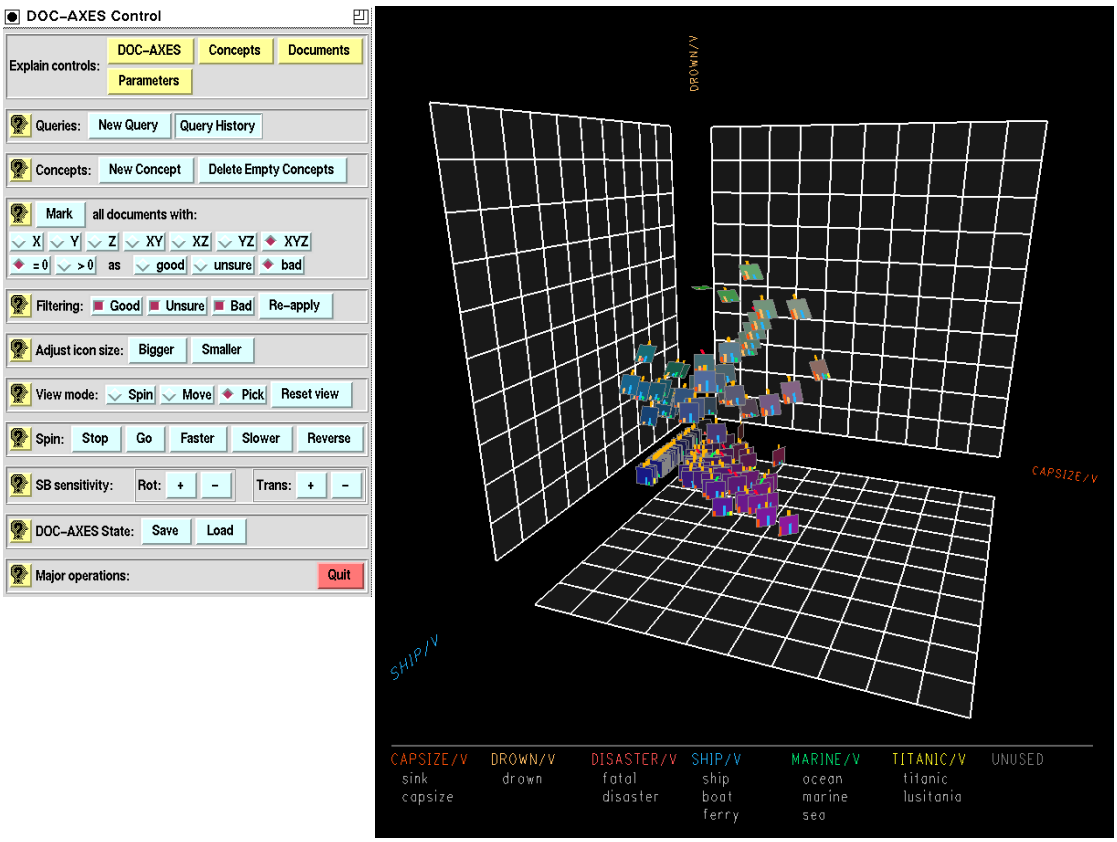


Abbildung 3.1: NIRVE Kontrollfenster und „3-D Axes Model“ [NIRVE03].

Beim *„3-D Axes Model“* werden die Relevanzen von drei dynamisch auszuwählenden Suchwörtern oder Suchgruppen auf die Achsen eines dreidimensionalen Koordinatensystems gelegt und die Dokumente an den entsprechenden Koordinaten positioniert.

Die Dokumente werden durch quadratische Icons, welche zum Ursprung hin ausgerichtet sind, repräsentiert. Jedes Icon besteht aus einem Balkendiagramm, welches über die relative Frequenz der Suchworte Auskunft gibt. Die Achsen selber werden nicht angezeigt, sondern sie werden nur durch leicht nach außen versetzte Gitternetze auf der xy-, xz- und yz-Ebene angedeutet.

Mithilfe des Kontrollfensters oder direkter Interaktion per Maus und Tastatur kann das Koordinatensystem gedreht oder der Betrachtungswinkel und der Abstand verändert werden.

Der angenommene Vorteil des „3-D Axes Model“ soll sich aus der Möglichkeit zur direkten semantischen Interpretation der räumlichen Dimension ergeben.

Bei der räumlichen Anordnung treten vor allem bei größeren Datenmengen Überdeckungen der Icons auf, wobei die Unterscheidung der Dokumente dadurch erheblich erschwert wird. Es kann beim „3-D Axes Model“ auch meistens keine offensichtliche sequentielle Folge erkannt werden, in der die Dokumente durchgegangen werden können.

John Cugini et. al. folgerten im Januar 2000, dass es bei räumlichen 3D-Modellen, wie das „3-D Axes Model“, unwahrscheinlich ist, dass sie einfach vom Anwender zu verstehen und anzuwenden sind, außer es handelt sich schon um sehr strukturierte Daten, die dargestellt werden.

3.2 ViSta

ViSta (Visual Statistics System) wird an der University of North Carolina am Department of Psychology unter der Leitung von Professor Forrest W. Young seit 1990 entwickelt [ViSt03]. Es wird vom Entwickler auf Grund des verwendeten Multiple-View-Konzeptes und weiterer Visualisierungstechniken als ein hoch dynamisches und sehr interaktives visuelles Statistiksystem bezeichnet.

ViSta ist frei verfügbar, akademisch orientiert und vor allem für die Lehre und Forschung im Bereich Datenanalyse, multivariate und graphische Statistik konzipiert. Der Programmcode ist offen und bietet Schnittstellen zur Erweiterung der Funktionalität und Automatisierung von Abläufen durch Applets, Scripte oder Plug-ins.

Die Oberfläche von ViSta wird von vier Visualisierungstechniken geprägt:

- „GuideMaps“ sind visuelle Phasen-Diagramme, die Nutzern mit wenig oder keinen Kenntnissen im Bezug auf Datenanalyse beim Erfüllen ihrer Aufgabe helfen.
- „WorkMaps“ visualisieren die Struktur einer laufenden Session und ermöglichen dem Anwender zu bereits vollzogenen Arbeitsschritten zurückzukehren um alternative Analysen durchzuführen.
- Dynamische statistische Visualisierungstechniken sollen dem Betrachter die Datenstruktur und Analyseresultate effizient und präzise repräsentieren. Hierzu werden Teilvisualisierungen, auch „Spreadplots“ genannt, wie Scatterplots, Spinplots, Boxplots, Histogramme, Listen und andere Darstellungen zu einer verknüpften, dynamischen und interaktiven Gesamtvisualisierung zusammengefügt (Multiple View). Die „Spreadplots“ können aber auch bei Bedarf einzeln betrachtet werden.
- „Statistical Re-Vision techniques“ erlauben Änderungen der Analyseparameter und deren Folgen visuell zu untersuchen.

Eine Teilvisualisierung von ViSta ist der *Spinningplot*. Es ist ein Würfel, in dessen Schwerpunkt der Ursprung eines dreidimensionalen Koordinatensystems positioniert ist. Der Würfel kann auch ausgeblendet werden, so dass nur noch das Koordinatensystem mit den Daten sichtbar bleibt.

Die Daten werden durch Würfel an der der Variablenausprägung entsprechenden Stelle im Koordinatensystem dargestellt. Bei selektierten Daten sind die Würfel ausgefüllt.

Die Achsen des Koordinatensystems besitzen keine Beschriftung und behalten auch beim Zooming ihre Länge bei. Daher sind die absoluten Werte der Datenrepräsentation nicht ersichtlich. Es kann nur die Position der Datenwürfel relativ zueinander verglichen werden.

Mit den Kontrollleisten im Spinningplot-Fenster kann der Würfel kontinuierlich über verschiedene Achsen gedreht, gezoomt und die Ansicht zurückgesetzt werden. Wird in den Mouse-Modus umgeschaltet, kann der Würfel per Maus über x- und y-Achse gedreht und einzelne oder per „Brushing“-Modus gleich mehrere Daten selektiert werden.

Beim „Brushing“-Modus wird ein Rechteck als Mauszeiger sichtbar und alle Datenwürfel, die sich in der zweidimensionalen Projektion, d.h. am Bildschirm, hinter dem Rechteck befinden, werden dynamisch selektiert.

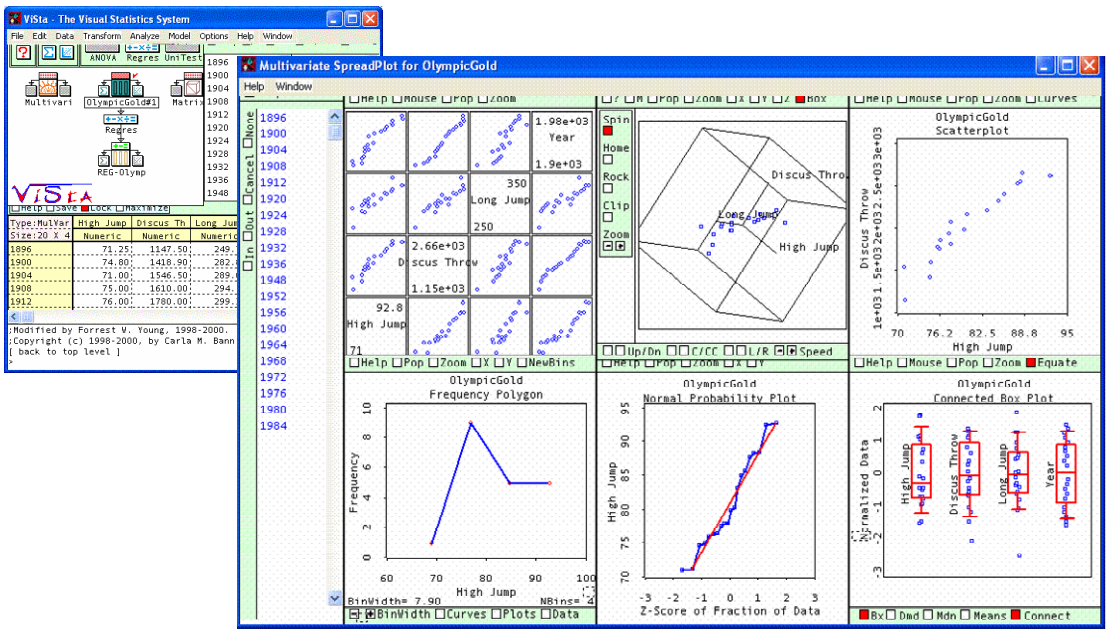


Abbildung 3.2: ViSta WorkMap mit multivariaten Spreadplots (OlympicGold-Daten).

3.3 WebWinds

WebWinds [EAGOW00] ist ein interaktives Visualisierungssystem für wissenschaftliche Daten und wurde am „Jet Propulsion Laboratory“ am „California Institute of Technology“, dem führenden Zentrum für Roboterforschung der NASA, seit 1998 entwickelt. WebWinds, wie auch sein Vorgänger LinkWinds, das jedoch nur für Unix- und Linux-Plattformen konzipiert ist, erhielten die interne Auszeichnung „NASA Software of the Year Award“ bzw. „Honorable Mention“.

WebWinds ist durch die Implementation in JAVA plattformunabhängig, kann sowohl lokale als auch Daten aus dem World Wide Web (WWW) einbinden und ist frei verfügbar. Die Haupteinsatzgebiete sind Wissenschaft und Lehre, vor allem im Bereich der atmosphärischen Forschung. Es wird aber auch von privaten und kommerziellen Anwendern zur Visualisierung und Analysierung von verschiedensten Daten eingesetzt.

Der modulare Aufbau von WebWinds soll eine einfache Benutzung per „drag and drop“ ermöglichen, wobei Tools, Applikationen, Visualisierungen und Datenbanken Objekte darstellen, die miteinander verbunden werden können. Als Visualisierungen stehen mehrere 2D- und 3D-Darstellungen zur Verfügung, wie zum Beispiel Histogram, Line Plot, IsoView, XYPlot, XYZPlot, Globe (Abbildung von 2D- oder 3D-Daten auf einen Globus) und andere.

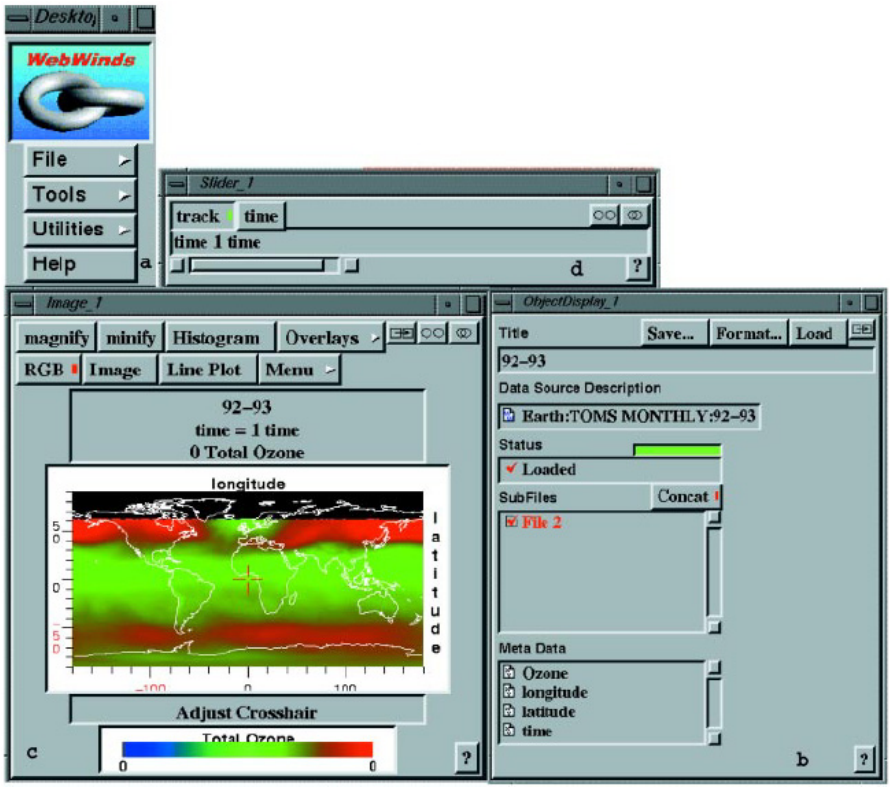


Abbildung 3.3: WebWinds mit Desktop-, Slider-, Image- und Display-Objekt [EAGOW00].

Der *XYZPlot* entspricht der allgemeinen Definition eines 3D-Scatterplots (siehe Abbildung 3.4). Er besteht aus einem dreidimensionalen Koordinatensystem, das durch beschriftete weiße Achsen auf schwarzem Hintergrund angezeigt wird. Die Daten, repräsentiert durch Kreuze, werden entsprechend ihrer Variablenausprägungen im Koordinatensystem positioniert. Die Farben der Kreuze entsprechen ihrer Position und sollen einen verstärkten räumlichen Eindruck beim Betrachter erwirken.

Zur Interaktion mit dem XYZPlot stehen drei Tools zur Verfügung: „Rotate“, „PanZoom“ und „Color“.

Das „Rotate“-Tool enthält zwei verschiedene Dialoge.
Beim „3AxisRotator“ kann das Koordinatensystem mit drei Slider, einer pro Achse, um die jeweilige Achse gedreht werden.

Der „2AxisRotator“ enthält einen zweidimensionalen Slider, mit dem je nach Bewegung des Sliderbuttons, das Koordinatensystem horizontal oder vertikal gedreht wird.

Beim „PanZoom“ kann mit dem zweidimensionalen Slider das Koordinatensystem verschoben und mit dem eindimensionalen Slider gezoomt werden. Mit dem „Color-Tool“ können die Farbpaletten verändert werden.

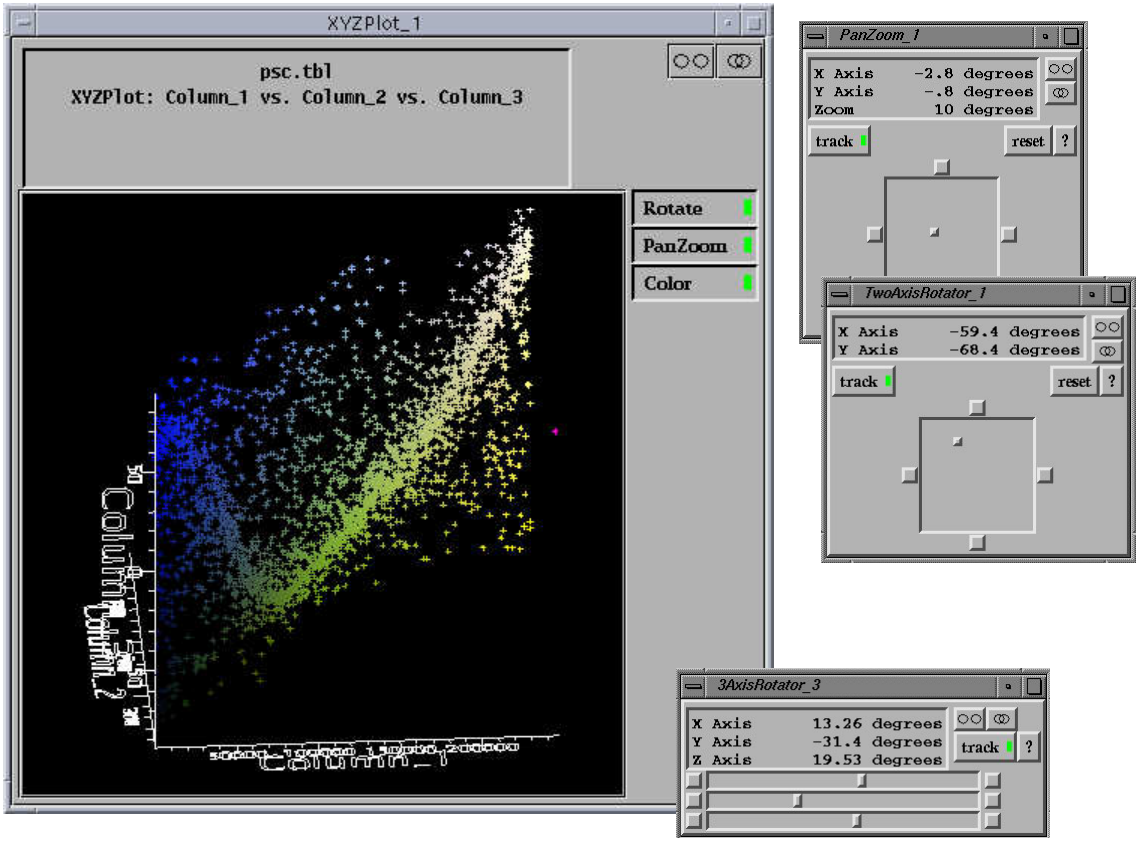


Abbildung 3.4: WebWinds XYZPlot mit PanZoom, TwoAxisRotator and 3AxisRotator [WebW03].

3.4 Spotfire

Spotfire [Spot03] ist ein kommerzielles Analysetool, welches die Datenanalyse und Entscheidungsfindung durch so genannte „Guides“ erheblich erleichtern und durch die Vielseitigkeit der interaktiven Visualisierungen verbessern soll. Für den gesamten Analysevorgang sind keine Programmier- und Datenbankkenntnisse, wie SQL oder die Anwendung von höherer Mathematik von Nöten, und kann bequem durch geführte Dialoge vollzogen werden.

Trotz der unterstützenden Bedienung ist Spotfire sehr anpassungsfähig und erweiterbar. Es wird in den verschiedensten Spezialgebieten, wie zum Beispiel Öl-, Chemie-, Pharma- und Halbleiterindustrie verwendet. Vielseitige, flexible Visualisierungen und direkte Kopplung erlauben eine schnelle und simultane Datenanalyse einer Vielzahl von Variablen.

IVEE (Information Visualization and Exploration Environment) wurde von Chris Ahlberg von der Chalmers University of Technology in Schweden entwickelt. Zum Vertrieb der Weiterentwicklung von IVEE namens Spotfire gründete C. Ahlberg 1996 die Spotfire Inc. Heute gilt Spotfire im Bereich „Guided Analytic Applications“ als Marktführer und wird auf der ganzen Welt von mehr als 80 international führenden Unternehmen eingesetzt. So verwenden zum Beispiel die 25 größten Pharmaunternehmen ausschließlich Spotfire.

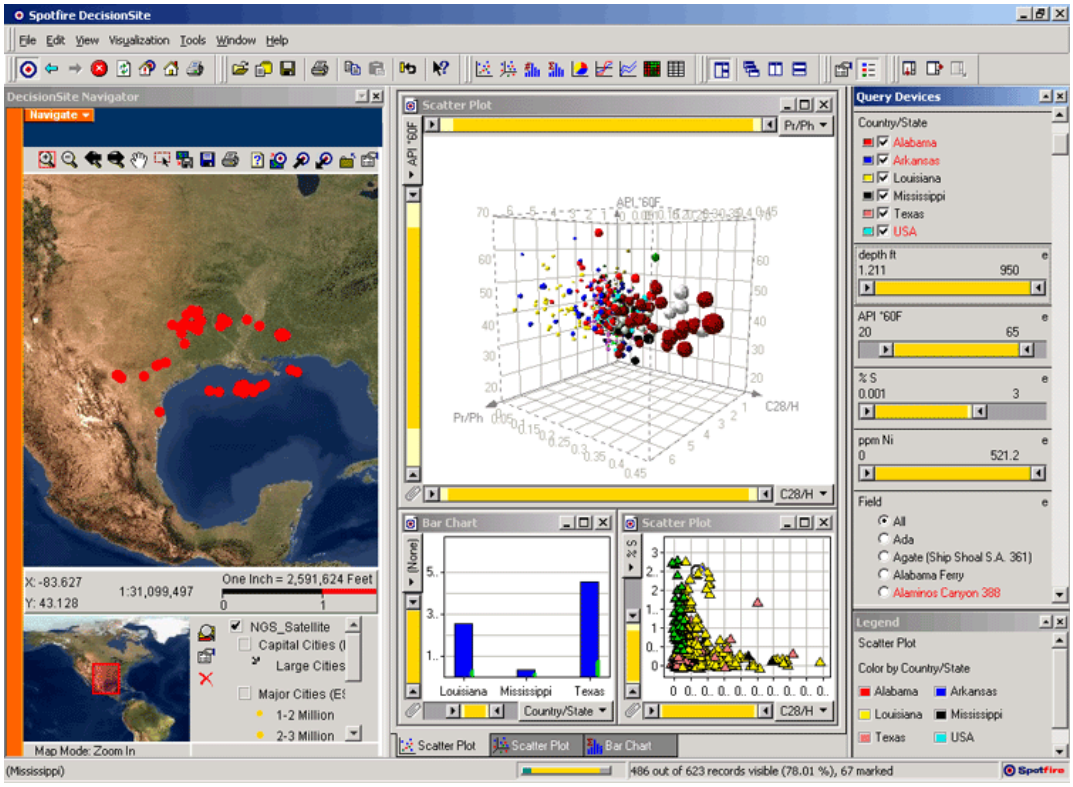


Abbildung 3.5: Spotfire beim typischen Einsatz in der Ölindustrie [Spot03].

Spotfire verbindet nach dem Multiple-View-Konzept mehrere Visualisierungen zu einer Gesamtdarstellung. Als Visualisierungen stehen 2D- und 3D-Scatterplots, Histogramme, Balken-, Kuchen- und Liniendiagramme, Profile charts, Trellis plots, Heat maps und Spreadsheet-Tabellen zur Verfügung. „Map Interaction Services“ zur räumlichen Einordnung von Geodaten oder Tools zur Bearbeitung und Anzeige von molekularen Strukturen sind weitere sehr spezielle Darstellungsinstrumente.

Der *3D-Scatterplot* wird als Würfel mit gestrichelten Kanten dargestellt, dessen hintere, untere und linke Fläche der xy-, xz- und yz-Ebene des inhärenten Koordinatensystems entsprechen. Diese Flächen werden durch Gitternetzlinien aufgeteilt und sind an den Kanten beschriftet (siehe Abbildung 3.6).

Die Daten werden durch kleinere Würfel oder Kugeln dargestellt. Ihre Farbe und Größe kann ebenso wie die Koordinatenachsen mit einer Variablen belegt werden. Durch die perspektivische Projektion des Würfels und der Datenrepräsentanten entsteht ein räumlich gut einschätzbarer dreidimensionaler Eindruck.

An der oberen, linken und unteren Seite des 3D-Scatterplots-Fensters befindet sich jeweils ein Pull-down-Menü zur Variablenbelegung der Achsen sowie ein Rangeslider, auch Alphaslider [AS94] genannt, mit welchen die angezeigte Datenmenge pro Achse im Wertebereich vom Maximal- und Minimalwert her eingeschränkt werden kann.

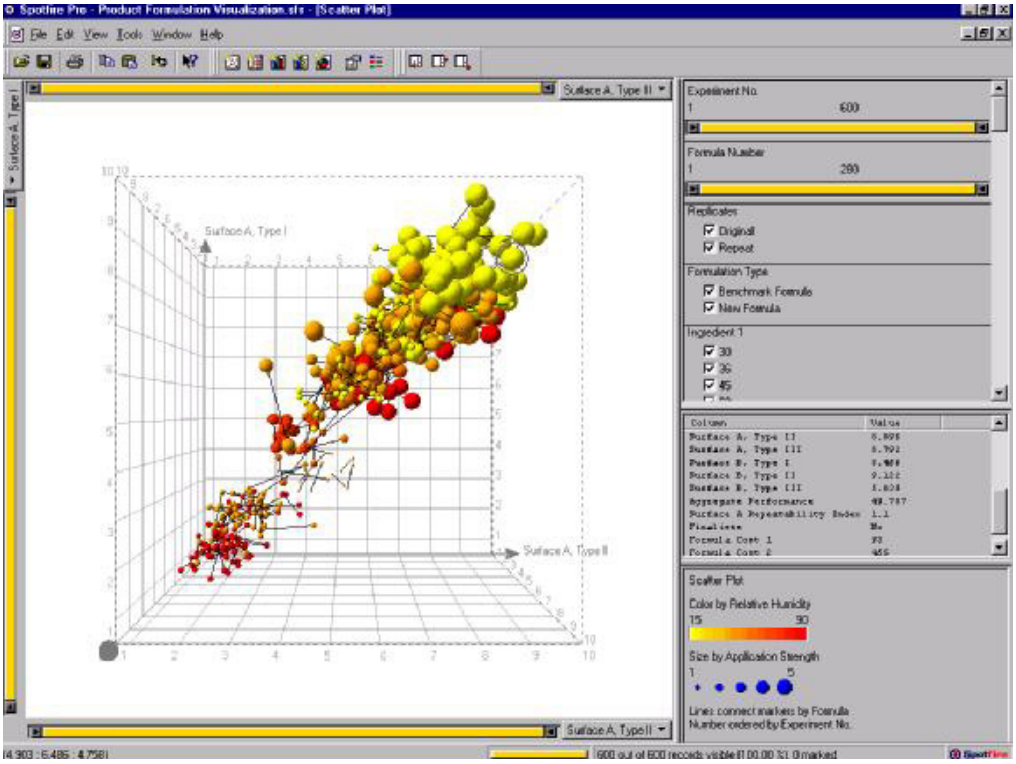


Abbildung 3.6: Spotfire, maximierter 3D-Scatterplot [Spot03].

3.5 Voxelplot

Der *Voxelplot* [VRVi03] ist ein aktuelles Forschungsprojekt des VRVis, dem Zentrum für Virtual Reality und Visualisierung in Wien. VRVis wurde im Januar 2000 als Kompetenzzentrum von mehreren Unternehmen und Universitäten zur disziplinübergreifenden Forschung gegründet und ist in Österreich bei der anwendungsorientierten Forschung im Bereich Virtual Reality und Visualisierung führend.

Der Voxelplot soll vor allem die Visualisierung von Daten unterstützen, welche den eigentlich bisher streng abgegrenzten Visualisierungsbereichen von Scientific und Infomation Visualization gleichermaßen zugeordnet werden können. Diese multidimensionalen Daten enthalten sowohl räumliche Informationen als auch abstrakte Daten, wie Temperaturen oder Druckverhältnisse (siehe Abbildung 3.7).

Die Daten werden durch mehrere koppelbare 3D-Scatterplots, welche verschiedene Achsenbelegungen aufweisen können, gleichzeitig dargestellt. Durch geeignete Interaktionsmöglichkeiten, wie farbliches Markieren und synchrone Bewegung, lassen sich Relationen in mehreren Darstellungen und somit auch in den Daten erkennen.

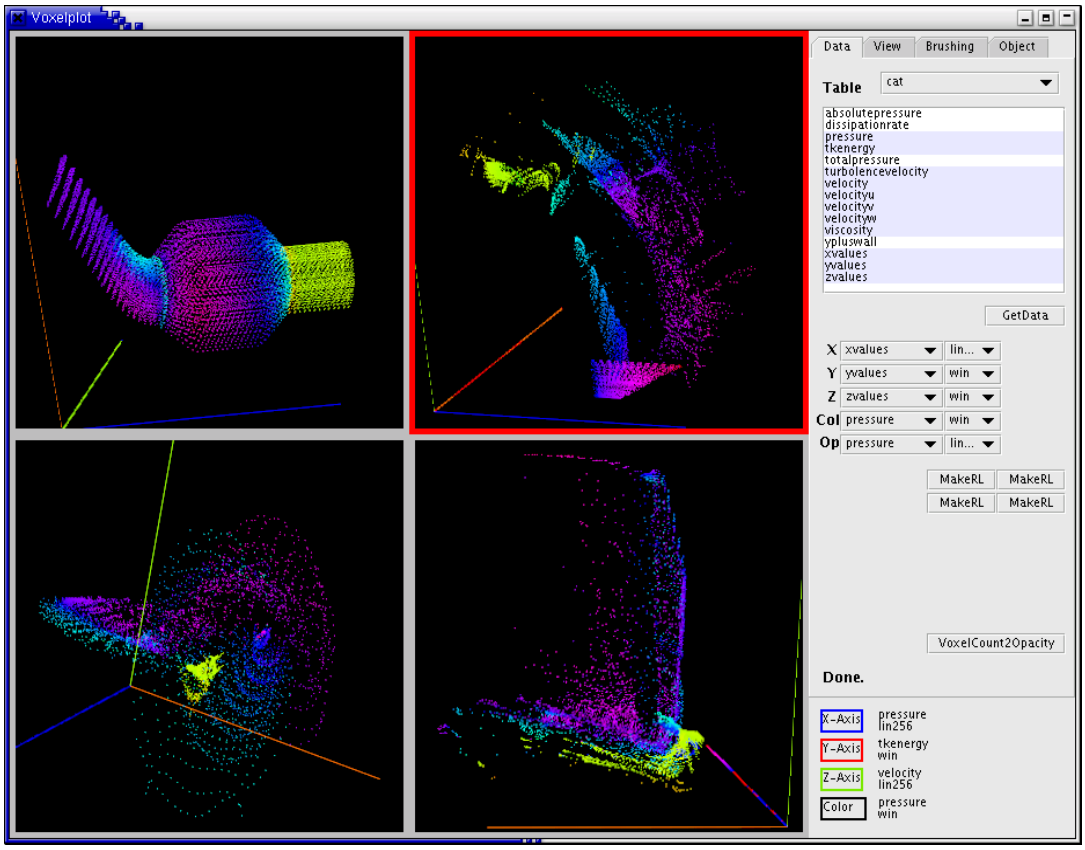


Abbildung 3.7: Voxelplot, Visualisierung von Messdaten an einem Katalysator [VRVi03].

Die *3D-Scatterplots* bestehen aus einem dreidimensionalen Koordinatensystem, dessen x-, y- und z-Achse farbig auf schwarzen Hintergrund dargestellt werden. Der Koordinatenraum wird in eine begrenzte Anzahl von Einheitswürfel, so genannte *Voxel*, eingeteilt. Die Anzahl der Voxel bestimmt die Auflösung der Visualisierung. Die Daten werden je nach Variablenausprägung auf die Voxel abgebildet und diese entsprechend der Datenwerte eingefärbt. Werden räumliche Daten visualisiert, kann der Betrachter gegebenenfalls das den Daten zugrunde liegende reelle Objekt in der Darstellung als Abbildung erkennen.

Die 3D-Scatterplots können manuell per „Drag and Drop“ oder animiert rotiert und die Rotationsgeschwindigkeit und -achse manipuliert werden. Mithilfe der Hilfstaste „Shift“ und vertikaler Mausbewegung kann die Ansicht gezoomt werden.

Beim so genannten „View slaving“ werden mehrere 3D-Scatterplots miteinander in ihrer Transformation verbunden. Die Rotation eines Scatterplots bewirkt beispielsweise in diesem Modus, dass auf alle gekoppelten Scatterplots dieselbe Interaktion automatisch angewendet wird. Die Scatterplots rotieren in diesem Falle entsprechend synchron.

Der Voxelplot bietet verschiedene „Brushing“-Methoden zum gleichzeitigen Markieren mehrerer Datenpunkte an (siehe Abbildung 3.8). Durch logische Operatoren können auch „Brushing“-Vorgänge miteinander kombiniert werden. Nicht markierte Voxel werden halbtransparent grau dargestellt, markierte hingegen behalten ihre ursprüngliche Farbe.
Beim „Range Brush“ werden die zu markierenden Daten durch Alphaslider [AS94] in ihrem Wertebereich eingegrenzt.
Der “Beam Brush“ markiert alle Voxel innerhalb eines durch den Mittelpunkt und Radius definierten Zylinders senkrecht zur Sichtebene.
Der „Cluster Brush“ versucht Cluster in den Daten zu erkennen und markiert diese.

Werden einzelne 3D-Scatterplots mittels „View Linking“ verbunden, wirken sich die aufgeführten „Brushing“-Methoden auf alle verlinkten Visualisierungen aus, ansonsten betrifft die Markierung nur die aktuelle Darstellung.

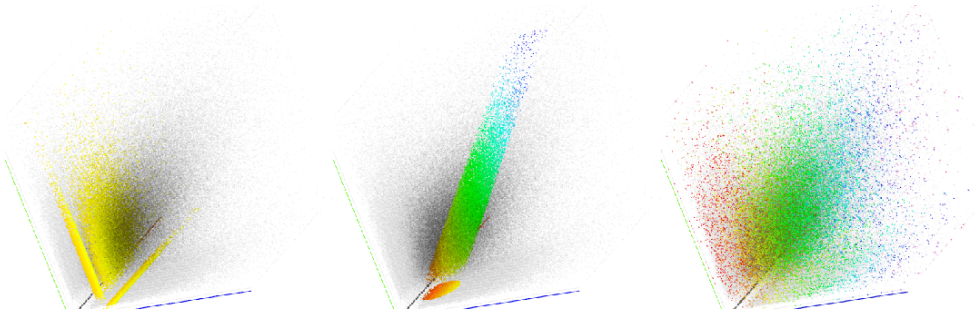


Abbildung 3.8: Voxelpot Brushing-Methoden,
Range Brush (links), Beam Brush (mitte), Cluster Brush (rechts) [Sahl02].

4 Projekt VisMeB

Bei der Recherche von relevanten Informationen auf großen Datenmengen können verschiedene Schwierigkeiten auftreten, mit denen ein Suchender konfrontiert und bei der Erfüllung seiner Aufgabe beeinträchtigt werden kann. Der Anwender muss dem Suchalgorithmus seinen Informationsbedarf in einer geeigneten Terminologie und Form artikulieren, um eine bedarfsspezifische Relevanzberechnung zu ermöglichen.

Mehrdeutigkeit und Ungenauigkeit können dabei die Güte der Treffermenge vermindern. Zur Extraktion der wirklich relevanten Informationen aus der berechneten Treffermenge muss der Anwender die Treffer einzeln explorieren und beurteilen. Je nach Art und Größe der Treffermenge kann sich dieser Prozess als zeitaufwendig und nicht trivial herausstellen.

Das Forschungsprojekt *VisMeB* [VisM03] wird von der AG Mensch-Computer Interaktion des Fachbereichs Informatik und Informationswissenschaft unter der Leitung von Prof. Dr. Harald Reiterer an der Universität Konstanz durchgeführt. Das wesentliche Ziel des Projektes ist, mithilfe eines visuellen Metadaten Browsers den Benutzer unabhängig von der Anwendungsdomäne bei der Suche und Extraktion von relevanten Daten aus einer großen Datenmenge effizient zu unterstützen.

VisMeB stellt dafür ein grafisches Userinterface zur Durchführung von Suchprozessen auf Metadaten zur Verfügung. Die erhaltene Ergebnismenge wird mithilfe verschiedenster Visualisierungstechniken, wie „LevelTable“, „GranularityTable“, „2D- und 3D-Scatterplot“, „CircleSegmentView“ und „Browser View“, dargestellt. Diese verwenden die Metadaten der Ergebnisobjekte und Gesamt- bzw. Einzelrelevanzen der Suchbegriffe, um dem Anwender die Möglichkeit zu geben, die einzelnen Treffer im Bezug auf seine Aufgabe visuell zu beurteilen und die für ihn relevanten zu isolieren.

Die Verwendung von Metadaten und nicht der Daten an sich gestatten einen domänenunabhängigen Einsatz von VisMeB. Darstellungen von nicht rein textuellen Daten, wie Molekularstrukturen oder Landkarten, sind schon bei geringer Datenmenge in einer Visualisierung schwer realisier- und auch vergleichbar. Standardisierte Metadaten hingegen können auch bei großen Datenmengen relativ leicht dargestellt und vom Anwender verglichen werden. Zudem bieten Metadaten zumeist eine schnell interpretierbare Zusammenfassung der eigentlichen Daten, was dem Anwender eine effizientere Einschätzung ermöglicht.

Das Projekt VisMeB kann sich auf die Ergebnisse und Erkenntnisse von zwei EU-Projekten stützen, die im Vorfeld am gleichen Lehrstuhl mit anderen internationalen Projektpartnern zusammen durchgeführt wurden und die Entwicklung von visuellen Suchsystemen zur Unterstützung von Rechercheaufgaben (Visual Information Retrieval Systems) zum Ziel hatten.

Das EU-Projekt *INSYDER* [INSY03] stellt eine Anwendung im Umfeld der so genannten Business Intelligence Systems zur Verfügung. Es soll Unternehmen bei der Suche und Analyse von Informationen aus dem World Wide Web durch geeignete Visualisierungen unterstützen [RMMH00].

Das Nachfolgeprojekt *INVISIP* [INVI03] soll Standortentscheidungen begleiten und nachfolgende Prozesse sowie beteiligte Parteien unterstützen. Hierfür wurde unter der Leitung von Prof. Dr. Harald Reiterer ein Metadaten Browser zur Suche und Analyse von Geometadaten entwickelt und evaluiert.

Ein typisches Einsatzszenario von VisMeB wäre zum Beispiel die Suche nach einem aktuellen Action- oder Comedy-Film auf einer Filmdatenbank. Dafür könnte der Benutzer mittels dem beim Starten der Anwendung erscheinenden textuellen Suchformulars die Datenmenge auf Filme der Kategorie Action und Comedy und der Veröffentlichung zwischen den Jahren 1998 und 2003 einschränken und die Suche starten. Anschließend würde dann der Anwender mithilfe der verschiedenen Visualisierungen die Filme zum Beispiel nach ihrem Rang, ihrer Länge und Sprache vergleichen und sich Informationen zu den Filmen anzeigen lassen. Die Datenmenge wird so von vielleicht mehreren Tausend auf ein paar wenige relevante Datensätze reduziert und der Suchende kann nun aus diesen die für ihn interessantesten Filme auswählen und gegebenenfalls in der Videothek ausleihen.

Die genaueren Möglichkeiten der einzelnen Visualisierungen zur Exploration der Daten werden in den folgenden Kapiteln näher erklärt. Dabei wird dieses Szenario und ein weiteres, welches die Suche auf einer Datenbasis mit Webdokumenten schildert, verwendet um die Funktionalitäten verständlicher darzustellen. Tatsächlich ist VisMeB im Bezug auf die Datenart sehr flexibel und ist im Moment für Entwicklungs- und Evaluationszwecke an drei verschiedene Datenbanken angeschlossen. Es können außer den in diesem Szenario beschriebenen Filmdaten noch Webdokumente und Geodaten visualisiert werden.

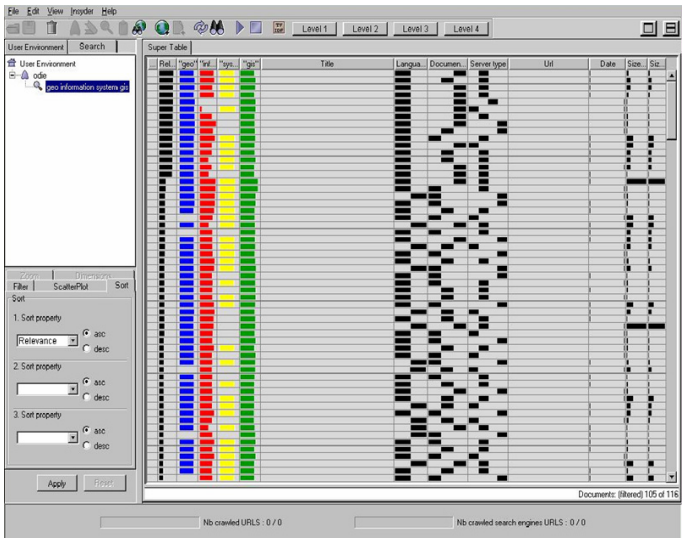


Abbildung 4.1: INVISIP, HTML Mockup.

4.1 Visual Configuration and Assignment Tool

Das Visual Configuration and Assignment Tool erlaubt eine benutzerfreundliche Anpassung von VisMeB an die jeweiligen Gegebenheiten. Es kommuniziert mit den gängigsten Datenbanksystemen und stellt ihr DB Schema in einer Baumhierarchie dar, wobei die Wurzel die Datenbank, die Knoten die Tabellen und Views und die Blätter die Spaltennamen darstellen. Die Datentypen der Spalten werden durch farbige Icons repräsentiert. Diese Baumstruktur wird im linken Teil der vertikal geteilten Visualisierung des Visual Configuration and Assignment Tool angezeigt (siehe Abbildung 4.2).

Auf der rechten Seite stellt ebenfalls eine Baumhierarchie die VisMeB Visualisierungen mit ihren möglichen Zuordnungen der Tabellenspalten bzw. Datentypen zu den Visualisierungen (z.B. BorwserView) dar. Durch einfaches „Drag and Drop“ werden die Tabellen- oder View-Spalten auf die Visualisierungsvariablen, welche eine feste oder dynamische Anzahl von Variablen aufnehmen können, gezogen und damit zugewiesen. Die Tabellen- und Visualisierungsdatentypen müssen übereinstimmen um eine gültige Zuordnung zu ermöglichen. Der Visualisierungsdatentyp „Any Type“ kann jeden beliebigen Datentyp aufnehmen.

Im Visualisierungsknoten „Initial Values“ werden die Standardbelegungen für die einzelnen Darstellungen beim Initialisieren von VisMeB festgelegt. Zum Beispiel kann beim 3D-Scatterplot die Rangfolge (rank) per Default auf der y-Achse abgetragen werden.

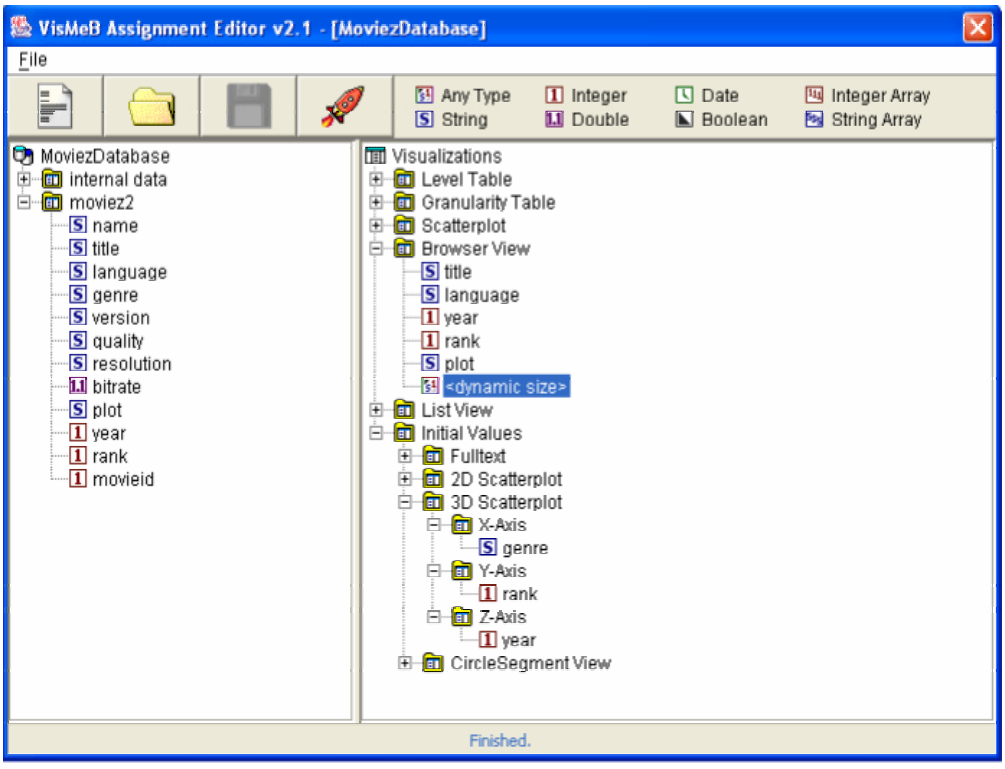


Abbildung 4.2: VisMeB Visual Configuration and Assignment Tool.

Innerhalb des Visual Configuration and Assignment Tool können neue Assignments (Zuordnungen) erstellt, bestehende geöffnet und editierte im XML-Format abgespeichert werden. Eine Grundkonfiguration, die nicht in diesem Tool stattfindet, ist die Anbindung von VisMeB an eine neue Datenbank. In diesem Falle muss eine Datenbank-Klasse in JAVA angelegt werden, in der der spezifische Datenbanktreiber, der Pfad, Benutzername, Passwort der Datenbank und View oder Table angegeben werden.

Das Visual Configuration and Assignment Tool wird im Normalfall nur bei Änderungen der Datenbasis oder bei grundlegenden Änderungen der Visualisierungen verwendet. Der Standardanwender kommt mit diesem Administrationstool weniger in Kontakt, da er meist eine vorkonfigurierte Umgebung vorfindet. Trotzdem erlaubt dieses Tool aufgrund der verwendeten Interaktions- und Visualisierungstechniken auch einem Erst- oder Gelegenheitsanwender ohne SQL-Kenntnisse Konfigurationen vorzunehmen.

4.2 Textuelle & graphische Suche

Mit VisMeB kann eine Suchanfrage durch zwei verschiedene Methoden definiert werden. Bei der textuellen Suche gibt der Anwender Suchbegriffe ein und grenzt gegebenenfalls die Grunddatenmenge durch Spezifizierung von Datenattributen für einen globalen Filter im Suchformular ein. So kann zum Beispiel bei Dokumenten die Datenmenge auf englische Dokumente der letzten 13 Jahre reduziert werden (siehe Abbildung 4.3).

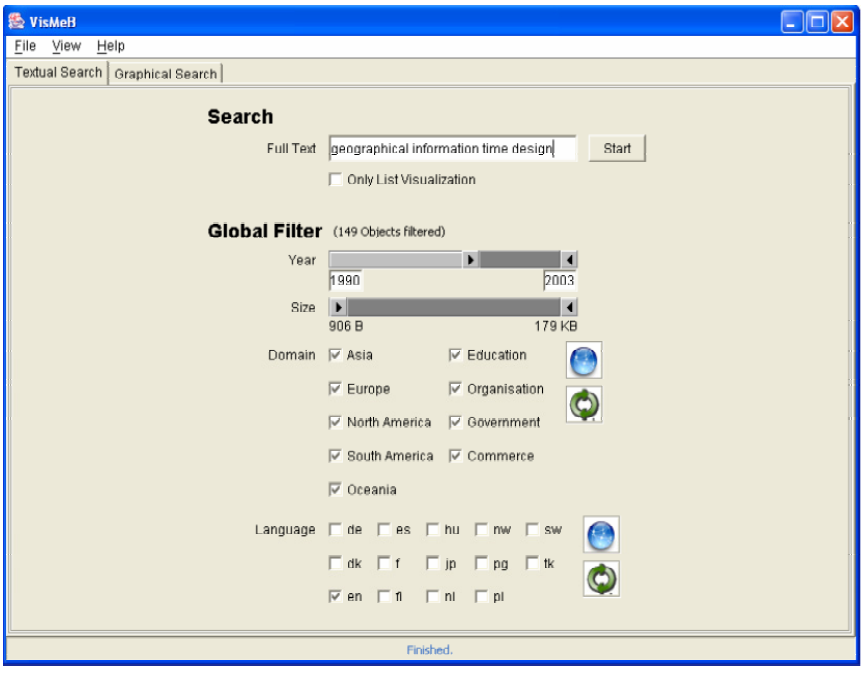


Abbildung 4.3: VisMeB textuelle Suche mit globalen Filter.

Mithilfe der graphischen Suche hat der Anwender die Möglichkeit, die Suchbegriffe und Filtereigenschaften durch die Interaktion mit der „CircleSegmentView“ (CSV, siehe Kapitel 4.8) nach den Dynamic Query [AWS92] und Query Preview [TPS00] Konzepten dynamisch zu erstellen und zu verbessern.

Dies erspart den konventionellen iterativen Suchprozess, bei dem einzelne Begriffe angegeben, das Resultat bewertet und gegebenenfalls weitere Suchvorgänge mit erweiterten bzw. vermeintlich besseren Begriffen gestartet werden. Des Weiteren wird die Komplexität von Suchanfragen mit booleschen Operatoren durch geeignete Darstellung im CSV vermieden.

4.3 SuperTable Konzept

Das Hauptanliegen des in VisMeB umgesetzten SuperTable Konzeptes gehört zu den klassischen Problemstellungen der Informations-Visualisierung: Einerseits soll die Visualisierung einer Dateneinheit sehr detailliert sein um eine vollständige Interpretation zu gewährleisten. Andererseits wird die Darstellung von so vielen Dateneinheiten wie möglich in der Ansicht angestrebt, um die Position einer Dateneinheit in den Kontext der gesamten Datenmenge bringen zu können [KRML03].

Zur Lösung dieser „Overview and Detail“-Problematik verwendet die SuperTable eine Kombination von verschiedenen Visualisierungen in einer Tabelle, wie z.B. „Bar charts“, „Tile bars“ und farblich kodierte Textstellen und Interaktionsmechanismen, wie z.B. „fisheye views“ [Furn86] und „Brushing and Linking“-Techniken [BC87]. So hat eine Selektion oder Fokussierung in einer Visualisierung direkte Auswirkung auf alle anderen Visualisierungen. Zusätzlich verfügt die SuperTable über ein so genanntes „Granularity“-Konzept, welches dem Anwender erlaubt, dynamisch den anzuzeigenden Detailgrad festzulegen. Damit hat der Anwender die Möglichkeit, interessante Daten immer detaillierter zu betrachten. Diese Methode wurde als „Focus of Interest“ [KMRE02] bezeichnet.

Jede Zeile der SuperTable entspricht einem Datensatz und jede Spalte einer Metadatenart. Bei der Suche auf Dokumenten enthält beispielsweise jede Zeile ein Ergebnisdokument und die Spalten können Metadaten, wie z. B. Sprache, Größe oder Domäne enthalten.

Zur Umsetzung des SuperTable Konzeptes wurden in VisMeB zwei Designvarianten der SuperTable, die Level- und GranularityTable, implementiert, die in Usability Tests miteinander und mit einer üblichen Listendarstellung verglichen werden soll.

4.4 LevelTable

Bei der LevelTable ist die Granularität, also der Detailgrad der angezeigten Informationen, nur global einstellbar, d.h. alle Zeilen der LevelTable besitzen die gleiche Granularität. Die Veränderung der Granularität ist in vier Levels möglich, wobei für jeden Level ein Button in der oberen rechten Ecke des Level Table Fensters positioniert ist. Je nach Level werden bestimmte Visualisierungen und Metadaten in den Tabellenzeilen angezeigt.

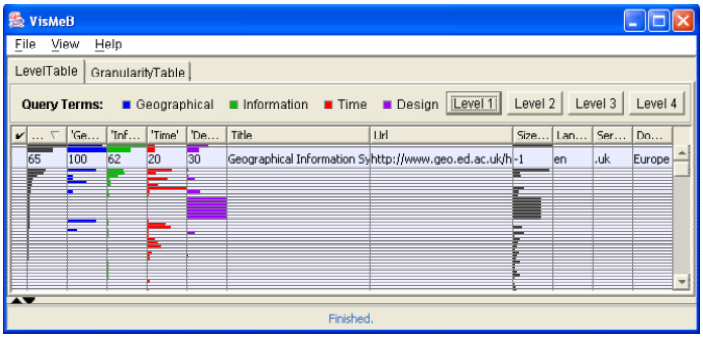


Abbildung 4.4: VisMeB LevelTable, Level 1 mit Overview- & Detail-Technik.

Der erste Level stellt durch farblich kodierte horizontale Balken numerische Metadaten, wie die Gesamt- und Einzelrelevanzen der Suchbegriffe im Bezug auf den Datensatz der jeweiligen Zeile, dar. Jede Zeile ist nur wenige Pixel hoch, um eine maximale Anzahl von Dateneinheiten anzeigen zu können. Befindet sich der Mauszeiger über einer Zeile, wechselt diese Zeile in Level 2, in welchem die Spalten gleich wie in Level 1 belegt sind, aber die Zeilenhöhe so definiert ist, dass die Metadaten als Text lesbar werden. Alle anderen Zeilen bleiben von dieser Veränderung unberührt (fisheye views [Furn86]).

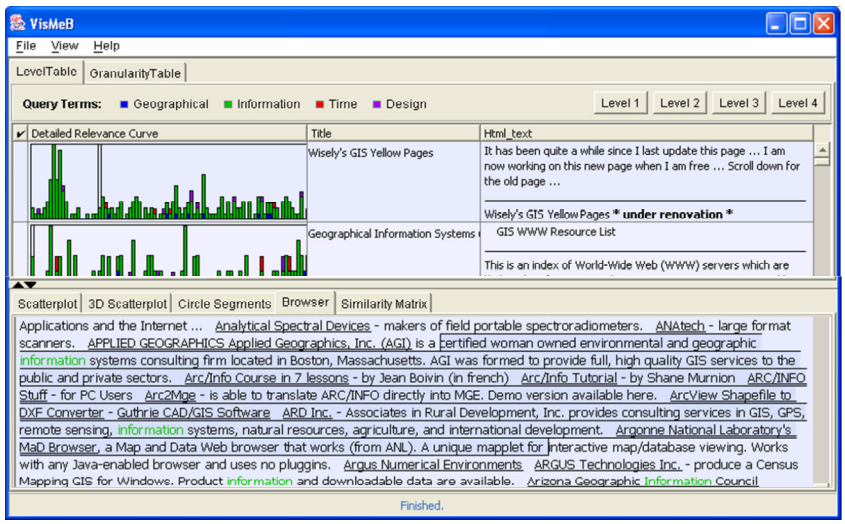


Abbildung 4.5: VisMeB LevelTable, Level 4 und Browser View.

Der dritte Level weist eine veränderte Spaltenbelegung im Gegensatz zu Level 1 und 2 und eine neue Visualisierung, die „Relevance Curve“, auf. Diese ähnelte einem Balkendiagramm, welches in der Darstellungsbreite die Dokumentenlänge repräsentiert. Die vertikalen Balken stellen die einzelnen Textsegmente dar, wobei die Gesamtrelevanz des jeweiligen Segmentes auf die Balkenhöhe abgetragen wird. In Level 4 werden die vertikalen Balken anteilsmäßig entsprechend der Teilrelevanzen der Suchbegriffe farblich kodiert und die Browser View mit dem jeweiligen fokussierten Dokument mit angezeigt. Berührt der Mauszeiger einen vertikalen Balken, wird in der Browser View das Textsegment, welches durch den Balken repräsentiert wird, automatisch fokussiert und farblich markiert. Diese Art der Visualisierung nennt sich „Detailed Relevance Curve“.

4.5 GranularityTable

Die zweite Designvariante der SuperTable ist die GranularityTable. Sie gleicht der LevelTable weitestgehend. Bei dieser wird aber versucht, einen weichen Übergang zwischen Übersichts- und Detaildarstellung zu verwirklichen, um den Anwender nicht durch zu harte Schnitte zwischen den Detailgraden kognitiv zu belasten. Idealerweise sollten keine Übergänge mehr sichtbar sein. Da dies aber technisch schwer umzusetzen ist, findet eine Annäherung durch sechs Granularity Levels statt, die mit Slider einstellbar sind.

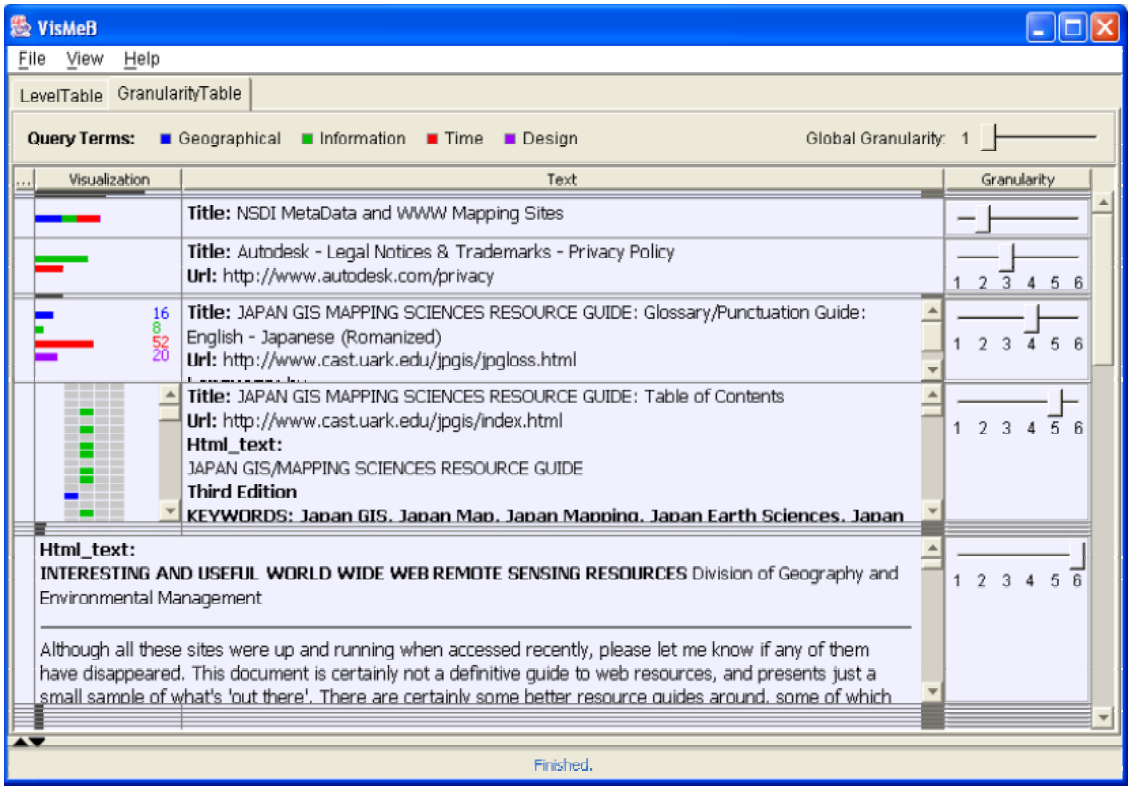


Abbildung 4.6: VisMeB GranularityTable, lokale Granularity Levels 1 – 6.

Im Gegensatz zu der LevelTable können die Granularity Levels der Tabellenzeilen unterschiedlich sein. In jeder Zeile befindet sich zur lokalen Veränderung der Granularität ein Slider und oben links im GranularityTable Fenster, anstatt der vier Buttons in der LevelTable, ein globaler Granularity Slider, der die Granularität für alle Zeilen festlegt.
Die GranularityTable besitzt nur vier Spalten: Selektion, Visualisierung, Text und eine Spalte für die eben beschriebenen lokalen Granularity Slider. Nur die Visualisierungs- und Textspalte verändern sich je nach Granularität.

In Level 1 sind, wie auch bei der LevelTable, die Zeilen bis auf wenige Pixel in der Höhe reduziert. Die Visualisierungsspalte repräsentiert mit einem horizontalen Balken die Gesamtrelevanz der in der Zeile beinhalteten Daten. Befindet sich der Mauszeiger über einer Zeile, wird deren Zeilenhöhe vergrößert und der Text wird einzeilig lesbar. Auch wird zum horizontalen Balken die Gesamtrelevanz als Zahl angezeigt. In Level 2 verwendet die Visualisierungsspalte zur Repräsentation der Einzelrelevanzen bezüglich der Suchbegriffe einen „Stacked Colored Bar Graph“, bei dem der horizontale Balken für die Gesamtrelevanz aus Level 1 je nach Einzelrelevanz anteilmäßig farblich kodiert ist.

In Level 3 werden die Einzelrelevanzen übereinander abgetragen und in der Textspalte mehr Informationen bereitgestellt. Zum Beispiel könnte bei Dokumentendaten in der Textspalte zusätzlich zum Titel, welcher schon im Level 2 präsent war, die Sprache und die Größe angezeigt werden. In Level 4 werden nochmals weitere Informationen zur Textspalte hinzugefügt und die Einzelrelevanzen in der Visualisierungsspalte werden mit ihren Werten beschriftet.

Die Visualisierungsspalte verändert sich erstmals in Level 5 grundsätzlich. Das durch die Zeile repräsentierte Dokument wird nun in dieser Spalte durch vertikal angeordnete „TileBars“ dargestellt. TileBars für Relevante Textsegmente werden farblich markiert, alle anderen ausgegraut. Die Textspalte beinhaltet nun das gesamte Dokument. Beim Anklicken einer TileBar wird das entsprechende Textsegment in der Textspalte fokussiert und markiert. In Level 6 wird die Visualisierungsspalte zu Gunsten der Textspalte ausgeblendet (siehe Abbildung 4.6).

4.6 Browser View

In der Browser View (siehe Abbildung 4.5) kann ein gerade fokussiertes oder eine beliebige Anzahl von ausgewählten Dokumenten angezeigt und gegebenenfalls miteinander verglichen werden. Hier werden nicht mehr die Metadaten eines Dokumentes, sondern das gesamte Dokument an sich visualisiert. Die Browser View kann die dargestellten Dokumente zoomen, nebeneinander anordnen und beinhaltet einen HTML-Interpreter zur grundlegenden Formatierung des Textes.

4.7 Scatterplot

Der Scatterplot (siehe Kapitel 2.1) stellt eine Visualisierung zur Verfügung, in der die dargestellten Daten an Hand von zwei auswählbaren Attributen miteinander verglichen werden können, um Cluster, Ausreißer oder Tendenzen in den Daten zu erkennen oder einen Gesamtüberblick zu gewinnen.
Die Daten werden durch Icons in einem zweidimensionalen kartesischen Koordinatensystem, welches durch die entsprechenden Datenattribute aufgespannt wird, repräsentiert. Einzelne Daten werden als Kreise, MultiDataPoints (MDP, siehe Kapitel 2.2) als Quadrate dargestellt. Bei einzelnen selektierten Daten sind die Kreise farblich ausgefüllt, bei zum Teil selektierten MDPs ist nur eine Hälfte des Icons und bei komplett ausgewählten MDPs das ganze Quadrat ausgefüllt. Fokussierte Icons werden vergrößert dargestellt. Über ein Kontextmenü zu den Icons kann der jeweilige Selektionszustand der Daten verändert und bei MDPs nähere Informationen zu den enthaltenen Daten abgerufen werden. Ein Kontextmenü zu den beschrifteten Achsen erlaubt die Belegung bzw. das Assignment derselbigen zu verändern.

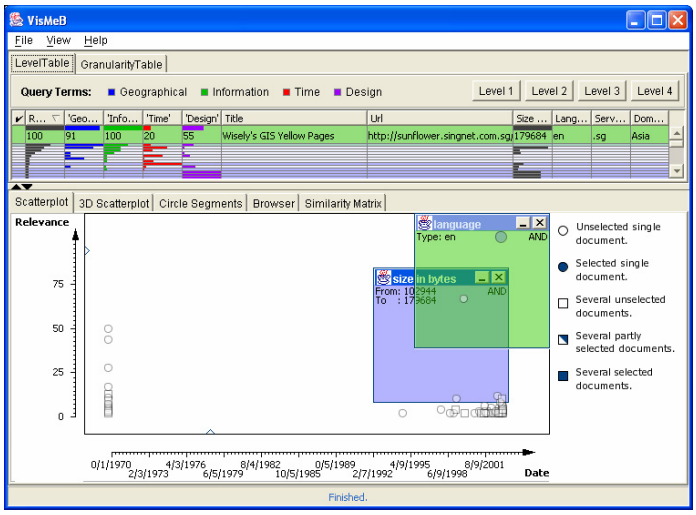


Abbildung 4.7: VisMeB 2D-Scatterplot mit Movable Filters.

Des Weiteren kann die Ansicht gezoomt und mit Movable Filter [FS95] die Daten lokal gefiltert werden. Beim Zooming zieht der Anwender über den Bereich des Scatterplots, den er näher betrachten möchte, ein Rechteck und vollzieht über das Kontextmenü eine Vergrößerung dieses Ausschnittes, die er auch wieder über dieses Kontextmenü durch „Zoom Out“ bzw. „Zoom Out Full“ rückgängig machen kann. Das Zooming wirkt wie ein globaler Filter, da auch in den anderen Visualisierungen von VisMeB nur noch die gezoomten Dokumente sichtbar bleiben.
Der Movable Filter ist ein lokaler Filter, der alle Punkte in einem definierten Bereich ausblendet, die nicht den vorher definierten Filterregeln entsprechen. Die gefilterten Datenobjekte werden auch in der Level- und GranularityTable farblich markiert.

4.8 CircleSegmentView

Die CircleSegmentView (CSV) wird als graphische Suche (siehe Kapitel 4.2) zur Definition von Such- und Filtereigenschaften und als eigene Visualisierung innerhalb von VisMeB eingesetzt. Auf Grund der verwendeten Dynamic Query [AWS92] und Query Preview [TPS00] Konzepte kann mit der CSV die Grunddatenmenge dynamisch eingeschränkt werden, wobei der Anwender immer unmittelbares Feedback auf die Veränderungen erhält.

Die CSV besteht hauptsächlich aus zwei „Pie Charts“, welche jeweils die gesamte Datenmenge repräsentieren. Die verschiedenen Segmente spiegeln die Verteilung der Daten in Bezug auf eine bestimmte Metadatenart wieder. Diese Metadatenart kann für jedes „Pie Chart“ aus einer Drop-down-Liste ausgewählt werden. Die Daten werden durch kleine farbige Kreise dargestellt.

Zusätzlich zur Einteilung nach Segmenten bestimmt eine weitere Metadatenart die Entfernung der Datenrepräsentanten zum Kreismittelpunkt und eine dritte den Winkel innerhalb des entsprechenden Segments. Die letzteren Metadatenarten können mittels Alphaslidern [AS94] in ihrem Wertebereich eingeschränkt und damit einen Filter auf die Datenmenge des jeweiligen „Pie Chart“ angewendet werden. Die abzubildende Metadatenart wird an den Alphaslidern per Kontextmenü festgelegt.

Die beiden „Pie Charts“ können mit booleschen Operatoren miteinander verknüpft werden, um komplexere Such- und Filtereigenschaften generieren zu können.

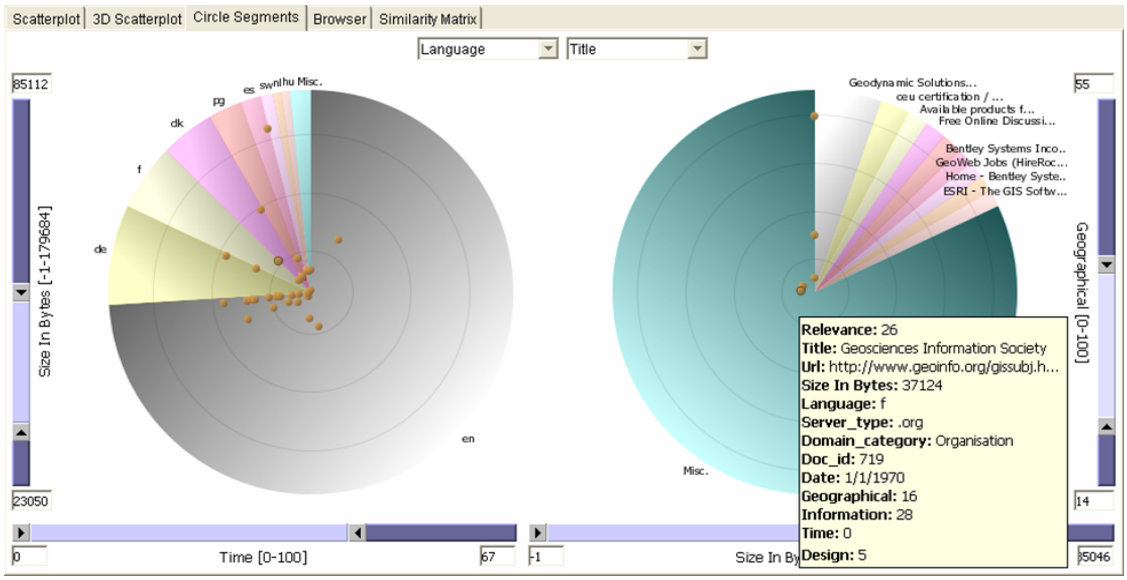


Abbildung 4.8: VisMeB CircleSegmentView mit ToolTip.

5 VisMeB 3D-Scatterplot

Im Folgenden wird die Aufgabenstellung für das Projekt „Konzeption und Implementation eines 3D-Scatterplots zur Visualisierung von Metadaten“ und anschließend der daraus entstandene aktuelle 3D-Scatterplot des Forschungsprojektes VisMeB [VisM03] mit den verwendeten Visualisierungs- und Interaktionsmechanismen näher beschrieben.

5.1 Projektdefinition

Zum Zeitpunkt der Aufgabenstellung bestand der Visuelle Metadaten Browser, damals noch im Projekt INVISIP [INVI03] verankert, aus der Level- und GranularityTable und dem zweidimensionalen Scatterplot. Ein 3D-Scatterplot wurde schon im Rahmen einer Vorlesung von drei Studenten[1] unabhängig vom INVISIP Projekt entwickelt. Aufgabe war nun, diesen 3D-Scatterplot weiterzuentwickeln und in die bestehende Projektumgebung zu integrieren. Das Projektteam zur Umsetzung dieser Aufgabe bestand aus zwei Studenten[2].

Bei der Integration des bisherige 3D-Scatterplots in INVISIP stellte sich heraus, dass die Datenmodelle nicht in vertretbarer Zeit vereinbar waren, da das INVISIP Datenmodell weitaus generischer aufgebaut wurde, um den verschiedenen Visualisierungen Rechnung zu tragen. Aufgrund der engen Verbindung der Darstellung mit dem eigenen Datenmodell beim 3D-Scatterplot, konnte auch diese nicht getrennt portiert werden.

Dies hatte zur Folge, dass das Team eine komplette Neukonzeption und –implementation beschloss, um so auch die Möglichkeit zu haben, neue und weitergehende Interaktionsmechanismen und Darstellungsarten grundlegend modellieren zu können.

5.1.1 Zielbestimmung

Ziel des Projektes ist, einen 3D-Scatterplot zu konzipieren, welcher die bestehende Umgebung von VisMeB mit Datenmodell, Visualisierungsstandards und verwendete Interaktionsmechanismen unterstützt. So soll der 3D-Scatterplot als Teilvisualisierung von VisMeB in einem JAVA JSplitPane, einem zweigeteilten Fenster, optional zum zweidimensionalen Scatterplot integriert werden und die „Brushing and Linking“-Konzepte zwischen den Visualisierungen berücksichtigen.

[1] A. Ross, S. Mayer und K. Gertz, Universität Konstanz

[2] P. Liebrenz [Lieb03] und W.A. König (Autor), Universität Konstanz

Der 3D-Scatterplot soll ein dreidimensionales Koordinatensystem mit Datenobjekten visualisieren, welche selektiert und zu welchen nähere Informationen abgerufen werden können. Die Manipulation des Koordinatensystems im 3D-Raum und der Effekt von übereinander liegenden Datenrepräsentanten sind weitere Schwerpunkte. Die Implementation findet in der Programmiersprache JAVA[3] statt.

Wünschenswert wäre eine gute Performance, so dass auch bei größeren Datenmengen noch mit dem 3D-Scatterplot interagiert werden kann. Die Gesamtperformance ist aber sehr vom Datenmodell und den weiteren Visualisierungen in VisMeB abhängig.

5.1.2 Produkteinsatz

Der Anwendungsbereich des 3D-Scatterplots ist durch den Einsatz von VisMeB definiert, welches generisch für die Suche auf Metadaten in den unterschiedlichsten Anwendungsdomänen konzipiert wurde. Vorläufig dürfte die Verwendung in Testzenarien für Evaluationsstudien die Hauptanwendung sein. Zielgruppe wären damit Versuchspersonen und alle Projektbeteiligten von VisMeB. Dementsprechend können durchschnittliche Büroumgebungen und Laborbedingungen als Betriebsumgebung angenommen werden.

5.1.3 Produktumgebung

Der 3D-Scatterplot bzw. VisMeB wird in der Programmiersprache JAVA entwickelt und ist aufgrund der plattformunabhängigen Konzeption auf allen gängigen Betriebssystemen einsetzbar, für die eine JAVA-Laufzeitumgebung, die JAVA Virtual Machine (JVM), verfügbar und installiert ist. Die JVM ist im frei verfügbaren JAVA Runtime Environment Paket enthalten, wobei für VisMeB mindestens die Version J2RE 1.3.1 Vorrausetzung ist.

Als Hardware können Standard-PCs verwendet werden. Für eine flüssige 3D-Interaktion mit größeren Datenmengen sind Personal Computer mit gängigen Prozessoren ab 1 GHz, mindestens 256 MB Arbeitsspeicher und einer üblichen 3D-Grafikkarte mit mindestens 16 MB Speicher zu benutzen. VisMeB kann im Offline- oder Online-Modus betrieben werden. Für den Online-Modus muss eine Internetverbindung bestehen, um die benötigten Daten vom Datenbankserver zu laden. Die Ladezeit bei der Initialisierung von VisMeB ist abhängig von der Verbindungsgeschwindigkeit, daher ist bei größeren Datenmengen eine Breitbandverbindung zu bevorzugen.

[3] JAVA, Sun Microsystems, http://java.sun.com/

5.1.4 Produktfunktionen

Innerhalb des 3D-Scatterplots können einzelne oder mehrere Daten selektiert und exploriert werden. Die Koordinatenachsen können mit verschiedenen Datentypen belegt werden, wobei die Datenrepräsentanten dementsprechend im dreidimensionalen Raum angeordnet werden. Das Koordinatensystem kann per Maus oder per Button rotiert und gezoomt werden.

5.1.5 Produktdaten

Der 3D-Scatterplot ist an das VisMeB Datenmodell angeschlossen, welches die Daten lokal aus speziellen Dateien auslesen oder online von einem beliebigen Datenbank-Server beziehen kann (siehe Kapitel 4.1). Die Daten selber bestehen aus Metadaten, auf denen mittels VisMeB gesucht werden kann. Die den Metadaten zugrunde liegenden Daten müssen nicht, können aber in der Datenbank vorhanden sein und gegebenenfalls bei entsprechendem Format auch von VisMeB visualisiert werden.

5.1.6 Benutzungsoberfläche

Der Benutzungsoberfläche des 3D-Scatterplots wie auch der von VisMeB liegt eine WIMP[4]-Oberfläche zugrunde. Die Interaktion mit der Anwendung soll per Maus oder per Tastatur erfolgen, wobei die Maussteuerung bei der 3D-Interaktion vorrangig ist.

5.1.7 Entwicklungsumgebung

Die Entwicklungsumgebung besteht aus dem Borland JBuilder[5] in den Versionen 7 bis 9 in Verbindung mit dem Versionsverwaltungssystem CVS[6]. Als Compiler wird der JAVA 2 SDK Standard Edition in der Version 1.3.1_07 verwendet. Die Entwicklungs-Hardware entspricht den Angaben für die Produkt-Umgebung (siehe Kapitel 5.1.3). Microsoft Windows XP und Linux kommen als Betriebsysteme zum Einsatz.

[4] WIMP, Windows, Icons, Menus, Pointers als Basiselemente der Interaktion

[5] Borland JBuilder, Borland Software Corporation, http://www.borland.com/jbuilder/

[6] CVS, Concurrent Versions System, http://www.cvshome.org/

5.2 Vorstellung des implementierten 3D-Scatterplots

Aufgrund der im letzten Kapitel beschriebenen Projekt-Anforderungen wurde vom Projektteam ein 3D-Scatterplot konzipiert, implementiert und in das bestehende Projekt VisMeB integriert. Dieser Entwicklungsprozess wird in Kapitel 7 näher betrachtet.

Hier wird nun der aktuelle VisMeB 3D-Scatterplot in seiner Funktionsweise und Darstellung vorgestellt und die verwendeten Interaktionskonzepte werden erläutert. Das anschließende Anwendungsbeispiel beschreibt einen beispielhaften Einsatz des 3D-Scatterplots, wobei die grundlegenden Elemente bei der Durchführung erklärt werden.

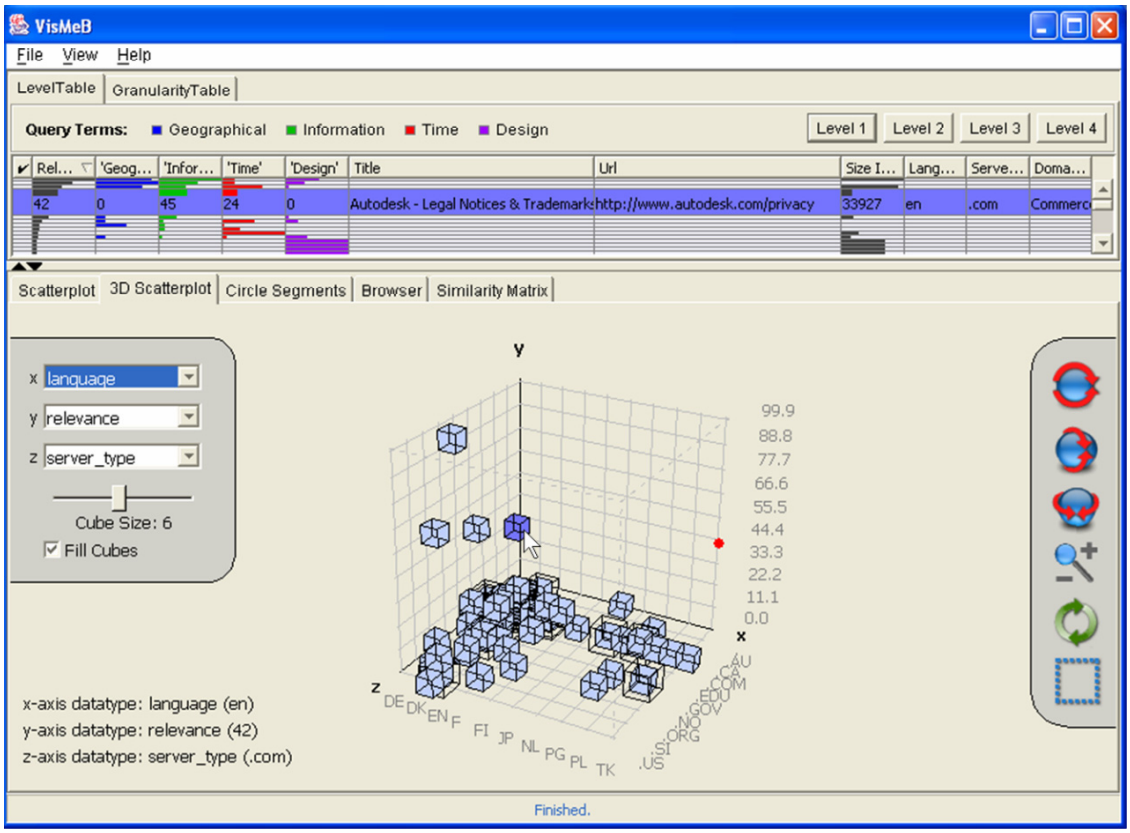


Abbildung 5.1: VisMeB 3D-Scatterplot mit fokussiertem Datenwürfel.

5.2.1 Anwendungsbeispiel VisMeB 3D-Scatterplot

Ein mögliches Anwendungsbeispiel für den 3D-Scatterplot könnte die Suche eines Anwenders auf einer Datenmenge mit gesammelten Webseitendokumenten nach Dokumenten eines bestimmten Themenbezugs sein. Dafür könnte er bei der textuellen Suche (siehe Kapitel 4.2) in VisMeB die für seinen Informationsbedarf passenden Suchbegriffe „Geographical“, „Information“, „Time“ und „Design“ eingeben und die Suche starten.

Anschließend wird das VisMeB Fenster horizontal geteilt, wobei im oberen Teil die Level- und GranularityTable in einem Register und im unteren Teil auch in einem Register 2D- und 3D-Scatterplot, CircleSegmentView, BrowserView und SimilarityMatrix angeordnet sind. Die SimilarityMatrix visualisiert Ähnlichkeitsbezüge, ist aber derzeit noch in der Entwicklungsphase und wird deshalb in dieser Arbeit nicht weiter thematisiert.

Nun könnten den Anwender vor allem englischsprachige Dokumente von einem kommerziellen Unternehmen mit einer hohen Relevanz bezüglich der Suchbegriffe interessieren. Daraus ergibt sich eine Suche nach drei Datendimensionen: Sprache, Server Typ und Relevanz. Für diese Aufgabe ist der 3D-Scatterplot prädestiniert, da die drei Datendimensionen auf die Koordinatenachsen abgebildet und so die Dokumente entsprechend ihrer Position im Raum herausgesucht und interpretiert werden können. Im 2D-Scatterplot müssten entweder immer zwei Dimensionen nacheinander verglichen oder mittels Movable Filters (siehe Kapitel 4.7) eine Hilfsdimension eingeführt werden.

Der Benutzer aktiviert nun den Registerreiter „3D-Scatterplot“ und wählt mit den Links die im so genannten „Option-Panel“ platzierten Drop-down-Menüs die Achsenbelegungen aus. In diesem Fall wird die x-Achse mit „language“, die y-Achse mit „relevance“ und die z-Achse mit „server_type“ belegt. Beim Betrachten und Rotieren des nun angepassten 3D-Scatterplots stechen in diesem Fall sofort vier Datenpunkte heraus, die oberhalb der restlichen Objekte platziert sind. Da auf der y-Achse die Relevanz abgetragen wird, kann der Anwender daraus schließen, dass diese Objekte eine überdurchschnittliche Relevanz besitzen (siehe Abbildung 5.1).

Weiterhin interessiert sich der Anwender für Dokumente englischer Sprache, welche sich demnach in Richtung der x-Achse auf Höhe der Beschriftung „EN“ befinden sollten. Beim Berühren der Datenwürfel mit der Maus erscheinen Tooltips, die nähere Informationen über diese bereitstellen. Auch werden links unten hinter den Angaben der aktuellen Achsenbelegung die Werte der Würfel, die gerade berührt werden, angezeigt.

Aus diesen Angaben kann der Anwender schließen, dass drei der vier oberen Dokumente englischer Sprache sind. Der ähnliche Prozess wird nun in Bezug auf den Server Typ vollzogen. Daraus resultiert, dass nur der in Abbildung 5.1 dunkelblau markierte Würfel den Vorgaben entspricht. Er hat eine überdurchschnittliche Relevanz, ist in der Sprache englisch gehalten und stammt von einer Website mit der Top Level Domain „.com“. Somit sollte dieses Dokument das für die Aufgabenstellung relevanteste Dokument aus der zugrunde liegenden Datenmenge sein.

Um diesen Treffer näher zu begutachten, kann der Anwender zum Beispiel die GranularityTable auf den höchsten Level setzen und dann das im 3D-Scatterplot fokussierte Dokument in seiner vorliegenden Form in der Table lesen.

5.2.2 3D-Koordinatensystem

In dem gerade beschriebenen Anwendungsbeispiel wurden schon einige Elemente des 3D-Scatterplots kurz beschrieben. Hier werden nochmals alle Visualisierungselemente und Interaktionsmechanismen aufgeführt und näher in ihrer Funktionalität erklärt.

Die VisMeB 3D-Scatterplot-Ansicht ist geprägt durch ein in der Mitte platziertes dreidimensionales Koordinatensystem, dessen x-, y- und z-Achse durch schwarze durchgehende Linien und durch die jeweilige Beschriftung an den Enden der Linien gekennzeichnet sind. Die Linien treffen sich im Koordinatenursprung. Die xy-, xz-, yz-Ebenen werden durch graue Gitternetzlinien durchzogen, um dem Betrachter die Zuordnung von Positionen im Koordinatenraum auf die Achsenwerte zu erleichtern.

Am Ende der Gitternetzlinien gegenüber den Achsen sind die entsprechenden Beschriftungen angebracht. Die Anzahl der Gitternetzlinien korrespondiert mit der Datenverteilung. Gibt es zum Beispiel insgesamt vier verschiedene Datenwerte bei einer Variablen, werden nur vier Gitternetzlinien angezeigt.

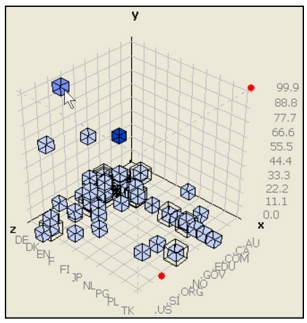
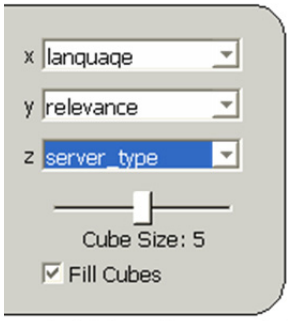


Abbildung 5.2: 3D-Koordinatensystem.

Die maximale Anzahl ist aber wegen der Übersichtlichkeit auf zehn Linien begrenzt. Um dem Betrachter einen besseren 3D-Eindruck zu ermöglichen, wird das Koordinatensystem durch gestrichelte graue Linien zu einem Würfel erweitert.



x-axis datatype: language (en)
y-axis datatype: relevance (100)
z-axis datatype: server_type (.sg)

Abbildung 5.3: Option-Panel.

Links neben dem 3D-Würfel befindet sich das so genannte Option-Panel, welches Einstellmöglichkeiten für den Scatterplot bereithält. So beinhaltet es auch drei Drop-down-Menüs zur Belegung der Achsen mit Datenvariablen der Datenmenge oder berechneten Relevanzen. Standardmäßig sind die im Visual Configuration and Assignment Tool (siehe Kapitel 4.1) festgelegten Assignments den Achsen zugeordnet. Die aktuelle Belegung der jeweiligen Achsen wird unter dem Option-Panel angezeigt.

5.2.3 Datenrepräsentation

Jeder Datensatz der zugrunde liegenden Datenmenge wird innerhalb des 3D-Koordinatensystems durch einen kleinen Würfel repräsentiert und entsprechend den Datenwerten im Koordinatensystem positioniert. Die Größe der Würfel ist durch einen Slider im Option-Panel dynamisch in zehn Stufen einstellbar. Hier kann auch per Checkbox festgelegt werden, ob die Würfel ausgefüllt oder nur durch ein Gittermodell dargestellt werden sollen. Die Farbe der Würfel hängt davon ab, ob sie fokussiert, selektiert oder nicht selektiert sind. Für jeden Status kann im VisMeB Option-Dialog (Menüpunkt „View“) eine beliebige Farbe definiert oder ein vordefiniertes Farbschema übernommen werden.

Ein Datenrepräsentant wird fokussiert, wenn die Maus diesen im 3D-Scatterplot oder, nach dem „brushing and linking“-Prinzip, das korrespondierende Datenobjekt in einer anderen VisMeB Visualisierungen, z.B. in der LevelTable, berührt. Erhält ein Würfel im 3D-Scatterplot den Fokus, werden seine Datenwerte bei der Beschriftung für die Achsenbelegung (unterhalb des Option-Panels) in Klammer angeführt. Zusätzlich werden rote Kreise an den Achsenbeschriftungen im Koordinatensystem angezeigt, um dem Anwender die Position und den Datenwert des Repräsentanten zu verdeutlichen. Behält ein Würfel mindestens zwei Sekunden lang den Focus, werden zu dessen Datenobjekt nach dem „details on demand“-Konzept zusätzliche Informationen in einem erscheinenden Tooltip angezeigt.

Der Status der Selektion kann für einzelne Repräsentanten durch einen Klick mit der linken Maustaste auf den Würfel oder in dem bei Rechtsklick erscheinenden Kontextmenü geändert werden. Die gleichzeitige Selektion von mehreren Daten ist im Mark-Modus (siehe Kapitel 5.2.6) möglich.

5.2.4 MultiDataPoints

Liegen mehrere Objekte genau auf der gleichen x-, y-, und z-Position, werden diese zu einem MultiDataPoint (MDP, siehe Kapitel 2.2) gebündelt und nur dieser wird noch angezeigt. Diese Methode hilft auch bei größeren Datenmengen die Übersichtlichkeit zu wahren.

Komplett selektierte oder nicht selektierte MDPs kann man an einem zusätzlichen Drahtgitter um den Würfel erkennen. Besitzt mindestens ein Objekt innerhalb des MDP einen anderen Selektionsstatus als die anderen, wird anstatt dem Würfel eine Pyramide in dem Drahtgitterwürfel gezeichnet, um den Anwender einen klaren Statusüberblick zu gewährleisten.

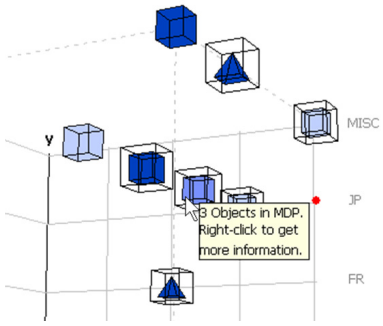


Abbildung 5.4:
MDP & einzelne Objekte.

Behält ein MDP mindestens zwei Sekunden lang den Focus, wird ein Tooltip angezeigt, der Auskunft über die Anzahl von Dokumenten im MDP gibt.

Bei Rechtsklick auf den MDP erscheint ein Kontextmenü, mithilfe dessen alle inhärenten Objekte entweder selektiert oder deselektiert werden können. Bei Linksklick auf den MDP oder auch über das Kontextmenü kann der MDP in der MultiDataPoint-View geöffnet und exploriert werden.

Die MultiDataPoint-View wird in Kapitel 5.2.7 näher beschrieben.

5.2.5 Direkte Manipulation

Die Datenobjekte können einzeln per Klick mit der linken Maustaste direkt selektiert oder deselektiert werden. Die Funktion zur gemeinsamen Festlegung des Selektionsstatus aller Objekte stellt ein Kontextmenü zur Verfügung, welches beim Rechtsklick auf den Hintergrund erscheint.

Der Menüpunkt „Reset View“ gehört zu jedem Kontextmenü des 3D-Scatterplots. Dieser setzt die Ansicht wieder in ihre Ausgangsposition und -größe zurück. Der Selektionsstatus der Objekte bleibt davon unberührt.

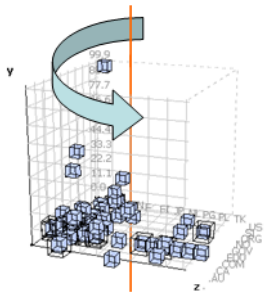


Abbildung 5.5: Vertikale Rotation.

Wird die Maus über den Scatterplot mit gedrückter Maustaste gezogen, rotiert das Koordinatensystem bzw. der Würfel entsprechend der Mausbewegung anteilsmäßig über die horizontale und vertikale Achse um den Würfelmittelpunkt. So kann der Würfel von allen Seiten betrachtet und bisher verborgene Objekte entdeckt werden. Durch die Rotation des Koordinatensystems werden Unklarheiten im Bezug auf die Positionen der Datenrepräsentanten klarer und der dreidimensionale Eindruck verstärkt.

Bei einer vertikalen Bewegung der Maus mit gedrückter mittlerer Maustaste wird das Koordinatensystem gezoomt. Eine Bewegung nach oben vergrößert die Ansicht, eine Bewegung nach unten verkleinert sie. Das gemeinsame Drücken der Shift-Taste und einer beliebigen Maustaste in Verbindung mit einer vertikalen Mausbewegung ermöglicht das Zooming auch ohne die mittlere Maustaste.

5.2.6 Interaction-Panel

Das Interaction-Panel stellt unter anderem die Möglichkeit zur Verfügung, das Koordinatensystem per Buttons über vertikale, horizontale und Tiefenachse zu rotieren. Solange der Anwender einen der Rotationsbuttons gedrückt hält rotiert es mit einer konstanten Geschwindigkeit. Die Buttons sind zweigeteilt und bestimmen je nach Position der gedrückten Maus im Button die Rotationsrichtung. Zum Beispiel verursacht die obere Hälfte des ersten Buttons in Abbildung 5.6 eine Rotation im Uhrzeigersinn, die untere dementsprechend die gegensätzliche Drehung. Ebenso verhält sich der Zoom-Button. Der Reset-Button (in Abbildung 5.6 zweiter von unten) besitzt die gleich Wirkung wie der „Reset View“-Menüpunkt in den Kontextmenüs. Er setzt das Koordinatensystem nach Rotation oder Zooming wieder in dessen Ausgangsposition und -größe zurück. Der Selektionsstatus der Objekte bleibt davon unberührt.

Abbildung 5.6: Interaction-Panel.

Der unterste Button in Abbildung 5.6 aktiviert den Mark-Modus, mit welchem gleichzeitig der Selektionsstatus von mehreren Datenobjekten verändert werden kann. Beim Klicken auf diesen Button verändert der Mauszeiger sich in ein Zielkreuz. Nun markiert man mit einem Klick im 3D-Scatterplot den oberen linken Eckpunkt eines Rechteckes und zieht dieses nun mit der Maus auf. Der untere rechte Eckpunkt wird durch das Loslassen der Maus bestimmt. Alle Datenobjekte, die sich in der Projektion unabhängig von der Tiefe in diesem Rechteck befinden, verändern ihren Selektionsstatus. Bereits selektierte Repräsentanten werden deselektiert und noch nicht selektierte werden selektiert.

5.2.7 MultiDataPoint-View

Die MultiDataPoint-View (MDPView) ist eine Visualisierung innerhalb des 3D-Scatterplots zur Exploration von Daten, die zu einem MultiDataPoint (MDP, s. Kapitel 2.2) zusammengefasst wurden. Wird im 3D-Scatterplot ein MDP mit der linken Maustaste angeklickt, nimmt die MDPView den Platz des 3D-Scatterplots im VisMeB-Fenster ein (s. Abbildung 5.7). Die Datenobjekte des angeklickten MDP werden als abgerundete Rechtecke auf einer Ellipsenbahn angeordnet, wobei sich maximal drei Objekte in der vorderen Hälfte befinden können und die restlichen in der hinteren aufgestaut werden. Vorne in der Mitte der MDPView markiert ein rötliches Rechteck ein Fokusfeld. Nach dem „details on demand“-Prinzip werden im „InfoPanel“ am linken Rand des Fensters nähere Informationen zu dem Datenobjekt, welches sich gerade zumindest teilweise im Fokusfeld befindet, angezeigt. Die visualisierten Metadatentypen entsprechen denen im Tooltip bei einzelnen Datenobjekten im 3D-Scatterplot. Diese Konformität soll dem Betrachter den Vergleich und die Orientierung erleichtern.

Zusätzlich informiert das „Infopanel“ über die aktuelle Achsenbelegung des Koordinatensystems und den dazugehörenden Werte des MDP, da diese für alle Objekte in dem MDP per Definition identisch sein müssen. Oberhalb des „InfoPanels“ befindet sich ein Drop-down-Menü, welches den Datentyp festlegt, nach dem die Objekte auf der Ellipsenbahn sortiert werden sollen. Standardmäßig wird nach der Gesamtrelevanz der Daten in Bezug auf die Suchbegriffe sortiert. Die zwei Checkboxen „Asc“ und „Desc“ bestimmen über die Sortierreihenfolge, auf- oder abwärts.

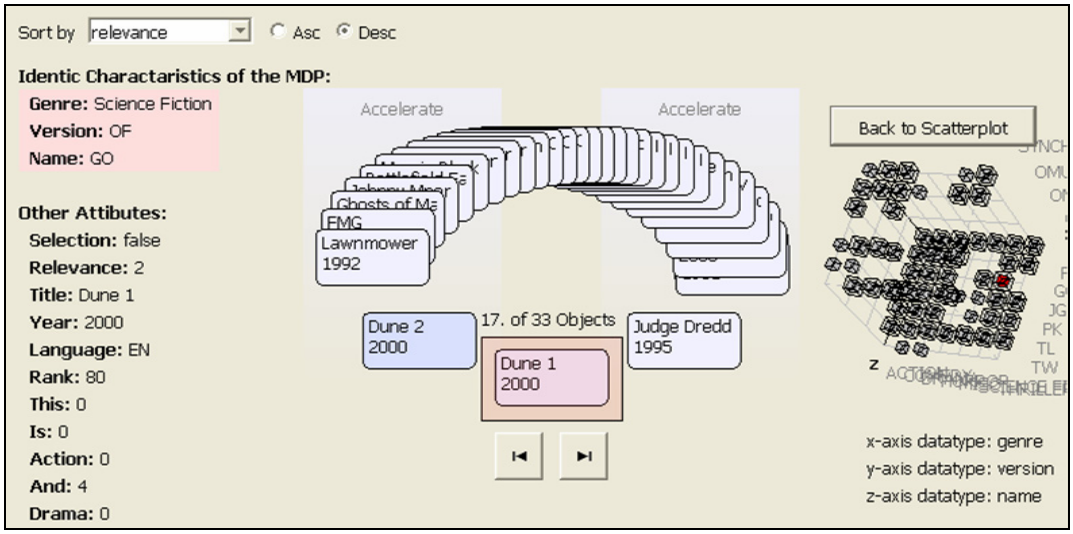


Abbildung 5.7: VisMeB 3D-Scatterplot, MDPView mit Filmdaten.

Sowohl auf der linken als auch auf der rechten Seite der Ellipse befindet sich jeweils ein Beschleunigungsfeld. Berührt die Maus zum Beispiel das Rechte, bewegen sich die Objekte auf der Ellipsenbahn im Uhrzeigersinn. Das Linke verursacht dementsprechend eine Bewegung gegen den Uhrzeiger. Die Geschwindigkeit der Bewegung ist abhängig von der Position der Maus im Bewegungsfeld. Je höher die Maus im Feld steht, umso schneller dreht sich die Ellipse. Verlässt die Maus das Feld, dreht sich die Ellipse noch insofern, dass das dem Fokusfeld nächste Objekt in dessen Mitte zur Ruhe kommt. Die Anzahl der Datensätze in dem MDP und die Nummer des aktuell fokussierten Objektes werden oberhalb des Fokusfeldes angezeigt. Unter dem Fokusfeld befinden sich zwei Buttons, mit denen die Ellipse Objekt für Objekt nach links oder rechts gedreht werden kann.

In der MDPView wird sogar ein zweifaches „overview and detail“-Konzept angewendet. Im „InfoPanel“ werden Details zu den im Überblick dargestellten Datenobjekten auf der Ellipsenbahn visualisiert. Diese sind wiederum Details zu dem rechts neben der Ellipse verkleinert angezeigten Koordinatensystem. In diesem sind alle Datenrepräsentanten außer dem aktuellen MDP, der rot markiert ist, in grau gehalten. Das Koordinatensystem kann wie im 3D-Scatterplot rotiert werden. Beim Klick auf das Koordinatensystem oder den „Back to Scatterplot“-Button wird die MDPView geschlossen und der normale 3D-Scatterplot wieder sichtbar.
Nähere Ausführungen bezüglich der MDPView sind unter [Lieb03] zu finden.

6 Mathematische Grundlagen von 3D-Visualisierungen

Bei graphischen Visualisierungen stehen dem Anwender zumeist verschiedene Interaktionsmöglichkeiten zur Verfügung um die dargestellten Objekte zu manipulieren. So können 3D-Objekte zum Beispiel verschoben (Translation), ihre Größe geändert (Skalierung) oder gedreht (Rotation) werden.

Diese Transformationen werden allgemein durch mathematische Operationen auf den Definitionspunkten der Objekte beschrieben. Die Definitionspunkte stellen in der Regel die Knoten- oder Eckpunkte der geometrischen Primitiven, aus welchen das Objekt zusammengesetzt wird, dar.

Im Folgenden werden nur die für diese Arbeit wesentlichen Grundlagen der Computergraphik zur 3D-Visualisierung näher beschrieben. Eine vollständige und detaillierte Abhandlung dieses Themas kann im Rahmen dieser Ausarbeitung leider nicht gegeben werden. Eine sehr ausführliche Aufarbeitung der Thematik ist in dem Standardwerk „Computer graphics: principles and practice“ von James D. Foley [FDFH90] zu finden.

6.1 Homogene Koordinaten

Homogene Koordinaten ermöglichen Transformationen, wie die Translation, Skalierung und Rotation, durch einheitliche Matrizenmultiplikationen auf die Definitionspunkte anzuwenden. Dies gestattet wiederum die Zusammenfassung von verschiedenen Transformationsschritten in eine gemeinsame Transformationsmatrix, indem die einzelnen Transformationsmatrizen als 4x4-Matrix miteinander multipliziert werden. So muss nicht mehr jeder Definitionspunkt mit jeder Transformationsmatrize, sondern nur noch mit einer einzigen Matrix multipliziert werden.

Beispielsweise benötigt eine Anwendung von vier Rotationen auf zehn Punkten ohne eine gemeinsame Transformationsmatrix 4*10 = 40 Matrizenmultiplikationen und verursacht damit eine Laufzeit von *O(m*n)*, wobei *m* die Anzahl der Transformationsschritte und *n* die Anzahl der zu transformierenden Punkte darstellt. Wird nun zuerst die gemeinsame Transformationsmatrix berechnet und dann erst diese mit den einzelnen Punkten multipliziert, reichen 3+10 = 13 Matrizenmultiplikationen aus und benötigt somit nur eine lineare Laufzeit von *O(n)*.

Homogene Koordinaten zeichnen sich durch eine den Punkten zusätzlich hinzugefügte Koordinate *w*, der homogenen Komponente, aus. Die Punkte in 3D werden entsprechend nicht mehr mit drei Koordinaten *P(x, y, z)*, sondern mit vier Koordinaten *P(x, y, z, w)* ausgedrückt, wobei zumindest eine der Koordinaten verschieden von Null sein muss.

Gleichung 6-1:

$$P(x, y, z) = P(x \cdot w, y \cdot w, z \cdot w, w) = P(x, y, z, 1)$$

Ist eine homogene Koordinate, z.B. *(2, 4, 6, 8)*, gerade das Vielfache einer anderen homogenen Koordinate, z.B. *(1, 2, 3, 4)*, so spezifizieren diese denselben Punkt. Dies hat zur Folge, dass jeder Punkt eine Menge von verschiedenen homogenen Koordinaten besitzt, die diesen repräsentieren.

Liegen die homogenen Koordinaten mit $w \neq 0$ *und* $w \neq 1$ vor, werden im Allgemein die Koordinaten durch *w* dividiert und die Werte *x/w*, *y/w* und *z/w* als kartesische Koordinaten des homogenen Punktes bezeichnet.

Zur Vereinfachung kann man der homogenen Komponente eines Punktes grundsätzlich den Wert *w=1* zuweisen. So entsprechen die homogenen Koordinaten *x*, *y* und *z* immer den kartesischen Koordinaten des Punktes. Die echte Konvertierung der kartesischen in homogene Koordinaten kann daher vermieden werden.

6.2 Translation

Unter Translation versteht man eine geradlinige Verschiebung eines Objektes bzw. seiner Definitionspunkte. Die neue Position eines Punktes wird durch die Addition des Verschiebungswertes mit den Punktkoordinaten berechnet.

Eine Translation des Punktes *P(x, y, z)* um *dx* Einheiten parallel zur x-Achse, um *dy* Einheiten parallel zur y-Achse und um *dz* Einheiten parallel zur z-Achse ist wie folgt definiert:

Gleichung 6-2:

$x' = x + dx$ $\quad y' = y + dy$ $\quad z' = z + dz$

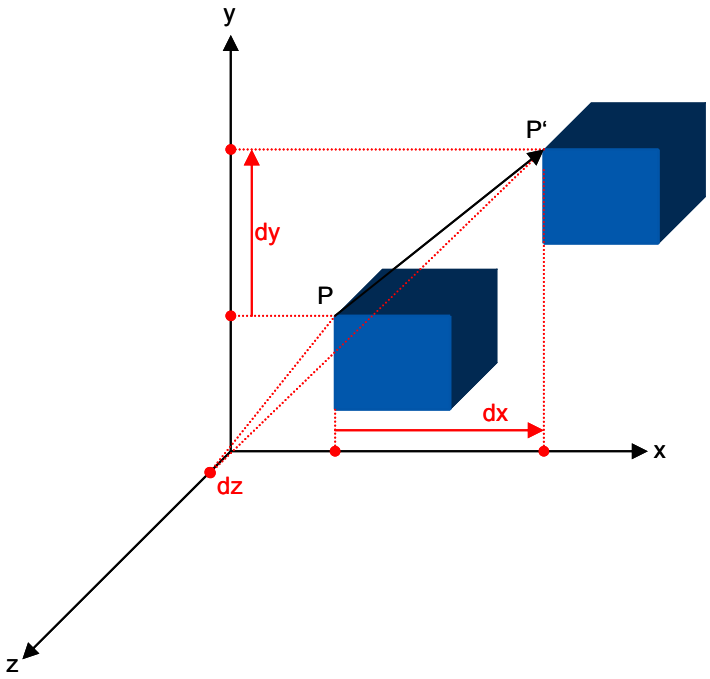


Abbildung 6.1: Translation eines 3D-Objektes um dx und dy.

Für den dreidimensionalen Raum unter Verwendung homogener Koordinaten ergibt sich die folgende Translationsgleichung:

Gleichung 6-3:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Die linke Seite entspricht den neuen Koordinaten des verschobenen Punktes, die rechte Seite beschreibt die ursprünglichen Koordinaten, welche mit der Translationsmatrix multipliziert werden.

Bei der vorher besprochenen Kombination von Transformationen wird die Translationsmatrix mit der einheitlichen Transformationsmatrix multipliziert und diese auf die dargestellten Objekte angewendet.

6.3 Skalierung

Die Skalierung bezeichnet die Vergrößerung bzw. Verkleinerung von Objekten. Dabei variiert man die Abstände der Eckpunkte des zu skalierenden Objektes durch Multiplikation mit dem Skalierungsfaktor.

Gleichung 6-4:

$$x' = sx \cdot x \qquad y' = sy \cdot y \qquad z' = sz \cdot z$$

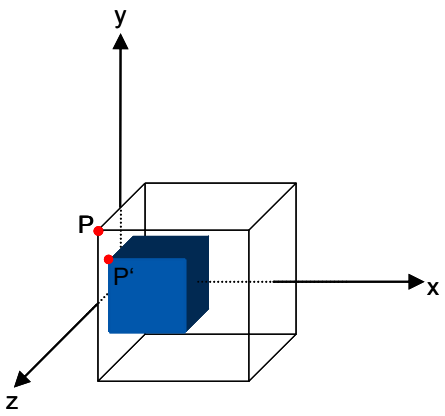


Abbildung 6.2: Skalierung eines 3D-Objektes um Faktor 0,5.

Für den dreidimensionalen Raum unter Verwendung homogener Koordinaten ergibt sich so die folgende Skalierungsgleichung:

Gleichung 6-5:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Sind die Faktoren *sx*, *sy*, *sz* betragsmäßig gleich, so erfolgt durch diese uniforme Skalierung eine proportionalitätserhaltende Vergrößerung bzw. Verkleinerung des Objektes. Andernfalls wird das Objekt verzerrt.

6.4 Rotation

Bei der einfachen Rotation eines Objektes wird dieses um einen Winkel α um die x-, y- oder z-Achse des Koordinatensystems, gedreht.

Soll um eine beliebige Achse gedreht werden, muss die Drehachse und damit auch das Objekt so verschoben und rotiert werden, dass die Drehachse durch den Ursprung verläuft und auf einer Koordinatenachse liegt. Das Objekt wird nun um die jeweilige Achse gedreht und abschließend die Drehachse mit rotiertem Objekt durch inverse Transformation an die ursprüngliche Position zurück gedreht bzw. verschoben.

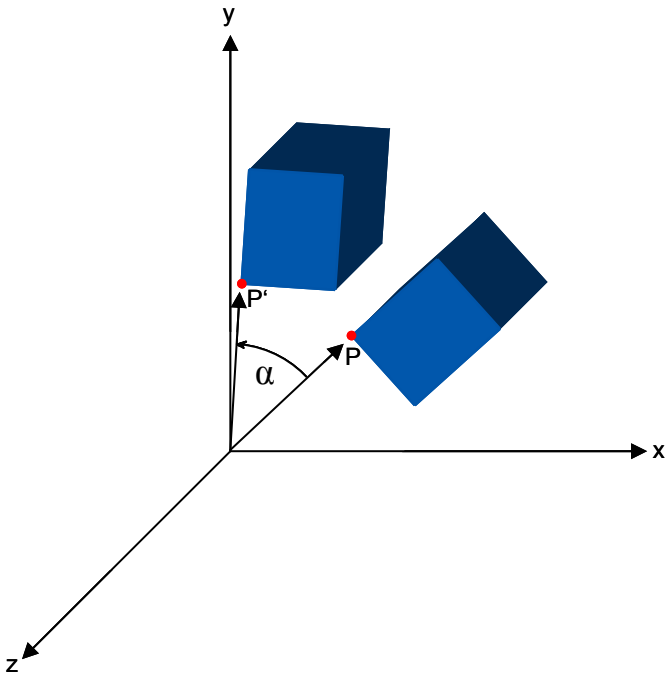


Abbildung 6.3: Rotation eines 3D-Objektes um α um die z-Achse.

Folgende Gleichung gilt für die Drehung um α um die x-Achse:

Gleichung 6-6:

$$x' = x$$
$$y' = y \cdot \cos(\alpha) - z \cdot \sin(\alpha)$$
$$z' = y \cdot \sin(\alpha) + z \cdot \cos(\alpha)$$

Für den dreidimensionalen Raum unter Verwendung homogener Koordinaten ergibt sich für die Rotation um die x-Achse folgende Gleichung:

Gleichung 6-7:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & \sin(\alpha) & 0 \\ 0 & -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Analog für die Rotation um die y-Achse:

Gleichung 6-8:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Analog für die Rotation um die z-Achse:

Gleichung 6-9:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & \sin(\alpha) & 0 & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

6.5 Projektion

Als Projektion wird die Abbildung von Punkten eines *n*-dimensionalen Koordinatensystems in ein Koordinatensystem mit *m* Dimensionen für $m<n$ verstanden. Auf Grund der Thematik dieser Arbeit bezieht sich dieses Kapitel ausschließlich auf die Abbildungsverfahren zur Darstellung von dreidimensionalen Objekten durch zweidimensionale Abbildungen.

Die Projektion ist durch verschiedene Parameter gekennzeichnet. Vom Projektionszentrum (Projection Reference Point, PRP) verlaufen die Projektionsstrahlen, auch Projektoren genannt, durch die Punkte des zu projizierenden Objektes und schneiden anschließend die Projektionsebene.

Der resultierende Schnittpunkt entspricht der Projektion des jeweiligen Punktes, welcher auf dem Projektionsstrahl liegt, auf die Projektionsebene (View Plane, siehe Kapitel 6.6).

Je nach Lage des Projektionszentrums kann die Projektion in zwei Arten – perspektivische und parallele Projektion – unterschieden werden. Ist der Abstand von Projektionszentrum zu Projektionsebene endlich, liegt eine perspektivische Projektion vor; andernfalls spricht man von einer Parallelprojektion.

Beide gehören zu den planaren geometrischen Projektionen, da die Abbildungen durch gerade und nicht gekrümmte Projektoren auf eine Ebene, anstatt auf eine gekrümmte Fläche, projiziert werden. Bei kartographischen Anwendungen, wie zum Beispiel bei der Realisation einer Weltkarte, werden eher gekrümmte Flächen, wie Kugel oder Zylinder, als Projektionsebene verwendet, um eine vollständige Repräsentation des Objektes zu ermöglichen.

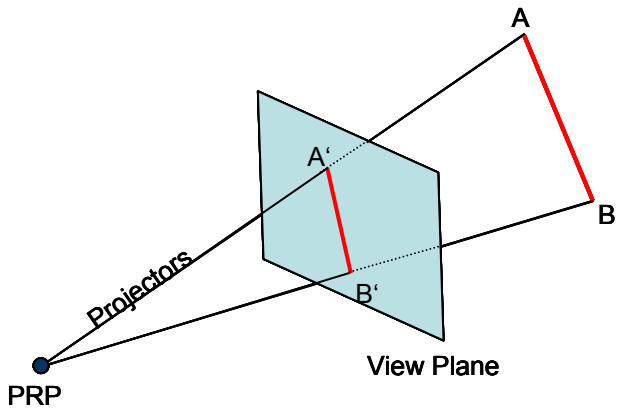


Abbildung 6.4: Perspektivische Projektion der Linie AB auf die View Plane [FDFH90].

Der erzeugte visuelle Effekt der perspektivischen Projektion ist dem des menschlichen Sehsystems und der herkömmlichen Fotographie sehr ähnlich und wird daher auch als natürlich empfunden. Objekte, die einen geringeren Abstand zum Projektionszentrum besitzen, werden größer als weiter entferntere dargestellt.

Diese so genannte perspektivische Verkürzung trägt erheblich zum Realismus der Abbildung bei, beeinflusst dabei aber Form, Winkel, Abstände und Parallelität. So bleibt die Parallelität von ursprünglich parallelen Strecken des Objektes bei der Abbildung nicht unbedingt erhalten.

Für Anwendungen, wie zum Beispiel CAD, bei denen es auf realitätsgetreue Maße der Objekte ankommt, ist die Parallelprojektion eher geeignet.

Hier bleiben bei der Projektion die Form, Abstände und Parallelität erhalten. Nur die Winkeltreue ist, wie auch bei der perspektivischen Projektion, abhängig von der Lage des Objektes zu der Projektionsebene.

Verlaufen die Projektoren parallel zur Normalen der Projektionsebene, wird von einer orthogonalen, ansonsten von einer schiefen Parallelprojektion gesprochen.

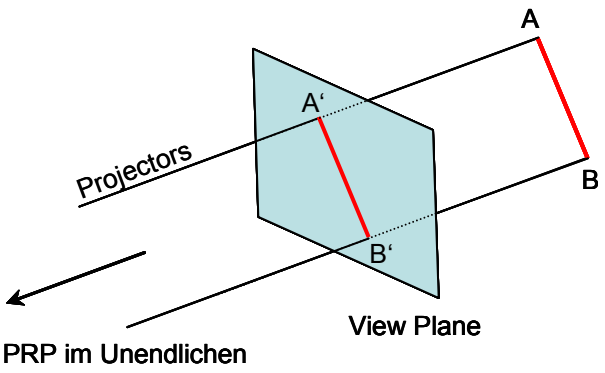


Abbildung 6.5: Parallelprojektion der Linie AB auf die View Plane [FDFH90].

6.6 Projektionstransformation

Die Projektionstransformation bildet Objekte von ihren einzelnen Objektkoordinatensystemen in das Beobachterkoordinatensystem ab. Hierbei findet eine Abbildung des dreidimensionalen Raumes auf eine zweidimensionale Fläche statt. Diese Fläche wird als Projektionsebene (View Plane) bezeichnet und kann direkt auf den Viewport des Anzeigegerätes, wie zum Beispiel Monitor oder Beamer, transformiert werden. Der Viewport beschreibt die Anzeigefläche auf dem Ausgabegerät, in welchem schlussendlich die Projektion dargestellt werden soll.

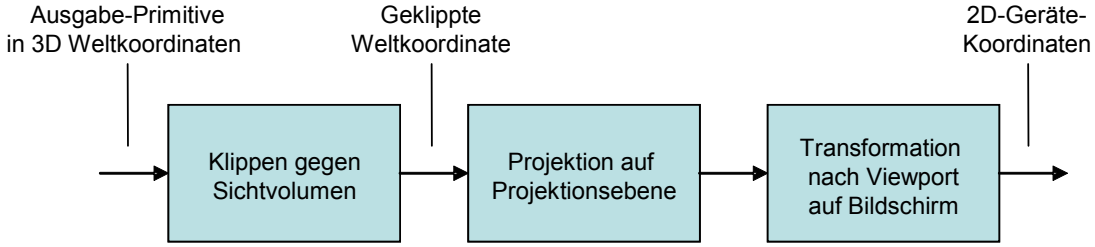


Abbildung 6.6: Konzeptionelles Modell der 3D-Sichtbarkeitstransformation, GKS [FDFH90].

Die Projektionsebene wird durch den View Reference Point (VRP) und die View Plane Normal (VPN) definiert.
Der View Reference Point gibt einen Punkt auf der Projektionsebene an und bestimmt daher ihre Position im Raum.
Die View Plane Normal stellt die Normale der Ebene und damit den Blickwinkel des Betrachters dar.

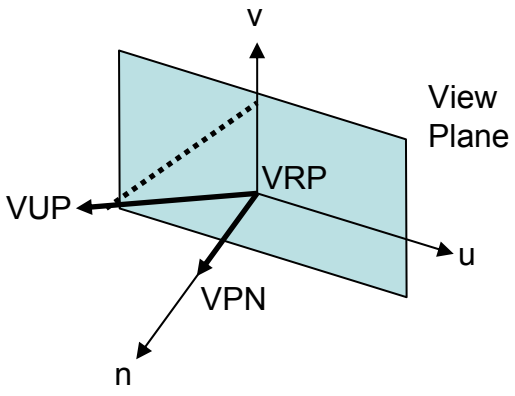


Abbildung 6.7: u-v-n-Koordinatensystem [FDFH90].

Da die Projektionsebene im Raum frei positioniert werden kann, besteht die Möglichkeit, dass Objekte aus dem Betrachtungsfeld gelangen und dann nicht mehr berechnet, projiziert und angezeigt werden sollen.

Um diese Objekte zu entfernen, wird ein View Volume (Sichtkörper) definiert, an welchem die Objekte meist noch vor der Projektionsphase abgeschnitten (Clipping) werden. Das heißt, dass die Objekte, die sich ganz oder nur zum Teil außerhalb dieses definierten Sichtraumes befinden, an den Grenzen des View Volume abgeschnitten und nicht weiter für die Projektion betrachtet werden. Nur der Teil des Raumes, welcher vom View Volume eingeschlossen ist, bleibt Gegenstand der Betrachtung.

Um dieses View Volume angeben zu können, wird ein u-v-n-Koordinatensystem, auch Viewing Reference Coordinate (VRC) System genannt, festgelegt, in welchem der VRP den Ursprung und die VPN die n-Achse darstellt.

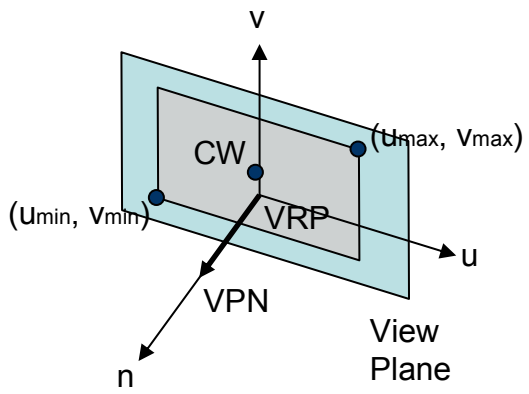


Abbildung 6.8: Viewing Reference Coordinate System,
CW als Mittelpunkt des Windows [FDFH90].

Die v-Achse stimmt mit der Projektion des View Up Vectors (VUP) parallel zur VPN auf der View Plane überein. Dabei definiert der VUP die vertikale Ausrichtung, also das „Oben“, der Projektionsebene.

Die u-Achse des VRC wird so gelegt, dass u, v und n ein rechtshändiges Koordinatensystem ergeben.

Zur Festlegung des View Volumes wird zuerst ein rechteckiges Fenster (Window) auf der View Plane mit dem Mittelpunkt CW definiert. Dieses Fenster grenzt den Betrachtungsraum in Richtung der u- und v-Achse mit n = 0 ein.

Die letztendliche Form des View Volumes ist abhängig von der Projektionsart:

- Bei der perspektivischen Projektion stellt das View Volume eine Pyramide dar, dessen Spitze durch das Projektionszentrum (PRP) repräsentiert wird und dessen Kanten durch die Ecken des Windows verlaufen.

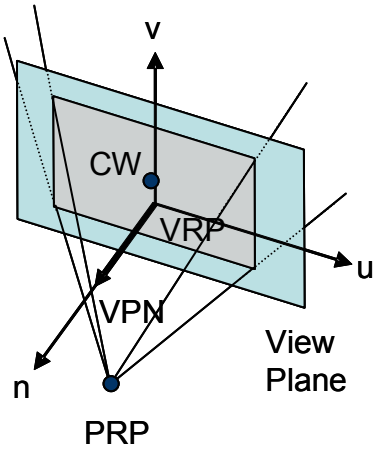


Abbildung 6.9: Pyramide als View Volume bei der perspektivischen Projektion [FDFH90].

- Liegt eine Parallelprojektion vor, ist das View Volume ein unendlicher Quader, dessen Seiten parallel zu der Projektionsrichtung liegen. Die Projektionsrichtung verläuft vom Projektionszentrum PRP zum Fenstermittelpunkt CW.

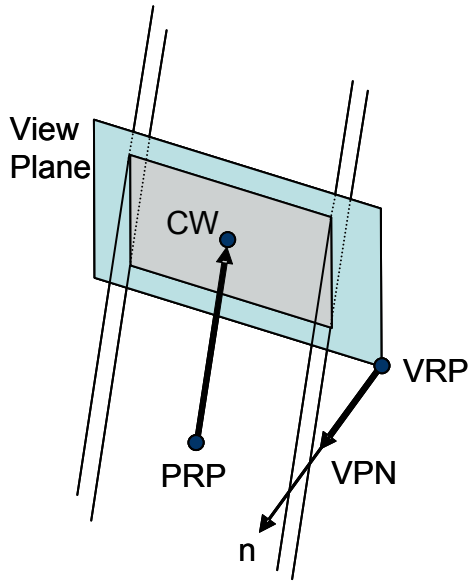


Abbildung 6.10: Unendliches Quader als View Volume bei der Parallelprojektion [FDFH90].

Um das View Volume in Richtung der n-Achse einschränken zu können, werden zwei zur View Plane parallele Ebenen, die Front und Back Clipping Plane, definiert, welche das View Volume schneiden und damit den Raum in der Tiefe begrenzen.

Diese Ebenen werden über die vorzeichenbehaftete Distanzen Front Distance (F) und Back Distance (B) relativ zum VRP spezifiziert. Um eine positives Sichtvolumen zu gewährleisten, muss F > B gelten.

Auf die verschiedenen Clipping- und Projektions-Algorithmen wird in dieser Arbeit nur projektspezifisch bei der Implementationsbeschreibung in den folgenden Kapiteln näher eingegangen.

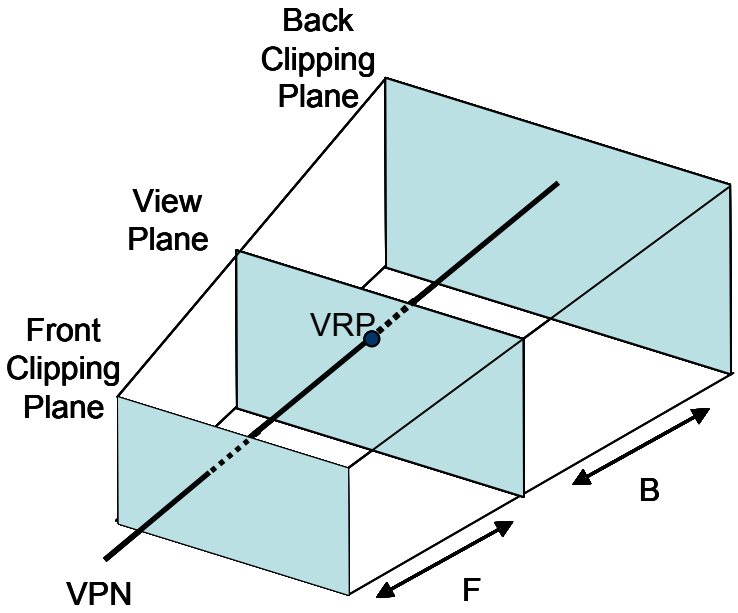


Abbildung 6.11: Abgeschnittenes View Volume bei perspektivischer Projektion [FDFH90].

7 Implementation des 3D-Scatterplots

7.1 Projektablauf

Zu Beginn des Projektes stand ein 3D-Scatterplot, der im Rahmen einer Vorlesung implementiert wurde, als Grundlage zur Verfügung. Wie schon in Kapitel 5.1 beschrieben, stellte sich heraus, dass dieser nicht in die bestehende Projektumgebung integriert werden konnte, da die Differenzen in den Datenmodellen nicht in vertretbarer Zeit hätten behoben werden können. Daher wurde eine komplette Neukonzeption und -implementation nötig, um der Aufgabenstellung gerecht zu werden. Die Konzeption und ihre Umsetzung werden in den folgenden Kapiteln beschrieben.

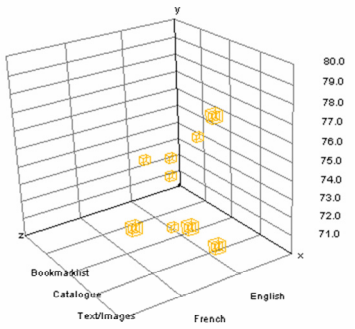


Abbildung 7.1:Erste Implementation[1].

7.2 VisMeB – Umgebung

Der 3D-Scatterplot ist als zusätzliche Visualisierung in das bestehende VisMeB Projekt integriert. Er verwendet das zugrunde liegende Datenmodell und unterstützt die Interaktion zwischen den verschiedenen Darstellungen. Aus diesem Grund wird hier kurz die Umgebung geschildert, in welche der 3D-Scatterplot integriert ist.

Über das Visual Configuration and Assignment Tool (siehe Kapitel 4.1) wird bestimmt, welche Felder aus der Datenbank ausgelesen und den Visualisierungen zugeordnet werden. Dabei stellt das Tool sicher, dass nur solche Felder den Visualisierungen zugeordnet werden, die diese auch verarbeiten können. Dem VisMeB Datenmodell stehen fünf verschiedene Datentypen zur Verfügung: String, Integer, Double, Boolean und Date.

Für jeden Datensatz in der angebundenen Tabelle oder View der Datenbank wird in einem Array der Klasse *DocumentManager* ein Objekt der Klasse *Document* angelegt. Jedes Document beinhaltet wieder einen Array, der für jedes Feld die Daten des Datensatzes enthält. Diese geschachtelte Array-Struktur kann man sich vereinfacht als Tabelle vorstellen, in der die Zeilen die Documents im Array des DocumentManagers und die Spalten die Daten an einer bestimmten Position in jedem Array der Documents sind.

Möchte man beispielsweise die Daten aus der dritten Zeile in der zweiten Spalte auslesen, muss man vom DocumentMangager die Instanz der Klasse Document an der Position 3 anfordern. In dieser stehen im Array an der Position 2 die gewünschten Daten. Die Datenfelder bzw. Spalten entsprechen also immer genau einer Position in allen Document-Arrays. Die Positionen werden in der Klasse Assignment den Konstanten für die möglichen Belegungen in den Visualisierungen zugeordnet. Die Visualisierungen benötigen also keine Informationen über die Indexe, sondern können einfach über die Konstanten die benötigten Daten anfordern. Die Spaltennamen und Minimal- bzw. Maximalwerte pro Spalte werden in der Klasse *FieldManager* gespeichert.

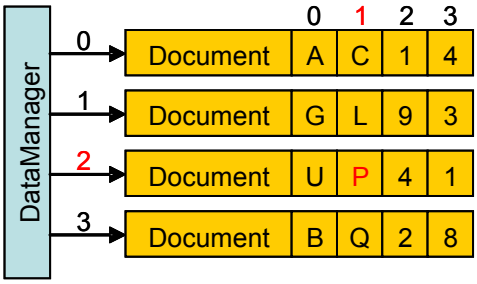


Abbildung 7.2: Vereinfachtes Datenmodell - Doppelter Array.

Alle Visualisierungen in VisMeB müssen gleichartige Methoden zur Initialisierung, Update, Repaint und Statusänderung in Bezug auf Selektion und Fokus bereitstellen. Diese Konformität wird durch das Interface View, welches von allen Visualisierungen implementiert wird, gewährleistet. Bei der Initialisierung der Visualisierungen melden sie sich beim *ViewNotifier* an. Gibt es eine Änderung in einer Darstellung, die auch auf die anderen Auswirkungen haben soll, gibt diese eine Meldung an den ViewNotifier weiter, der wiederum alle angemeldeten Visualisierungen benachrichtigt. Diese Methodik ermöglicht erst die eingesetzten „brushing and linking“-Techniken.

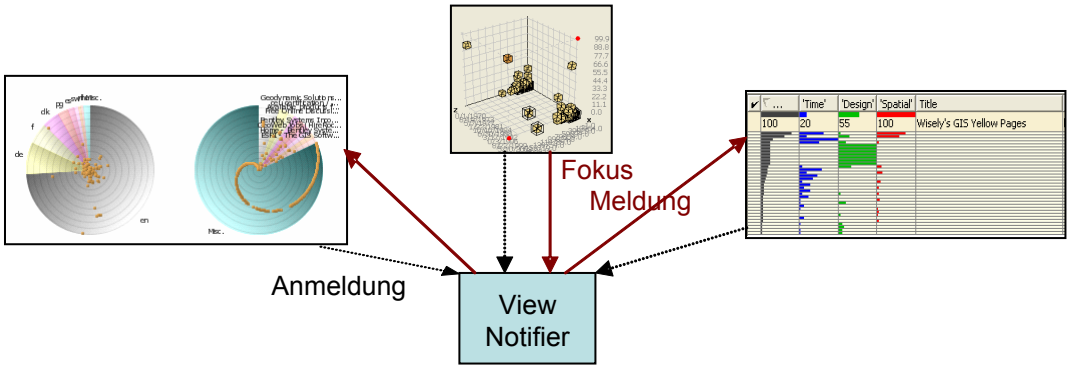


Abbildung 7.3: Interaktion - ViewNotifier und Visualisierungen.

7.3 3D-Scatterplot – Konzeption

Der Implementation des 3D-Scatterplots ging eine ausführliche Konzeptionsphase voraus, da Fehler im Konzeptdesign später nur noch schwer korrigierbar gewesen wären. Durch detaillierte Vorüberlegungen wurden Problemstellungen schon vor der Entwicklung sichtbar und konnten grundlegend gelöst werden. Der Schwerpunkt lag auf einem effizienten und dennoch generischen Design nach den Prinzipien der Objektorientierung. Tatsächlich musste das entworfene Konzeptdesign während der Implementation nicht mehr verändert werden und ermöglichte eine effektive Programmierung, da parallel im Team entwickelt werden konnte.

Der 3D-Scatterplot sollte weitestgehend unabhängig von VisMeB entwickelt werden und nur durch schmal definierte Schnittstellen mit dem Datenmodell und dem ViewNotifier (siehe Kapitel 7.2) kommunizieren. Dies ermöglicht eine Wiederverwendung in anderen Projekten und minimiert die Fehleranfälligkeit und den Supportaufwand. Änderungen in anderen Visualisierungen oder im Datenmodell verursachen keine Anpassungen des 3D-Scatterplots, solange die Schnittstellen unberührt bleiben.

Dieses Konzept der Kapselung von Aufgabeneinheiten wurde auch innerhalb des 3D-Scatterplots weiter verfolgt. Die Gesamtproblemstellung wurde in mehrere Teilprobleme bzw. Klassen, die durch definierte Schnittstellen mit einer Controller-Klasse kommunizieren, aufgeteilt (siehe Abbildung 7.5). Dies hilft vor allem die Komplexität zu reduzieren, da jede Klasse nur Methoden zur Lösung ihres „Problems“ zur Verfügung stellen muss und von anderen Vorgängen unberührt bleibt. Die Controller-Klasse ist das Bindeglied der verschiedenen Klassen und ist für die Initialisierung und Koordination zuständig. Dieses Konzept ist vereinfacht in der Informatik als Model-View-Controller-Konzept (MVC) bekannt, wobei hier die Visualisierung (View) und die Funktionseinheit (Model) von einander getrennt und durch den Controller verbunden werden.

Ein weiterer Vorteil der Kapselung von sinnvollen Einheiten mit definierten Schnittstellen liegt in der Möglichkeit zur parallelen Entwicklung im Team. Die Klassen können unabhängig voneinander implementiert und erst später über die in der Konzeptionsphase definierten Schnittstellen verbunden werden.

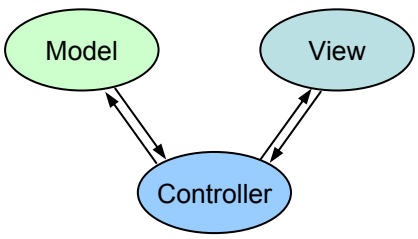


Abbildung 7.4: Model-View-Controller-Konzept.

Die Controller-Klasse im 3D-Scatterplot ist die Klasse *ScatterplotView3D*. Diese wird von VisMeB bei der Initialisierung gestartet. Zur Interaktion mit dem Datenmodell und den anderen VisMeB Visualisierungen implementiert sie das Interface *View* und meldet sich beim ViewNotifier an. ScatterplotView3D initialisiert alle anderen 3D-Scatterplot Klassen und stellt zwischen ihnen die Verbindung her. Sie selbst erbt die Container-Funktionalitäten des JPanels und kann daher von VisMeB direkt in die Gesamtvisualisierung eingebettet werden.

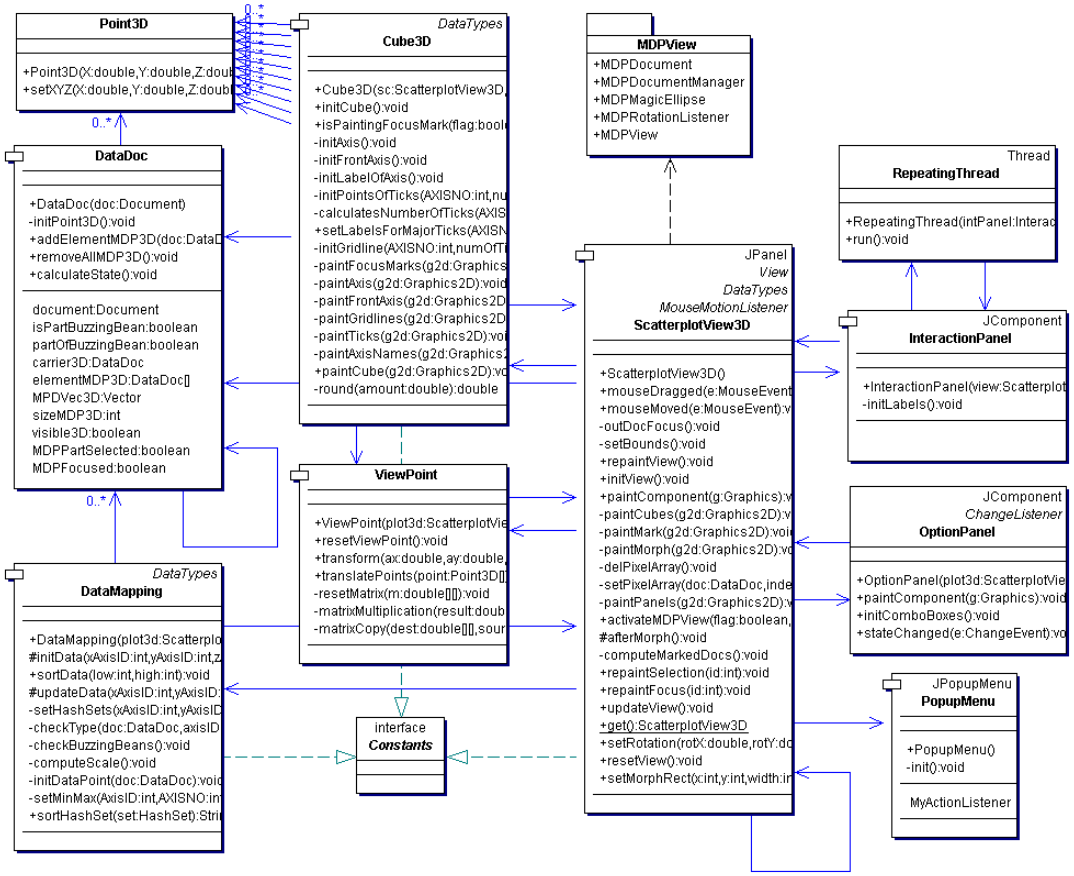


Abbildung 7.5: UML-Diagramm VisMeB 3D-Scatterplots.

Der 3D-Scatterplot besitzt ein eigenes Datenmodell, um die speziellen Anforderungen dieser Visualisierung zu unterstützen. Für jeden Datensatz bzw. jede Instanz der Klasse *Document* im VisMeB Datenmodell wird hier eine Instanz der Klasse *DataDoc* erstellt, welche eine Referenz auf das Document und weitere Eigenschaften für diese Visualisierung enthält. Die DataDocs werden in einem Array in der Klasse *DataMapping* gespeichert, welche auch den DataDocs entsprechend ihren Werten bezüglich der Achsenbelegung eine Position im 3D-Raum zuweist. Die Position wird im DataDoc durch ein Array mit 18 Instanzen der Klasse Point3D gespeichert, welche Definitionspunkte für die Repräsentation des DataDocs als Würfel oder Pyramide in der Visualisierung darstellen.

Die Klasse *Cube3D* stellt Methoden zur Berechnung und Darstellung des 3D-Koordinatensystems mit Gitternetz und Beschriftung bereit. Diese werden, wie die DataDocs, mittels Instanzen der Klasse Point3D im 3D-Raum positioniert.

Zur Rotation und zum Zooming des Koordinatensystems stellt die Klasse *ViewPoint* Transformationsmatrizen zur Verfügung und projiziert die transformierten 3D-Punkte auf die zweidimensionale Ebene. Das *InteractionPanel* und *OptionPanel* stellen Buttons und Drop-down-Menüs bereit um die Koordinatenachsen zu belegen oder die Visualisierung zu manipulieren. Sie erben die Container-Funktionalitäten des JPanels und können daher direkt zur ScatterplotView3D-Klasse hinzugefügt und angezeigt werden. Der *RepeatingThread* wird zur kontinuierlichen Ausführung der Manipulation des Koordinatensystems über die Buttons verwendet. Dieser Vorgang wird in einem parallelen Prozess ausgeführt und beansprucht daher nicht die Ressourcen des VisMeB Prozesses.

Im Package *Scatterplot3D* sind außer den bisher schon vorgestellten Klassen noch das Interface *Constants* und die Klasse *PopupMenu* enthalten. Constants definiert, wie der Name schon sagt, 3D-Scatterplot spezifische Konstanten. Die Klasse PopupMenu stellt die jeweiligen Kontextmenüs bereit.

Das Package *MDPView* beinhaltet weitere Klassen für die MultiDataPoint-View, die aus dem 3D-Scatterplot heraus aufgerufen werden kann. Die Konzeption und Implementation dieser Teilvisualisierungen ist nicht Teil dieser Arbeit, kann aber unter [Lieb03] eingesehen werden.

Die aktuelle Implementation des 3D-Scatterplots umfasst elf Klassen nur für den Scatterplot und nochmals fünf Klassen für die MultiDataPoint-View. Insgesamt besteht der Quellcode aus über 4000 Zeilen und ist als Java Bytecode 135 KB groß.

Es wurde bei der Konzeption und Implementierung sehr viel Wert auf effiziente Algorithmen und Ressourcenschonende Datenhaltung gelegt, da Java einmal belegten Speicher nur unbefriedigend wieder frei gibt und graphische Prozesse nicht auf die Grafikkarte ausgelagert werden können, sondern direkt von dem CPU berechnet werden. Auch ist der von Java erzeugte Bytecode, welcher von der Java Virtual Machine ausgeführt wird, zum Teil langsamer als ein von beispielsweise C++ erzeugter Maschinencode, der direkt vom Prozessor interpretiert werden kann.

7.4 Datenmodellierung

Wie schon beschrieben, besitzt der 3D-Scatterplot ein eigenes Datenmodell, wobei die aus der Datenbank ausgelesenen Daten nicht doppelt gespeichert, sondern nur referenziert werden. Die benötigten Speicherressourcen erhöhen sich dadurch nur unerheblich und man erhält die Möglichkeit unabhängig von anderen Visualisierungen das Datenmodell entsprechend den speziellen Bedürfnissen des 3D-Scatterplots zu modellieren.

Für jede Instanz der Klasse Document im VisMeB Datenmodell wird von der Klasse DataMapping im 3D-Scatterplot bei der Initialisierung eine Instanz der Klasse DataDoc (Quellcode 1) erzeugt und in einem Array gespeichert. Diese Instanz beinhaltet eine Referenz auf das entsprechende Document, damit die eigentlichen Daten für die Visualisierung ausgelesen werden können. DataDoc enthält nur Daten im Bezug auf die Darstellung, wie den x-, y- und z-Datenwert und ein Array mit 18 Instanzen der Klasse Point3D (Quellcode 5) zur Positionierung der Definitionspunkte der Datenrepräsentation.

Zur Behandlung der MultiDataPoints (siehe Kapitel 2.2) werden im DataDoc auch ein Array vom Typ DataDoc, eine Referenz auf den MDP-Repräsentanten und mehrere Booleanwerte gespeichert. Liegen mehrere DataDocs mit identischem x-, y- und z-Datenwert, also als MDP vor, wird das erste als Repräsentant aller anderen ausgewählt. In der Visualisierung wird nur noch der Repräsentant als MDP dargestellt.

Alle DataDocs in diesem MDP speichern eine Referenz auf den Repräsentanten und dieser speichert in dem genannten Array die Referenzen aller inhärenten DataDocs. Die gegenseitige Referenzierung ermöglicht einen direkten Rückschluss auf alle DataDocs im MDP.

Wählt der Anwender im 3D-Scatterplot eine neue Achsenbelegung aus oder führt er eine neue Suche durch, wird das 3D-Scatterplot Datenmodell aktualisiert. Dabei werden den x-, y- und z-Datenwerten der DataDocs die entsprechenden Werte zugewiesen. Integer- und Double-Werte werden direkt abgespeichert, Boolean-Werte werden auf 0 oder 1 abgebildet, Date-Werte in Sekunden umgerechnet und String-Werten wird ihr Rang bei einer alphabetischen Sortierung der Datenmenge ohne Duplikate zugewiesen (Quellcode 4).

Anschließend werden die Werte linear normalisiert und auf die Achsenlänge des globalen Würfels bzw. Koordinatensystems skaliert. Alle Datenpunkte liegen damit in dem Würfel, wobei der Ursprung des Koordinatensystems den Minimalwert darstellt. Anhand dieser normalisierten Werte werden zu den einzelnen DataDocs die Definitionspunkte für die Repräsentation im Raum berechnet.

Acht Punkte für den „normalen“ Datenwürfel, nochmals acht Punkte für das äußere MDP-Gitter, einen für das Zentrum und einen für die Spitze der Pyramide bei teilweise selektierten MDPs, also insgesamt 18 Punkte, werden im Array des DataDocs als Point3D gespeichert. Das Zentrum des Datenwürfels liegt dabei genau auf der berechneten Position für den Datenwert. Durch die Vielzahl von Definitionspunkten kann gewährleistet werden, dass die Datenrepräsentation als Würfel oder Pyramide auch noch bei einer Rotation perspektivisch richtig im Raum liegt und verzerrt wird.

Die berechnete Position eines Definitionspunktes wird im Point3D als untransformierte Koordinate gespeichert. Diese bleibt bei einer Transformation oder Projektion unverändert. Zusätzlich werden die transformierten x-, y-, und z-Werte und die projizierten x- und y-Koordinaten im Point3D gespeichert, die sich bei jeder Rotation oder bei jedem Zooming verändern.

7.5 Visualisierung

7.5.1 Transformation & Projektion

Jedes visuelle Element des 3D-Scatterplot wird in seiner Position, Ausrichtung und Größe durch Definitionspunkte bestimmt. Ein Würfel entsteht beispielsweise beim Verbinden seiner Definitions- bzw. Eckpunkte. Die Definitionspunkte sind, wie im vorherigen Kapitel schon beschrieben, vom Typ Point3D, welcher die untransformierten, die transformierten und die projizierten Koordinaten speichert.

Die untransformierten Koordinaten beschreiben die Ausgangsposition eines Objektes im Raum. Bei einer Rotation oder Zooming des 3D-Scatterplots wird die transformierte Position der Definitionspunkte berechnet und im Point3D als transformierte Koordinaten abgespeichert. Da die Visualisierung auf einem zweidimensionalen Ausgabegerät angezeigt werden soll, müssen die dreidimensionalen Koordinaten auf eine zweidimensionale Fläche projiziert werden. Daraus ergeben sich x- und y-Koordinaten, die ebenfalls in Point3D abgespeichert werden. Bei der Zeichnung des 3D-Scatterplots wird die projizierte Position der Definitionspunkte ausgelesen und entsprechend verbunden.

Die Methoden zur Berechnung der verschiedenen Transformationen und der Projektion stellt die Klasse ViewPoint zur Verfügung. Hier werden nach den Grundlagen der Transformation (siehe Kapitel 6) Matrizenmultiplikationen auf homogenen Koordinaten ausgeführt. In Folge eines Transformationsimpulses durch den Anwender, wird eine Einheitsmatrix nacheinander mit der jeweiligen Transformationsmatrix für Skalierung und Rotation um die jeweilige Achse multipliziert (Quellcode 7).

Die daraus resultierende Matrix wird anschließend mit der globalen Transformationsmatrix multipliziert. Diese gleicht bei der Initialisierung der Einheitsmatrix und wird bei jeder Transformation durch die eben genannte Multiplikation verändert. Soll der 3D-Scatterplot wieder in seiner Ausgangslage angezeigt werden, reicht die Zurücksetzung der globalen Transformationsmatrix auf die Einheitsmatrix.

Bei der Projektion der Definitionspunkte auf die zweidimensionale Darstellungsfläche werden zuerst für jedes Objekt der Klasse Point3D, also für alle Definitionspunkte, die transformierten Koordinaten berechnet. Dafür werden die untransformierten Koordinaten mit der globalen Transformationsmatrix multipliziert und das Ergebnis in Point3D als Variablen *wx*, *wy* und *wz* abgespeichert.

Anschließend werden *wx* und *wy* mit einer Konstanten multipliziert und durch die Summe aus *wz* und einer Distanzkonstanten dividiert. Daraus resultiert die x- und y-Koordinate des Definitionspunktes bezüglich der eingesetzten perspektivischen Projektion, welche den Variablen *x2d* und *y2d* der Point3D-Objekte zugewiesen werden (Quellcode 6).

7.5.2 Cube3D – das 3D Koordinatensystem

Die Klasse Cube3D ist verantwortlich für die Berechung und Darstellung des Koordinatensystems mit Hilfslinien und Beschriftung im 3D-Scatterplot.

Wie die Datenrepräsentanten werden auch die Visualisierungselemente des Koordinatensystems durch Definitionspunkte vom Typ Point3D definiert. Dabei ist die Größe der Visualisierung des Koordinatensystems abhängig von der Größe des VisMeB-Fensters. Eine weitere Abhängigkeit besteht zwischen der Anzahl von Hilfslinien bzw. Beschriftungen und den Ausprägungen der Variablen, welche auf die jeweilige Achse abgebildet wird. Gibt es weniger als zehn verschiedene Datenwerte pro Achse, werden für jeden Wert die Beschriftung und eine Hilfslinie dargestellt. Ansonsten ist die Anzahl auf genau zehn begrenzt. Tritt dieser Fall ein, müssen die anzuzeigenden Beschriftungen zu den in gleichen Abständen verteilten Hilfslinien berechnet werden (Quellcode 2).

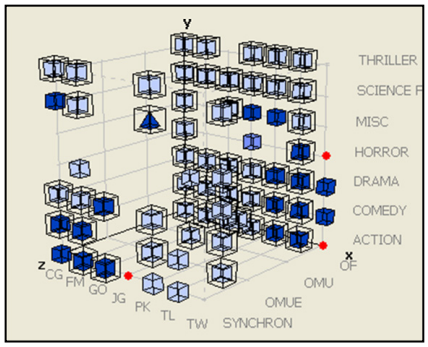


Abbildung 7.6: 3D-Koordinatensystem.

Bei Integer und Double Datenwerten ist die lineare Zuweisung ein triviales Problem. Bei Datumswerten wird die Zeitdifferenz in Sekunden ermittelt und ermöglicht so auch eine relativ einfache lineare Verteilung. Bei Strings werden die Werte sortiert, Duplikate entfernt und in einem Array angeordnet. Nun können die Indizes des Arrays wie Integerwerte behandelt und die jeweiligen Strings der linear ermittelten Indize den Beschriftungen zugeordnet werden. Die Beschriftung wird leicht versetzt zu den Hilfslinien dargestellt, um eine Überscheidung mit den Datenrepräsentanten zu vermeiden. Weiterhin werden auch zu lange Beschriftungstexte abgeschnitten und reelle Zahlen auf eine Dezimalstelle gerundet (Quellcode 3).

Zum Koordinatensystem wird in der unteren linken Ecke des 3D-Scatterplot-Fensters die aktuelle Achsenbelegung angezeigt. Befindet sich der Mauszeiger über einem Datenrepräsentanten, wird hinter der Achsenbelegung für x-, y- und z-Achse der jeweilige Wert des fokussierten Datenrepräsentanten angezeigt. Zusätzlich visualisieren drei rote Punkte an den Achsenbeschriftungen dessen Position direkt im Koordinatensystem.

Die Klasse Cube3D gibt dem Anwender durch Beschriftungen, Gitterlinien und Hervorhebungen wesentliche Hilfsmittel zur Hand um die dargestellten Daten besser zu lokalisieren, interpretieren und sich im Raum zu orientieren.

7.5.3 Datenrepräsentation

Wie schon im Kapitel 7.4 beschrieben, werden jedem Datensatz bzw. DataDoc 18 Definitionspunkte zur Visualisierung des jeweiligen Datenpunktes zugewiesen. Diese Punkte dienen je nach Selektionsstatus und Art der Datenwerte als Eckpunkte eines normalen oder eines geschachtelten Würfels oder einer Pyramide. Diese Datenrepräsentanten werden bei jeder Veränderung der Visualisierung direkt von der ScatterplotView3D-Klasse neu gezeichnet.

Aufgrund der gewollten Nichttransparenz der Objekte können sich diese überlappen und damit überdecken. Java hat nun die Eigenschaft, die Objekte nach der Reihenfolge ihrer Zeichnung auf der Projektionsfläche in der Tiefe bzw. z-Richtung anzuordnen.

Würden nun die Objekte nach immer der gleichen Reihenfolge gezeichnet werden, könnte es sein, dass nach einer Rotation ein Objekt ein anderes überdeckt, obwohl es eigentlich hinter dem anderen Objekt anzuzeigen wäre. Dieser Effekt vermindert erheblich den dreidimensionalen Eindruck und verwirrt den Anwender bei der Lokalisierung von Datenrepräsentanten.

Um diesen Effekt zu umgehen, wird ein modifizierter z-Buffer-Algorithmus [Catm74] verwendet. Die Idee ist, die Objekte nach ihrem transformierten z-Wert absteigend zu sortieren und dann nacheinander in dieser Reihenfolge zu zeichnen. So wird das hinterste zuerst und das vorderste Objekt zuletzt gezeichnet und die Überdeckung entspricht der tatsächlichen Position im Raum.

Dies wurde umgesetzt, indem die Indize der DataDocs des Datenarrays in der Klasse DataMapping in einem neuen Integer Array (z-Buffer Array) der gleichen Größe gespeichert werden. Diese Array wird bei jeder Transformation nach dem transformierten z-Wert des Mittelspunktes des Datenrepräsentanten mittels Quicksort [Hoar62] absteigend sortiert. Anschließend werden die DataDocs aufsteigend nach der Reihenfolge der Indize im z-Buffer Array gezeichnet und damit richtig in Raum positioniert (Quellcode 10).

Dieser modifizierte Algorithmus berücksichtigt nicht, dass Objekte, die von anderen Objekten zum Teil oder ganz überdeckt werden, eigentlich nicht mehr komplett oder gar nicht mehr gezeichnet werden müssten. Da die Objekte nicht Pixel für Pixel, sondern auf einer höheren Programmebene als Polygone und Linien gezeichnet werden, wäre der Aufwand um diese Überdeckungen zu berechnen größer als der Mehraufwand beim Zeichnungsprozess.

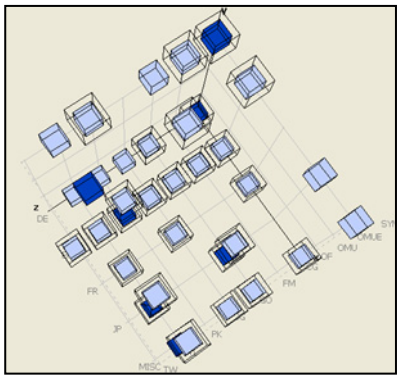


Abbildung 7.7: Z-Buffer & Überdeckungseffekt.

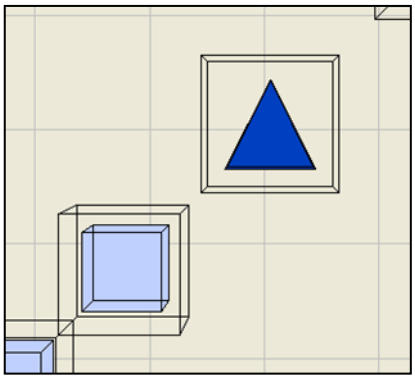


Abbildung 7.8: Würfel und Pyramide als Datenrepräsentanten.

7.6 Interaktion

7.6.1 Direkte Manipulation

Der VisMeB 3D-Scatterplot besitzt sehr ausgeprägte Möglichkeiten zur direkten Manipulation, welche grundsätzlich in zwei Gruppen aufgeteilt werden können. Zum einen kann das gesamte Koordinatensystem per Maus vergrößert oder verkleinert und über die horizontale und vertikale Achse gedreht werden. Zum anderen kann der Selektionsstatus der einzelnen Datenrepräsentanten direkt per Mausklick geändert und zusätzliche Informationen als Tooltip angezeigt werden.

Für die Drehung des Koordinatensystems wird die vertikale und horizontale Differenz des Mauszeigers bei einer Bewegung ermittelt und daraus die Winkel für die Rotation um die Achsen berechnet. Diese werden dann der Klasse ViewPoint übergeben, welche daraus die entsprechende Rotationsmatrizen erstellt und mit der globalen Transformationsmatrix multipliziert. Bei der anschließenden Projektion und Neuzeichnung der Definitionspunkte werden diese um die genannten Winkel gedreht angezeigt (Quellcode 9).

Da dieser Vorgang optimalerweise für jede pixelweite Veränderung der Mausposition wiederholt wird, entsteht beim Anwender der Eindruck einer stufenlosen dynamischen Drehung, welche er direkt beeinflussen kann.

Ähnlich verhält es sich auch mit der Vergrößerung und Verkleinerung des Koordinatensystems, wobei hier aber nur anhand der Mausbewegung ermittelt wird, ob es sich um eine positive oder negative vertikale Bewegung handelt. Dementsprechend wird der Klasse ViewPoint pro messbare Bewegung ein Zoomfaktor von 0.9 oder 1.1 übergeben.

Für die direkte Manipulation der Datenrepräsentanten muss das System wissen, an welcher Stelle des 3D-Scatterplot-Fensters sich ein Objekt befindet und ob andere Objekte in der Tiefe vor oder hinter diesem liegen, da Datenrepräsentanten nur für diese Pixel sensitiv sein dürfen, welche sie auch zur Darstellung verwenden. Hierfür wurde ein doppelter Integer Array *pixel[x][y]* angelegt, wobei *x* die horizontale und *y* die vertikale Auflösung des Fensters beschreibt und alle Felder des Arrays mit dem Wert -1 belegt. Dieser Array stellt also für jedes Pixel im Fenster ein Feld zur Verfügung (Quellcode 8).

Da nach der Reihenfolge des z-Buffers (siehe Kapitel 7.5.3) die Objekte gezeichnet werden, bietet es sich an, bei diesem Schritt auch für den Datenrepräsentant eine Begrenzungsbox zu errechnen. Nun wird für jeden Pixel, der von der Box im Fenster belegt wird, im Array *pixel[][]* jeweils der Index für das dem Datenrepräsentant zugrunde liegende DataDoc abgespeichert. Durch die Reihenfolge der Bearbeitung ist gewährleistet, dass immer das in der Tiefe vorderste Objekt auch im Array registriert ist, da die anderen Indexe überschrieben werden.

Bewegt sich nun der Mauszeiger im Fenster, kann für jeden Pixel im Array *pixel[][]* überprüft werden, ob sich ein Datenrepräsentant an dieser Stelle befindet. Wenn ja, wird das entsprechende DataDoc aus dem Datenarray mit dem Indexwert aus dem Pixelarray geladen. Ist es ein MultiDataPoint, wird mit der linken Maustaste die MultiDataPoint-View und mit der rechten ein Kontextmenü aktiviert. Ansonsten kann mit der Linken der Datenrepräsentant direkt selektiert werden und mit der Rechten erscheint ebenfalls ein angepasstes Kontextmenü.

Wenn sich kein Objekt an der Position des Mauszeigers befindet, kann mit der rechten Maustaste ein Kontextmenü aufgerufen werden, welches die Möglichkeit bietet, die Ansicht in den Ausgangszustand zurück zu versetzen. Weiterhin können auch alle Objekte im 3D-Scatterplot selektiert oder deselektiert werden.

Beim Berühren der Datenrepräsentanten werden sie fokussiert und damit farblich hervorgehoben. Über diese Fokussierung werden auch alle anderen aktiven Visualisierungen von VisMeB informiert und reagieren ebenfalls durch eine Hervorhebung. So können die Informationen der verschiedenen Visualisierungen verknüpft werden und der Anwender kann den fokussierten Datenpunkt in einen Gesamtbezug setzen.

7.6.2 Option- & Interaction-Panel

Das Option-Panel wird links am Rande des 3D-Scatterplot-Fensters angezeigt und stellt Drop-Down-Menüs zur Belegung der Achsen zur Verfügung. Dafür werden alle möglichen Variablen, die durch das Visual Configuration and Assignment Tool (siehe Kapitel 4.1) für den Scatterplot festgelegt wurden, aus den Konstanten der Klasse Assignment (siehe Kapitel 7.2) ausgelesen. Weiterhin kann die Größe der Datenrepräsentanten und die Darstellung verändert werden. Anstatt der gefüllten Würfel oder Pyramiden werden nur noch farbige Drahtgitter der gleichen Form visualisiert.

Das Interaction-Panel bietet, wie schon in Kapitel 5.2.6 beschrieben, Möglichkeiten zum Zooming und zur Rotation des Koordinatensystems über alle drei Achsen an. Diese Bewegung ist sehr rechenintensiv und wird daher in einem eigenen Thread[7] ausgeführt, um die Gesamtperformance nicht zu beeinflussen. So wird zum Beispiel das Koordinatensystem solange gedreht, wie der Anwender die Maus über dem entsprechenden Button gedrückt hält. Der Drehwinkel und Zoomfaktor sind festgelegte Konstanten, was eine kontinuierliche Rotation mit gleich bleibender Winkelgeschwindigkeit ermöglicht.

Weiterhin stellt das Interaction-Panel eine Möglichkeit zur Mehrfachselektion von Datenrepräsentanten zur Verfügung. Im Mark-Modus kann ein Rechteck über dem Koordinatensystem aufgezogen werden, wobei der Selektionsstatus aller Objekte, die sich in der Projektion innerhalb dieses Rechtecks befinden, verändert wird.

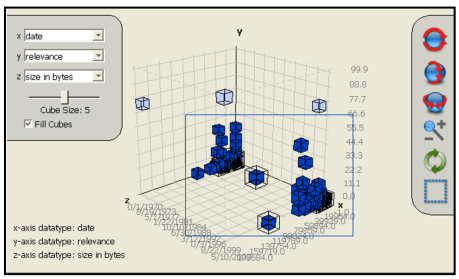


Abbildung 7.9: Mark-Modus.

[7] Java Thread, einzelner isolierter sequentieller Prozess innerhalb einer Anwendung.

8 Richtlinien zur Konzeption von 3D-Scatterplots

Aufgrund der Erfahrungen mit der Konzeption und Implementation des vorgestellten 3D-Scatterplots und einer im Vorfeld dieser Arbeit angefertigten Seminararbeit [Koen03], welche den Vergleich von 2D- und 3D-Scatterplots und ihre Vor- und Nachteile zum Thema hatte, werden hier Richtlinien zur Konzeption von 3D-Scatterplots vorgestellt. Es soll ein Leitfaden zur geeigneten Realisation gegeben werden, um dem Anwender eine sinnvolle, effiziente und intuitive Visualisierung zu bieten.

8.1 Vorbedingungen

Erst seit kurzem werden 3D-Scatterplots als zusätzliche Visualisierung in Spezial- und Standardprodukten in den verschiedensten Gebieten eingesetzt. Daher sind die Anwender meist noch skeptisch gegenüber dieser Art der Darstellung, besonders in Bezug auf Effektivität und Effizienz. Oft wird der 3D-Scatterplot auch noch als „Toy-Application“ empfunden, da die Dreidimensionalität und die Art der Interaktion bisher nur aus 3D-Spielen bekannt waren.

Dieser mangelnden Akzeptanz von Seiten des Anwenders muss eine sinnvolle Umsetzung als produktive Arbeits-Visualisierung gegenüberstehen. Der 3D-Scatterplot sollte zusätzlich zu herkömmlichen Visualisierungen, wie zum Beispiel Tabellen, eingesetzt werden und durch geeignete „brushing and linking“-Techniken verbunden und kombiniert werden. So fällt es dem Anwender leichter, sich mit dem 3D-Scatterplot vertraut zu machen und erhält zusätzliche Interaktionsmechanismen, die er zur Erfüllung seiner Aufgabe einsetzen kann.

Allerdings darf der dem 3D-Scatterplot fast immanente „Joy of Use“ nicht vernachlässigt, sondern muss sogar gefördert werden. Ein Anwender, der eine gewisse Freude bei der Bedienung empfindet wird sicherlich nicht unproduktiver. Es ist sogar anzunehmen, dass Anwender, welche diese Empfindungen bei der Arbeit erfahren, länger leistungsbereit sind, wenn nicht sogar eine Leistungssteigerung erleben. Dieser „Joy of Use“ kann durch eine ansprechende Darstellung und durch geeignete Interaktionsmechanismen verstärkt werden. Auf diese Aspekte wird in Kapitel 8.2 und 8.3 näher eingegangen.

Der 3D-Scatterplot ist für ein weites Spektrum an Datenarten beliebiger Dimension geeignet. Sicherlich der nahe liegendste Einsatz ist die Visualisierung von räumlichen Daten, wie zum Beispiel technische oder medizinische Daten aus dem Bereich der Scientific Visualization. Aber auch für andere Daten mit zumindest drei Dimensionen bietet der 3D-Scatterplot durch die intuitivere und natürlichere Visualisierung und der Möglichkeit der gleichzeitigen Darstellungen von drei Merkmalen einen Mehrwert gegenüber Visualisierungen, die nur zwei Dimensionen in Bezug setzen können.

Eine wesentliche Vorbedingung zum Einsatz einer Visualisierung wie dem 3D-Scatterplot, ist die Erfüllung der benötigten Hard- und Software-Voraussetzungen. Durch die räumliche Darstellung in Verbindung mit den Transformationsmöglichkeiten beansprucht der 3D-Scatterplot vor allem bei großen Datenmengen nicht zu unterschätzende Speicher- und Prozessorperformance. Daher muss schon bei der Konzeption des 3D-Scatterplot-Frameworks auf eine effiziente Modellierung geachtet werden.

In der Umsetzungsphase sind effiziente Algorithmen und die verwendete Programmiersprache sehr von Bedeutung. Softwareschnittstellen wie OpenGL[8] und DirectX[9] ermöglichen die Auslagerung von Berechnungen auf den Grafikprozessor und entlasten damit die CPU. Auch enthalten sie vorgefertigte Bibliotheken für Standardaufgaben wie Transformationen und Projektion. Diese Schnittstellen werden durch verschiedenste Programmiersprachen unterstützt.

Für Java wurde ein 3D-API[10] verabschiedet, welches die Verwendung von OpenGL-Befehlen auch mittels der Java-Syntax erlaubt. Dieses API war zur Zeit der Umsetzung des VisMeB 3D-Scatterplots noch nicht für alle Plattformen als stabile Version verfügbar. Daher wurden alle graphischen Methoden vom Projektteam implementiert, wobei so auch völlig freie Gestaltungsmöglichkeiten gegeben waren.

8.2 Visualisierung

Zur Verstärkung des dreidimensionalen Charakters des 3D-Scatterplot sollte die perspektivische Projektion verwendet werden, welche erheblich realistischer erscheint als die Parallelprojektion, da weiter entfernte Objekte verkleinert dargestellt werden. Weiterhin hilft eine Erweiterung der Visualisierung des Koordinatensystems zu einem Würfel, da so der Betrachter weitere Kanten und Knotenpunkte bekommt, die ihm helfen die Lage des Koordinatensystems im Raum leichter zu bestimmen.

Die im Raum parallel verlaufenden Hilfslinien an den Koordinatenachsen verstärken durch ihre Ausrichtung zum Projektionszentrum in der Projektion ebenso den dreidimensionalen Eindruck und unterstützen den Betrachter bei der Positionsbestimmung von Datenpunkten im Koordinatensystem. Die Beschriftungen zu den Hilfslinien sollten abgesetzt aber noch gut zuordenbar positioniert werden, um Überschneidungen der Beschriftungen mit den Datenpunkten zu vermeiden. Die Anzahl der Hilfslinien und ihrer Beschriftungen muss auf einen Maximalwert begrenzt werden, damit die Übersichtlichkeit der Visualisierung noch gewährleistet werden kann.

[8] OpenGL, http://www.opengl.org/

[9] DirectX, www.microsoft.com/directx/

[10] Java 3D-API, Application Programming Interface, Java Schnittstelle für OpenGL und DirectX

Die Repräsentanten der Datenwerte im Koordinatensystem sollten aus einfachen dreidimensionalen Objekten bestehen, welche klar identifiziert und bei verschiedenen Objekten deutlich unterschieden werden können. Für eine reale Darstellung und eine einfache Lokalisierung durch den Anwender müssen die Objekte sich auch naturgemäß überdecken, das heißt, Objekte, die hinter anderen positioniert sind, werden auch von diesen überdeckt (z-Buffer, siehe 7.5.3).

Eine dynamische Veränderung der Größe der Objekte ist vor allem bei sehr unterschiedlichen Datenmengen von Vorteil. Werden zum Beispiel große Datenmengen von räumlichen Gegenständen mit sehr hoher Auflösung dargestellt, ist es sinnvoll die Objektgröße sehr klein zu wählen, um der hohen Auflösung Rechnung zu tragen. Aus der Vielzahl vielleicht nur noch pixelgroßer Objekte kann dann beispielsweise im 3D-Scatterplot wieder der ursprüngliche dreidimensionale Gegenstand wahrgenommen werden.

Können die Datenobjekte verschiedene Status, wie Selektion oder Fokus, annehmen, müssen diese durch Farbe und / oder Form klar kommuniziert werden. Ihr Schwerpunkt bzw. Zentrum sollte genau auf der errechneten Position im Koordinatensystem liegen, wobei die Werte innerhalb des Koordinatensystems so normalisiert werden, dass keine Position außerhalb dessen belegt werden kann.

Um die Übersichtlichkeit des 3D-Scatterplots zu fördern, können MultiDataPoints (MDP, siehe Kapitel 2.2) eingesetzt werden. Dabei werden Datenobjekte mit identischen Merkmalsausprägungen und daher gleicher Position in der Visualisierung zu einem MDP zusammengefügt. So können gleiche Datensätze nur durch ein Objekt repräsentiert werden, welches dann auf Abruf weitere Informationen über die eingeschlossenen Datensätze bereithält. Die inhärenten Daten können dann mithilfe dieser interaktiven MDPs dynamisch exploriert werden. Diese Technik kann dadurch erweitert werden, dass auch Datenobjekte mit einer bestimmten Abweichung zu einem MDP zusammengefasst und somit Cluster gebildet werden.

8.3 Interaktion

Die Navigation im Raum ist für die meisten Anwender eine nicht triviale Problematik. Die völlige Freiheit sich in allen drei Dimensionen zu bewegen überfordert oft den Benutzer und ruft zumeist ein Gefühl der Verlorenheit hervor [SVMCL99]. Auch wird die Navigation im Raum nur unzulänglich von den herkömmlichen zweidimensionalen Eingabegeräten unterstützt und 3D-Eingabegeräte, wie zum Beispiel Spaceballs, besitzen derzeit noch einen geringen Verbreitungsgrad.

Um dem Anwender beim 3D-Scatterplot trotz der Dreidimensionalität eine einfache Navigation bzw. Interaktion zu ermöglichen, sollte der Betrachtungspunkt fixiert werden und nur das Koordinatensystem rotierbar und im Abstand veränderbar ausgelegt werden. Dabei sind feste Orientierungspunkte und Hilfslinien, wie auch schon im letzten Kapitel beschrieben, sehr wichtig für die Orientierung.

Die Beschränkung der Rotation per Maus auf die horizontale und vertikale Sichtachse hilft ebenso die Komplexität zu reduzieren. Dabei ist zu beachten, dass sich die Rotation in Winkelgeschwindigkeit und Drehrichtung intuitiv verhält. Der Anwender sollte das Gefühl haben, das Koordinatensystem bzw. den Würfel greifen und manipulieren zu können. Zusätzlich zur Interaktion per Maus sind Buttons für alle Interaktionsmöglichkeiten und vor allem für eine Zurücksetzung der Ansicht sehr hilfreich.

Der Anwender kann nur ungenau die Datenwerte eines Objektes anhand der Position im Koordinatensystem ermitteln. Daher werden Techniken benötigt, die eine genaue Identifikation ermöglichen. Eine Möglichkeit ist, die Position eines fokussierten Objektes mit Markierungen auf den Koordinatenachsen zu verdeutlichen. Zusätzlich sollten auf alle Fälle die reellen Werte für zumindest die Variablen ausgegeben werden, welche auf die Dimensionen abgebildet sind. Bei längerer Fokussierung eines Objektes können auch in einem kleinen dynamisch zuordenbar positionierten Pop-up-Fenster (vgl. Tooltip-Technik) alle wichtigen Daten zu dem fokussierten Objekt angezeigt werden.

Hilfreich ist auch eine Kopplung des 3D-Scatterplots mit anderen Visualisierungen mittels „brushing and linking“-Techniken, wobei fokussierte oder selektierte Datensätze auch in den anderen Visualisierungen hervorgehoben werden. So können zum Beispiel bei Dokumentdaten in einer gekoppelten Textansicht die fokussierten Dokumente im Volltext angezeigt werden.

Ebenfalls ist die Möglichkeit zur direkten Manipulation der Datenobjekte bei der Interaktion mit dem 3D-Scatterplot wichtig. Diese sollten per Mausklick oder per Kontextmenü selektiert oder wieder deselektiert werden können. Auch sollte zu jeder Zeit eine Zurücksetzung der Ansicht über ein Kontextmenü möglich sein. Bei großen Datenmengen ist auch eine Mehrfachselektion wünschenswert, um die Interaktion effizienter gestalten zu können.

9 Ausblick

In dieser Arbeit wurde die Konzeption und Implementation des VisMeB 3D-Scatterplots thematisiert, welcher zusätzlich zu einem 2D-Scatterplot im Projekt eingesetzt wird. Auf Grund der bisher noch spärlichen Evaluationen in Bezug auf 3D-Scatterplots wäre ein Vergleich von 2D- und 3D-Scatterplots sehr aufschlussreich.

In der derzeitigen Implementation ist der 3D-Scatterplot auf zweidimensionale Ein- und Ausgabegeräte ausgerichtet und emuliert den dreidimensionalen Raum und die damit einhergehenden Interaktionsmöglichkeiten. Bei einer Evaluation sollte noch eine weiterführende Umsetzung des 3D-Scatterplots in den Vergleich miteinbezogen werden. Diese sollte 3D-Eingabegeräte, wie Spaceballs oder 3D-Stifte, und die dreidimensionale Anzeige durch Shutter-Brillen oder ähnliches unterstützen. Somit würde der Betrachter von der mentalen Transformation zwischen den Dimensionalitäten befreit und könnte von einer noch zu verifizierenden besseren Auffassung der Informationen im Raum und einer intuitiveren Interaktion profitieren.

Weitere Vorschläge zur Verbesserung des bestehenden 3D-Scatterplots sind eher kosmetischer Art. So könnte der 3D-Eindruck durch geeignete Beleuchtungs- und Schattierungsmodelle verstärkt werden, wobei die Performance der Anwendung dabei nicht negativ beeinträchtigt werden sollte. Außerdem könnten Datenrepräsentanten durch ihre Farbsättigung ihre Wertigkeit verdeutlichen. Objekte mit geringer Sättigung würden somit eine gewisse Nähe zum Koordinatenursprung visualisieren. Der Einsatz von Anti-Aliasing-Techniken würde weiterhin eine klarere Wahrnehmung der dreidimensionalen Objekte ermöglichen. Dies wurde bisher schon in diesem Projekt versucht, scheiterte aber leider an der Performance der von Java bereitgestellten Algorithmen. Ein spezifisch modellierter Algorithmus könnte dabei die Effizienzprobleme beheben. Weiterer Effizienzgewinn bei starker Vergrößerung des Koordinatensystems wäre von Front- and Back-Clipping-Verfahren (siehe Kapitel 6.6) zu erwarten. Damit wäre auch ein virtueller „Durchflug“ durch den Scatterplot in Richtung des Koordinatenzentrums realisierbar, welcher bisher durch Maximal- und Minimalwerte eingeschränkt wird.

Die gesammelten Erkenntnisse aus Recherchen, Konzeption, Implementation und Anwendung wurden in Kapitel 8 zu Richtlinien zusammengefasst, an welchen man sich bei einer Neukonzeption eines 3D-Scatterplots orientieren kann. Idealerweise können die vorgestellten Merkmale für eine „gute“ Umsetzung als Qualitätskriterium bei einer Bewertung bestehender 3D-Scatterplots Anwendung finden.

10 Anhang

10.1 Anhang A: Code-Beispiele

Quellcode 1: Klasse DataDoc, Variablendefinition

```
public class DataDoc{

  private Document doc;
  protected DataDoc carrier3d;
  protected Vector MDP3d = new Vector();
  protected boolean partOfBuzzingBean = false;
  public double valueX3d, valueY3d, valueZ3d
  protected boolean visible3d = true;
  public Point3D[] point3d = new Point3D[18];
  private boolean partSelectedMDP = false;
  private boolean focusedMDP = false;

  public DataDoc(Document doc) {
    this.doc = doc;
    initPoint3D();
  }
}
```

Quellcode 2: Klasse Cube3D, Berechnung Anzahl Hilfslinien bei Integer-Werten

```
if(scatterplotView.getType(AXISNO) == INTEGER){
      String DiffStr = String.valueOf(diff);
      if (diff >= 0) {
        numOfTicks = Integer.valueOf(String.valueOf(DiffStr.charAt(0))).intValue();
      }
      else {
        int index = DiffStr.indexOf('.');
        double rounded = diff * (java.lang.Math.pow(10, (DiffStr.length() - 1) - index));
        DiffStr = String.valueOf(rounded);
        numOfTicks = Integer.valueOf(String.valueOf(DiffStr.charAt(0))).intValue();
      }

      if(numOfTicks < 1){
        numOfTicks = 1;
      }
      else if ((numOfTicks > 1) & (numOfTicks <= 5)){
        numOfTicks = numOfTicks * 2;
      }
      else {
        numOfTicks = 10;
      }
```

Quellcode 3: Klasse Cube3D, Bestimmung der Beschriftungen bei String-Werten

```
else if (scatterplotView.getType(AXISNO)==STRING){
    int maxLength = 30;
    switch (AXISNO){
     case X_AXIS:{
      String[] noDupSortX=scatterplotView.getNoDupSortX();
      if (noDupSortX.length < 10){
       int length = noDupSortX.length;
       for(int k=0; k < length; k++){
        if(noDupSortX[k].length() > maxLength){
         temp[k] = noDupSortX[k].substring(0,maxLength)+"...";
        }
        else{
         temp[k] = noDupSortX[k];
        }
       }
      }
      else{
       double divisor = ((double)noDupSortX.length-1)/9;
       for(int k=0; k < 10; k++){
        int i = new Double((double)k*divisor).intValue();
        if(noDupSortX[i].length() > maxLength){
         temp[k] = noDupSortX[i].substring(0,maxLength)+"...";
        }
        else{
         temp[k] = noDupSortX[i];
        }
       }
      }
      break;
     }
```

Quellcode 4: Klasse DataMapping, Zuweisung von String-Datenwerte aus dem VisMeB Datenmodell

```
if(plot3d.getType(AXISNO)==STRING){
    String[] noDupSort = null;
    if(AXISNO==X_AXIS) noDupSort = noDupSortX;
    if(AXISNO==Y_AXIS) noDupSort = noDupSortY;
    if(AXISNO==Z_AXIS) noDupSort = noDupSortZ;
    try{
     String str = doc.getDocument().getData(axisID).getString().toUpperCase();
     int length = noDupSort.length;
     for(int j = 0; j < length;j++)
     {
      if(str.compareTo((String)noDupSort[j]) == 0)
      {
      value=j;
      break;
     }
     }
    }catch(Exception e){
     value = 0;
    }
   }
```

Quellcode 5: Klasse Point3D

```
public class Point3D {

  double lx, ly, lz;
  double wx, wy, wz;  //
  int x2d, y2d

  public Point3D(double X, double Y, double Z) {
    this.lx = X;
    this.ly = Y;
    this.lz = Z;
  }

  public void setXYZ(double X, double Y, double Z){
    this.lx = X;
    this.ly = Y;
    this.lz = Z;
  }
}
```

Quellcode 6: Klasse ViewPoint, Berechnung der transformierten Koordinaten und Projektion

```
public void translatePoints(Point3D[] point){

    int length = point.length;
    for(int i=0;i<length;i++){
      point[i].wx =   point[i].lx*mm[0][0] + point[i].ly*mm[1][0] + point[i].lz*mm[2][0]
                      + mm[3][0];

      point[i].wy = -(point[i].lx*mm[0][1] + point[i].ly*mm[1][1] + point[i].lz*mm[2][1]
                     + mm[3][1]);

      point[i].wz = -(point[i].lx*mm[0][2] + point[i].ly*mm[1][2] + point[i].lz*mm[2][2]
                     + mm[3][2]);

      point[i].x2d = (int)(1000 * point[i].wx / (point[i].wz+distance) + xoffset);
      point[i].y2d = (int)(1000 * point[i].wy / (point[i].wz+distance) + yoffset);
    }
  }
```

Quellcode 7: Klasse ViewPoint, Rotation um die z-Achse und Zoom

```
public void transform(double ax, double ay, double az, double zoom){
   double cost,sint;
   resetMatrix(wm);
   if (ax!=0.0){
     cost = Math.cos(ax);
     sint = Math.sin(ax);
     mtemp[0][0]=1;   mtemp[0][1]=0;       mtemp[0][2]=0;      mtemp[0][3]=0;
     mtemp[1][0]=0;   mtemp[1][1]=cost;   mtemp[1][2]=sint;   mtemp[1][3]=0;
     mtemp[2][0]=0;   mtemp[2][1]=-sint;   mtemp[2][2]=cost;  mtemp[2][3]=0;
     mtemp[3][0]=0;   mtemp[3][1]=0;       mtemp[3][2]=0;       mtemp[3][3]=1;

     matrixMultiplication(mres,mtemp,wm);
     matrixCopy(wm,mres);
   }

…

   if (zoom!=1.0){
     if (((zoom>1) && (controlZoomFactor<3.0) || (zoom<1)
&&(controlZoomFactor>0.3))){
       controlZoomFactor= controlZoomFactor*zoom;
       mtemp[0][0]=zoom;   mtemp[0][1]=0;        mtemp[0][2]=0;        mtemp[0][3]=0;
       mtemp[1][0]=0;         mtemp[1][1]=zoom;  mtemp[1][2]=0;        mtemp[1][3]=0;
       mtemp[2][0]=0;         mtemp[2][1]=0;        mtemp[2][2]=zoom;  mtemp[2][3]=0;
       mtemp[3][0]=0;         mtemp[3][1]=0;        mtemp[3][2]=0;        mtemp[3][3]=1;
       matrixMultiplication(mres,mtemp,wm);
       matrixCopy(wm,mres);
     }
   }
   matrixMultiplication(mres,mm,wm);
   matrixCopy(mm,mres);
 }
```

Quellcode 8: Klasse ScatterplotView3D, Events für MouseClicked, MousePressed, MouseReleased

```
addMouseListener(new MouseAdapter(){
  public void mouseClicked(MouseEvent e) {
    if (isMDPViewON) {
      activateMDPView(false, null);
    } else {
      if (!markModus) {
        if (pixelXY[e.getX()][e.getY()] != -1) {
          DataDoc doc = dataMapp.getZBufferDoc(pixelXY[e.getX()][e.getY()]);
          if (doc.isPartOfBuzzingBean()) {
            if (e.getModifiers() == e.BUTTON1_MASK) {
              activateMDPView(true, doc);
            } else if (e.getModifiers() == e.BUTTON3_MASK) {
              popmenu.setDataDoc(doc);
              popmenu.setType(PopupMenu.MDP_MENU);
              popmenu.show(ScatterplotView3D.get(), e.getX(), e.getY());
            }
          } else {
            if (e.getModifiers() == e.BUTTON1_MASK) {
              doc.getDocument().setSelected(!doc.getDocument().isSelected());
              ViewNotifier.notifySelection(ScatterplotView3D.get(),
                                           doc.getDocument().getID());
            }
            if (e.getModifiers() == e.BUTTON3_MASK) {
              popmenu.setDataDoc(doc);
              popmenu.setType(PopupMenu.DOC_MENU);
              popmenu.show(ScatterplotView3D.get(), e.getX(), e.getY());
            }
          }
        } else {
          if (e.getModifiers() == e.BUTTON3_MASK) {
            popmenu.setType(PopupMenu.VIEW_MENU);
            popmenu.show(ScatterplotView3D.get(), e.getX(), e.getY());
          }
        }
      }
    }
  }
  public void mousePressed(MouseEvent e){
    setOldMousePos(e.getX(), e.getY());
    setIsMousePressed(true);
    markX = e.getX();
    markY = e.getY();
    markWidth = markHeight = 1;
  }
  public void mouseReleased(MouseEvent e){
    setIsMousePressed(false);
    outDocFocus();
    if (markModus) {
      computeMarkedDocs();
      markModus = false;
      setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
      repaintView();
    }
…
```

Quellcode 9: Klasse ScatterplotView3D, Events für MouseDragged, MouseMoved

```
public void mouseDragged(MouseEvent e){
  if (markModus) {
    if(e.getX()>markX && e.getY()>markY){
      markWidth = e.getX() - markX;
      markHeight = e.getY() - markY;
    }
  } else {
    if (!e.isControlDown()) {
      double xrot = (e.getY() - oldMouseY) * 2 * Math.PI / getSize().height;
      double yrot = (e.getX() - oldMouseX) * 2 * Math.PI / getSize().width;
      setRotation(xrot, yrot, 0);
    }
    if (e.isControlDown() && !isMDPViewON()) {
      if (e.getY() < oldMouseY) {
        setZoomfactor(1.1);
      } else if (e.getY() > oldMouseY) {
        setZoomfactor(0.9);
      }
    }
    oldMouseX = e.getX();
    oldMouseY = e.getY();
  }
  repaintView();
}

public void mouseMoved(MouseEvent e){
  try{
    if (pixelXY[e.getX()][e.getY()] == -1) {
      outDocFocus();
    } else {
      DataDoc doc = dataMapp.getZBufferDoc(pixelXY[e.getX()][e.getY()]);
      cube.isPaintingFocusMark(true);
      if (!isMousePressed()) {
        if (doc != lastMovedDoc) {
          if (lastMovedDoc != null) {
            lastMovedDoc.getDocument().setFocused(false);
            if (!lastMovedDoc.isPartOfBuzzingBean()) {
              ViewNotifier.notifyFocus(this, lastMovedDoc.getDocument().getID());
            }
          }
          if (doc.isPartOfBuzzingBean()) {
            setToolTipText("<html><font face=" + Preferences.FONT +
                    " size=2>" + (doc.getSizeMDP3D()) +
                    " Objects in MDP.<br>Right-click to get<br>
                    more information.</font></html>");
          } else {
            setToolTipText(doc.getDocument().getToolTip());
          }
          doc.getDocument().setFocused(true);
          cube.setFocusMarks(doc);
          lastMovedDoc = doc;
          repaintView();
          if (!doc.isPartOfBuzzingBean()) {
            ViewNotifier.notifyFocus(this, doc.getDocument().getID());
          }
```

Quellcode 10: Klasse ScatterplotView3D, Methode zur Zeichnung der Datenrepräsentanten

```
private void paintCubes(Graphics2D g2d){
   DataDoc doc;
   Document document;
   int size = dataMapp.getZBufferSize();
   for(int i=0; i < size; i++){
    doc = dataMapp.getZBufferDoc(i);
    document = doc.getDocument();
    if(doc.isVisible3D() && document.isAvailable()){
     if(isMDPViewON){
      if(doc.equals(mdpView.getDoc()))
       g2d.setColor(colorMDP);
      else
       g2d.setColor(colorNotMDP);
     }else{
      doc.calculateState();
      if(document.isFocused() || doc.isMDPFocused()){
       g2d.setColor(Preferences.FOCUSED_SP3D_COLOR);
      }else{
       if(document.isSelected() || doc.isMDPPartSelected()){
        g2d.setColor(Preferences.SELECTED_SP3D_COLOR);
       }else{
        g2d.setColor(Preferences.DEFAULT_SP3D_COLOR);
       }
      }
     }
     vpoint.translatePoints(doc.point3d);
     setPixelArray(doc,i);
     if(isCubeFilled){
      if(!doc.isMDPPartSelected()){
       for(int j=0;j<4;j++){
        rect1x[j] = doc.point3d[ar1[j]].x2d;      rect1y[j] = doc.point3d[ar1[j]].y2d;
        rect2x[j] = doc.point3d[ar2[j]].x2d;      rect2y[j] = doc.point3d[ar2[j]].y2d;
        rect3x[j] = doc.point3d[ar3[j]].x2d;      rect3y[j] = doc.point3d[ar3[j]].y2d;
        rect4x[j] = doc.point3d[ar4[j]].x2d;      rect4y[j] = doc.point3d[ar4[j]].y2d;
        rect5x[j] = doc.point3d[ar5[j]].x2d;      rect5y[j] = doc.point3d[ar5[j]].y2d;
        rect6x[j] = doc.point3d[ar6[j]].x2d;      rect6y[j] = doc.point3d[ar6[j]].y2d;
       }
       g2d.fillPolygon(rect1x,rect1y,4);
       g2d.fillPolygon(rect2x,rect2y,4);
       g2d.fillPolygon(rect3x,rect3y,4);
       g2d.fillPolygon(rect4x,rect4y,4);
       g2d.fillPolygon(rect5x,rect5y,4);
       g2d.fillPolygon(rect6x,rect6y,4);
       g2d.setColor(colorCubeBorder);
…
```

10.2 Anhang B: CD-ROM

Dieser Arbeit ist eine CD-ROM mit folgendem Inhalt beigefügt:

- Ausführbare Version von VisMeB (Stand 23.09.2003)
- Bachelor-Arbeit im PDF-Format
- Quellen
- Abbildungen

11 Referenzen

11.1 Quellenverzeichnis

[AWS92] C. Ahlberg, C. Williamson, B. Shneiderman, „Dynamic queries for information exploration: An implementation and evaluation“, Proceedings ACM CHI'92 Conference, 1992.

[AS94] C. Ahlberg, B. Shneiderman, „The Alphaslider: A Compact and Rapid Selector“, Proceedings ACM CHI'94: Human Factors in Comp. Systems, 1994.

[BC87] R. A. Becker, W. S. Cleveland: “Brushing Scatterplots.” Technometrics, 1987.

[Catm74] E. Catmull: “A Subdivision Algorithm for Computer Display of Curved Surfaces”, PhD Thesis, Dept of Computer Science, University of Utah, Salt Lake City, Utah, U.S.A., 1974.

[CLS00] J. Cugini, S. Laskowski, M. Sebrechts, "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), San Jose, CA, 23-28 Januar 2000.

[DCM03] Dublin Core Metadata Initiative**,** http://dublincore.org, online am 23.09.2003.

[EAGOW00] L. Elson, M. Allen, J. Goldsmith, M. Orton, W. Weibel, „An example of a Network-Based Approach to Data Access, Visualization, Interactive Analysis, and Distribution“, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 2000.

[FS95] K. Fishkin, M.C. Stone: “Enhanced Dynamic Queries via Movable Filters”, Proceedings of the CHI’95 Conference, New York, ACM Press, 1995.

[FDFH90] J. Foley, A. van Dam, S. Feiner, J. Hughes, Computer graphics, principles and practice, 2. Auflage, Addison-Wesly, 1990.

[Furn86] G. W. Furnas, “Generalized Fisheye Views”, Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems, ACM Press, 1986.

[Hoar62] C. A. R. Hoare: "Quicksort." The Computer Journal, vol.5(1), pp.10-15, 1962.

[INSY03] INSYDER, Internet Système de Recherche, Europäische Kommission, ESPRIT, Projekt Nr. 29232, http://www.insyder.com, online am 23.09.2003.

[INVI03] INVISIP, Information Visualization for Site Planning, Europäische Kommission, Projekt Nr. IST-2000-29640, http://www.invisip.de, online am 23.09.2003.

[KMRE02] P. Klein, F. Müller, H. Reiterer, M. Eibl, „Visual Information Retrieval with the SuperTable + Scatterplot", Proceedings of the 6th International Conference on Information Visualisation (IV 02), IEEE Computer Society, 2002.

[KRML03] P. Klein, H. Reiterer, F. Müller, T. Limbach, „Metadata Visualization with VisMeB", IV03, 7th International Conference on Information Visualization, London, 2003.

[Koen03] W. A. König: "2D- versus 3D-Scatterplots: Vorteile & Nachteile", Seminararbeit Human-Computer Interaktion, Universität Konstanz, 2003.

[Lieb03] P. Liebrenz: „Visualisierung von MultiDataPoints in einem 3D-Scatterplot; Konzeption & Implementierung innerhalb eines Metadatenbrowsers", Bachelor-Arbeit, Universität Konstanz, 2003.

[Niel98] Jakob Nielsen: „2D is Better Than 3D", Alertbox, http://www.useit.com/alertbox/981115.html, online am 23.09.2003, 1998.

[NIRV03] NIRVE, http://zing.ncsl.nist.gov/~cugini/uicd/nirve-home.html, online am 23.09.2003.

[PRIS03] NIST PRISE Search Engine: http://www.itl.nist.gov/iaui/894.02/works/papers/zp2/main.html, online am 23.09.2003.

[RMMH00] H. Reiterer, G. Mußler, T. Mann, S. Handschuh, „Insyder - An Information Assistant for Business Intelligence", Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Developement in Information Retrieval, Athens, 2000.

[Sahl02] G. Sahling, „Interactive 3D Scatterplots – From High Dimensional Data to Insight", Master Thesis, Wien, 2003. http://www.vrvis.at/vis/resources/DA-NSahling/, online am 23.09.2003.

[SVMCL99] M. Sebrechts, J. Vasilakis, M. Miller, J. Cugini and S. Laskowski: "Visualization of Search Results: A Comparative Evaluation of Text, 2D, and 3D Interfaces", Proceedings of 22nd ACM SIGIR conference on Research and development in information retrieval, 1999.

[Shne03] Ben Shneiderman: „3D or Not 3D: When and Why Does it Work?", Web3D Symposium, Phoenix, 2003.

[Spot03] Spotfire Inc., http://www.spotfire.com, online am 23.09.2003.

[TPS00] E. Tanin, C. Plaisant, B. Shneiderman, "Browsing large online data with query previews", Proceedings of the Symposium on New Paradigms in Information Visualization and Manipulation (NPIVM), ACM Press, 2000.

[VisM03] VisMeB, A Visual Metadata Browser for Visual Data and Text Mining, AG Mensch-Computer Interaktion, Universität Konstanz, http://www.inf.uni-konstanz.de/iw_is/Forschung/vismeb/, online am 23.09.2003.

[ViSt03] ViSta, the Visual Statistics System, http://forrest.psych.unc.edu/research/, online am 23.09.2003.

[VRVi03] VRVis, Zentrum für Virtual Reality und Visualisierung Forschungs-GmbH, http://www.vrvis.at, online am 23.09.2003.

[WebW03] WebWinds, http://www.openchannelsoftware.com/projects/WebWinds, online am 23.09.2003.

11.2 Abbildungsverzeichnis

11.3 Abkürzungen

Abkürzung	Bedeutung
API	Application Programming Interface
CAD	Computer Aided Design
CW	Mittelpunkt des Projektionsfensters (Window)
DB	Datenbank
INSYDER	Internet Système de Recherche
INVISIP	Information Visualization for Site Planning
IVEE	Information Visualization and Exploration Environment
MDP	MultiDataPoint
MVC	Model-View-Controller-Konzept
NIRVE	NIST Information Retrieval Visualization Engine
NIST	National Institute of Standards and Technology
PRP	Projection Reference Point (Projektionszentrum)
VisMeB	Visueller Metadaten Browser
ViSta	Visual Statistics Sytem
VPN	View-Plane Normal
VRC	Viewing-Reference Coordinate
VRP	View-Reference Point
VRVis	Zentrum für Virtual Reality und Visualisierung
VUP	View Up Vectors
WWW	World Wide Web