# *A Generic Framework for the Development of 3D Information Visualization Applications*

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades
eines Doktor-Ingenieurs (Dr.-Ing.)
genehmigte

**Dissertation**

von
Diplom Informatiker

**Martin Leissler**

aus Neu-Isenburg

Referent: Prof. Dr. E. J. Neuhold
Koreferent: Prof. Dr. Veljko Milutinovic
Tag der Einreichung: 29. Januar 2004
Tag der mündlichen Prüfung: 23. März 2004

Darmstadt 2004
D 17
Darmstädter Dissertation

# Preface

This thesis was written during my time as scientific associate at the Fraunhofer Institute for Integrated Publication and Information Systems (IPSI) (formerly GMD IPSI). It is based on the results and the experience I have gained during the development of several information visualization systems including LyberWorld, cyber:gallery, Virgilio and others.

I want to thank my colleagues Heinz-Dieter Böcker, Ulrich Thiel, Peter Fankhauser, and Prof. Klaus Mätzel who have significantly triggered my interest for visual interfaces to information systems and specifically for the cognitively efficient design of such interfaces, and who have advised and encouraged me during my scientific work at Fraunhofer IPSI.

I also want to thank all colleagues and students of the DELITE and OASYS divisions of Fraunhofer IPSI for the numerous and interesting discussions and especially Andreas Müller for the tireless software development support during his master thesis.

My fellow Ph.D. colleagues Gerald Jäschke, Xiangchuan Wang, Fathi Husein, Subramaniam Krishnasamy, Thomas Klement, and Thomas Risse for the great collaboration in our joint scientific project work at IPSI and their motivating role model with respect to finishing the thesis work in a timely manner.

My very personal thank goes to my mentor Matthias Hemmje for the tireless support and advisory at formal problems especially in the last phase of completing the thesis, for all the organizational support making the completion possible, and for thoroughly reading and commenting, contributing significantly to the quality of this work.

Furthermore, I want to thank Professor Erich Neuhold who assisted and influenced the development of the work at hand through many valuable suggestions and advices as well as Professor Veljko Milutinovic (Fellow of the IEEE) for taking over the position of 2nd referee.

Additional thank goes to Prof. Nicolas Georganas, Prof. Ben Shneiderman, Dr. Frank Nack and Basawaraj Patil for interesting and highly inspiring discussions.

Darmstadt, January 2004

Martin Leissler

# Abstract

The relatively new scientific discipline of information visualization is a promising approach to the problem of the ever-increasing information flood we are facing in today's world of tightly connected, widely spanned networks and large data collections. The advantage of visualizing data stems from its ability to shift load from a human's cognitive system to his more efficient perceptual system. The main trend in this scientific area clearly goes in the direction of 3D visualization techniques for information, since 2D visual interfaces and techniques have been thoroughly researched in the past.

One of the challenges in this field lies in the efficient development and execution of applications supporting 3D information visualization fostering rapid development of experimental systems and their analysis and evaluation against real user requirements. In the area of 2D user interfaces and visualization techniques the WIMP (Windows, Icons, Menus, Pointer) paradigm, Model/View/Controller (MVC) paradigm, and declarative approaches using XML broadly dominate the application development and execution process. However, the structure of 3D information visualization applications is inherently more complex than in the 2D case. Trying to directly apply such models, techniques and paradigms to the development of 3D information visualization applications is at least questionable and a relatively unexplored field.

Therefore, the goal of this thesis is to develop a model that is applicable to the development and execution of 3D information visualization applications in a cost effective and computationally efficient way, covering all aspects of information visualization, including data modeling, transformations, and presentation modeling in an integrated and declarative way.

This is achieved by means of constructing a process model, developing an architecture, and an experimental implementation of a framework for the generic development and execution of 3D information visualization applications, including an evaluation of the computational efficiency of the execution.

# Content

# 1 Introduction

## 1.1 Historical background

In the beginning the world was three-dimensional and so was our spatial perception. No one would ever think of trying to change this natural factor. Wrong! Rather it seems a natural habit since the beginning of mankind that we try to reduce concepts to their basic principles just in order to be able to better comprehend their reality, sometimes without foreseeing the impact of such simplified models. Already in the Stone Age people started to draw real life scenes such as hunting of animals on flat stone cave walls using naturally prepared colors and therefore reducing the dimensionality of perceived reality in order to simplify the communication of content. Higher levels of abstraction were invented such as written symbols for numbers or letters, forming words and sentences and therefore reflecting our natural communication medium of speech. Again for reasons of technical simplicity flat two-dimensional media were used such as paper and ink. Today we still use these techniques and even if we talk of art the first thing that naturally comes to our mind are two-dimensional flat paintings. The painting and drawing techniques have become more advanced over time allowing an artist to create the illusion of three-dimensionality using perspective, lighting, and color. With the technology of photography it is even possible to chemically recreate stunningly realistic pictures of real life scenes. The medium, however, has always stayed flat.

With the revolutionary discovery that rapidly blending a sequence of single flat still pictures allows the illusion of scenes moving over time the age of cinematic media arose. A new dimension of time flow was added to the medium but still one spatial dimension was missing. Later the invention of the cathode ray tube (CRT) further revolutionized this development and enabled the technology to become smaller, more portable, and available to a growing number of people.

With the end of the industrial revolution came the first computers and the necessity for a full-grown interaction cycle between man and machine. There was a need to design input devices to enable humans to communicate with the given machine (and its underlying architectural model) and output devices to enable machines to communicate their output to human beings. The existing technologies of CRT and typewriter keyboard seemed perfectly suitable (or should we say immediately available?) for this purpose and were naturally adapted. Therefore, interaction was the next dimension introduced within a new medium with the third spatial dimension still being left out. Today's by far most popular input device is the pointer mouse. It originates from the first graphical user interfaces utilizing the desktop metaphor introduced to the mass market on the legendary Apple Macintosh and has become widely spread with the worldwide domination of the Microsoft Windows operating system.

Over the years huge mainframe computers were replaced by desktop PC's and portable laptops with rapidly increasing multimedia features. In fact, today the multimedia features and performance are a main sales factor for computer hardware. Nowadays a relatively low cost and simple desktop machine is capable of displaying multicolored photo-realistic still images and high quality movies, playing multi channel sound, and even replacing entire video labs with off-the-shelf desktop video editing software.

The next revolutionary technology was the Internet with its globally networked infrastructure. Born in 1982 from the ARPA-net (see e.g., http://www.dei.isep.ipp.pt/docs/arpa.html) the Internet today links together approximately 5 Mio. computers involving an estimated number of 40-50 Mio. people from about 200 countries across the world. Even cautious estimations from analysts extrapolate a number of more than 190 Mio. Internet users worldwide by the end of the year 2003. In other words, the Internet is virtually exploding. The most popular services provided within the Internet are electronic mail and the World Wide Web (WWW). The WWW allows a server to provide content to users as so called “Web Pages” which contain a mixture of different information media such as text, bitmap images, video and sound. People use Web browser software installed on their machines to browse such pages. Web pages are often classified as hypertext medium, which means that they can be linked to each other by so called hyperlinks. By clicking on an element of a web page the browser displays a new web page defined by the destination address of the hyperlink. As can already be derived by their name, web pages use the real world metaphor of the well-known flat book or magazine pages to provide their content. Again this is convenient because of the flat medium these pages are displayed on (CRT) and the flat input device (mouse) to navigate through a system of linked web pages.

By now the greater scheme should be obvious: The chain of human achievements from cave paintings to modern computers irresistibly led to the omission of one spatial dimension in the presentation and interaction with these media.

Throughout recent history science fiction authors have in many cases been the foreseers of groundbreaking technological and scientific advances (although their time predictions are often rather based on fiction than on science). In 1984 William Gibson’s novel “Neuromancer” [Gibson,1984] coined a new sub-genre in science fiction literature called Cyberpunk. Many follow up authors took up this example and covered different aspects of the new genre. However, there seems to be one common concept across all these fictional novels and stories, which Gibson has named "Cyberspace" (and which uses different names but similar concepts throughout the Cyberpunk literature genre). Gibson described Cyberspace with his own characteristic words:

*"Cyberspace. A consensual hallucination experienced daily by billions of legitimate operators, in every nation, by children being taught mathematical concepts… A graphic representation of data abstracted from the banks of every computer in the human system"*

Cyberspace (or today more often called "The Matrix") is the vision of a world-spanning network of computers, which is represented by a three-dimensional metaphor and is simultaneously experienced by all connected users through neural interfaces, while representing users themselves as "residual self-images" or "the mental projection of your digital self." (today often referred to as "avatars").

Thinking about Gibson's vision, it seems intriguing to be able to move in a virtual environment, which partially resembles our own well-known reality for the sake of familiarity. But since Cyberspace is artificially generated, it is even more fascinating to enable its participants to go well beyond the limitations of our every-day physical constraints.

As a result of the high popularity of Gibson's visionary ideas and its followers, many scientific and industrial research groups work hard today on the task of bringing back the "lost" third dimension to human-computer interfaces experimenting with Virtual Reality, novel visualization techniques, graphical user interfaces, immersive devices, and 3D special effects for media. However, since the human computer interfaces have been flat for so many years now, there is considerably more experience with 2D user interfaces than with the use of 3D techniques in applications. This manifests in the various, significantly different approaches to 3D interfaces in experimental applications, which emphasizes the still immature nature of 3D.

Many groups working in this area believe today that the key success factor for introducing the third dimension to mass acceptable applications is a smooth introduction of such techniques. 3D interfaces cannot be introduced suddenly by simply replacing well-known 2D user interface metaphors but rather smoothly by carefully mixing some well tested and well-thought 3D elements into existing 2D applications, in order to allow them to prove their applicability in real working situations.

Although the long term end goal of such research might be to one day have humans interact with computers as envisioned by Gibson or by Gene Roddenberry's famous HoloDeck (which will unlikely be achieved without advanced research results in the field of neural interfaces), it is likely that the traditional 2D screens and input devices won't disappear for a long time. We are still at the early beginnings of a new era of interaction with machines, which will solely be designed around the human being right from the start, throwing aboard the limitations from the days when technology did not permit the use of all perceptive dimensions.

## 1.2 The situation today

The coming of the "Information Age", unbroken rapid growth of the Internet, and the development of one generation after another of increasingly powerful computers have resulted in more and more data being produced and stored in shorter and shorter periods of time. In companies the data produced in operative processes taking place in various divisions is stored in electronic databases. Some of this information constitutes a critical resource for corporate decision-making and is drawn upon company management.

The mere possession of these, in some cases extremely large amounts of data does not necessarily lead to an expansion of personal knowledge, particularly if there is a lack of methods and mechanisms with which to analyze the excess supply of information and to extract the more relevant data. This fact becomes clear, for instance, when an Internet search engine produces a flood of data consisting of several hundred thousand unsorted and unevaluated documents. A further indication of this problem is provided by the mere fact that the NASA is still evaluating the data gathered by its very first space probes.

Once the desired data has been found, presenting it in such a way that users will be able to understand the inherent knowledge is a factor of key importance. If the results of a series of measurements are presented merely as a column of numbers it takes a great deal of time and mental effort to understand the information contained in them. If, on the other hand, the results are presented as the graph of an underlying mathematical function, then the information can be understood quickly and intuitively. This comes from the capability of shifting load from the user's cognitive system to the perceptual system. Information needs to be visualized in an information space in order to be successfully retrieved by users. This visualization can either be carried out by the users in their own mind, in which case it is essentially the users' conceptualization of that information, or it could be accomplished by an external system, in which case the visualization is generated on some kind of display device.

Since the early 1990s, inspired by rapid advances being made in the fields of graphics hardware and algorithms, numerous systems have been built and described, which aim at analyzing and structuring the information visualization process with a view to making it possible to visualize various types of data. The focus of interest has been on three-dimensional visualization for the most part, given that the two-dimensional user interface has been used successfully for decades now and has been extensively researched (cf. retrieval interfaces: InfoCrystal [Spoerri,1993]; Fisheye/Lens Views: [Sarkar,1992]; scalable GUIs: PAD++[Bederson,1996]; etc.). Several of these systems are described in more detail within the chapters 2 and 3.

While some of the theoretical concepts from preceding work have found immediate use in later work and have served as a basis for further development of information visualization systems and prototypes, only very few of them have found their way to the end-user in the form of commercially available software. One of the reasons is a mere technical one. Since most of these systems were only designed as proof-of-concept prototypes, the low quality and missing modularity of the code didn't allow for easy reusability and modification. Furthermore, the shift from a UNIX world to a more Windows oriented world would have required a considerable amount of re-design and re-implementation.

# 1.3 Problem description and motivation

As a result of today's ever increasing flood of data in the on-line as well as off-line worlds, the challenge of finding relevant key information needed for solving problems becomes ever harder. Assuming a continuous growth of available data, an increasing number of scientists forecast an information-dead-end for the near future resulting in a near impossibility to find any relevant information in a timely manner.

One of the scientific areas that promises a way out of this dead-end is the relatively new area of information visualization. The question, however, is why the mere visualization of data collections would help to cope with the growing data-flood problem. In other words: Why should we visualize data?

To answer this question let us consider an arbitrary set of "information objects" (the exact definition of this term will be postponed to a later chapter), like, e.g., a set of database records, a collection of Web pages, or a group of related e-mail messages. Such collections are potentially valuable to humans because they can be used to support the solving of problems. Their value lies buried either *within individual items* or in *the relationship between the items* or in both.

Harvesting this value from individual items is the classical *information retrieval* problem: find and examine one ore more items with a certain property. To harvest value from relationships among items is, in contrast, an *information analysis* problem. Humans perform cognitive analysis by means of comparison of various properties of items with one-another and comparison of item properties with a-priori knowledge. However, with increasing complexity of information our ability to perform mental comparisons decreases rapidly.

Visualization techniques can effectively dampen this effect by using a kind of "graphical-memory" to mentally represent relationships making it easier to compare and analyze them. From a cognitive psychology point-of-view the advantage of visualizing data stems from its ability to shift load from a human's cognitive system to his perceptual system. This assumption is based on the conviction that the human mind can be understood as a supreme pattern recognition machine and that visual perception is the most efficient interface with the highest bandwidth to that machine. It comes now naturally that the two steps of leveraging this powerful machine for problem solving are deriving relationships among information in a semantically meaningful way and then to represent these relationships in a cognitively efficient graphical form.

It follows that visualization of information can be a powerful mechanism for information analysis enabling human beings to make fast and effective comparisons. Therefore, it seems generally advisable to design visualization techniques in such a way that they support easy comparison of the significant information.

A apparent trend today in the scientific area of information visualization as well as in some industrial branches is the use of 3D visualization of information, mainly because 2D techniques have been thoroughly researched and three-dimensional display of information seems to be a novel and promising approach, although various problems and challenges remain untackled and several questions are open (see also Section 2.10). Modern video games are among the most popular applications to heavily utilize real time 3D visualization techniques.

One of these challenges of this field lies in the efficient development and execution of applications supporting 3D information visualization techniques. Taking a closer look at how 2D information visualization applications are developed should allow us to learn something about the 3D case, since the development of 2D visualization applications has gone a long way already and lots of experience has been gained in this field.

The main design and development paradigm used today for 2D visualization applications (see section 2.9.1 for several examples) and 2D traditional WIMP-style (Windows, Icons, Menus and Pointer) graphical user interfaces is the Model-View-Controller (MVC) paradigm. MVC splits up an application into the concepts of a Model, a View, and a Controller. User inputs, model of the external world, and visual presentation to the user are handled by controller, model, and view objects, respectively. The controller interprets all user inputs and maps these actions to commands that are applied to the model and/or view to perform the appropriate modifications. The model represents and manages the data elements, can be queried for its state, and responds to state-changing instructions. The view manages a (typically rectangular) display area and presents data to the user through text and graphics. In the terms of object oriented software design there is a well-known and frequently used design pattern for MVC-model based applications [Buschmann,1997]. One of the greatest benefits of the MVC model is the easy exchangeability of either of the model, view, or controller components leading to an extremely flexible and efficiently modifiable architecture.

Most recent and increasingly popular approaches to 2D GUI and visualization design and development employ a declarative approach using mainly the XML and XSLT description languages along with XML based schema definition languages, such as XDR or XSD (for more details see sections 2.1 and 2.4). These approaches, however, can easily be looked at as declarative MVC approaches in the sense that the model and the views are described through a description language (e.g. using XML/XSD) and the controller (mapping back and forth between model and view) is described by a transformation language (as e.g. XSLT).

In the field of 2D GUI and visualization design and development the MVC paradigm has clearly proven its applicability, cost effectiveness, and computational performance and scalability, as a large number of applications today employ this model.

However, taking a closer look at the structure of such applications it becomes clear that 2D visualization techniques and 2D GUI's have relatively few parameters encapsulated within the view onto which data from the model could be mapped, dramatically limiting the overall complexity of such applications. In other words, the complexity of the "presentation space" (the exact definition of this term is given at a later time) is relatively low. This becomes evident when observing the visual elements of traditional windows-style GUIs (such as buttons or scrollbars) or at the most popular 2D data visualization techniques presented in section 2.10.

Observing, in contrast, the situation in the field of design and development of 3D information visualization applications and GUI's, the picture is quite different. Since 3D information visualization is a relatively young and emerging discipline, there are no such standard GUI elements and interaction paradigms like WIMP to offer reuseable and rapidly recognizable interaction facilities to a user like in the 2D world. Therefore, 3D information visualization applications still have a relatively low acceptance among the broad mass of users, which in turn leads to quite limited budgets for the development of experimental 3D applications among the scientific community and within company research labs. In consequence most 3D information visualization applications have merely prototypical status and are implemented by scientists in a proprietary way not sharing a standard design and usage paradigm (see section 2.9.2 for several examples).

One of the challenges in this field is therefore to find a commonly usable and acceptable design and development paradigm for 3D visualization applications, making development, testing, and modification according to feedback from evaluations and customer requirements cost effective and to keep the execution model acceptably performant, while at the same time trying to leverage the experience gained from the MVC paradigm in the 2D application development field. To successfully tackle this challenge could initiate a major breakthrough in the acceptance of 3D information visualization applications by establishing consistency in development workflows and interaction paradigms.

The specific question that arises is now if the MVC (or a similar) model can be applied to 3D information visualization in a similarly successful manner as it has been applied to the development of 2D GUIs and applications. What turns this (obviously looking) approach into a problem is the fact that on the one hand the semantic distance between the data representation and the presentation is potentially high (abstract data) and at the same time the structure of the presentation space can be potentially (by several orders of magnitude) more complex than in the case of 2D visualization, especially when interaction and navigation with 3D elements is involved.

The well established MVC model has in fact been applied to 3D applications in several cases like, e.g., 3D video games and scientific visualization. However, in these cases the semantic distance between data (geospatial coordinates, movement vectors, 3D objects etc.) and the presentation of the data is relatively low, meaning that only few transformations have to be applied to the data in order to turn them into a visual appearance. In other cases either the data or its structure is fairly simple or the structure of the presentation space is only comparably complex to the techniques used in 2D visualization. In fact, very few systems exist that support the design and development of 3D information visualization applications in its full potential complexity involving complex data, complex presentation, and a mapping beteeen these two (which would naturally be also potentially complex) in a similar fashion as MVC works for the 2D applications. And we daresay that there are - at the time being - no applications, which support these aspects in a generic way employing a declarative approach with all its benefits, as in the case of using XML and XSLT.

Therefore, the goal of this thesis is to find out if a model similar or equal to MVC is applicable to the development and execution of 3D information visualization applications in a cost effective and computationally efficient way, covering all aspects of information visualization, including data modeling, transformations, and presentation modeling in an integrated and declarative way.

In the remainder of this thesis it will become apparent that this is tackled by means of constructing a process model, an architecture, and an experimental implementation of a framework for the development and execution of 3D information visualization applications, including an evaluation of the efficiency of the execution.

# 1.4 Explanation of concepts

A number of methods designed and implemented in the past are described in Chapters 2 and 3 for the purpose of formulating the objective of the present thesis. Before that, however, there is a need to explain the meaning of a number of basic concepts, as they are indispensable for understanding the concepts in question.

## 1.4.1 Data and information

In ordinary language the terms "data" and "information" are often used to mean the same thing. In the context of this thesis the meaning of these two terms needs to be defined more specifically, since their distinction is of key importance in the field of computer science, which sees itself as the science of systematic information processing.

In computer science a single item of data is an addressable unit that can be stored electronically. There are specific data types such as whole numbers, floating decimal numbers, or character strings. Basic data types can be stored in abstract data structures such as sets, lists, or trees or they can be combined to form higher, user-defined data categories. The content of a single item of data is, taken by itself, empty of meaning, for instance the number 42 or the alphanumeric character string "X#25:13b", and must first be interpreted by a program or a user to make sense of it. An item of data becomes an item of information when it is related to other items of data and viewed in a specific context (more about information theory in Chapter 4).

One possible definition of information can be found in [Engesser,1993] (we will, however, use a different one later in this thesis), in which information is defined as an extremely complex concept consisting at least of the following three elements:

- a syntactical element that describes the admissible structure of the components the information is made up of,
- a semantic element that indicates the meaning of the information,
- a pragmatic element that makes it possible to derive the purpose of the information and the action hoped for.

The syntactical element is always visible. The semantic element is usually an indirect result of processing the information while the pragmatic element normally remains hidden. In order to guarantee the syntactic correctness of information when it is selected, processed, and stored we make use of a "schema". A schema is a data structure definition formulated in a data description language. Its function can be compared to the grammar of a natural language, which governs the structure of valid sentences formed in that language. There are various data models for the storage of data, including the hierarchical data model, the network data model, and the relational data model. This issue is also referred to in more detail in Chapter 2 in the discussion of XML.

## 1.4.2 Information Visualization

Visualization in its original sense means the visible representation of a fact or circumstance. Since human beings perceive the largest part of their environment visually they are able to take in and process optically encoded information quite well shifting the main processing load from mind to the perceptual system. Even though recent studies in the psychology of perception have shown that people are best able to take in and store information when the largest possible number of senses is involved [Craik,1972][Bransford,1979] and [Huitt,2003], the sense of sight plays a special role in the transmission of information. While it may take a large number of sentences and a long time to explain the content of a painting to someone in a conversation, the same person can take in the same information content in a split second by looking at the picture just one time. The fact that complex and abstract interrelationships can be depicted understandably by means of sketches and illustrations underscores the special status of visual perception. Since the first cave paintings human beings have communicated with each other by means of visually depicted messages and visualization has continued to be a very important factor throughout history, right on up to its role in today's information technologies.

Numerous definitions of the term "information visualization" are found in the literature written on the subject. The following are a few examples:

*"The focus of Information Visualization is on effective and user-validated ways of presenting and interacting with data and creating an internal representation of the information in the mind of the user." [Jern,1996]*

*"The use of computer-supported, interactive, visual representations of abstract data to amplify cognition." [Card,1999,2]*

*„Information visualization uses computer graphics and interactive animation to stimulate recognition of patterns and structure in information. It does so by exploiting the human perceptual system in ways similar to Scientific Visualization, which allows scientists to perceive patterns in large data collections. Information visualization works on the structure of information inherent in large information spaces.“ [Crossley,1997]*

*"The overall goal of a visualization is to see patterns and understand any underlying relationships in the data." [Brinkschulte,1996]*

In [Däßler,1999] it is stated, that like in every new research area there is no exact definition of the term information visualization. Three more quotes regarding the objectives and goals of information visualization are supposed to describe the objective of information visualization in more detail:

*Information Visualization is the use of computer-supported interactive visual representation of abstract data to amplify cognition (XEROX PARC, User Interface Research Group).*

*Information Visualization combines aspects of scientific visualization, human computer interfaces, data mining, imaging and graphics. (Gershon und Eick, 1997).*

*Information visualization is the process of binding information that is not inherently spatial into a visual medium which the user can view and perceive. (Medialab, University of Georgia).*

# 1 INTRODUCTION

According to [Jern,1996] the term "information visualization" implies interactivity, simple and intuitive usability, multidimensionality, flexibility, extendibility, and collaboration. No restrictions are to be imposed on the definition at this point so we can summarize information visualization as the set of all techniques used for the presentation of abstract information through modes of visualization that can be, but do not necessarily have to be, generated by a computer. The use of a computer makes it possible to add the element of interactivity and, as such, the development of electronic information systems with a user interface consisting of a virtual graphics information space in which the user can navigate. In the course of this thesis information visualization will be viewed in the context of 3D visualization. According to [Jäschke,2000] the virtual-reality-inspired approach constituted by freely navigable information spaces offers the following further advantages compared with traditional graphical user interfaces:

- *The world is information and we can navigate in it:* If we are able to manipulate information the same way we do objects and ideas in real life we will be able to spend more time understanding things than looking for ways of managing them. Well-designed user interfaces can help us to see things in larger contexts and to extract information from complex interrelationships.
- *The world is experience and we can learn from it:* Virtual worlds are, for the most part, models of the reality around us. It is possible today to create virtual environments, in which we are able to act, communicate, and learn in the real world.
- *Human society is based on communication:* The fact that people live and work in groups and communicate with each other are normal characteristics of human society. Collaborative, virtual environments provide a framework for new forms of cooperation and joint action on a global scale.

At the beginning of the computer era, when mainframes that filled entire rooms were programmed with punch cards, visualization was not yet a significant factor and costly computing time was used exclusively for the solution of numeric problems. In the 1970s and 1980s, when vector displays and then raster displays came onto the market, the visualization factor began to move more and more into the foreground. Initially two-dimensional user interfaces were used. Then, in the mid-1980s, the first powerful workstations became available, enabling individual scientists to present their research results with three-dimensional graphics. Initially it was primarily physicists and chemists who took advantage of this opportunity. At the time, graphic representation was limited to the rendering (ray tracing) of individual images, so that interaction was not an option. This changed in the early 1990s, when progress in hardware and software development as well as a growing consumer interest in 3D games brought advanced visualization technologies onto the PC market at affordable prices, thus making them available for commercial applications as well. Powerful PCs, equipped with accelerated 3D graphics cards, made it possible to generate interactive 3D graphics in real time and, since the mid-1990s, to address entirely new aspects of information visualization.

For the next chapters of this thesis we will assume our quite intuitive and informal definition of information visualization presented in this section.

A more formal model for information visualization based on the set theory is then presented later in chapter 4. However, in order to already introduce some of the formal terms involved in the information visualization process, we will state at this point that:

The information visualization process can be described as a function, mapping a given set of information objects (often called information space) including their structure to as set of presentation objects (often called presentation space). This short definition including the intuitive no-tion what the terms ‘information object’, ‘structure’, and ‘presentation object’ represent is enough for now to be able to understand the information visualization techniques, frameworks, and systems described in chapters 2 and 3.

### 1.4.3 Metaphors

In [Paradiso,1997] the term "metaphor" is defined as follows: „A metaphor is a rhetorical figure, the purpose of which is to make one thing understandable by referring to another thing.“ A metaphor makes it possible to understand a new and possibly complex set of concepts by means of another set of concepts already known. Metaphors have been used successfully in designing user interfaces for computer systems. For instance, the "desktop" metaphor, which relates the computer screen to the surface of a desk, is one of the best known and most widely used.

The aim pursued in connection with the systematic application of metaphors was to make it possible even for inexperienced users to carry out operations with complex computer systems on the basis of easily and intuitively understandable interface functions. Taking the desktop metaphor as an example, users can erase a file by dragging an icon representing the file in question and dropping it on another icon representing a waste-paper basket. This metaphor consists of using a source concept, i.e. the act of throwing waste paper away in a waste-paper basket, to express a target concept, i.e. erasing a file on a hard disk. Metaphors can thus be related to an action, as in the previous example, and the target concept can express abstract data. Thus, by way of example, the size of a sphere may represent the number of employees a company has and the number of projects the company is carrying out can be expressed by the color of the sphere (on a scale from green to red). A large green sphere would mean for the company represented that it still has production capacities free and can take on further orders, while a small red sphere would mean that the company's production capacities are currently being fully utilized. In this way it is possible to compare the production capacity utilization levels of several companies at a glance instead of having to go to the trouble of comparing evaluation criteria individually. We will encounter this specific metaphor further in the context of the „corpogram“ sample application.

From a more formal view (which will lead to a formal model later in this thesis) an information visualization metaphor is the graphical representation of the presentation space combined with the information how the data is mapped to the visual elements. For example, in the well-known bar chart metaphor the information objects may be the cell contents of an Microsoft Excel spreadsheet and the presentation space consists of the colored bars, the coordinate axes, and the bar legend.

## 1.5 Contents overview

Within Chapter 1 first the terms "data modeling", "information visualization", and "metaphors" are defined. Then the problem of modeling, designing, and implementing a generic information visualization framework, which is able to express mappings from arbitrary data sets to presentation objects is described and the object of the thesis formulated abstractly.

In the first part of the 2nd chapter basic research and technologies are presented that were used in the research carried out in the context of this thesis, e.g. XML, DTDs, schemata (XDR), VRML, X3D, DOM, XML transformations (XSLTs), Visual Basic, and ASP. Furthermore, existing technical extensions for the support of implementing 3D visualization are presented, including

VRML Server Side Includes, SQL Runtime Node, Trigger Server, and the VRMLet technology, which all have been developed over the past years.

The second part of chapter 2 introduces a commonly used taxonomy for information visualization and describes several well-known 2D and 3D information visualization techniques. It concludes with a critical view on 3D information visualization.

Chapter 3 continues with a discussion of the related work on 3D information visualization published in the past, including well known milestone systems of information visualization history as well as recent work, including some work on which the author of this thesis has previously worked on.

The chapter concludes with an analysis of the techniques and systems presented earlier, and takes a closer look at their deficits. After that, some general requirements are formulated for a generic 3D information visualization system.

Chapter 4 presents model of the general information visualization process and abstract design of a system able to perform such a process. From this further requirements for generic information visualization are derived and supplemented and then the necessary components are specified.

In chapter 5 a further analysis of implementation-specific requirements is given, followed by a description of the technical architecture. Finally, key details for implementation of the prototype resulting from the design description are presented on the basis of components programmed for this purpose.

In chapter 6 the selected approach and the implementation are subjected to critical evaluation based on a number of exemplary application scenarios. This is followed by a detailed performance and a scalability analysis with the goal to find and delineate weaknesses.

The thesis concludes in chapter 7 with a summary of what has been achieved thus far and outlines prospects for future research work. Particular attention is focused on the possibilities of converting from VRML to X3D and from XDR to XML schemata.

# 2 State of the art

This chapter presents basic technologies, some advanced technologies and well-known information visualization techniques, an understanding of which is presupposed in the remainder of this thesis. The detail with which these technologies are presented varies in accordance with their importance for the thesis. Readers who are already acquainted with one or more of the technologies described here can skip over the sections in question or the entire chapter.

## 2.1 XML, DTDs, and schemata

First a description of the concepts behind "mark-up languages" and "meta-languages" is provided. A look is then taken at the historical development and the precursors of XML. Finally, DTDs and schemata are presented. These technologies are important prerequisites for understanding the design and implementation of a generic information visualization framework.

### 2.1.1 Markup languages, SGML, and XML

XML is the abbreviation for „extensible markup language“ and stands for a platform-independent data-description language. The term „markup language“ stems originally from the printing industry. Authors would mark their manuscripts with handwritten instructions that told typesetters things like what font to use or gave other production-related parameters. A set of instructions like this, used to define differences from the rest of the written text in some form, e.g. color, can be conceived of as a language in and of its own right with specific rules of syntax and grammar. The concept of marking a text with additional instructions has remained in the case of electronic documents. Instructions of this kind are specifically defined in a set of "tags". The latter determine how computers display text on screens and in print. Thus, by way of example, headings may be displayed in a larger font or individual words may be displayed in bold type or in italics. A markup language generally serves two different purposes. Either it determines the format and graphic layout of a document or it defines the structure and meaning of sections of text marked by tags.

XML is based on the concepts of SGML, which in turn developed out of the first modern markup language, GML („Generalized Markup Language“). The latter was developed in 1969 by IBM as a relational language for the structured administration of any data and documents. It is a metalanguage, i.e. it can be used to describe other languages as well as their grammars and vocabularies. In 1986 the *International Standards Organization* (ISO) made SGML the international standard for information management. This standard was intended to provide a means of creating platform-independent and application-independent documents that retain their formatting, directory paths, and references to other documents. By means of a mechanism similar to a grammar users can determine the structure of their documents with self-defined tags. SGML is a powerful but also a complicated markup language that was and still is used extensively by the U.S. government, major corporations, and publishers of technical information. The high costs involved in using this language, due to its complexity, has effectively prevented its use from spreading to include smaller companies and private individuals.

### 2.1.2 HTML and XML 1.0

HTML, the „Hypertext Markup Language“, is one of the best known applications of SGML. It is a web-content-description language which determines how a document is displayed. It does not make any statement about what kind of document is involved. With the help of separation symbols, referred to as „delimiters“, a fixed set of tags is established that can be used to model the appearance of a web page. The following figure shows an example of a simple HTML document and how it is displayed within an HTML capable browser.

```
<HTML>
    <HEAD>
        <TITLE> Sample Page </TITLE>
    </HEAD>
    <BODY>
        This is an example of <FONT size="5"><B>
        markup. </B></FONT>
    </BODY>
</HTML>
```

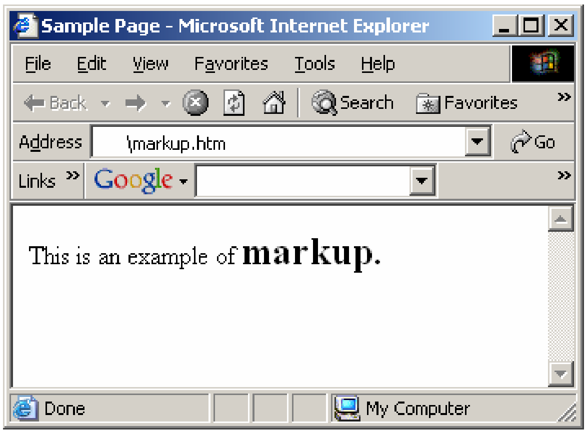



**Figure 1** Example of markup using HTML

Although the primary use of HTML is to format documents for presentation, it also determines the structure and sequence in which elements occur. Every valid HTML document must contain a `<BODY></BODY>` section, in which the other tags are embedded. However, no new user elements such as `<SALUTATION>`Hello.`</SALUTATION>` can be introduced.

In contrast to HTML, XML is not just an SGML application but rather a subset that can be used to describe new languages. While SGML specifications are up to 150 pages long, those for XML are only 35 pages long. This makes XML a metalanguage which is less complex than SGML and, as such, much easier to learn and apply. In 1996 the *World Wide Web Consortium* (W3C) began to design XML, which was intended to combine the flexibility and power of SGML with the popularity and widespread use of HTML. Due to its many optional features SGML is so complex that it is difficult to write generic parsers[1], whereas XML parsers are much easier to develop. XML also makes use of existing Internet protocols and software, simplifying the processing and transmission of data. Since XML is a subset of SGML, the backward compatibility with existing SGML oriented systems is guaranteed. The following ten design objectives led in February 1998 to a W3C recommendation with regard to specifications for XML 1.0 [W3C,2000]:

- *XML shall be straightforwardly usable over the Internet:* This does not mean the use of XML is to be limited to the Internet, but rather that the complexity of SGML is to be reduced. This simplification facilitates parser design and the use of XML on the Internet or on corporate intranets.
- *XML shall support a wide variety of applications:* XML is intended to be used both on the web as well as in traditional client/server environments. Since special consideration was not given to any particular technology in designing the language, XML can be used with any programming language.
- *XML shall be compatible with SGML:* All features of XML also exist in SGML so that SGML parsers can continue to be used.
- *It shall be easy to write programs that process XML documents:* The development of parsers is to be made as easy as possible so that XML will rapidly become widely used and accepted.
- *The number of optional features in XML is to be kept to the absolute minimum, ideally zero:* Optional features that are rarely used result in the development of specialized parsers. If XML provides no options then every parser will be able to read all XML documents.
- *XML documents should be human-legible and reasonably clear:* Since XML is text-based and follows strict but simple formatting rules, it is easy for people to understand the meaning of a document. XML is intended to describe the structure of the content and not the type of presentation of a document.
- *The XML design should be prepared quickly:* XML is to be developed as quickly as possible so that no single commercial dealer will be able to dominate the market with rapid proprietary developments.
- *The design of XML shall be formal and concise:* XML specifications use the „Extended Backus-Naur Form“ (EBNF). This is a standardized form for the description of programming languages and keeps the specifications concise and unambiguous.
- *XML documents shall be easy to create:* The intention is to make it possible to create XML documents with a text editor. A number of XML editors have become available. As a result of the simple structure, applications can generate XML in runtime situations if needed.
- *Terseness in XML markup is of minimal importance:* Ambiguities are not permitted in XML. SGML and HTML allow for abbreviations such as the dropping of closing tags, making it more difficult to read the document in question. The requirement that opened tags must be closed rules out ambiguities.

1 A parser is a program that assesses the syntactic correctness of a character string based on a grammar.

### 2.1.3 Data modeling, well-formedness, and validity

XML documents can be self-describing if tag names (elements) are carefully chosen. For instance it is obvious that `<CustomerFirstName>Walther</CustomerFirstName>` describes a customer's first name and not his or her address. An XML tag can have attributes as in `<Customer FirstName='Walther'></Customer>`, which obviously represents the same data item. The question arises for XML users as to whether data should be modeled as content of elements or by using attributes. There are various ways of answering this question. There is no one single and generally valid answer. Analogous to object-oriented modeling, objects and features of these objects can be identified in the matter to be described. The objects are then noted as elements of XML, while the features constitute attributes of elements. There are the extremes of exclusively element-based modeling, on the one hand, and the description of data by means of a number of empty elements with a series of attributes, on the other, as well as any number of variations in between, as the following example shows:

```
<Car>
  <Manufacturer>BMW</Manufacturer>
  <Model>M5</Model>
  <Color>Silver</Color>
  <Doors>5</Doors>
<Car>
```

```
<Car Manufacturer='BMW' Model='M5' Color='Silver' Doors='5'/>
```

From the point of view of performance there is no difference between the various modeling paradigms for the parser, as we will see later on in our discussion of the „Document Object Model“ (DOM). When using attributes, a list of permitted values can be defined such as „yes“, „no“, and a default value which is automatically added by the parser. What speaks for the use of elements is the fact that they can contain strings of other elements of any level of complexity for which a certain sequence may be defined. In this case the XML document requires more storage space, since tags have to be closed in accordance with specifications. On the other hand, the use of elements makes it possible to refine the scheme at a later point in time. By contrast, the content of an attribute, once defined, is fixed.

```
<Person>
  <Name></Name>
  <Address></Address>
</Person>
```

Refinement

```
<Person>
  <FirstName></FirstName>
  <LastName></LastName>
  <Address>
    <Street></Street>
    <HouseNumber></HouseNumber>
    <TelNumber></TelNumber>
  </Address>
</Person>
```

```
<Person Name=''
 Address=''/>
```

No possible refinement

## 2.1 XML, DTDS, AND SCHEMATA

There are two fundamental criteria an XML document needs to satisfy: well-formedness and validity. Well-formedness requires the document to correspond to certain basic structural principles common to all XML documents. According to [Martin,2000] the fundamental rules for a well formed documents are the following:

- As in the case of SGML and HTML the "less than" and "more than" signs (<...>) are the standard delimiters.
- Every opening tag must have a corresponding closing tag. The only exception to this rule is an empty tag which, by definition, has no content. In this case the opening and closing tag can be merged into one: <EmptyTag/>
- In contrast to HTML, attributes must be enclosed in quotation marks: <FONT Size='3'/>
- Elements must be properly related. Each XML document may contain one and only one root element and all other elements are children of this element.
- Element names are case sensitive. <Tag>, <tag>, and <TAG> are interpreted as three different tags by a parser.

Valid XML documents must obey specific rules established in a so-called „Document Type Definition“ (DTD) or a schema. DTDs and schemata define what an element may contain and the structural design of the document. This is similar to a database schema with the difference that elements in XML may contain other elements. DTDs have a syntax of their own to describe the structure of the documents, while schemata are themselves noted in XML. When a parser validates an XML document against a DTD or a schema it checks to see that every individual element is in compliance with the specified rules.

### 2.1.4 DTDs and schemata

XML makes it possible to define a vocabulary of tags of one's own choosing that describes specific problems or situations. The question arises as to how it can be guaranteed that not every user of XML will have to develop his or her own specific vocabulary for dealing with the same problem and how it can be ensured that the user of an existing vocabulary will adhere to the prescribed structure and the available tag names. So that an existing vocabulary can be used by others there must be a standard option for formalizing the selected tag names as well as the related rules of syntax. DTDs provide a mechanism of this kind as part of the XML 1.0 specifications. A DTD uses a formal grammar to define the structure and the possible values of an XML document. An XML document may refer to a related DTD, to which a parser can then refer back to when validating the elements and structure of the document. DTDs may be defined either as being „inline“ (part of an XML document) or external (a separate file). In essence a DTD consists of the declarations of elements, attributes, and rules which determine the sequence and frequency in which the elements may occur in a valid document. An opportunity to define macros as well as to integrate external files such as photos is provided by so-called "entity declarations". However, the latter will not be a further subject of discussion here.

The basic structure of an element declaration is as follows:

```
<!ELEMENT ElementName Rule >
```

There is a choice of rule options `ANY`, `EMPTY`, `#PCDATA,` and mixed declarations. `ANY` means the element may contain no elements at all or any other elements, to the extent that they have been defined in the DTD. The `EMPTY` rule says, on the other hand, that it may not contain any data, something that only makes sense in combination with attribute declarations, while #PCDATA stands for „parsed character data“ and means that the declared element may contain only text and no other parsable elements. The following example shows the application of mixed declarations, in which elements are separated by the „|“ sign indicating a choice of alternatives or by a comma indicating occurrence in a specific sequence. The different rules can be related with one another an any way desired by using parentheses.

```
<!ELEMENT Parent (Child | #PCDATA) >
<!ELEMENT Child (ChildA, ChildB) >
<!ELEMENT ChildA (#PCDATA) >
<!ELEMENT ChildB (#PCDATA) >
```

In addition to symbols that define the sequence of elements there are also those that determine the number of occurrences of elements (occurrence symbols). If a symbol of this kind does not follow the name of an element in the definition of a rule then it must occur exactly one time in the specified place in the document instance. The element is followed by a question mark (Element?), the element may either occur not at all or exactly one time only. Element* stands for any number, including zero, while Element+ indicates any number, but at least one occurrence. Accordingly, the known structure of an e-mail can be formalized in the following way:

```
<!ELEMENT EMAIL (To+, From, CC*, Subject?, Body?) >
```

In HTML almost all directives that determine the appearance of the text inside the tags are indicated in the form of attributes. The syntax for attribute definitions inside a DTD looks like the following:

```
<!ATTLIST ElementName AttributeName AttributeType Defaults >
```

The attribute type can be selected from a number of fixed-definition types such as CDATA, ENTITY, ID, IDREF, or Enumerated. ENTITY attributes are used to refer to external files. ID and IDREF are used to define links inside the document. Enumerations restrict the selection of possible values for the attribute, while in the case of a CDATA attribute any string can be given as a value. #IMPLIED can be given as the default value, making the indication of a value for this attribute in the document instance optional. #REQUIRED instructs the parser to view the indication of a value as a necessary condition. Finally, a default value can be defined which the parser automatically inserts into the XML document instance if no value is indicated there for the attribute. When attributes are used it needs to be taken into account that in the case of a multiple occurrence of the same attribute for the same element only the first occurrence of the attribute is valid. The sequence in which the attributes occur is not important. The following are two examples of attribute declarations:

```
<!ATTLIST Address country CDATA ’US’ >
<!ATTLIST Employee gender (male | female) #REQUIRED >
```

Although DTDs have been used successfully for many years in connection with SGML there are nonetheless a number of serious limitations:

- *DTDs do not have a syntax of their own:* One of the most obvious disadvantages of DTDs is the fact that an XML parser has to use a second parser technology to read and to validate DTDs, since the latter are written in a language different from XML.
- *A DTD is a self-contained entity:* This makes it impossible to combine DTDs that describe the same problem.
- *DTDs do not support inheritance:* Object-oriented design makes it possible to continue to use and expand one or more existing DTDs instead of having to develop completely new ones.
- *DTDs do not contain datatype definitions:* Since all element and attribute content is made up of character strings a program has no way of knowing what datatypes it is dealing with. Either information on the datatypes is an integral part of the program logic or it must be modeled by the document itself.

With a view to addressing these shortcomings of DTDs the W3C group has, since 1999, accepted ideas and proposals for new methods of describing the structure and content as well as the semantics of XML documents. Several proposals for such schema languages addressing the aforementioned problems have been submitted to the W3C (including XML Schema, XML Data, Document Content Description (DCD) [DCD,1998], XML Data-Reduced (XDR) [XDR,1998]). Although Microsoft was the first to have a fully available implementation of its XDR schema language as early as in 2000, XML-Schema is an official W3C recommendation since May 2001.

One of the most important concepts is that of the so-called „namespaces“ which make it possible to mix elements from different information models in one document. The intention is to be able to choose a prefix for each of the information models without having to change the designations or meanings of the tags. In this way it is possible to use `<Ornithology:wing>`, `<Architecture:wing>` as well as `<Aviation:wing>` in the same document even though they are from entirely different domains.

During the implementation work for this thesis the XDR Schema Language proposed by Microsoft was the only one for which an implementation was available and was therefore used. XDR is based on XML data specifications as well as on DCD, has a slightly modified grammar, and concentrates on describing the syntax of documents. Mechanisms for object-oriented design and inheritance have not been implemented. A more detailed description of XDR is given in Section 2.5.

# 2.2 Modeling languages for 3D content

## 2.2.1 VRML

VRML is the acronym for „Virtual Reality Modeling Language“ and is a description format for the description of interactive 3D objects and worlds. Based on the Silicon Graphics „OpenInventor“ file format, VRML 1.0 Standard was published in May 1995 as a simple, text-based, and platform-independent scene-description language, intended to make possible the description of three-dimensional web pages analogous to HTML. It soon became evident that VRML 1.0 still had deficits with regard to interactivity features, object behavior, and animation. This ultimately led in several steps to ISO VRML 97 Standard (VRML 2.0), which eliminated the indicated deficits and made it possible to integrate multimedia data consisting of texts, images, sounds, and video by means of URLs. In addition, VRML 2.0 has an extension module which makes it possible to define new and reusable objects, as well as the description of an interface between VRML worlds and external applications – the *External Authoring Interface* (EAI).

According to [Leissler,1997] the development of the VRML 2.0 specifications aimed at giving the format the following performance profile:

- *Authorability:* The ability to develop programs, with the help of which entire VRML applications can be created, modified, and maintained, as well as to import data from other formats that are widely used in industry.
- *Completeness:* The claim to supply all the information needed for successful implementation of the standard and to provide a full set of capabilities for broad acceptance by industry.
- *Combinability:* The ability to use elements of VRML as needed as well as to combine them with one another and, as such, to create the prerequisite for their renewed use.
- *Extendibility:* The ability to add new elements to the VRML elements explicitly defined in the standard.
- *Implementability:* The foresight to design the standard so sensibly that it can be implemented on a large number of computer platforms.
- *Multiuser potential:* The openness needed so as not to exclude the possibility of later implementations of multiuser environments.
- *Orthogonality:* Independence to use elements of VRML as needed or at least to provide precise descriptions of their dependencies.
- *Performance:* The claim to design elements of VRML with the aim of achieving good interactive performance independent of the computer platform.
- *Scalability:* The ability to combine elements of VRML to form dynamic 3D worlds of any size (in theory at least).
- *Good design:* The claim to give every element of VRML a well defined interface, to make its use independent of specific conditions, and to rule out side effects.

3D objects and worlds are described in VRML using a hierarchical scene graph. VRML 2.0 offers more than 50 different node types (e.g. geometric primitives, sensor nodes, group nodes), a sufficiently large number to be able to model complex scenes. If needed, further node types can be added, using a prototyping mechanism. For their part the nodes consist of type-standardized data fields that determine the specific functionality of the node. For a detailed description of the different VRML concepts (e.g. event handling, route mechanism, sensor nodes, interpolator nodes, animation, prototyping, scripting) see [Carey,1997], [Roehl,1997] and [Ames,1997].

Browsers that support VRML 2.0 have in the meantime become available for a large number of computer platforms, both as plugins for web browsers as well as in component form for embedding in one's own applications. Most VRML viewers offer a Java implementation of EAI. Particularly noteworthy here is the *Cortona VRML Client* from *Parallel Graphics* [ParallelGraphics,2001] for Windows and Apple Macintosh platforms. It offers a large number of proprietary extensions such as the display of Nurbs, textures with environment mapping, and a drag-anddrop mechanism from the embedding web page into the displayed 3D scene. In addition, the *Cortona* browser is a COM component that can be easily embedded into COM enabled deveopment environments (e.g., *Visual Basic* projects) and offers its own language-independent API („*VRML Automation Object Model for Cortona ActiveX Control*“), with which displayed scenes can be manipulated in a way similar to an extended EAI, e.g. directly from a web page using *JavaScript*. With the alternative use of *OpenGL* or *DirectX* and related access to 3D accelerated graphics hardware *Cortona* offers excellent performance on a Windows platform.

In the meantime a number of proprietary 3D plugins have been developed for use on the Internet which have their own file formats and APIs. The decision to use VRML for the implementation part of this thesis is based primarily on the fact that it is the only real standard for 3D content description. This will ensure the further development and the availability of viewers and tools for the creation of content as well as their widespread use. Widely used professional software for 3D modeling (e.g. *3D-Studio MAX*) has import and export filters for VRML and can be used to create VRML content.

### 2.2.2 X3D

Recent efforts have been aimed at embedding the time-tested and widely used VRML standard in the XML description language. This development is known under the name X3D and is being promoted and coordinated by the *Web3D Consortium* [Web3D,1997]. The two best-known specification proposals for X3D have come from the companies *Blaxxun* and *Shout3D*. They are largely identical. Viewers already exist for both proposals. They are implemented in the form of Java applets and differ only in the syntax and functionality of their API, the „Scene Authoring Interface“ (SAI). X3D will be a strict subset of the VRML specifications, based on its concepts, and in written in XML notation. The core of X3D will provide a very small set of 3D scene elements and a powerful API for programming and expansion. An expansion module for the language is called „Profiles“.

The following objectives are given for *Blaxxun's* „Core X3D Specification“ [Blaxxun,2001].X3D is to:

- define a set of requirements so small that a minimal viewer can be transferred to the client together with the X3D content.
- be compatible with the VRML97 standard so that existing authoring tools and 3D content can continue to be used.
- be based on specifications that can be implemented by a large number of companies and individuals without a great deal of time, effort, and expense.
- contain only such requirements as will make implementation possible on a wide variety of platforms.
- offer sufficient functionality so that a minimal implementation will be able to satisfy the requirements most needed by producers of 3D content.
- offer flexibility and extendibility.
- constitute a basis for the development of extension profiles (e.g. to support VRML97 and MPEG4).

There is no need to discuss X3D concepts here, since they correspond to those of VRML. However, they are more restricted in number. By way of example, *scripting* and *prototyping* mechanisms for the generation of new nodes were not included. On the other hand, this functionality can be attained in X3D with the help of SAI (more powerful than EAI) and profile extensions. The number of nodes was reduced to the most important 3D primitives such as the `IndexedFaceSet`, with which more complex objects such as *boxes*, *cones*, *spheres* and *cylinders* can be reproduced with the help of new profiles.

## 2.3 Document Object Model (DOM)

Next we will look at a way of accessing an XML document from inside a program. The Document Object Model (DOM) is a set of interface definitions that allow programmers to load an XML document into a tree structure and to operate within the latter. Originally the DOM concept was used in connection with web browsers. Objects such as windows, documents and links were considered parts of the browser's object model. Needless to say, the type of implementation varied for the different browsers. With a view to offering a standardized type of implementation for accessing and manipulating documents on the Internet the *W3 Consortium* created specifications that eventually became what is now W3C DOM. W3C DOM is a collection of language-independent and platform-independent interface definitions for which no implementation details are predefined, so that implementation is possible with any programming language. DOM objects make it possible for program developers to export parts of a document, to carry out searches in it, to modify it, and to add or delete elements. DOM offers standard functionality for navigating in documents and manipulating the content as well as the structure of HTML and XML documents.

When used to manipulate an XML text file DOM first imports the file and parses it. In this process the file is broken down into its constituent parts such as elements, attributes, commentaries etc. which are then assembled in a tree structure in memory. This object structure represents the XML document (see Figure 2):

```
<Invoice>
  <Customer Name='John Doe'
      Address='Anystreet 17'
      City='Anytown'
      ZIP='34567'
      Telephone='0815-55555'/>
  <InvoiceItem Product='TelevisionSet'
      Quantity='1'/>
  <InvoiceItem Product='VideoCassette'
      Quantity='5'/>
  < InvoiceItem Product='VideoRecorder'
      Quantity='1'/>
</Invoice>
```

## 2.3 DOCUMENT OBJECT MODEL (DOM)

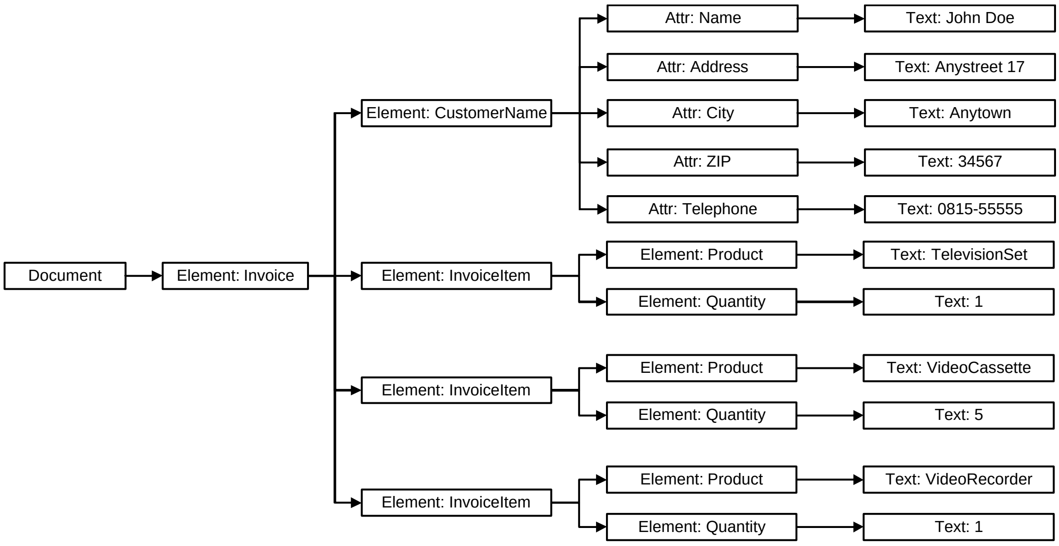



**FIGURE 2** An XML Document as DOM Structure

The use of DOM for the manipulation of XML documents offers a number of advantages over other mechanisms that can be used to generate XML, e.g. writing directly to an XML stream:

- *DOM guarantees a valid grammar and good document structure:* This avoids problems such as unclosed tags and incorrectly related tags. By way of example, an attribute in DOM can never be a child of another attribute node.
- *DOM abstracts the content of the document with its grammar:* A node tree is a logical representation of an XML file and contains information and its relations without being bound by the XML grammar. A developer does not necessarily have to be familiar with the grammatical details of XML in order to use DOM.
- *DOM simplifies the manipulation of documents:* With DOM it is very easy to add an element in the middle of the document. Global operations, such as the deletion of all elements with a certain tag name, can also be carried out with a small number of commands, instead of having to use a "brute force" method to search the file and remove the tags in question.
- *DOM mirrors typical hierarchical and relational database structures:* The way in which DOM represents the relations between data elements is very similar to that of modern hierarchical and relational databases. This makes it possible to use DOM to convert data between databases and XML files.

At the present time DOM Level 1 specifications have W3C recommendation status. However, specifications also exist for DOM Level 2. They are in conformity with Level 1 specifications and also contain extensions for namespace support, stylesheets, filter mechanisms and an event model. As was mentioned before, DOM specifications describe mechanisms and interface definitions, but do not describe specific implementations. Different implementations exist for different programming languages as well as for different hardware and software platforms. Microsoft's DOM implementation was used for this thesis and is presented in Section 2.5.1.

## 2.4 XML transformations

From one point of view XML is a document format. From another point of view it is a hierarchical model for the storage of data. From yet another point of view XML is a means of transferring data between applications connected to one another by a network. All these points of view take into account the common factor that an XML document is a set of elements organized in accordance with a specific schema. When XML is used to create flexible applications that communicate with other applications through networks and independently of specific platforms it must be assumed that different applications will make use of different structures to represent the same data. This chapter deals with the possibility of transforming an existing data structure into another XML vocabulary or of merely reorganizing the data structure. Transformations of this kind fall into one of the following three categories:

- *Structural transformations:* These are transformations from one XML vocabulary into another, equivalent to translations (e.g. transformations from X3D to VRML).
- *Dynamic generation of documents:* This makes it possible for users to filter or sort a document dynamically. Thus, by way of example, sort criteria could be selected by clicking on the headings of table columns.
- *Transformation into a render language:* This transforms the structure of data so that can be used for presentation, e.g. HTML.

### 2.4.1 XSL

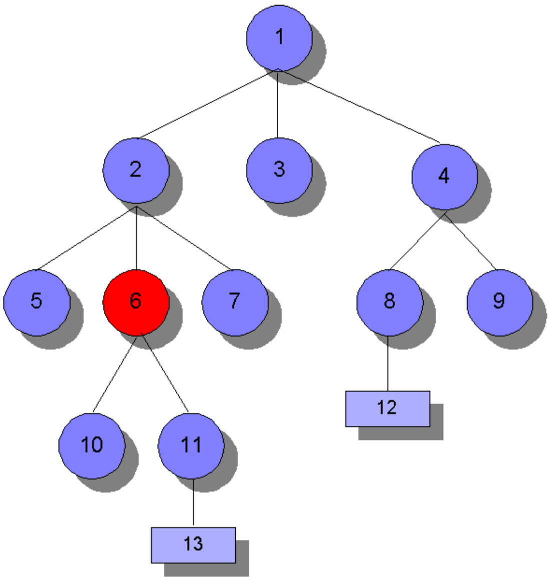
„eXtensible Stylesheet Language“ (XSL) is an XML-based language used to transform XML documents into other document formats or into a render language. XSL has its roots both in CSS („Cascading Style Sheets“) as well as in DSSSL („Document Style Semantics and Specification Language“) and is basically made up of the following two languages: XPath and XSLT. Both will be dealt with in the following two sections.

### 2.4.2 XPath

Given that increasing amounts of information are being stored in the form of XML documents, a way is needed to access this information (or parts thereof) in a structured manner. Specifications exist for languages which make it possible to express relations between documents, select sections of documents, as well as carry out specific searches for content (XLink, XPath, XPointer, XML Fragment Interchange). XPath was chosen for the implementation section of this thesis [W3C,1999,1], since an implementation of it exists and it can also be used to carry out content searches in an document instance.

XPath is a query language that was developed specifically for the purpose of addressing and selecting XML document sections. A set of elements or, very specifically, a single element of a document and/or its content can be selected with the help of so-called path expressions. Path expressions give the position of elements being searched for in relation to a so-called context node which, in accordance with established standards, is the root of the document. The notation is declarative and refers to the XML document tree that represents the content of the document as a hierarchy of nodes. The syntax is based on „Uniform Resource Identifiers“ (URIs), which are used to navigate in directories. Xpath queries make it possible to conduct searches in accordance with specific patterns at selected context nodes in order to carry out additional operations relative to the resultant set of nodes. This notation gives XPath expressions extraordinary flexibility for searches in a document tree. A path expression consists of a so-called „location step" or a number of such „location steps" which always have the following form and are separated from one another by a „/" sign:

```
axis::node_test[predicate1][predicate2]...
```




Axes relative to Node #6:

| | |
|---|---|
| self:: | **6** |
| child:: | **10,11** |
| parent:: | **2** |
| descendant:: | **10, 11, 13** |
| descendant-or-self:: | **6, 10, 11, 13** |
| ancestor:: | **2, 1** |
| ancestor-or-self:: | **6, 2, 1** |
| preceding-sibling:: | **5** |
| preceding:: | **5, 2, 1** |
| following-sibling:: | **7** |
| following:: | **10, 11, 13, 7, 3, 4, 8, 9, 12** |

**FIGURE 3** Axis dimensions of Xpath expressions

Each of these location steps produces a number of nodes that constitute a new context node set, to which the next location step is applied. The axis determines the relation between the nodes to be selected and the current context node set. The following largely self-explanatory expressions can be chosen for the axis: `ancestor`, `ancestor-or-self`, `attribute`, `child`, `descendant`, `following`, `following-sibling`, `parent`, `preceding`, `preceding-sibling`, `self` (see Figure 3).

The node test can be used to limit the node set produced by a location step to a certain type; the node tests `comment()`, `node()`, `processing-instruction()` and `text()` are used for this purpose. The node set on the selected axis can also be filtered with regard to an element name that is defined as a node test value. When the „*" sign (wildcard) is used, all elements are selected, i.e. there is no filter function.

The currently selected node set can be limited further by using a number of predicates. A predicate is a Boolean expression calculated for every node in the node set resulting from the axis and the node test. XPath offers a number of functions that can be used to select the desired nodes. These functions range from a simple determination of position inside a node set on down to string concatenations. Complex expressions can be formed with the help of comparison operators and mathematical functions. Some of the more important Xpath functions are: `count()`, `position()`, `last()`, `name()`, `concat()`, `contains()`, and `substring()`.

The following abbreviations exist for frequently used Xpath expressions. „`child::`" axis is the default setting when no axis is specified. „@" can be used in place of „`attribute::`" axis.

The expression „/descendant-or-self::node()/“ can be abbreviated „//“. „.“ is the equivalent of „self::node()“ and „..“ stands for „parent::node()“. Figure 4 provides an overview of these abbreviated notations:

| Path expression | Abbreviated expression |
|---|---|
| child::Book[position() = 1] | Book[position() = 1] |
| Book[position() = 1]/attribute::color | Book[position() = 1]/@color |
| /descendant-or-self::node()/Title | //Title |
| self::node()//Title | .//Title |
| parent::node()/Title | ../Title |

**FIGURE 4** Xpath expressions and their abbreviations

The following are more complex examples of path expressions. They are accompanied by explanations intended to enhance the reader's understanding of the language being presented:

- /bookstore[@specialty="textbooks"]: Select all bookstore elements at the root of the document that have a specialty attribute with the content „textbooks“.
- book[/bookstore/@specialty=@style]: Select all books whose style corresponds to the specialty attribute of the bookstore element at the root of the document.
- book[abstract][title]: Find all books that have an abstract and a title element.
- author[not(last-name[1]="Bob")]: Find all authors directly below the context node whose first last name is not „Bob“.
- //book[author/degree]: Select all books below the context node whose authors have at least one degree element.

### 2.4.3 XSLT

XSL is an application of XML, with which stylesheets can be developed for the formatting of XML files. XSLT is an XSL extension which makes it possible to modify the structure of XML documents. It can be used, for example, to delete elements or to change the sequence of elements. This makes it possible to transform an XML document into any other XML vocabulary or into any other structure. The XSLT stylesheet is a well structured and valid XML document whose elements consist, on the one hand, of XSL instructions in the XSL namespace and, on the other, of output elements which are exported directly into the transformed document. XSLT is a declarative language, i.e. it determines what the result of a transformation should look like but not how this result is achieved. An XSL stylesheet consists of a number of „templates“ which specify the way in which a node from the source tree is to be integrated in the transformed tree. The XSL processor traverses the tree of the untransformed document, starting from the root and trying to find an appropriate <xsl:template> instruction for each node in the stylesheet. If the processor finds an appropriate template expression it applies the specified rules to generate a representation of the untransformed node in the resultant tree.

The root of every stylesheet is the element <xsl:stylesheet version='' xmlns:xsl=''> with the attributes version and xmlns:xsl. This element declares the XSL namespace. The root element of the stylesheet can contain any number of <template match=''> instructions. The match attribute consists of an Xpath expression. If the

## 2.4 XML TRANSFORMATIONS

XSL processor finds a node from the source tree which is described by applying the match expression of a template to the current context, it applies the content of the template to the node. Each node of an untransformed document is allowed to have one and only one match expression. XSLT has the power of a computationally complete language. XSLT has more than thirty elements in its structural transformation rules covering the repeating, sorting, copying, and numbering of nodes as well as conditional instructions, only the most important of which are listed here. The instruction `<xsl:for-each select='XPath-Expression'>` is used for iteration over a previously selected node set. The element `<xsl:sort select='XPath-Expr.'>` is used to sort the elements in a "for-each" instruction. The instructions `<xsl:if test='Expr.'>`, `<xsl:choose>`, `<xsl:when>` and `<xsl:otherwise>` can be used for conditional processing of elements.

The following example demonstrates the use of a stylesheet. The original list of books is sorted by price and transformed into an HTML table. The structure of the source document is modified in the process. The sequence of elements is changed and elements have been deleted.

The original XML document:

```
<booklist>
  <book title='Wintertime'>
    <author>Hans Maier</author>
    <type>novel</type>
    <price>24.90</price>
  </book>
  <book title='Professional XML'>
    <author>Peter Müller</author>
    <type>nonfiction</type>
    <price>99.90</price>
  </book>
  <book title='Death in the Bathtub'>
    <author>Klaus Schmidt</author>
    <type>murder mystery</type>
    <price>19.90</price>
  </book>
</booklist>
```

The XSLT stylesheet:

```
<?xml version='1.0'>
<xsl:stylesheet version='1.0'
       xmlns:xsl='http://www.w3.org/1999/XSL/Transform'>

  <xsl:template match='book list'>
    <HTML><BODY><TABLE Border='1'>
    <TR><TH>author</TH><TH>title</TH><TH>price</TH></TR>

    <xsl:for-each select='book'>
      <xsl:sort select='price'/>
      <TR>
        <TD><xsl:value-of select='author'/></TD>
        <TD><xsl:value-of select='@title'/></TD>
        <TD><xsl:value-of select='price'/></TD>
      </TR>
    </xsl:for-each>

    </TABLE></BODY></HTML>
  </xsl:template>
</xsl:stylesheet>
```

The HTML document resulting from the transformation:

```
<HTML>
  <BODY>
    <TABLE Border='1'>
      <TR><TH>author</TH><TH>title</TH><TH>price</TH></TR>
      <TR>
        <TD>Klaus Schmidt</TD>
        <TD>Death in the Bathtub</TD>
        <TD>19.90</TD></TR>
      <TR>
        <TD>Hans Maier</TD>
        <TD>Wintertime</TD>
        <TD>24.90</TD></TR>
      <TR>
        <TD>Peter Müller</TD>
        <TD>Professional XML</TD>
        <TD>99.90</TD></TR>
    </TABLE>
  </BODY>
</HTML>
```

# 2.5 MSXML and XDR

The implementation section of this thesis is based in large part on XML and the related technologies. Since the developed application also had to access XML schemata and Microsoft's XML SDK (MSXML) was at that time the only implementation that supported schemata in XML notation in addition to DTDs it was the product chosen. Version 3.0 of the MSXML implementation was used, shipped with Windows 2000 and the Internet Explorer version 5.5 SDK It includes a parser with DOM implementation and an event-based parser SAX („Simple API for XML“). The DOM implementation supports namespaces, contains an XSLT processor, and can evaluate XPath expressions. The software package contains a full implementation of Microsoft's XML schema proposal, XDR. The current version of MSXML is 4, which fully supports the XML-Schema language.

## 2.5.1 DOM API, XPath, and XSLT

The MSXML package is implemented in the form of a COM component. The latter can be used by a number of programming and script languages such as *JavaScript*, *Delphi*, *Visual Basic*, *VisualCOBOL*, *VBScript*, *PerlScript*, *PythonScript* and *C++*. The DOM implementation deviates slightly from the W3C specifications for Level 1 DOM, but offers a number of very useful proprietary extensions. In Section 2.3 it was explained that an XML document is represented in the *Document Object Model* in the form of a tree. A total of twelve node types are available for the creation of a tree structure, the most important of which are: `IXMLDOMDocument`, `IXMLDOMElement`, `IXMLDOMAttribute`, `IXMLDOMComment`, `IXMLDOMText`. The interface definitions of all node types inherit from the object `Node`.

`IXMLDOMDocument` is located at the root of the document tree. Via the `validateOnParse` property the programmer can indicate whether or not the document is to be validated, if a DTD was indicated. The `resolveExternals` property determines whether or not default attributes are to be inserted automatically by the parser; with the `asynch` property the parser can be instructed to parse and validate in the background and to continue with the rest of the program parallel to that. The parsing process is initiated with the `load` method, in which the XML document is exported directly from a text file; in the case of the `loadXML` method the XML document is a string in the form of a transfer parameter.

After the parser has generated a tree structure in memory the program can navigate the tree and change both structure and content with the help of the functionality of the `Node` interface. The type, name, and content of a node can be determined and changed with the properties `nodeType`, `nodeName` and `nodeValue`. The tree can be navigated with properties such as `firstChild`, `nextChild` and `lastChild` and can be manipulated with methods such as `appendChild`, `removeChild`, `replaceChild`, `insertBefore` and `cloneNode`. The container objects `NodeList` and `NamedNodeMap` continue to exist, with which it is possible to iterate over a set of nodes or attributes. The methods `xml` and `save` return the document tree as an XML string or save it directly as a file in the file system.

Xpath expressions can be evaluated directly on any node representing a context node. To do this one of the methods `selectSingleNode` or `selectNodes` is called up with a path expression as a parameter on any `IXMLDOMNode` which then directly returns either the selected node or a `NodeList` object. The XSLT processor is also directly implemented in the form of method calls on the DOM node. The XSLT stylesheet only needs to be loaded as `IXMLDOMDocument` and then to be applied to a node of another XML document through one of the methods `transformNode` or `transformNodeToObject`. These methods then supply the transformed document either as an XML string or as a further document tree structure.

### 2.5.2 XDR

„XML Data - Reduced“ (XDR) is the only XML schema proposal for which an implementation existed at the time the framework for this thesis was developed. More recent versions of Microsoft's MSXML SDK support XML schema. However, applications can be developed with XDR, making use of the XDR schema, which can easily be adapted to the official W3C recommendation of XML schema. XDR schemata are noted in XML, essentially have the functionality of traditional DTDs, and support namespaces as well as datatypes. The object-oriented design and the inheritance mechanisms of XML Schema are, however, not supported by XDR.

The root of an XDR schema is always the `<Schema>` element. It may contain any number of declarations and applications of elements and attributes. Declarations are made via the `ElementType` and `AttributeType` elements. The latter have the attributes `name`, `content`, `dt:type`, `model` and `order`. `empty`, `textOnly`, `eltOnly` or `mixed` can be selected for the `content` attribute. `dt:type` determines the datatype and the `model` attribute indicates whether or not any elements will be permitted as content (`open`) or only those defined in the schema itself (`closed`). In an `ElementType` declaration, previously declared elements can be used to model content (*content model*) in the manner specified in the `order` attribute (`one`, `seq`, `many`). With the help of the `<group>` elements parenthetic expressions can be reproduced in the *content model* of DTDs.

The elements `<attribute>` and `<element>` are used for the employment of previously declared elements in the *content model* of another element. The `type` attributes must agree in each case with the `name` attribute of an `ElementType` or `AttributeType` declaration. Value ranges can be indicated with the `minOccurs` and `maxOccurs` attributes of the `element` tags, in contrast to the occurrence symbols of DTDs. With regard to attribute declarations, a defined standard value can be set via the `default` attribute, while the `required` attribute determines whether the parser will view the occurrence of the attribute as mandatory or optional.

An element declaration in DTD notation and its equivalent in XDR are given in the following to illustrate the above-described concepts:

```
<!ELEMENT Foo ((X | Y) | (A, B?, C))>

<ElementType name='Foo' content='eltOnly' order='one'>
  <group order='one'>
    <element type='X'/>
    <element type='Y'/>
  </group>
  <group order='seq'>
    <element type='A'/>
    <element type='B' minOccurs='0'/>
    <element type='C'/>
  </group>
</ElementType>
```

## 2.6 ASP and VB

„Active server Pages“ (ASP) is a technology that provides a server-side scripting environment in connection with Microsoft's *Internet Information server* (IIS) by means of which dynamic web pages and applications can be created. ASP pages are files that contain HTML tags and script commands which access databases and dynamically generate the content of HTML pages in run-time situations. The script parts of ASP pages can be written in *JavaScript* and *VBScript* and make use of *ActiveX* components to carry out complex calculations and keep business logic separate from presentation by HTML pages. The execution of precompiled components is also more powerful than runtime script interpretation. *Active Server Pages* also makes it possible to manage so-called „sessions“, i.e. for every client who connects with the server an environment is created in the server in which the status of the connection and certain types of information, such as user input, is stored and managed.

Like traditional HTML pages, ASP pages are called up by a URL. The server loads the page, executes the script code it contains, and sends the produced HTML page back to the client. Objects for session and application management are standard features on the server. *ActiveX* components can be instantiated by calling up the `CreateObject` method of the `Server` object. Script code execution as evoked by client callup can be configured via the `Request` object. Commands for this purpose are encoded in the URL and evaluated in the script code with the help of `Request` objects.

The following is an example of the callup of an ASP page via a URL:

```
http://localhost/test.asp?Command=getPicture&Filename=logo.jpg
```

The following example of *JavaScript code* describes the evaluation by the ASP page:

```
<% @ LANGUAGE = "JScript" %>
<% Response.ContentType = "text/HTML" %>
<%
  Command = Request.QueryString("Command");
  if(Command==’getPicture’) {
    %>
    <HTML><BODY><IMG SRC=’
  <%Response.Write(Request.QueryString("Filename"));%>
    ’></BODY></HTML>
    <%
  }
  elseif(Command==’...’) {
  }
%>
```

*ActiveX* components that are instantiated and used in an ASP page can be implemented with any programming language that is able to generate an *ActiveX* DLL (*Visual Basic*, *VisualC++*, *Java*). The *Visual Basic* programming environment, part of Microsoft's *Visual Studio,* is designed to provide application developers a powerful programming language for what is referred to as „Rapid Application Development“ (RAD) on the Windows platform. *Visual Basic* (VB) is intended to make it possible to realize the development of graphics applications rapidly, efficiently, and with user friendliness based on familiarity with program elements from the Windows operating system. Visual Basic is one of the most frequently used programming languages in the history of computers. Its outstanding features are that it is easy to learn and easy to use. The "dialect" constituted by *MS Visual Basic* is an object-oriented variant of the original language that has been expanded to include numerous elements for direct interaction with the Windows operating system. The object-oriented features in the Version 6.0 are, however, still recognizable as later additions and do not make it possible to engage in purely object-oriented design work.

Since only a limited amount of time and human ressources was available for the implementation part of a thesis, and *Visual Basic* held out the prospect of being able to complete development work in a relatively short period of time, it was chosen as the programming language. A large number of ready-to-use components such as "Treeview", "FileDialog", etc. are supplied with VB or are freely available on the Internet so that development time can, indeed, be reduced in comparison with other programming languages. Further details on VB will not be discussed here and can be read in [Koffler,1998],[Appleman,1999], and [Baartse,2000].

## 2.7 VRML extensions for data driven 3D-visualization

The Virtual Reality Modeling Language (VRML) [Web3D,1997] has proven to be a powerful platform for the creation of three-dimensional objects and scenes. However, the current specifications do not provide for mechanisms that would make it possible to access and manage external data sources (e.g. relational databases). This capability is crucial for the use of VRML in more complex applications. Without mechanisms of this kind a VRML scene remains isolated and is not able to present external data in the scene nor update it in the virtual world on the basis of changes that have taken place. The concepts described below are intended to make it possible to construct a VRML scene on the basis of database queries and to synchronize VRML content dynamically with database content in runtime situations. More details can be found in [Leissler,1999].

The *Server Side Include* mechanism (SSI) can modify a VRML scene called up by a client before it is delivered. These changes are made exclusively on the server side. In the approach taken by [Risse,1998] and further described in [Leissler,1999] a prepositioned server component examines the VRML scene for embedded database queries, connects to the database, and inserts the results into the VRML scene to be delivered. The instructions for the database accesses to be undertaken are included in the VRML file by means of an extension of VRML, a so-called `SQLServer-Includes` node. In principle the enrichment of the original scene with database content could be formulated with any syntax. The selected approach proved to be advantageous, particularly in the case of errors, since `SQLServer-Includes` nodes are preserved in the scene, contain the SQL statement, and may contain an error message. The multiple instantiation of visual objects is provided for, but requires the presence of all positioning data for the objects in the database.

The *Runtime SQL Node* was motivated by the same considerations as the *Server Side Includes*, but it takes an alternative approach. The VRML event model makes it possible for 3D objects and worlds to show independent behavior. Scene description nodes generate events that, when transmitted to other nodes, produce reactions in the scene. In VRML script nodes make possible the specification of complex program logic. SQL scripting, also known as embedded SQL, specifies a script node for accessing databases via SQL instructions. The idea behind this approach was developed by the VRML Database Working Group [DBWORK] and the *Runtime SQL Node* constitutes an implementation of the described concept. The script node is activated by an event in a runtime situation. It causes the client to connect to a database and execute an SQL instruction. The result, in turn, is transmitted by events to other nodes. The *Runtime Node* makes it possible to execute any SQL instructions when a VRML scene is in runtime mode. The instances in which a database is accessed are based on commands in the program logic of the VRML world as carried out by the client.

The functionality of the *Runtime Node* goes beyond that of the SSI mechanism, in that the persistence of a VRML scene can be achieved by means of SQL UPDATE instructions. In other words the changes in a scene affected by a user can be written back into a database and preserved for the future.

The third and last mechanism is that of the *Trigger Server*. The open structure of the VRML standard opens up a large number of options for generating a VRML scene dynamically. The *Server Side Include* and *Runtime Node* technologies are two approaches of this kind. Regardless which of the above-indicated ways is used to generate a VRML scene dynamically on the basis of changeable data, the following problem exists: changes in the underlying data do not automatically lead to changes in the generated VRML scene, i.e. the virtual scene is not synchronized with the data. Without additional mechanisms it will become obsolete. The *TriggerSserver* described in [Leissler,1999] and [Leissler,2000] creates a 1-to-n relationship between a database system and running VRML applications so that changes in database status result in all related VRML scenes being notified. This enables the latter to adjust their appearance to current database status.

The architecture consists of the components *Trigger Server* and `External Sensor`. The *Trigger Server* is directly linked to an active database management system (ADBMS) whose database is monitored. External sensors that want to be notified of changes in the ADBMS can register (and unregister) with the Trigger Server. SQL3 triggers cause the ADBMS to initiate the *Trigger Server* notification mechanism whenever changes take place in the database. The latter, in turn, notifies all external sensors that are registered with it of any changes that have taken place. The `External Sensor` defines a new VRML node type that can be used in VRML scenes to forward external notifications in the form of events.

The implementations of all three mechanisms are realized as VRML nodes and, as such, are very closely connected to the use of VRML. Instead of encapsuling program code in VRML nodes in the form of Java classes it is also conceivable to manage database connections and data transfers to external components (*JavaApplet*, *ActiveX-Control*) and to manipulate the VRML scene from the outside via the EAI. This would also simplify changeover to other render technologies (e.g. X3D). This factor is taken up again in Section 7.2.2 and discussed there in more detail.

# 2.8 A taxonomy for Information Visualization

A Taxonomy has the general goal of classifying objects. In our case the objects are applications using the process of information visualization. This allows us to properly discuss some upcoming questions on data types and information visualization techniques. In [Englberger,1995] several general information visualization (IVIS) techniques and different approaches for a taxonomy of IVIS techniques are presented, all which were published until the year 1995. However, in the following we will look about on the taxonomy published in "A Task by Data Type Taxonomy for Information Visualizations" (TTT) by Ben Schneiderman [Shneiderman,1996], which is not included in [Englberger,1995].

Within the TTT 7 data types and 7 tasks are described.

The 7 data types are:

- **1-dimensional**: Like, e.g., a text file (line by line) or an alphanumeric list of names.
- **2-dimensional**: Like, e.g., a geographical map or a book layout
- **3-dimensional**: 3D-Objects, like e.g, chemical molecules or buildings.
- **Temporal**: Time-Series, e.g., scientific measurement rows or
- **Multi-dimensional**: Typical within relational databases
- **Tree**: A tree structure is a set of information objects IOi in which every IOi has a father object except the root node.
- **Network**: All other structured object sets, which do not have the tree constraints.

The TTT data types impose a certain level of abstraction, so that more specific data-types (e.g. 2.5 D) should be treated as subordinates.

The 7 tasks of the TTT are:

- **Overview**: Overview over an information space. Techniques to achieve this are, e.g., zoom-out or fisheye view.
- **Zoom**: Enlarge the interesting details.
- **Filter**: Filter out the uninteresting details.
- **Details on demand**: Detailed information will be presented, if requested by the user. This reduces the cognitive load to the user and the load on the computing resources.

- **Relate**: Display the relations between information objects.
- **History**: Remember the user actions, such that functions as undo and redo are possible.
- **Extract**: Extraction of specified areas of the information space, in order to store this area (e.g. in a file) or to communicate it (e.g. via e-mail).

For example, the often-used term of Focus+Context can be described within the TTT as a combination of overview, relate and zoom. This means that the area selected by the user is enlarged and presented in more detail whereas the context (the structure) is scaled down in size and displayed with less details.

Chris North maintained a list entitled "A Taxonomy of Information Visualization User-Interfaces" during his time at the Human-Computer-Interaction Laboratory at the University of Maryland, which contained published IVis-Applications and visualization techniques to be fitted into the TTT. His work is now continued at Olive[2] (On-line Library of Information Visualization Environments).

## 2.9 Visualization techniques for information spaces

In this section we will present some well known visualization techniques, which mainly apply to the tree data type of the TTT or the network data type (see section 2.8). Although this thesis deals with the process of information visualization rather than with specific IVis techniques, it is nonetheless important to describe and analyze some of the important techniques in order to get an understanding for the general requirements of software implementations of such techniques and the components involved.

In order to gain an understanding why such IVis techniques are important we shall use the example of an idealized HTTP server based Web site. The uppermost page of the WWW server is the root page. From this page all other pages, which are typically grouped together in a logical way, are reachable directly or indirectly. If a WWW document (a node) does not contain any more links to other pages it is called a leaf. With growing data to be published within the web site, the complexity of the site increases. For the management of such complex WWW-sites appropriate visualization and manipulation tools are needed to administer the huge amount of information represented by the site. A graphical view of the web site in the form of a tree diagram seems most appropriate. (see Figure 5)

[2] http://otal.umd.edu/Olive/

## 2.9 VISUALIZATION TECHNIQUES FOR INFORMATION SPACES

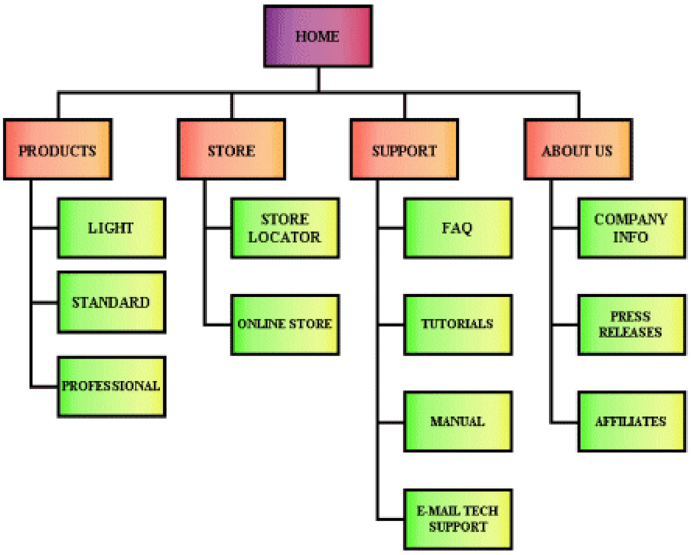



**FIGURE 5** A tree diagram for a hierarchical web site structure

The figure displays the visualization of the site structure in the 2D plane. Real WWW servers, however, have a significantly higher complexity. One can easily imagine, that the given visualization reaches its limits quite fast as complexity increases. That means it is important to find more appropriate visualization techniques, which are able to visualize highly complex hierarchical structures in a concise way.

For the IVis on hierarchical structures 2 major goals can be derived, which are supported by corresponding visualization strategies.

- Analysis of the information space. This goal is typically achieved by using visual correlations to represent semantic correlations.
- Navigation within the information space. This can be achieved by providing advanced visual navigation tools to graphically represent the user navigation paths.

In the following two sections we will take a look at visualization techniques used to visualize hierarchical or arbitrarily structured information spaces in 2D and in 3D.

### 2.9.1 Visualization techniques in the 2D plane

#### 2.9.1.1 Treeview

The Treeview visualization as displayed in Figure 6 (Source: University of Maryland: http://www.cs.umd.edu/hcil/treemap-history/index.shtml) is the best known and traditional way of displaying hierarchies. Figure 6 depicts a part of a satellite management system for Hughes Network Systems. The three-level hierarchy shows each node of a network at a fixed size and color was used to indicate available capacity. However, for the vast amount of data occurring within today's applications this technique seems no longer appropriate for most cases.

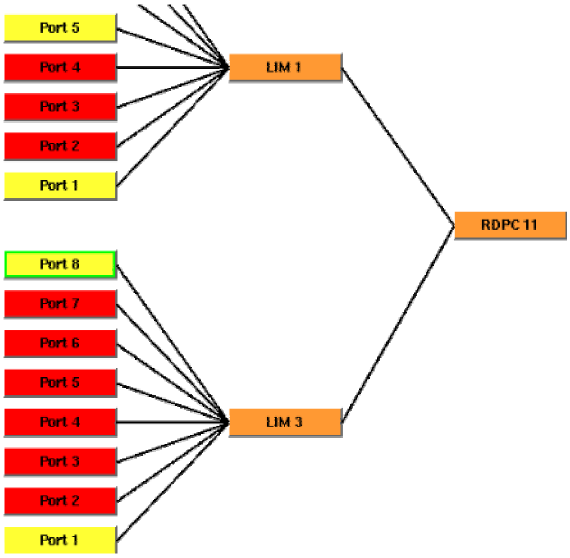



**FIGURE 6** A Treeview

### 2.9.1.2 Fisheye View

In [Furnas,1981] the Fisheye View (FEV) technique is presented as a Focus+Context technique. The main idea of the FEV is the detailed display of a local area (focus) as well as important parts of the global information structure as a context. FEV's can be generally divided into two classes. The *Filter FEV* and the *Distortion-FEV*. With the Filter FEV all information below a given relevance threshold is filtered out of the display.

The Distortion-FEV simulates an optical fisheye lens. This means that objects in the focus (of the lens) are enlarged and displayed in more detail, whereas objects outside of the focus are shrunk and displayed in less detail.

The Filter FEV is the original concept presented in [Furnas,1981]. However, in contrast to the Filter FEV the Distortion FEV can be combined with nearly every other visualization technique because it is applied to the presentation process.

Dependent on the application specific goal there have been several different realizations of the FEV, which can be enumerated as follows:

- Filter FEV
    - Clustering [Fairchild,1988]
    - 3D Point Perspective [Fairchild,1988]
    - Sampling Density [Fairchild,1988]
- Distortion FEV
    - For Graphs [Sarkar,1992]
    - Fisheye [Misue,1991]
    - Orthogonal [Misue,1991]
    - Biform [Misue,1991]
    - Tree [Furnas,1981]

In [Furnas,1981] this technique was applied to structured files (e.g. computer programs), where the FEV is defined as a mathematical function DOI (Degree of Interest). This function is calculated from the parameters of focal point, distance of a point to the focal point, and level of detail of a given node. This means that the value of the DOI function (the degree of interest) for a given node is dependent on his distance to the root node as well as to the focal node. Furthermore a threshold value k is defined, which decides if a given node is displayed.

In [Noik,1993] a FEV is described, which can neither be classified as Filter FEV nor as Distortion FEV. It deals with a layout independent FEV of structured graphs with multiple focal points, which is used as a tool for the exploration of large structured hypertext networks.

In [Mukherjea,1995,1] a Focus+Context technique is described, which is very similar to the Filter FEV by calculating the degree of interest of a given node of a hierarchy to display. This technique also uses a threshold value for deciding upon the visibility of a node.

It shall be explicitly noted, that the term "Fisheye View" describes an abstract concept rather than a specific visualization technique. This term is also used as metaphor for other techniques, which have features similar to the Fisheye lens.

#### 2.9.1.3 Treemap

Treemap [Shneiderman,1992] is a development of the Human-Computer-Interaction Laboratory of the University of Maryland. In the treemap visualization technique a 2D rectangle is recursively subdivided in order to display large hierarchies.

The algorithm for the generation of the treemap visualization is fairly simple: An initial rectangular surface is cut along the horizontal or vertical direction in order to display level 1 of the hierarchy. Then the resulting surfaces are subdivided in the respective other direction to display level 2. This step is now successively repeated for every hierarchy level. The area of the subdivison surfaces are used to display information.

The treemap technique is used in the TreeViz [Johnson,2000] and WinSurfer [Teittinen,1994] applications. Both applications serve as an interface to the hard disk involving display as well as interaction. The Figure 7 displays the application of the treemap technique within WinSurfer. With this application the users consuming the most space on a hard disk or the largest files on the disk can easily be detected.

In [Jungmeister,1992] the treemap visualization technique is used for share portfolio management. This application seems downright predestined for using this technique. In [Turo,1994], furthermore, the combination of the treemap display and a distortion function is described.

For the TreeViz application a realization for more than 5000 information objects has been presented. The main disadvantage, however, of this technique is, that in many cases surfaces may have the same size but completely different forms, which clearly makes a comparison of objects using treemap techniques difficult.

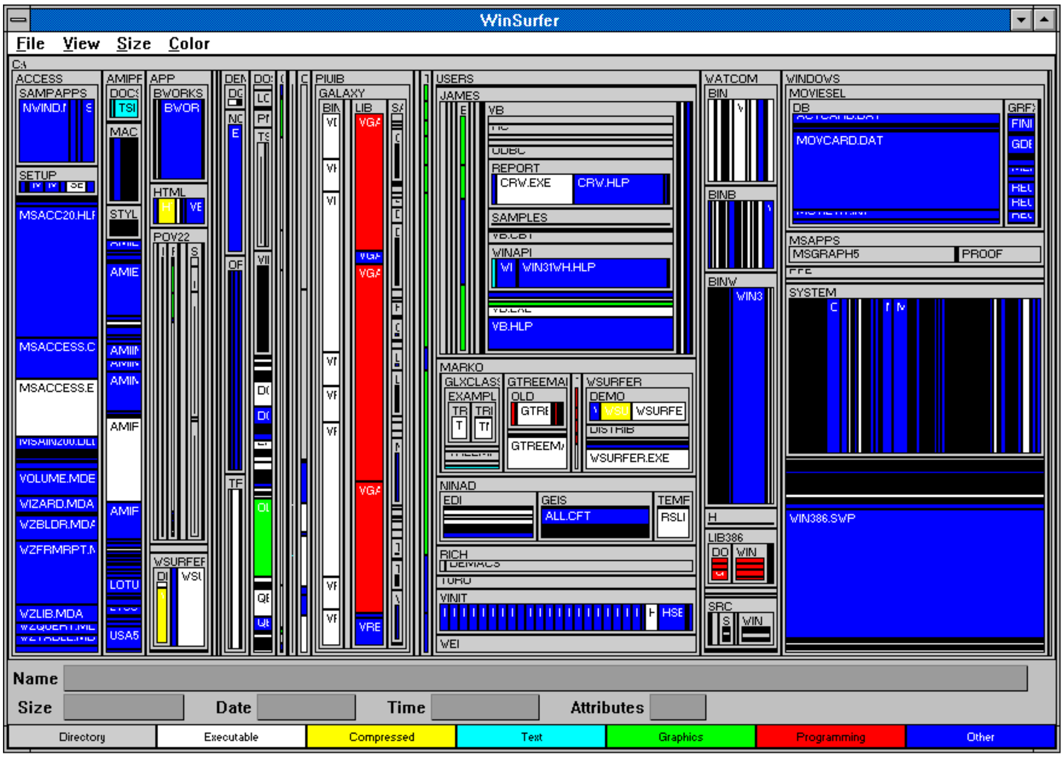


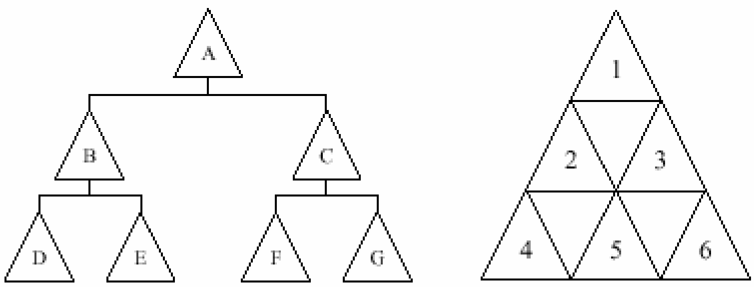

**FIGURE 7** A Treemap visualization in WinSurfer

### 2.9.1.4 Cheops

Cheops [Beaudoin,1996] is a visualization technique specially developed to compactly display large hierarchical information spaces. Cheops aims at the following goals:

- Keep context in large hierarchies
- Easy access to details
- No grouping mechanism (e.g., LOD)
- No degree of interest function (DOI) (therefore no repeated recalculation of the display is necessary)

The basic element of the Cheops method is the triangle. Figure 8a displays a binary tree in its usual view. Figure 8b displays the same tree using the Cheops view.




(a) binary hierarchy (b) Cheops form

**FIGURE 8** The Cheops principle

The Cheops form of Figure Figure 8b can be seen as a compressed form of the binary tree of Figure 8a. The triangle with the No. "5" in Figure 8b can either represent node "E" or node "F", depending on the selection of the father triangle being either "2" or "3". This means that the visual representation is the result of a process of unfolding, depending on the superior hierarchy level.

For large hierarchies with more than 1000 information nodes the compression of the Cheops method will show up stronger. A single triangle can then represent multiple information objects as displayed in Figure 9 with a ternary tree.

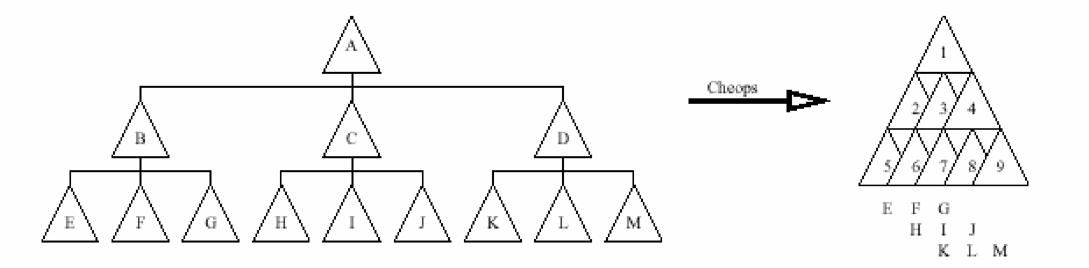



**FIGURE 9** Cheops: Displaying the compression

The total number of cheops objects in a given display can be described by the following formula:

$$\sum_{i=1}^{k}[(n-1)(i-1)+1]$$

whereas *n* is the fixed number of child nodes of the n-ary tree and *k* is the depth of the tree hierarchy.

This is a dramatic improvement over the traditional n-ary tree display, which has $\sum_{i=1}^{k} n^{i-1}$ nodes.

The following table relates growing binary trees (increasing in *n* and *k*) and their traditional and Cheops representation to each other in number of objects (first table) and spatial span (second table).

| *n* | *k* | *No. of logical nodes* | *No of Cheops objects* |
|---|---|---|---|
| 3 | 3 | 13 | 9 |
| 3 | 6 | 364 | 36 |
| 6 | 6 | 9331 | 81 |
| 8 | 9 | 19.173.961 | 261 |
| 10 | 10 | 1.111.111.111 | 415 |

| *n* | *k* | *No. of logical nodes in the bottom level* | *Span of Cheops objects on bottom level* |
|---|---|---|---|
| 3 | 3 | 13 | 5 |
| 3 | 6 | 243 | 11 |
| 6 | 6 | 7776 | 26 |
| 8 | 9 | 16.777.216 | 57 |
| 10 | 10 | 1.000.000.000 | 82 |

If triangles with a side length of 1,2 cm are used, the spread of a Cheops visualization of a binary tree with n=10 and k=10 would be 12 cm. With the traditional view from Figure 8 it would already be 12.000 km!

In [Beaudoin,1996] besides the Cheops method also the visual characteristics, the selection as well as the browsing of Cheops structures are discussed in some detail.

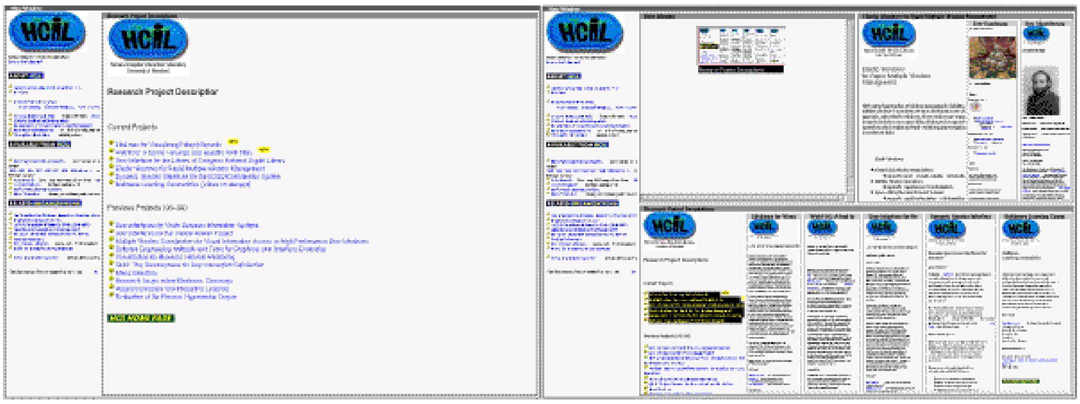
#### 2.9.1.5 Elastic Window

One of the most popular ways of displaying information spaces is the use of a workspace within a window-based system, such as MS Windows or X11, wherein the information objects are contained within the windows. Such a window can contain e.g. a shell, text documents or compound multimedia documents. A typical user task requires users to open multiple windows. The semantic grouping of the windows, e.g. different views on a 3D object displayed in separate windows, demands a significant cognitive effort from the user using the widow systems available today. This is due to the fact that relations, groups, and hierarchies of windows are not graphically displayed and the windows can even overlap each other. As a result users often forget to complete unfinished work, lose overview, or need significant amounts of their available time to organize the windows and the operations to be applied to them. Studies on the topic of working with window systems are summarized in [Kandogan,1996].

In [Kandogan,1996] the concept of Elastic Windows is proposed to solve the described problems implying a more efficient layout and faster window operations. The elastic window principle is made up of 3 components:

- Hierarchical window layout according to their semantics
- Space filling partitioning of the windows (similar to the Treemap technique)
- Operations across multiple windows


(a) (b)



(c) (d)

**FIGURE 10** Elastic Windows for the display of a WWW document hierarchy

Through the special layout of the windows particularly the hierarchical structure is displayed. The operations defined on windows can be applied to any node within the hierarchy, meaning that operations can affect a whole group of windows at once.

This feature is mainly interesting for the purpose of navigating a hierarchically structured information space as displayed in Figure 10 (Source: Human Computer Interaction Laboratory, University of Maryland[3]). (a) displays the access to a web page. (b) Clicking into the first page brings up a second page to the right. (c) Because the second page is of greater interest, the user magnifies it. (d) Displays a later state of the navigation within the hierarchy of hypermedia documents. With the help of a hierarchy icon multiple documents can be pooled. The Elastic Window homepage also contains an MPEG video for the demonstration of this dynamic process.

Because of the necessary spatial requirements of the hierarchy nodes (windows) this technique is appropriate for relatively small hierarchical information spaces containing less than 100 information objects, whereas the set of information objects changes dynamically over time.

### 2.9.1.6 Hyperbolic Tree

In [Lamping,1996] and [Lamping,1995] the Hyperbolic Tree information visualization technique is introduced (under the name Hyperbolic Browser) as a focus+context technique for large hierarchical information structures.

The principle of the hyperbolic browser is made up of 2 components:

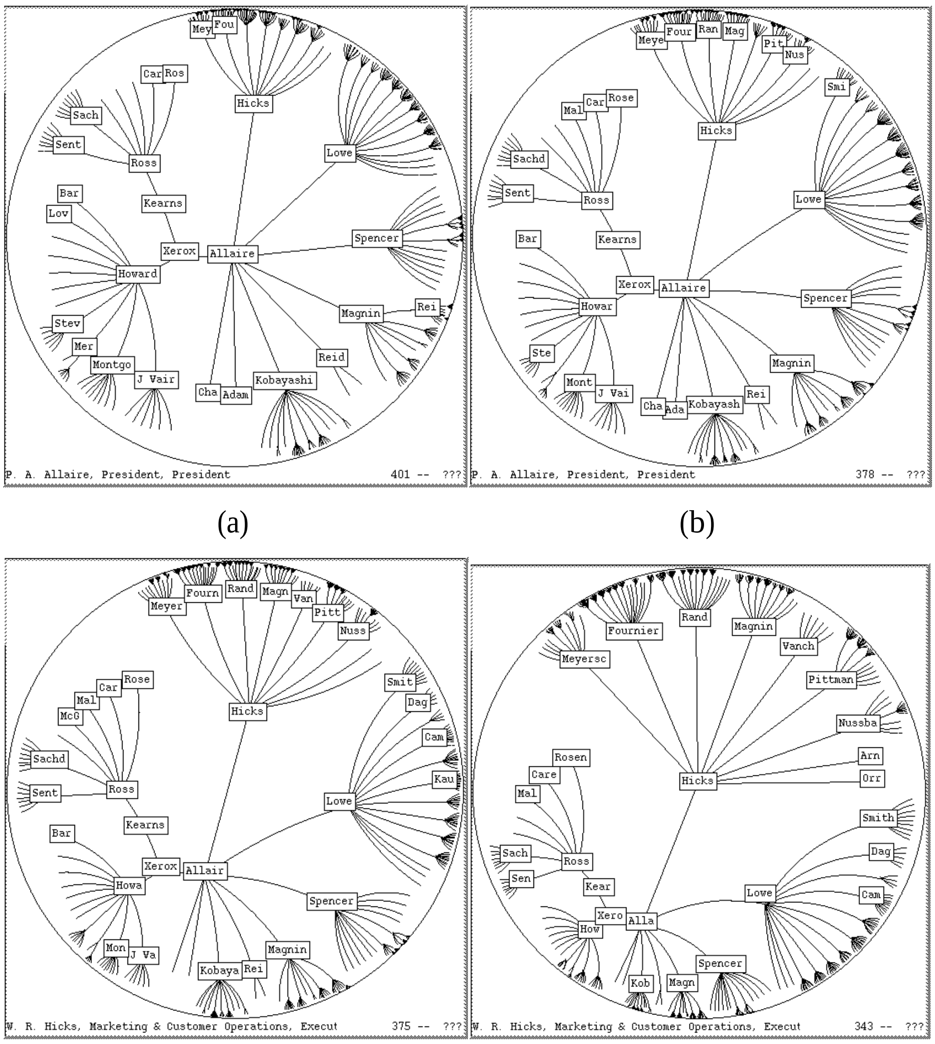
- Star-shaped layout of the hierarchy on a hyperbolic surface with a calculation of the position and span of given hierarchy components.
- Mapping from the hyperbolic surface to an Euclidian circular area.

Figure 11 (Source: [Lamping,1995]) depicts the display of a hierarchy using the hyperbolic browser.

The display of a tree on a hyperbolic surface has the feature that the size of the displayed components and the space between them decreases with the distance from the center in the direction of the circles edge. In the hyperbolic browser the root node is initially at the center and the com-

[3] http://www.cs.umd.edu/hcil/elastic-windows/

plete hierarchy is displayed. An interaction with the hyperbolic browser is achieved by clicking and dragging the node of interest to the center of the circle. Thereby the node itself and its near environment will be displayed in more detail, without losing the context. The user controlled tree view is achieved through transforming and scaling the initially calculated components of the hierarchy. Figure 11 (Source [Lamping,1995] [4]) demonstrates the navigation in a hierarchy in which the node "Hicks" is dragged to the center.




(a) (b)

(c) (d)

**FIGURE 11** Navigating a hierarchy with the hyperbolic browser

The authors of [Lamping,1995] state that with the hyperbolic browser in comparison to 2D display techniques in the Euclidian surface an increase in the number of nodes by factor 10 can be achieved at the same time maintaining the ability to effectively navigate in the hierarchy. In their test environment they have successfully presented hierarchies with 2000 information objects.

[4] http://www.acm.org/sigchi/chi95/Electronic/documnts/papers/jl_bdy.htm

## 2.9 VISUALIZATION TECHNIQUES FOR INFORMATION SPACES

The display of hierarchical information spaces using the hyperbolic browser has become a well-known technique in the meantime and its application can be found in multiple products, such as VizControls[5] from inXight, a subsidiary company of Xerox.

On its web pages inXight demonstrates the display of various hierarchies as a hyperbolic trees realized through a Java Applet[6] that allows the navigation of hierarchic web site structures and direct access to the pages. VizControl was licenced by Softsquad (http://www.softsquad.com) in order to integrate the technique in its products HoTMetaL Intranet Publisher and HoTMetaL PRO. The inXight technique is also used in the Microsoft Site Server and Netscape's WWW server.

A Java-Applet[7] developed by Jeff White allows some experiments with hyperbolic trees. This applet allows a user to switch between hyperbolic and Euclidian surface with the difference between the two growing with the size of the hierarchical structure to display.

### 2.9.1.7 Fractal View

Fractal View was presented in [Koike,1993] as an alternative display technique for large hierarchies. The basic principle is similar to the Filter FEV and is based on geometric self-similarity feature of fractals. Figure 12 displays a binary tree displayed as a fractal. Every edge is larger than its predecessor by the factor *r*. A value *D*, which represents the fractal dimension [Foley,1990], describes the relation between the branch factor *N* and the scale factor *r* as

$$D = -log(r*N)$$

In the Fractal View technique a fractal value $F_v$ is calculated for every hierarchy node x named $F_{vx}$ in order to get a measure for the relevance of a node.

$$F_v(focus) = 1$$

$$F_v(child\ of\ x) = rx * F_{vx}$$

$$rx = C * Nx^{-1/D}$$

The formulas describe the calculation of $F_{vx}$ . $Nx$ hereby is the branch factor of node $x$ and $C$ is a constant value with $C \in [0 \ldots 1]$. An important fact about fractal view is that the maximum number of displayed nodes M can be controlled with the following formula:

$$F_v = r^n = N^{-n/D}$$

$$M = \frac{F_v^{-D} - \frac{1}{N}}{1 - \frac{1}{N}}$$

With growing $N$ the maximum number of nodes $M$ will converge against $F_v^{-D}$ , whereas $M$ already comes near to $F_v^{-D}$ for values of $N$=3, $N$=4. Because of the self-similarity feature the hier-

---

[5] http://www.inxight.com/inprodvz.htm

[6] http://www.inxight.com/products/st_sdk/online_demos_st.html

[7] http://www.cs.brown.edu/people/jsw/java/hypertree.html

archy presents itself to the user in an equal distributed layout. The principle of fractal view layout is depicted in Figure 12 [8] (Source: [Koike,1993]). The principles are:

- Calculation of $F_{vx}$ for every node x.
- Definition of a threshold value n.
- Display if $F_{vx} > n$.

Through the selection of nodes depending on the value of $F_{vx}$ only nodes in the surrounding environment of the focus are displayed. If the threshold value is kept constant and the focus changes within the hierarchy the number of displayed nodes (M) will also remain nearly constant. Therefore, the maximum number of displayed nodes can be specified even with large hierarchies, such that the cognitive load on the user and the load on the computing resources will not become too high.

In principle the selection of nodes via the value of $F_{vx}$ corresponds with the principle of the DOI function in the Filter FEV (Section 2.9.1.2).

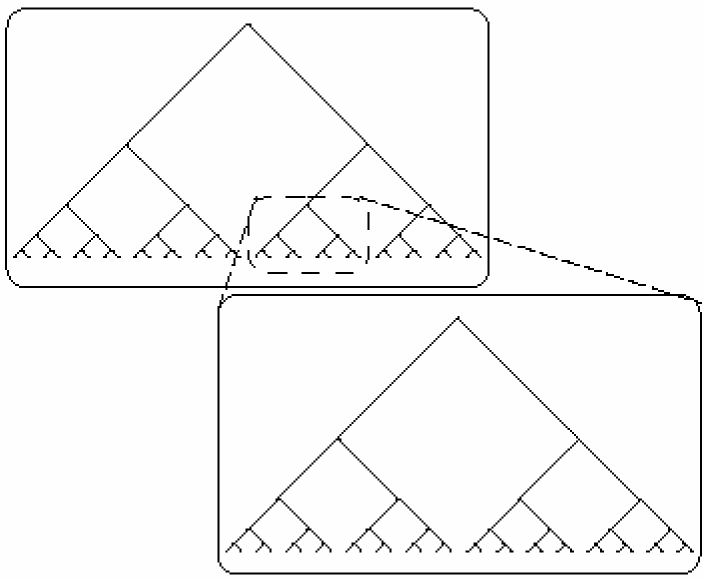


**FIGURE 12** The concept of the fractal view layout

#### 2.9.1.8 WebTOC

WebTOC[9] [Nation,1997] is a tool developed at the University of Maryland Human-Computer Interaction Laboratory (HCIL) that displays a hierarchy (e.g the structure of a WWW server) similar to the table of contents in a book as a 1D list. By clicking with the mouse the user can unfold or hide areas of the hierarchy. WebTOC displays attributes, such as number of files in a directory or memory usage, as color coded bars.

---

8 http://130.153.158.203/~koike/papers/vl93/vl93.pdf

9 http://www.cs.umd.edu/hcil/webtoc/

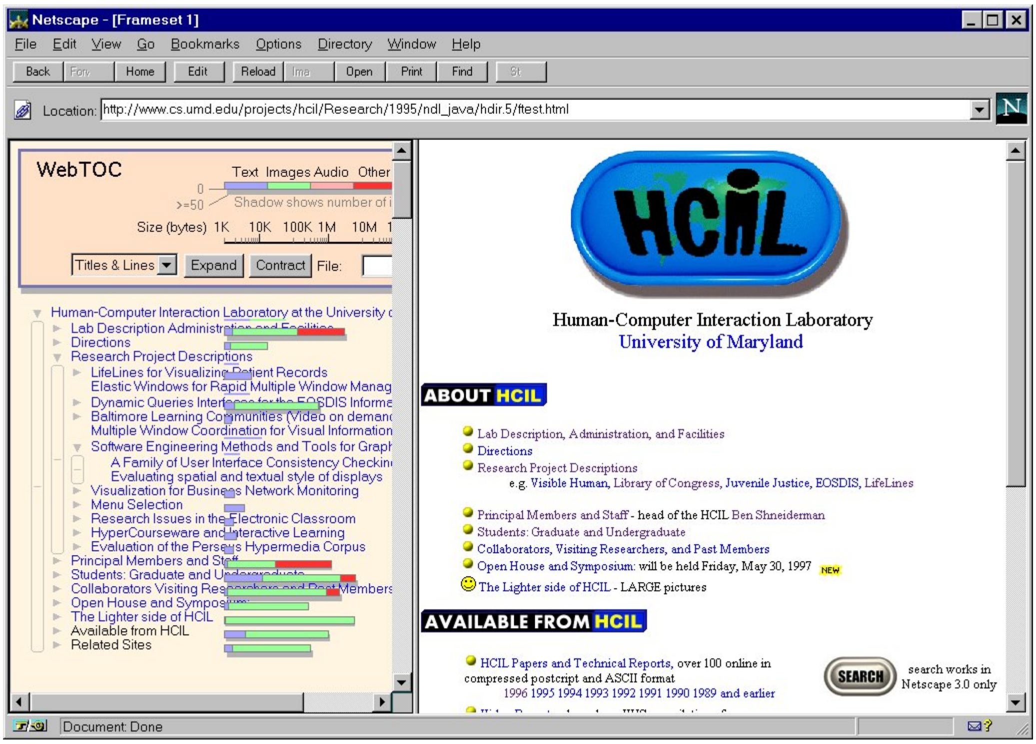



**FIGURE 13** The WebTOC display

Figure 13 (Source: WebTOC Homepage) depicts a WebTOC Java-Applet in the left frame of a browser. WebTOC is used for navigation while the right frame displays the selected contents.

Each document is represented by a colored bar with its length corresponding to the size of the file. The color of the line represents the type of file (e.g. HTML, image, video). Links to other documents can be collapsed into bigger bars that display the total size of the document subtree it references. The size of the shadow under a bar indicates the number of items subordinate in the document hierarchy. This allows a user to easily distinguish between items with few or many links.

The number of information objects displayable with WebTOC is estimated to 10000 by the developers.

### 2.9.1.9 Mapdisplay

A good example for the application of algorithms for self-organizing information spaces are so-called Mapdisplays[10], as developed in 1997 at the University of Kentucky. The algorithm for the generation of a document map as shown in Figure 14 is partially implemented in Java. Therefore, multiple parameters can be manipulated interactively by the user on a client machine, such as number of documents and number of terms on a map. With this kind of information visualization some interesting semantic relations can be displayed:

- Coherent areas can be interpreted as concept areas, i.e., adjacent topics can therefore be grouped to form topic groups.
- The size of the term areas directly correlate with the frequency of occurrence of terms.

10 http://www.uky.edu/~xlin/asis95.htm

- Adjacent areas are often an indicator for a semantic correlation.

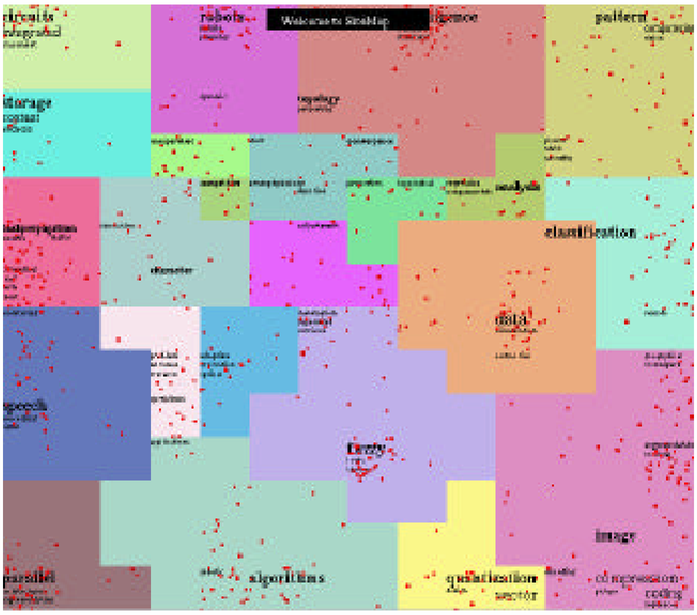



**FIGURE 14** A document map using the Mapdisplay

The above figure displays a set of documents which represent the hits of a database query to the INSPEC database. The documents have been structured and visualized within a semantic map. The size of the term areas correlates with the frequency of occurrence of the given term within the global document space. The dots represent single documents, which are assigned to certain term areas.

#### 2.9.1.10 Newsmap

In connection with the MVAB-Project (Multidimensional Visualization and Advanced Browsing) the scientists of the Pacific Northwest Laboratories (PNL) developed new technologies for the visual exploration of extreme large (>20000) document sets during the mid of the 90's. As a result the visualization tools Galaxy and ThemeScape, also summarized under the name SPIRE (Spatial Paradigm for Information Retrieval and Exploration) .The metaphors of starfield simulation and topographic 3D-map were chosen to visualize data, because they embody a multitude of effective cognitive features. Star field simulations generate dot patterns, which can be comprehended easily and intuitively. Topographic maps simulating mountains and valleys can also be associated with everyday experience. The latest commercial product spawned from these developments is a graphical user interface from the company Cartia, which generates document maps in the style of geographic height profiles (Figure 15).


**FIGURE 15** The Newsmap News Browser

The figure displays the Newsmap news browsers visual interface for finding news. Single news items are categorized and then mapped as topic areas into the document map. In this display mountains represent higher concentrations of documents within a given topic. Height lines, furthermore, connect certain interrelated term domains.

### 2.9.2 Visualization techniques in 3D space

Nearly every technique in the 2D plane can be transferred to 3D space. This, however, is not true vice versa. For example, [Koike,1993] describes the application of fractal views in 3D space. In this section we present techniques which are specially designed for use in 3D space.

#### 2.9.2.1 Document Lens

In the beginning of the 90's a whole bunch of new visual metaphors for information spaces have been developed at XEROX PARC (Palo Alto Research Center) summarized in [Robertson,1993]. One of the first metaphors to gain popularity was the so-called Document Lens (Figure 16).

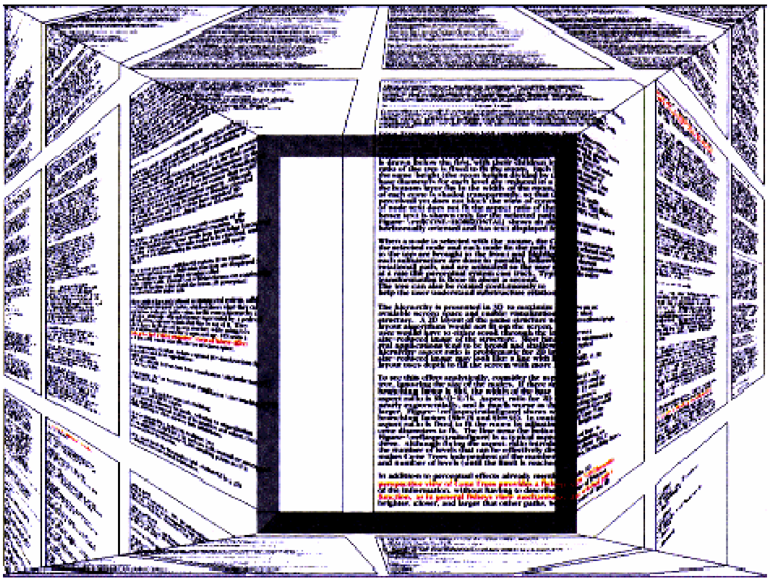


**FIGURE 16** The Document Lens from XEROX PARC

The document lens on the figure displays a text document containing of multiple text pages. The complete text can be interactively viewed through the interactively shiftable view window (framed black), where it is not distorted in perspective and therefore readable. The document lens has the purpose of displaying single text fragments in the context of a bigger documentation.

#### 2.9.2.2 Cone Tree

The geometric form of the cone plays a central role in the Cone Tree technique presented in [George,1991] and [George,1994,2]. A cone tree is a 3 dimensional graphical representation of a tree structure, in which the root node (e.g. displayed as a cube or a sphere) is positioned at the top of a semi-transparent cone. The children of the root node are located around the circular base of the cone. Each child can in turn be the root node of a sub-tree, which is recursively also displayed as a cone and whose root object is at the same time the child object of its father node. Figure 17 explains the process of generation of a cone tree.

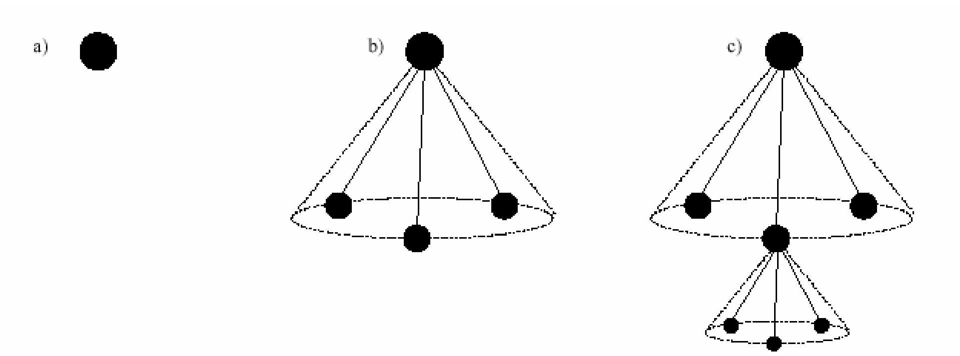



**FIGURE 17** Cone Tree principle: a) root node; b) child nodes; c) 1st recursion level

In [Najork,1995] the basic algorithm for generating cone trees has been written down:

```
let rec ConeTree =
  proc (tree)
    let r = GroupGO_New();
    let node = SphereGO_New([0,0.5,0],0.1);
    r.add(node);
    SurfaceGO_SetColor(node,color_hsv(real_float(tree[0]) * 0.01,1.0,1.0));
    if #(tree) > 1 then
      let cone = ConeGO_New([0,0,0],[0,0.5,0],0.25);
      r.add(cone);
      SurfaceGO_SetTransmissionCoeff(cone,0.8);
      SurfaceGO_SetColor(cone,"gray");
    end;
    for i = 1 to #(tree)-1 do
      let angle = 2.0 * math_pi * real_float(i) / real_float(#(tree) - 1);
      let ctr = [math_sin(angle)*0.25, 0, math_cos(angle)*0.25];
      let edge = LineGO_New(ctr,[0,0.5,0]);
      r.add(edge);
      LineGO_SetColor(edge,"gray");
      let child = ConeTree(tree[i]);
      r.add(child);
      GO_SetTransform(child,Matrix4_Translate(Matrix4_Scale(Matrix4_Id,
                         0.5,0.5,0.5),ctr[0],-0.25,ctr[2]));
    end;
    r;
  end;
```

Figure 18 displays the result of applying the function ConeTree on a given tree structure.

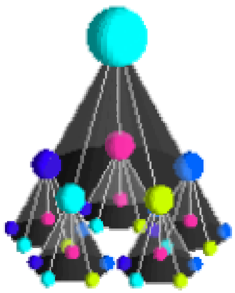


**FIGURE 18** Cone Tree display of a given tree structure.

The function ConeTree is invoked with an array representing the tree structure as an input parameter and outputs the graphical display of the tree structure as a cone tree. The sphere (SpereG0_new) represents the root node, which is color coded depending on the ID-value (SurfaceG0_SetColor). If a node has children a transparent cone is inserted (ConeG0_New) whose tip is positioned within the newest generated sphere. At last the algorithm iterates through all sub-trees in order to calculate and apply an even distribution across the cone base. The function ConeTree is called recursively for each child node and the corresponding sub-tree is uniformly scaled by a factor of 0,5 along the Cartesian coordinate axes.

The presented algorithm is quite restricted to the display of balanced trees with a relatively small number of child nodes per node. The Cone Tree technique presented by Robertson et al. is the basis for many other research efforts, which mainly aim at refining and improving the technique [Tversky,1993] [Najork,1995] [Carriere,1995].

### 2.9.2.3 Perspective Wall

One of the most popular developments from XEROX PARC is the Perspective Wall (Figure 19). The Perspective Wall represents a sequential display of files along a timeline. This visualization technique uses the room metaphor to display files created at a certain time in the foreground, while older files and newer files can be seen in an overview fashion. The wall can be interactively moved by a user bringing other time slices to the foreground.

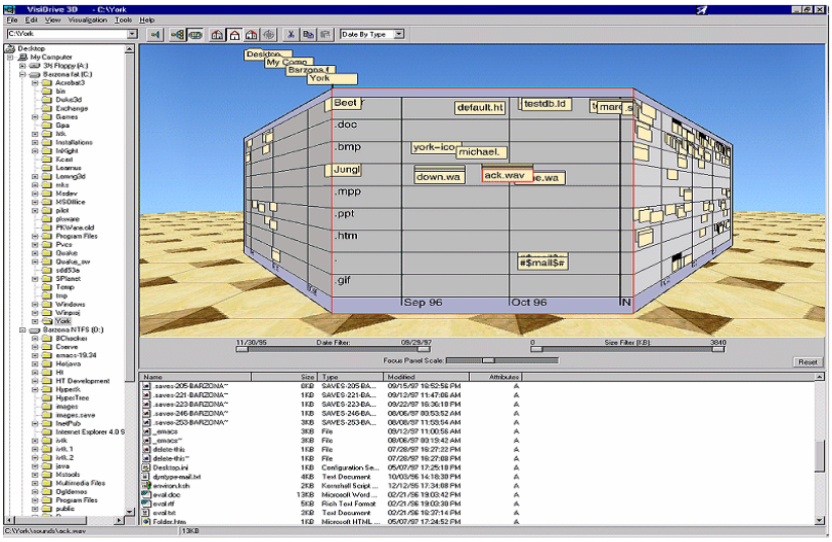



**FIGURE 19** The Perspective Wall

#### 2.9.2.4 Hyperspace

The visualization system Hyperspace [Wood,1995] has been developed in 1995 at the School of Computer Science at the University of Birmingham. The system allows displaying the hyperlink structure of parts of the Web. The layout of the web pages within the information space is calculated in such a way that related documents (i.e., web documents connected through hyperlinks) are positioned near to each other in 3D space. Other documents, which are not connected via hyperlinks, are placed far away from each other. Each web page is represented by a sphere with its size depending on the number of hyperlinks outgoing from the page. The hyperlinks are displayed as simple lines between spheres.

The overall structure is randomly distributed over a 3D space. With the help of non-linear optimization methods the objects are able to self-organize within the given space. While spheres repel each other, links will apply an attraction force. After applying the iterative algorithm to the information objects, interesting spatial structures will show up. In Figure 20 a clear separation of index files (web sites root nodes) can be seen, which look similar to pincushions.

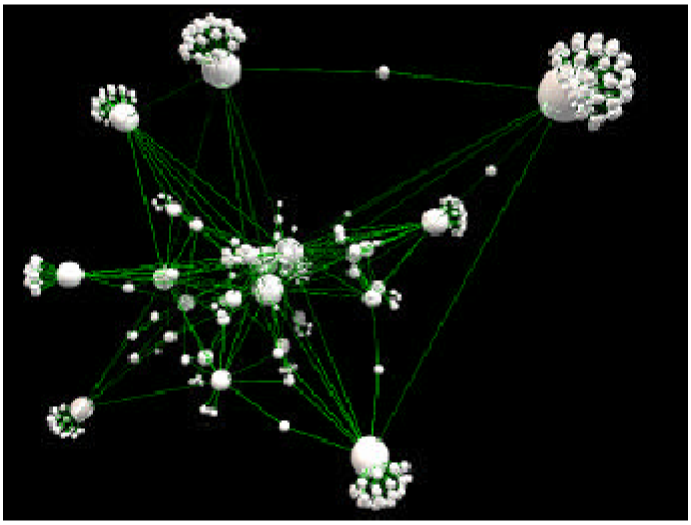


**FIGURE 20** A self-organizing information space with Hyperspace

#### 2.9.2.5 Hotsauce

In order to explore web sites and their link structure visually, a Web-browser plug-in with the name Hotsauce (a.k.a. ProjectX) from the Apple Computer Corporation (now to be found at http://www.xspace.net/hotsauce/) can be used (Figure 21). Hotsauce is an information and navigation system in 3D, which supports the interactive exploration of web contents. The user interface is based on a proprietary exchange format called the Meta Content Format (MCF) for the description of structured content.

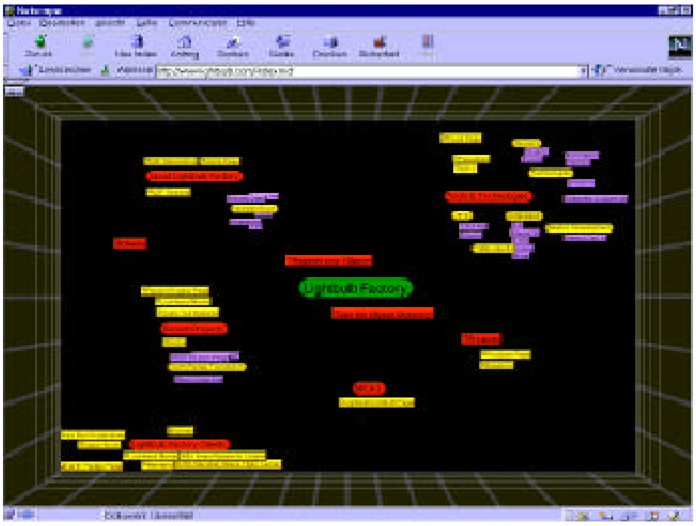


**FIGURE 21** The Hotsauce Browser

Through a simple fly-through navigation the user can penetrate deeper and deeper into the hyperlink schema. While zooming new subcategories become visible whereas other categories disappear from the view.

#### 2.9.2.6 LyberWorld

The 1994 „LyberWorld“ application [Hemmje,1994] was one of the early approaches to deal with the mapping of information to three-dimensional metaphors. Intended to express the essential characteristics of the application in one word, „Lyber“ was formed from the English words "library" and "cyberspace" (virtual, computer-generated space). This implies that the tool can be understood as a virtual computer-assisted library. The LyberWorld application consists of three different visualization metaphors:

- context tree (LyberTree)
- relevance sphere (LyberSphere)
- document room (LyberRoom)

The application makes it possible for users to formulate a database query to an underlying information retrieval system and to evaluate the results of the query visually. If a user enters a search concept a corresponding query is generated to the database. As a result the system produces a set of documents related probabilistically to the concept that was entered.

The LyberTree tool indicates existing interrelationships. The titles of the documents found are visualized as small elongated platelets running along the z axis on a cylindrical surface (see Figure 5). The root of the first document cylinder is the search concept. The connection between the document titles and the search concept is represented visually by a circular cone, the tip of which is on the word and the bottom of which is on the document roll. This indicates the interrelationship between two levels. The user can examine the set of located documents more closely by turning the roll in space to look at the titles that are on the back of the cylinder. The user can select a single relevant document platelet and then expand the title in question. The selected platelet is always displayed directly toward the user and is highlighted. The expansion of a document title creates a so-called concept level, containing all the concepts in the database that are related to the document. The concept levels are represented by platelets and rolls, analogous to the document levels, except that they are a different color to indicate the type of roll in question. In any given session users can expand as many concept and document levels in the database as they want to, expanding the context tree in the process. They can explore those areas of the database that appear important to them and systematically expand the set of desired results.

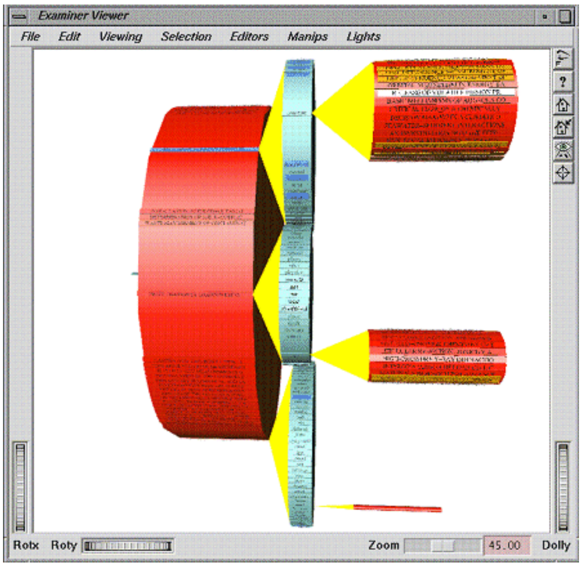



**FIGURE 22** The LyberTree

In order to be able to further evaluate the results derived from concept searches and related documents following the described context tree procedure LyberWorld offers a further visualization metaphor, the LyberSphere or "relevance sphere" (see Figure 23). The individual search concepts corresponding to the text platelets on the rolls at the concept level are now represented as small spheres (reference spheres) on the surface of the relevance spheres and define the reference points. The documents that correspond exactly to the platelets on the document cylinders of the context tree are inside the sphere and visualized as small book symbols. In this visualization process relevance values, each reflecting the strength of the relationship between a concept and a related document, are mapped to the reference sphere's strength of attraction. A reference sphere representing a search concept can be seen as a kind of magnet. Its size reflects the strength with which it is able to attract the documents in the overall sphere. Initially the reference spheres of the search concepts are spread evenly over the surface of the relevance sphere, but can be moved by the user on that surface to bring about a conceptually grouped arrangement of document symbols inside the sphere. As a further interactive option the user can change the strength of the magnetic attraction exerted by reference spheres on documents. The user can also determine levels of concentration in the relevance sphere, a factor that changes the magnetic strength of the overall relevance sphere surface. As a result of this it is possible to avoid strong concentrations of document symbols and to keep a clear picture of the documents at hand.

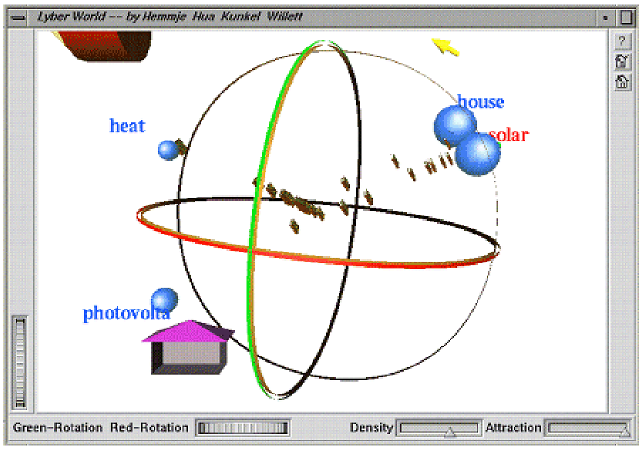
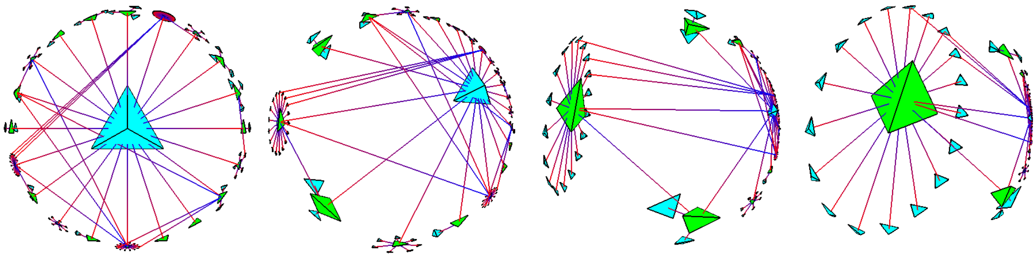



**FIGURE 23** The LyberSphere

The third and last visual tool from LyberWorld is the document room. This metaphor can be activated simultaneously with the context tree or the relevance sphere. The document selected in the context tree or the relevance sphere is projected onto the rear wall of the document room displayed in the background. The user can use the relevance sphere to study the content of individual documents during the search phase in the context tree or during the segmentation phase.

### 2.9.2.7 Hyperbolic Space

Since cone trees are displayed in Euclidian space the graphical object grows in size with the size of the information space to display. For larger information spaces a display in hyperbolic space has been developed. The 3D hyperbolic space [Munzner,1995] has makes it possible to display the whole information structure as well as interesting details at the same time. Furthermore, the hyperbolic isometry allows a mathematically elegant navigation. Figure 24 displays a navigation in hyperbolic space.



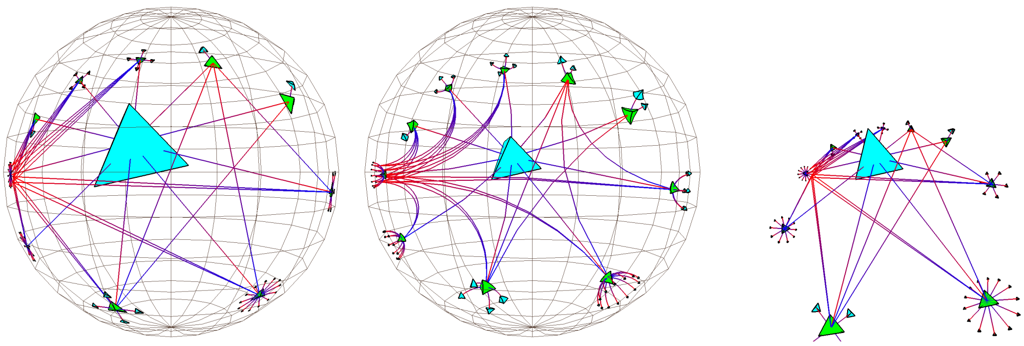
(a) (b) (c) (d)

**FIGURE 24** Interaction in hyperbolic space

The center triangle in (a) is dragged to the right (b). The left triangle in (c) is dragged to the center (d).

For displaying hyperbolic space mainly three different models can be used:

- **Projective or Klein Model:** This display can be imagined as projected to the inner surface of a sphere. This technique is well suited for display with modern workstations. (Figure 25a)
- **Conformal or Poincar Model:** This is based on the projective model. The lines are displayed as arcs and the surfaces as parts of the spherical surface. The cost for displaying with this technique are much higher than with the projective model. (Figure 25b )
- **Insider Model:** In this model the camera is inside the sphere and cannot escape it. (Figure 25c)



(a) (b) (c)

**FIGURE 25** The 3 models for displaying hyperbolic space

### 2.9.2.8 Information Cube

The Information Cube is a 3D visualization technique presented in [Rekimoto, 1993]. This technique uses a box-in-a-box metaphor, which is probably well known to most of us and using it should be intuitive. With the information cube all information objects are represented as a set of cubes, which are fully nested into each other, whereas the cubes surfaces are displayed semi-transparent. The root (Level 0) is displayed by the outermost cube. The information objects of the next hierarchy level are contained within this cube represented as smaller cubes. This rule is recursively applied across all hierarchy levels.


**FIGURE 26** The Information Cube

Figure 26 (http://www.csl.sony.co.jp/person/rekimoto/cube/) displays an Information Cube display. A cube representing a leaf node of the tree can also embed other 3D-objects such as, e.g., chemical molecules.

With the use of semi-transparency 2 goals are achieved: On the one hand it allows the user to see through the cube in order to recognize the box contents. On the other hand by summing up the transparency values of the respective cube surfaces the actually visible part of the hierarchy is decreased in order to limit the complexity of the displayed image. Therefore, through the use of semi-transparency an appropriate amount of information is displayed without hiding the information objects. The transparency value can be furthermore used to encode attributes, such as a certain color value mixed with a transparency value for the relevance (DOI) of an information object.

In this technique the display is strongly coupled with the user interaction. One of these interactions is rotating the selected cube around its axes. However, the more interesting interaction is the movement through the hierarchy, which is achieved through selecting a cube and triggering the action "Walk" to go into the cube. This action is graphically represented by a translation and a following scaling, such that the selected cube spans the complete viewport as initially the root cube did. The Information Cube is a visualization technique suited for complex hierarchies. In [Rekimoto, 1993] a prototypical implementation with 1500 information nodes was presented.

### 2.9.2.9 Landscape

The Landscape visualization technique uses the landscape metaphor. This technique was presented in 1992 by SGI with the FSN system[11] [Tesler,1992] and became famous when featured in the movie Jurassic Park.

fsn (pronounced “fusion”) is a file system navigator in cyberspace. It lays out the directories in a hierarchy with each directory represented by a pedestal. The height of the pedestal is proportional to the size of the files in the directory. The directories are connected by wires, on which it is possible to travel. On top of each directory are boxes representing individual files. The height of the box represents the size of the file, while the color represents the age.

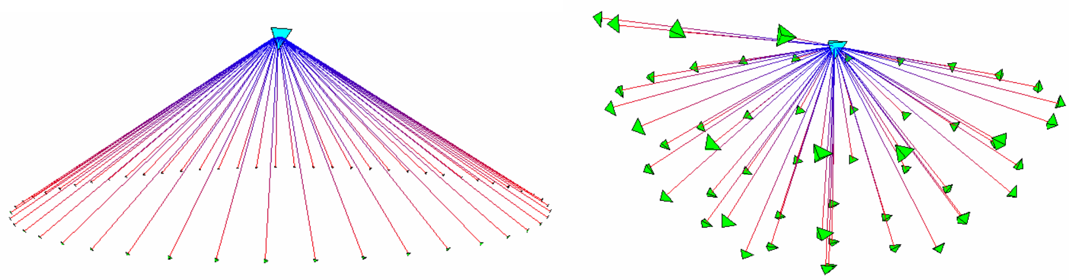
[11] http://www.sgi.com/fun/freeware/3d_navigator.html


**FIGURE 27** File System Navigator from SGI

The Landscape metaphor was later integrated into the Hyper-G client Harmony [Andrews,1995]. In [Andrews,1996,2] an extension of the landscape display is described, in which additional levels are introduced to be able to better display relations of the hierarchy.

### 2.9.2.10 H3

The H3 layout technique [Munzner,1997] was especially developed for large hierarchies. The display uses hyperbolic space. H3 is based on the Cone Tree technique, which was modified, such that the child nodes of a given node are not placed around the circular edge of the cones base (Figure 28 (a) ) but rather on a hemisphere sitting on the circular base surface of the cone (Figure 28 (b) ).



(a) (b)

**FIGURE 28** Navigation in H3

The hyperbolic navigation offers focus+context with a high degree of graphical overview. With this technique it is possible to display more than 20000 hierarchically structured information objects. Figure 29[12] displays a sequence of movement in an information space displayed using H3, in which a node is dragged from right to left. The H3 homepage also contains video sequences that give a much clearer impression.

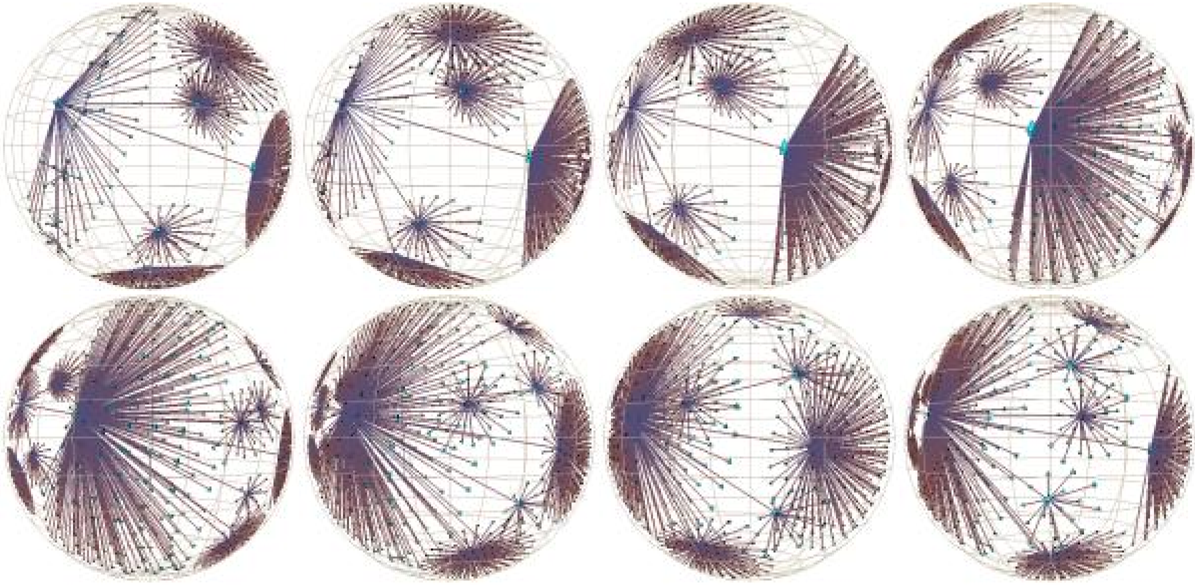


**FIGURE 29** A navigation sequence in H3

### 2.9.2.11 Document Finder

Searching information on the Web often brings up the problem that a key word based search returns an unstructured HTML result list with a large number of entries. In this case the Document Finder (FH Potsdam, Däßler and Palm, 1998) offers support. The Document finder structures the hit list of a search query in real time and dynamically generates a 3D information landscape. In the current version the Document Finder realizes a visual database interface to the bibliographic database Infodata, which is a quite large literature database covering information sciences and related topics. The database has more than 80000 records. Figure 30[13] displays the retrieval interface with an input mask, a retrieval path, and a virtual 3D information space for the interactive exploration of the hit documents.

[12] http://www-graphics.stanford.edu/papers/h3/

[13] http://fabdp.fh-potsdam.de/infoviz/projects.html

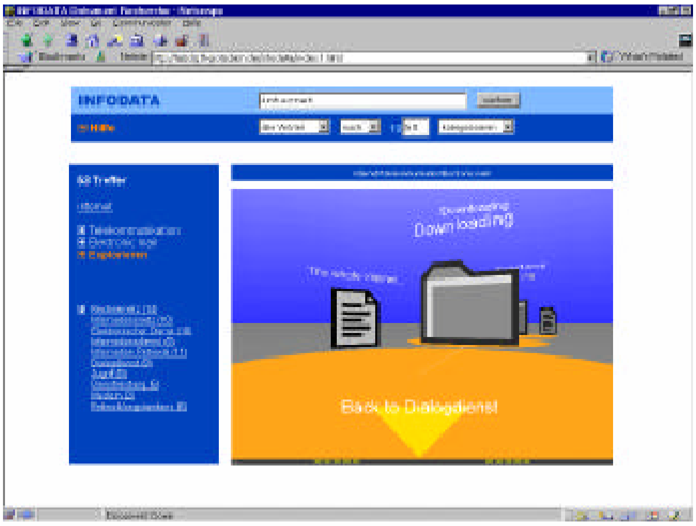


**FIGURE 30** The interface of Document Finder

Within the retrieval interface of the Document Finder documents can be browsed either txt based or visually explorative. The information space shows hierarchically positioned document clusters as a result of an automatic structuring of a search queries set of hit documents.

### 2.9.3 Other and combined techniques

It is definitely worth to note, that all of the above techniques are today classic information visualization techniques, which are not only often referenced in numerous publications but also used to derive other techniques. Besides the described techniques there are innumerable other techniques, which are mostly a combination or some variation, refinement, or improvement to the above ones. For example, nearly every visualization technique can be combined with the Filter FEV technique to offer additional focus+context ability to the user.

It can be informally derived that information visualization includes the display of the information objects themselves as well as the display of the structure of the information space. Both tasks can be supported by nearly any visualization technique, which in the future will be catalyzed further by ongoing improvements in the area of multimedia hardware and software.

A good example of a combined visualization technique is an application from the company inXight depicted in Figure 31, which visualizes a hierarchically structured information space using a hyperbolic browser and thumbnail images as nodes

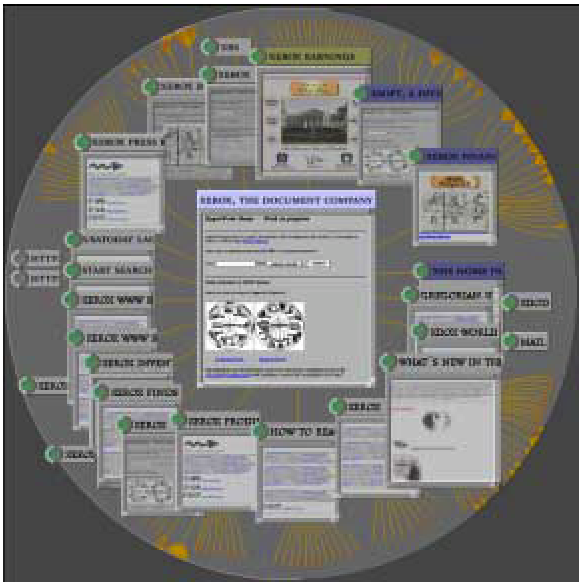


**FIGURE 31** Hyperbolic Browser with thumbnail images as nodes

Some of the numerous other techniques, which had a significant early impact on the field of information visualization but could not be mentioned here are:

SemNet [Fairchild,1988], which was developed by Kim Fairchild in 1988 and was one of the first visualization systems for information in the history of this young science branch. It spawned out of research going on in the mid of the 80's at the University of Singapore aiming at using new VR techniques to build virtual information spaces, in which a user could freely navigate through a network of abstract objects.

MineSet [Brunk,1997] implemented the visual metaphor of the information landscape in 1995 and was the follow up project of SGI's FSN.

Trend setting for new ways in the design of user interfaces to search the Internet was clearly the visual user interface for the Internet-Search service Gopher, called GopherVR [McCahill,1994]. It was developed in 1991 by the Gopher team of the University of Minnesota.

The KOAN system [Kreuseler,1998], which stands for context analyzer, was developed as a joint project between Siemens AG and the CG department of the university of Rostock, Germany. It visualizes arbitrary information objects, their attributes, and their context in a 3D graph layout.

Jim Foley's Navigational View Builder [Mukherjea,1995,1] is a tool which enables a user to create visualizations of a given information space in an interactive manner using the techniques of binding, clustering, filtering, and hierarchy building for structural and content analysis.

PerspectaView (Perspecta Inc., PerspectaView Software. http://www.perspecta.com) is a Java based 3D typographic map developed by Apple Inc. based on work from the Visual Language Workshop at the MIT Media Lab. It uses 3D fly-through text labels similar to Hotsauce, but adds links between the nodes. The graph is laid out in 3D space and can be rotated in a similar manner as cone trees. PerspectaView turned out to be more interesting with respect to information mining from non-Web sources such as Usenet groups.

The SPIRE Toolkit [Miller,1997] (which stands for Spatial Paradigm for Information Retrieval and Exploration) is a set of tools for exploring information, including query, subset, and trend analysis tools. It allows users to rapidly discover hidden information relationships by reading pertinent documents from data in many areas, including industrial competitiveness, strategic planning and medical research. SPIRE was recognized with a 1996 R&D 100 Award.

A good overview of the significant developments and tools in the field of information visualization is given by Däßler in [Däßler,1999] (Figure 32).

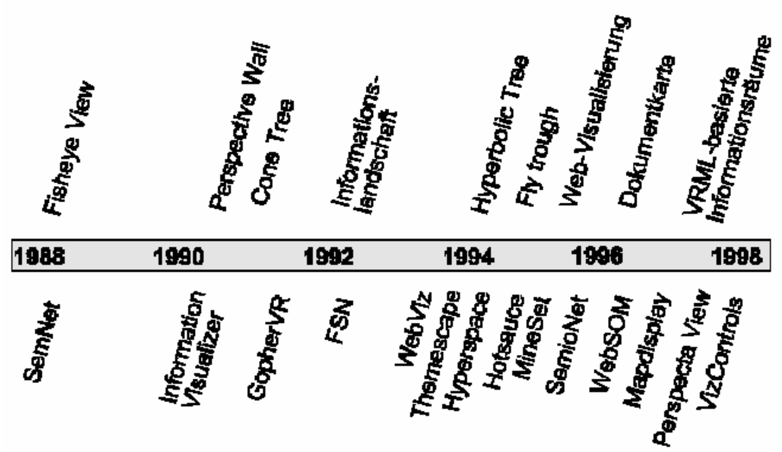



**FIGURE 32** Selected techniques from the field of information visualization [DÄßLER,1999]

## 2.10 Critics on Information Visualization

In the German language publication "Informationsvisualisierung: Stand, Kritik, Perspektiven" [Däßler,1999] Professor Däßler presents an insightful wrap up of the main problems and critical issues of information visualization. He observes that after a phase of hype information visualization is now looked upon with certain skepticism with respect to the goals and benefits of recent developments in this area. Although there is a large number of information visualization systems and prototypes as well as scientific projects, the number of visual metaphors used in commercial systems and so-called real-world-applications is relatively low. One of the reasons for this might be that the design of such systems is not user centered but rather driven by technology in the areas of visualization and computer graphics. The real problem of visualizing data with the goal to offer an added information value has been neglected too often. Furthermore, most systems haven't been evaluated by the users of the target group with respect to usability criteria and acceptance.

Däßler further summarizes the goals of information visualization, which were formulated in the mid 90's, around the following 3 focal points:

- The visual administration and visual research in large document collections.
- The development of inquiry interfaces for large document collections, online databases, and Internet search engines as a measure to cope with the ongoing information flooding.
- The visualization of complex data structures from relational or object oriented online databases (taking into account that approx. 80% of today's web applications are database driven).

Comparing these goals with the actual state-of-the-art in the field of information visualization, Däßler states that approximately 90% of all information visualization systems are simply hierarchical classification schemas using various visual metaphors. Most systems presented in the field are furthermore static and merely support an active user role, in which the user is forced to navigate through the information space on his own, leaving the added-value of 3D user interfaces ambiguous. Only very few approaches even have been adopted for real applications (e.g., Hotsauce, Hyperbolic Tree) even though there are numerous approaches for hierarchy visualization. The Windows Explorer style navigation is still used in many applications in contrast to other visual metaphors for hierarchies such as Hyperbolic Tree or Treemap. All attempts of establishing visual user interfaces for online search engines and catalogs (Exite3D, AltaVista-Refine, Yahoo3D) have failed. Däßler observes that since there is still no alternative to text display in applications to visualize content and semantic aspects. He directly questions the use of abstract geometric primitives, such as sphere or cube for information visualization.

On the basis of some experience with past and ongoing developments in the field, Däßler categorizes the main problems of information visualization to lye in the following areas:

- **Preparation and structuring of information**
  The process of mere visualization cannot enhance the quality of the data structure and data classification, but that in turn visualization often uncovers poor information structuring. Therefore, information visualization can only be as good as the underlying information structure. Tree-like information structures can be visualized effectively, whereas more complex structures impose severe problems on visual tools, especially when text documents are involved. In conclusion, a good information structure is a key factor for a successful visualization technique.

- **Visualization of large information sets**
  One of the major problems in information visualization is coping with large information sets. Well-known techniques such as scatter plot, information landscapes, or semantic maps, can give a good overview but fail when the goal is to drill-down to detailed information. Many examples have, e.g., shown that information visualization using semantic networks fails as soon as a certain amount of data is exceeded (e.g., network diagrams on large web sites).

- **Visual description of contents**
  Text documents are still encountered often in information visualization. However, graphical objects and symbols are in most cases not able to handle this complexity and abstract objects are not self-describing at all. Therefore, it is still necessary to display text, which confronts users with the well-known problem of reading text in a graphical display. Even simple text lists are often better readable. One approach is to use the graphical properties of text (e.g., size, color, transparency, etc.) to represent information, making the "graphical" text more self-explaining (e.g., Elastic Catalog from MIT [Murtaugh,1996]). Däßler also harshly criticizes the habit of using the adage that "a picture says more than 1000 words" to legitimize the use of visual concepts by observing that the set of known icons and pictograms is by no means able to encode text documents, which makes it hard to display textual terms as abstract graphical objects. Moreover, the relations between textual terms are highly complex semantic networks, which cannot be simply expressed through the well-known graphical associations between objects, such as connection lines.

- **2D versus 3D**
  Two main ideas for the use of 3D techniques can be identified in the early information visualization systems of the 90's. One is to increase the displayable information space through the use of spatial depth. The other is to use 3D navigation techniques (such as fly-through) to explore the information space in an intuitive manner, similar to a flight simulator. These approaches were often believed to be the next generation user interfaces. For 3D information spaces reality has shown that the use of a third spatial dimension often adds more problems than it can solve. Moreover, without the use of special 3D viewing devices (e.g., 3D goggles) it is hard to perceive spatial distances. This is a huge problem since many 3D infor-

mation visualization techniques use spatial distance of 3D objects as an expression of semantic similarity.

For 3D navigation it is still unclear if spatial navigation is better suited to the human experience, which would mean that we could better orient ourselves in 3D space than on a 2D map. In this way it would be easier to detect interesting information clusters laid out in 3D space. The thesis that humans have better orientation in 3D space because our every-day perception and experience also takes place in three dimensions is critically discussed in the thought provoking article "2D is better than 3D" written by Jakob Nielsen in the Nov. 1998 Alertbox (http://www.useit.com/alertbox/981115.html).

Nielsen claims that "2D is better than 3D because people are not frogs" and means that humans having the eyes in the front of their heads (unlike frogs) are optimized for wandering around on a 2D plane (the savannah) and "not swinging through trees". He underpins this statement by observing that the number of humans driving cars (2D orientation) is huge compared to the number of humans flying through the air with aircrafts (3D orientation). Furthermore the use of 3D navigation in computer applications imposes the problem that the current input and output devices (such as CRT and mouse) are of 2D nature making it difficult to control 3D applications with them. Additional problems include the increased cognitive load on users navigating in 3D, which distracts them from their primary task, a poor screen resolution making far-away objects (especially text) unrecognizable, and non-standard software forcing users to additional downloads.

Nielsen then provides scenarios for especially "bad use of 3D". These include "most abstract information spaces" of high dimensionality with 3D only taking away one more dimension than 2D, "navigation through hyperspaces" being too confusing for end users, and "virtual reality gimmicks" simulating the real world (e.g., 3D shopping malls) which are "putting the interface in the way of [the users] goal".

Nielsen then derives some very general rules "when to use 3D". He enumerates the "visualization of physical objects that need to be understood in their solid form", "abstract data sets that have exactly three attributes", and "entertainment applications" like 3D video games. Under http://www.useit.com/alertbox/981115_comments.html Nielsen has collected various comments and discussions on the 2D-vs-3D issue in response to his article. In one of Nielsens answer to a readers comment he explicitly warns "against using 3D except in the cases where the third dimension adds substantial value to the user's task".

- **Evaluation and real time visualization**
  One of the major remaining problems of human computer interaction is that user acceptance and user requirements are too often neglected. The progress in this area, however, can only be measured by broad acceptance and usage of visual metaphors in real application scenarios. Since the quality of visual tools for information seeking is mostly compared to the speed of Internet search engines, response time is a major criterion. However, a short response time can still only be achieved by using relatively simple algorithms for layout calculations.

In his conclusion Däßler states that information visualization has still not accomplished the main requirements of providing better ways for analyzing and interpreting mass data and of simplifying the complexity of the world, which resides in databases and distributed information resources in computer networks. The use of algorithms and visualization methods on abstract information has failed and a user-centered approach has been almost completely neglected in the design and development process of visual tools. Technological aspects have largely dominated the field.

## 2.10 CRITICS ON INFORMATION VISUALIZATION

Däßler describes the field of information visualization to be currently in a state of reorientation. In any case the success of the whole field is dependent on progress in the area of automatic information processing, filtering, and structuring. Only on the basis of good information structures, visual and analytic tools will have the potential to become successful as widely accepted user interfaces. Däßler sees the future perspectives of information visualization in the fields of data mining and knowledge discovery. Information visualization would therefore no longer be an isolated discipline but rather an integral part of an overall process of complex, interactive information analysis and search.

As a final conclusion Däßler leaves the reader with three, yet unanswered, questions, which all future developers and designers of information visualization applications and tools for the visual exploration of abstract data need to take into account.

- When is information visualization an appropriate instrument to visually represent relations, associations and other abstract concepts within databases or document collections?
- Is automatic structuring of information, which is currently based mainly on statistical methods, good enough to apply meaningful and value-added visual metaphors as known from scientific visualization?
- What is the added information value gained from the exploration of three dimensional information spaces when compared to traditional diagrams, graphics, or images?

Taking Däßler's critical view into account, it can be deduced that broad acceptance of information visualization techniques can only be achieved if a clearly observable added value of using 3D in such applications is given and if the applications using these techniques have common visualization and interaction paradigms.

It is therefore all the more necessary to perform thorough user tests and experiments on realistic application scenarios across the various application domains. However, the development of 3D information visualization applications in connection with data sources still requires a high amount of development work. In order to enable valid and thorough user tests a development technology is required that allows for the rapid development of highly complex visualizations on the basis of arbitrary data sources as well as the possibility to rapidly change the application according to the test results or in conformance with customer requirements.

An IVIS software framework as proposed in this thesis is an approach longing for the fulfillment of these requirements and therefore potentially opening the door for a broad scientific and commercial acceptance of 3D information visualization within everyday applications.