Three-Dimensional Information Visualisation: The Harmony Information Landscape

Diplomarbeit in Telematik

Peter Wolf

Technische Universität Graz
Institut für Informationsverarbeitung
und Computergestützte neue Medien (IICM)

Fertigstellung: 1. 5. 1996
Prüfungsfach: Informationsverarbeitung
Betreuer: Dipl.-Ing. Keith Andrews
Begutachter: O. Univ.-Prof. Dr. Dr. h.c. Hermann A. Maurer

Ich versichere, diese Arbeit selbständig verfaßt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.

Die Diplomarbeit ist in englischer Sprache verfaßt.

Acknowledgments

I would like to thank a number of people at IICM for their support during my work and preparation of this thesis:
Jürgen Schipflinger
Dipl.-Ing. Michael Pichler
Dipl.-Ing. Keith Andrews
O. Univ.-Prof. Dr. Dr. h.c. Hermann A. Maurer

All figures used in this thesis have been prepared by myself except for:
Figure 2.1 by Network Wizards [Wiz96].
Figure 2.2 by Keith Andrews.
Figure 2.3 by Keith Andrews.
Figure 2.5 by Keith Andrews.
Figure 3.1 by Jonathan Meyer [Mey96].
Figure 3.2 by Christopher Ahlberg [Ahl96].
Figure 3.4 by S. Feiner and C. Beshers [FB90].
Figure 3.7 by P. D. Kochevar and L. R. Wanger [KW95].

Abstract

The Harmony Information Landscape is a three-dimensional graphical interface for browsing hypermedia information on a Hyper–G server. It automatically generates a navigable spatial layout that visualises documents on the server, their attributes, hierarchy and hyperlink relationships between them. In the course of this thesis various new features have been added to a first, basic version of the Landscape. This thesis first describes the Landscape's underlying data model defined by Hyper–G, and compares it to other Internet information systems. Then various aspects of spatial information visualisation are discussed. The Landscape's new features and details of their implementation in C++ under the UNIX operating system are introduced. Finally, a user's guide provides a brief overview of the current user interface.

Contents

Chapter 1

Introduction

In 1945 Vannevar Bush published his vision of a 'library of a million volumes [...] compressed into one end of a desk' [Bus45]. In a way this vision came true, since ever growing capacity of mass storage devices and increasing computer performance made it possible to store large amounts of information on a single computer system. The Internet connects millions of computers and computer networks all around the world and thus further multiplies the amount of available information. Today, documents accessible in this world wide and networked database range from bible translations to commercial catalogues, from train timetables to the latest copies of newspapers from all around the world, from weather forecasts to research papers, and from encyclopedias to TV guides.

At the same time the Internet makes this information accessible for a constantly growing number of users who get connected to the net. Due to the variety of available information it is not only computer experts, but also researchers from different fields of science and other people with all kinds of professional and also personal interests who want to use this huge database. What they all need are information systems and interfaces to them, that are on one hand powerful enough to enable useful and effective information access, but on the other hand are as easy and intuitive to use as possible. Because no matter how important the use of those tools and databases may be for their work, in most cases it will be only one segment of this work.

The Harmony Information Landscape described in this thesis is a graphical user interface that is designed to meet those demands. By using spatial metaphors it presents information in an intuitive and compact way. The Landscape is part of Harmony, the Hyper–G client and authoring tool for the X11 windows system on UNIX platforms. Hyper–G in turn is a second generation information system

which, apart from providing clients like Harmony for accessing the information, features a server that supports full-text indexing and structuring of hypermedia information, comprehensive link management, multi-authoring and multilinguality. Moreover Hyper–G is compatible to first generation Internet information systems like Gopher or the World Wide Web.

The concept of the Harmony Information Landscape is to map attributes, hierarchical structures and link relationships of documents stored on Hyper–G servers to icons which are placed in a three-dimensional landscape. Structuring elements like collections and clusters are visualised as platforms or pedestals, and the documents they contain are represented by 3-D icons located on top of those platforms. The platforms themselves are placed on the surface of the landscape according to their position in the collection hierarchy. The icon used for a document can be defined by the user, but generally is intended to be a good metaphor of the document type it represents. For instance a book indicates a text document, while a gramophone represents a sound document, and a camera symbolises a film clip. The age of a document can be indicated by the brightness of its icon, and the size of an icon grows with the size of the document it represents. Above and below selected documents incoming and outgoing links of this document can be visualised. The user can dynamically add documents from the Hyper–G server to, or delete them from the Landscape. By clicking a document's icon in the Landscape, this document can be viewed with a corresponding viewer. The user can interactively move around in the generated information space and is thus encouraged to explore its structure and contents. This kind of exploration is quite effective because many tasks that are necessary to get a good overview of the servers contents are off-loaded from the users' cognitive system onto their perceptual system.

Chapter 2

Internet Information Systems

2.1 The Internet

The Internet [Kro94] is an information network connecting computers and local area networks all around the world. In the early 1980s it was developed from the American military ARPA net. Since then the number of hosts connected to the Internet has approximately doubled every year (Figure 2.1) to about 9.5 million in January 1996 [Wiz96]. All the computers connected to the net use the TCP/IP protocol [Hed87] to communicate. Based on this protocol the Internet provides the following basic services:

- *e-mail* for sending and receiving electronic mail.
- *FTP* for transferring files.
- *news* for discussions of various topics.
- *Telnet* for working on a remote computer.

As the amount of information available via FTP grew, it became nearly impossible to know how and where desired information could be found. So *Archie*, a searchable database containing names and location of all files publicly available via FTP, has been implemented. But information providers and users started to look for other ways of accessing data. Ways that are more comfortable than downloading files using FTP and then finding a matching viewer to actually browse them. This lead to the development of new and more sophisticated information

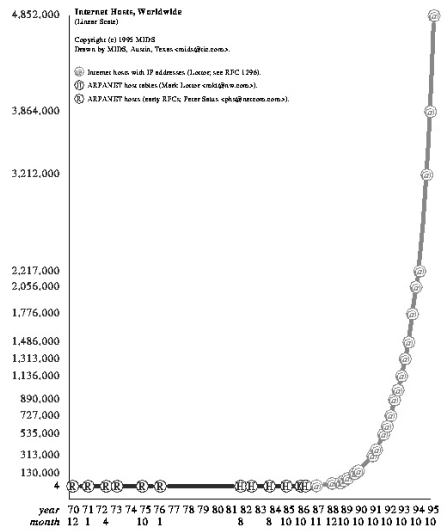

Figure 2.1: The growth of the Internet

systems, which are described in the following sections. They are all based on the client server model, where servers store and manage information and clients are basically tools for accessing and viewing this information. Additionally clients usually provide various aids for finding the desired information. Most of those information systems support hypertext or even hypermedia. So here is a short definition of the meaning of those terms:

Normal text is structured and written sequentially. Thus it can be read from the beginning to the end. Hypertext is structured non-sequentially. If for any word or phrase in a hypertext document further information is available, this part of the text is emphasised and called *source anchor*. A *link* connects the source anchor with a *destination anchor* in a new document which contains the additional information. By activating the source anchor, a reader follows the link and gets to the new document. The reader's location in the new document is determined by the location of the link's destination anchor. In hypertext the reading sequence is determined by the reader and no longer by the author.

Authors of hypertext documents do not only have to write the text, but they also have to define source and destination anchors of the links in their text. Moreover it is important that authors of hypertexts always bear the reader's great degree

of freedom in mind and structure their work accordingly. Authoring hypertext today is usually done by using the Hypertext Markup Language (HTML)[BL96]. A HTML-file contains texts, structuring information and the definition of the links of a hypertext.

Hypermedia documents are structured like hypertexts, but apart from text they can contain various other types of multimedia information like images, movies, sound sequences, etc.

The network of all interlinked hypertext or hypermedia documents is often called *hyperspace*. Basically hyperspace consists of two elements, nodes and links. Nodes are sequences of texts, or in case of *hypermedia* also other multimedia documents, and links connect those documents from source to destination anchors. The problem most commonly described by users of hypertext and hypermedia systems is the *Lost-In-Hyperspace syndrome*. When navigating the hyperspace users may at some point not know:

- their location in the hyperspace.
- where they came from and how to get back there.
- which nodes they already have visited.
- how to find something of which they know that it is there.
- if they have seen all the information that is important for them.
- how not to get sidetracked or how to get back to their main task from sidetracks.

The cause of these problems is very often that readers have no overview of structure and organisation of the documents they are accessing or, even worse, that there is no such structure. So it is very important for hypermedia information systems, both to support the structuring of documents and to provide tools that help the user to get an overview of this structure.

2.2 First Generation Information Systems

2.2.1 WAIS

Wide Area Information Servers (WAIS) enables the user to perform full-text searches of documents stored on WAIS servers. Every WAIS database contains a

full-text index of all documents and their addresses on the server. The starting point of a WAIS search is the main database which contains addresses and keywords of all other WAIS databases. The user searches this main database for a suitable server, connects to this server and then performs full-text searches on it. WAIS has a scoring algorithm and lists found documents sorted by their score.

What WAIS has not got is a mechanism for supporting links from one document to another. So a retrieved document is always isolated and it is not easy to find related documents.

2.2.2 Gopher

Gopher, developed at the University of Minnesota in 1991, uses menus to present available information to the user. For their search users connect to a local server. They then can browse the server's contents via menu trees. Some menu items connect the users to other Gopher servers. By browsing the menus, the users will eventually find the information they were looking for. Or they will not, which is bad luck, because the standard Gopher service does not include any search facilities. In this situation *Veronica*, a Gopher harvester which contains an index of all Gopher menu entries, can help. The problem of this system is that even those menu entries may not be sufficient for efficient searches. Their length is limited to 80 characters and so they may not describe the menus content accurately enough.

2.2.3 World Wide Web

The World Wide Web (WWW, W3) was the first system on the Internet that really supported hypertext and multimedia. It was this fact, together with the fact that the World Wide Web was the first system that could provide a 'flashy' client as an interface to its information, which encouraged even inexperienced computer users to navigate and browse WWW servers. Thus it became, in spite of some considerable disadvantages, the most popular and widespread information system on the Internet.

The only navigational concept for WWW servers is following links. As WWW documents are not structured hierarchically, the user can not systematically browse the contents of a server. To get around this problem, information providers have to create overview and menu pages manually.

WWW also does not support searching for keywords or attributes on one or more servers, because documents are not being automatically indexed and there is no standard for the definition of meta-information like attributes. So *WWW robots* like Alta Vista or Lycos constantly visit servers all around the world and build searchable databases by indexing document titles and special keywords like headlines that they find. However, due to the size of the Web, a single database can never contain a full-text index of all the information available. Moreover, if documents on a server are moved or deleted, those index-databases can of course not be notified and their index can not be updated.

Moving and deleting documents leads to another problem of WWW. On most WWW servers, links of documents are not available separately but are stored embedded in the documents. This makes it impossible to update links when the document they refer to is moved or deleted. Thus moving or deleting WWW documents creates *dangling links* that lead nowhere.

2.3 Hyper–G — A Second Generation Information System

Every information system described so far leaves one or more of the following problems for both information providers and users of the information:

- Systems that only connect documents with links make structuring of documents difficult for information providers, while users in turn get easily disorientated and *lost-in-hyperspace* when browsing unstructured documents.
- Efficient searching of the information space is only possible if a full-text index of all stored documents exists.
- When links point to documents that have been moved or deleted, they point nowhere and become dangling links.
- No information system effectively supports multilinguality.
- It is very difficult or even impossible to handle huge databases and guarantee data and link consistency.
- Clients can only be used for browsing documents but not for authoring. Readers can not even attach annotations for themselves to documents.

Gopher, WAIS and WWW all solved parts of those problems but they all have their weaknesses and leave a lot to be desired. Hyper–G [M+96, AKMSr94, AKM95, Flo95] is a modular, distributed hypermedia information system with an architecture and design that combines the best concepts of other information systems and, in addition, provides some completely new features. This enables Hyper–G to solve all the problems listed above and thus made it the first second generation information system.

2.3.1 Features of Hyper–G

Structuring with Collections and Clusters

Hyper–G's data model (Figure 2.2) is based on two structuring elements: *Collections* and *Clusters*.

Clusters are used to combine different documents in one logical unit. For example if a text and a sound document belong together and should also be presented together, they are put into the same cluster. Clusters are also used for storing the same document in different languages to support multilinguality. When the cluster is selected, the Hyper–G client automatically presents the cluster's information in the language that suits the user's preferences best. The basic item of a Hyper–G database is often a document cluster rather than a single document.

Collections are used for grouping related documents and clusters. A collection can be part of one or more parent collections, which creates the hierarchical information structure on Hyper–G servers. As collections, clusters and single documents can have more than one parent, the collection hierarchy is not a tree structure but a directed acyclic graph.

Identification and Access Control

Similar to the UNIX file system, Hyper–G can control the read and write access to all documents stored on a server. Access rights depend on the users, the group they belong to and their identification level. A user's identification level can either be *anonymous* or *identified*. Anonymous users can read all publicly available documents, but are not allowed to modify documents and they can not see any private documents or links to them. Identified users are known to the system and to all other users. They can have a home collection of their own for keeping private information.

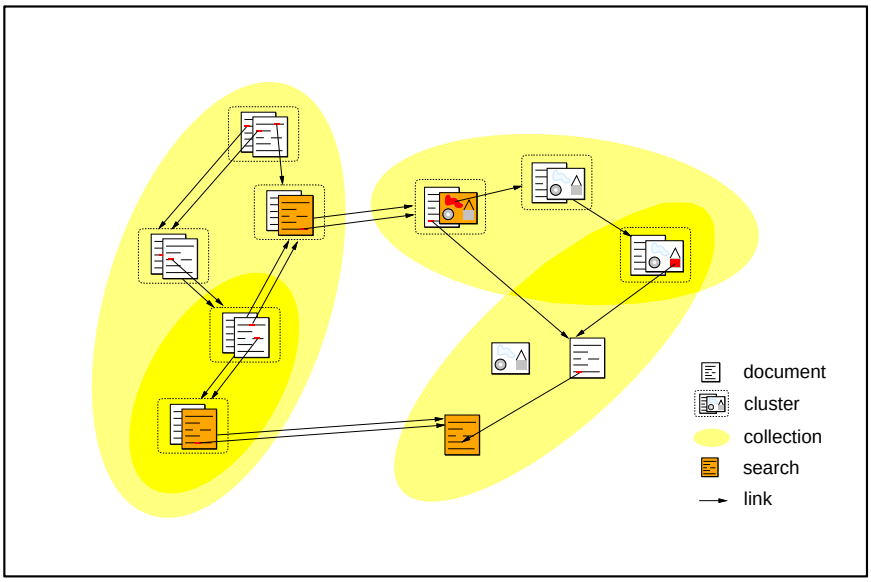

Figure 2.2: The Hyper–G datamodel

Annotations

The user of a Hyper–G client is not only a passive reader, but also can become an author by writing annotations to documents or adding new links between existing documents. These annotations may be made visible only to their author, to a particular user group or to all users.

Multilinguality

Hyper–G users can choose their language preferences. According to these preferences the language of user interface and the presented documents is changed. Multilingual documents are created by using clusters which contain the same document in different languages.

Bidirectional Links and Link Consistency

Links in Hyper–G can not only be followed from source to destination, but also in the reverse direction. This feature, sometimes called *backlink following*, helps

users to find out, which documents are referring to the document they are currently interested in. Hyper–G also performs automatic link maintenance which guarantees link consistency. This means that there are no dangling links and that, when documents are moved, link sources and destinations move with them.

Interoperability

As WWW is the most widespread information system on the Internet and there are still quite a number of Gopher and other servers around, interoperability between Hyper–G and these systems is very important. At the moment Hyper–G clients and servers are able to interact with WWW and Gopher servers and clients respectively. When accessed by a Gopher client, the Hyper–G server maps the collection hierarchy into a Gopher menu tree. When accessed by a WWW client, each level of the collection hierarchy is converted to an HTML document containing a menu of links to its members. Documents of WWW and Gopher servers can be integrated into the Hyper–G collection tree and thus are presented to the user within one large, consistent information space. Access to WAIS and FTP servers is planned for the future.

Search Facilities

Hyper–G offers two methods of searching. One is to search for attributes like titles, keywords, author, or creation time. The other is to use the full-text index that is generated from each document when it is inserted into the Hyper–G server.

Real Hypermedia

Hyper–G supports real hypermedia. This means that it is possible to create links to and from all types of multimedia documents including movies, audio clips, PostScript pages and even documents stored on read only devices like CD-ROM.

2.3.2 The Hyper–G Server

The Hyper–G server consists of three server processes:

- The *document server* stores all local documents and caches remote ones locally. The least recently accessed documents are deleted regularly from the cache. The problem of outdated documents in the cache is solved by assigning an unique object ID to each document. Each version of a document gets a new ID. When a client requests a document, it really requests this ID. So if there is an outdated version of a document in the cache, its ID does not match the one of the new version and the server thus fetches the new version. The old version will no longer be accessed and eventually it will be deleted from the cache.

- The *link server* is a database of all objects (e.g. anchors, collections, documents, etc.) and the relations between them. Those relations are either hierarchical ones, like which document belongs to which collection, or they describe links in terms of connected source and destination anchors. The link server addresses objects in with their unique object ID. It also stores additional attributes like title, author, creation time, keywords, etc. about all objects. This enables the user to browse the structure of an information space without fetching the actual documents from the document server.

- The *full text server* contains a full text index of all text documents on the server. Whenever a new document is stored or deleted or a document's contents are modified, this index gets updated.

During a Hyper–G session a client only talks to a single Hyper–G server. Should the client need information from a remote server, maybe with another protocol, the local server acts as a proxy and fetches the remote information for the client (see Figure 2.3).

2.3.3 Hyper–G Clients

There are three different native Hyper–G clients *hgtv* the terminal viewer, *Amadeus* the client for MS-Windows and *Harmony* the UNIX client for the X11 windows system. The Landscape is a part of Harmony, thus only this client is going to be further described.

The main process of Harmony is the *Session Manager* . It uses a connection-oriented protocol to talk to a single Hyper–G server [KP94]. The Session Manager's central part is a one-dimensional view of the collection hierarchy and the documents in it (Figure 2.4). There is a particular icon for each object type in the hierarchy. Icons of previously viewed documents are ticked. By clicking a collection icon, users can open and close this collection. By clicking a document's

icon they can start a native or external viewer for the document (Figure 2.5). At the moment Harmony viewers for hypertext (including inline images), audio, PostScript, 3-D scenes, images and movies are available. Each of these viewers provides tools for editing, adding and deleting links.

Moreover, the Session Manager contains dialog boxes for setting language preferences, user identification, for searching the database and presenting the results of the search. There is a button that starts a dialog box with a history list of previously visited nodes and another one for a window with a two-dimensional local map that presents an overview of incoming and outgoing hyperlinks, parent-child relationships, annotations, etc. for a selected document. The types of links and the depth of incoming and outgoing links can be determined by the user. In the history list, local map and search dialog, single-clicking an object gives the user location feedback, i.e. paths to the object are opened up in the collection tree and all displayed occurrences of the object are highlighted. Double-clicking any of the documents in those dialogs starts a viewer for the document and inserts it into the collection tree. Finally, the Session Manager contains a button to start the Harmony Information Landscape described from Chapter 4 on. Harmony is implemented in C++ [Str91] using the InterViews [LC+92] X11 user interface toolkit.

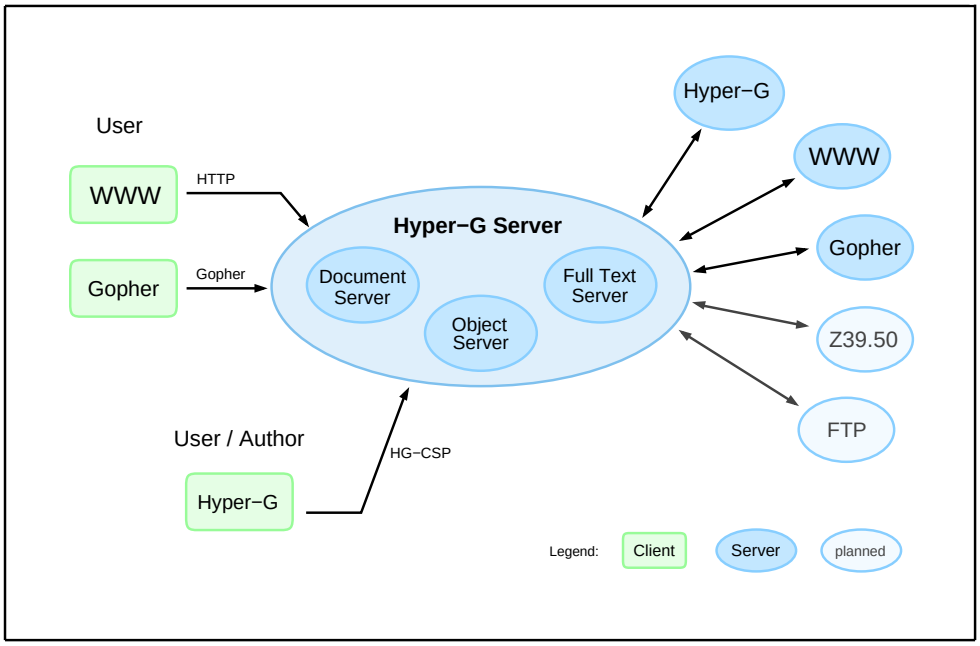

Figure 2.3: The Hyper–G architecture

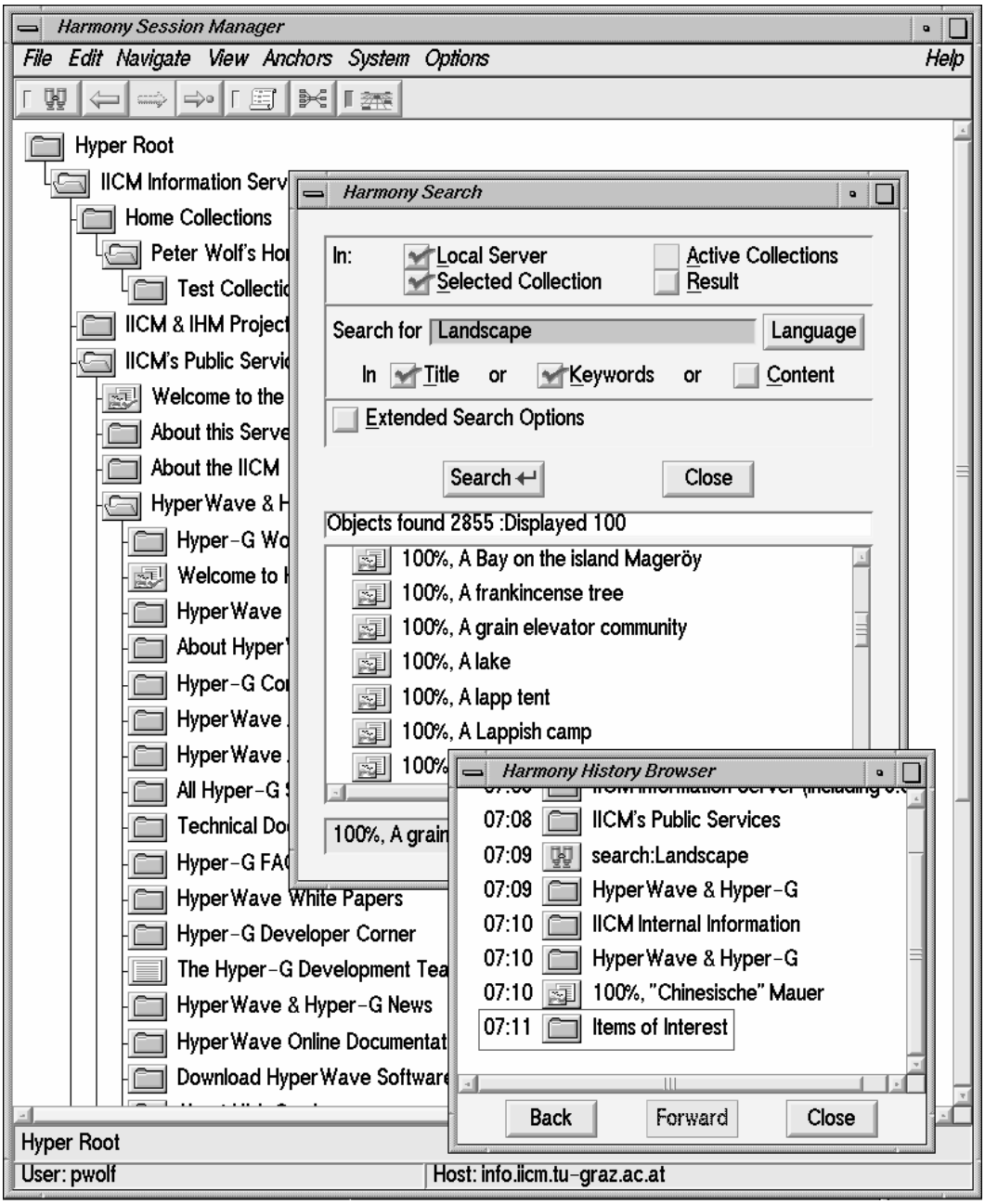

Figure 2.4: The Harmony Session Manager with history list and search dialog

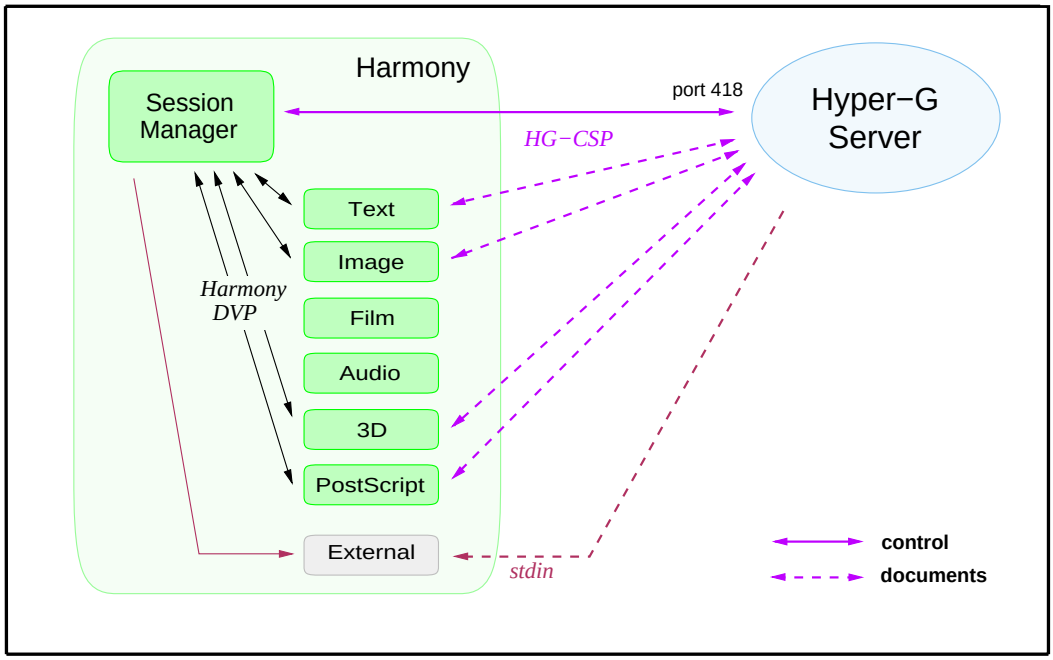

Figure 2.5: The Harmony architecture

Chapter 3

Spatial Information Visualisation

3.1 Spatial Metaphors

First visualisation systems concentrated on spatial scientific data, for example liquid flows over turbines, where it is quite obvious how the information can be mapped into three-dimensional spaces. With time, the importance of visualising more complex sets of information and their structure, like documents in hypermedia databases, grew. To visualise information which is not closely related to physical space, it it necessary to find appropriate spatial metaphors. A spatial metaphor is a likening of an actual physical space or object on which intuitively understood activities can be performed, to something that it describes [Tro94]. In the case of hypermedia databases, such metaphors have to be used for mapping type, attributes and contents of documents from the database to objects which can be placed in virtual environments according to the structure of the information they represent.

According to Heiner Benking et al.[BJ94], the following classes of spatial metaphors can be distinguished:

- *Geometric forms*: cubes, spheres, etc.
- *Artificial forms* : townscapes, houses, rooms
- *Natural forms*: landscapes, trees, etc.
- *Systemic structures*: highway systems, pathways, flow systems
- *Dynamic systems*: atomic, molecular, planetary, galactic systems

- *Traditional symbol systems*: mandalas, sand paintings, etc.

3.2 Spatial User Interfaces

Theoretical basics for the implementation of spatial information visualisation systems can be learned from spatial cognition research. This research focuses on the analysis of human spatial cognition and theories of cognitive mapping in real world spatial environments, i.e. the human process of building and memorising internal models of the three-dimensional world and constructing 'mental maps'. The knowledge of cognition mechanisms for navigation and orientation can help to build more coherent and navigable spatial environments.

Spatial user interfaces essentially try to make use of the human abilities to use space for organising objects. They use human skills in navigating three-dimensional structures like houses or cities and their skills in intuitively interacting with three-dimensional objects. Thus they can help to make structures explicit, to support navigation and organisation, or to facilitate data manipulation [DA94].

The structuring of the information presentation should as far as possible reflect the structure of the users' work and their mental model of this work, because then the visualisation makes sense for the users and they can do their tasks more easily. But finding this structure is often very hard as different users and different groups of users all tend to have their own ways of organising their work. Even single users will change and evolve their mental model of the information as they work with it. There are two different approaches to solve this problem. One allows users to customise the interface's model of the information space according to their current mental model of work. The other approach is to provide a more static model of the information and trust that its consistency makes users adapt their mental model to the model presented by the information system. The advantage of this static approach is that it prevents drastic layout changes which might confuse users and distract them from their task.

For the visualisation of large information spaces it is important that spatial metaphors for a particular chunk of information should not only be consistent with what they immediately represent, but they also should scale up. This means that they also should give some clues about what they contain and how they fit into their part of the information space, when viewed from a distance or in an overview. In this way a set of small-scale metaphors can easily be combined to create a descriptive higher- level representation. Such sets of representations that

always scale up to higher levels in smooth transitions enable users to zoom in and out. By zooming out, they can get an idea of the whole information structure and by zooming in on some detail they can view it from close up, while still retaining its context. This concept somehow reflects the infolded and fractal ordering of natural space.

3.2.1 Visualising Hyperspaces

There are similarities between navigating through everyday environments like cities or landscapes and navigating through information structures like hypertext or hypermedia systems [Shu90]. Thus it is quite obvious to use those real-life environments as metaphors for visualising hyperspaces. Virtual worlds built with such metaphors provide intuitive means for browsing information spaces. Spatial user interfaces using physical space as a metaphor for information space, create life-like virtual environments in which users can locate and select objects and perform actions. Those virtual environments can be explored with everyday knowledge and similar to the exploration of real space, by using navigational metaphors.

3.2.2 Navigating Hyperspaces

Virtual spaces created when visualising information are often larger or at least more complex than real-life environments. That is the reason why real-life navigational metaphors, like walking or flying, are often insufficient and tiresome in virtual environments. Thus almost all visualisation systems additionally use *magic* navigational features like teleportation which have no counterpart in real-life. They quickly move the user from one point of the virtual space to another point or even another space. In fact magic navigation is one major advantage of virtual environments, as they allow users to perform tasks that are not possible in the real world. Even though magic features are not understandable in terms of everyday 3-D experience, users find no difficulty in handling them, as long as they are informed what happens and where they are located after their magic movement [Tro94].

No matter how good metaphors for navigation in cyberspaces are, there will always be differences compared to real-life navigation. Thus the navigation and exploration of virtual spaces has to be learned by their users as the exploration

of physical space has to be learned by children. For most users, however, this learning is both challenging and fun.

3.3 Classification of Spatial User Interfaces

3.3.1 Hand-crafted versus Automatically Generated Layouts

According to Keith Andrews [And94] two ways of creating spatial layouts can be distinguished:

Hand-crafted layouts are modeled by a designer for very specific, sometimes quite complicated applications. The designers usually need some knowledge of architecture or graphic design. Users of hand-crafted systems often can use spatial relationships to communicate with other users or even with the system's database.

Automatically generated layouts are built on-the-fly by an algorithm which uses pre-fabricated components. They often have to be dynamically restructured, e.g. when the interface responds to a query or when the layout's underlying database is changed. On one hand this restructuring may be necessary and even desirable. But on the other hand users might get lost when the interfaces information model constantly changes, because they can not continually rebuild their mental model of the information space.

3.3.2 Dimensions: Two, Three or More ?

Spatial user interfaces usually are two- or three- dimensional. This leads to the general distinction of three-dimensional *spatial nodes* of the information used and two-dimensional *structure maps* of the information space as a whole [And94].

In **2-D space** all objects are placed on one plane and relationships can be expressed by their relative location to each others. Users look at this plane from above either with or without perspective. If no perspective is used, all objects are represented in the same distance to the user and thus perceived as equally important. Systems which use perspective can emphasise objects which are in the focus of the user's view as they are larger than objects which are further away. *Fisheye-views* can be used to visualise larger areas of the information space on limited

screen space. An advantage for users of two-dimensional information spaces is that they find it easy to communicate about locations in this space in terms of 'above', 'below', 'left' or 'right' of some landmarks. Common two-dimensional interface metaphors are desktops or tables.

3-D space always uses some kind of perspective or depth cues. There have to be some objects that obscure or even contain other objects. Desktop- or table-metaphors have to be replaced by new ones, like rooms, houses or landscapes. The advantage of three-dimensional metaphors is that there is really more space available for placing information objects. Therefore they are very useful for presenting large amounts of information. Moreover, users can move around without any limitations. They can zoom into objects or move away from them and thus gets all kinds of perspectives and views of the information space. Finally there is a wide range of transformations and animations that can be performed in three-dimensional space, like transforming the space into a map, various fly-to metaphors, etc.

To go **beyond the third dimension** would create a lot of new visualisation possibilities. Objects in 1-D space can be sorted and placed using one criterion, 2-D space allows two and 3-D space three criteria. Hyper-dimensional space could be used for placement according to even more criteria, which would be very useful, e.g. for the visualisation of search results. The problem with those multi-dimensional spaces is that of course only their projections into two- or three-dimensional spaces can be viewed. But intuitive metaphors for those projections are hard to find, and there are only very few systems that perform hyper-dimensional visualisations.

3.3.3 Single and Multi User Interfaces

For systems that are strictly limited to the visualisation of information, single-user interfaces usually are are sufficient. But in shared information systems, where many users can explore and modify the same information space, *multi-user* interfaces help to get an overview of who is viewing or changing what data. Multi-user interfaces of information systems use special icons called *avatars* to represent users in the information space. If users of multi-user interfaces can use *agents* (programs that automatically do some work for the user) which modify the shared information, it is necessary that those agents too are represented in the information space to enable other users to see what is going on [Ber94].

3.4 Credibility of Virtual Spaces

To be really useful, spatial interfaces and virtual spaces created by them need to be credible [Tro94]. Criteria for credibility are:

- **Apparent Presence**
  Research has proved that the sense of presence in an information space is an important factor for effective task performance. Presence is defined as the extent to which a participant in the virtual space reports the illusion of being in the alternative environment rather than in his real environment. The factors involved in improving the degree of presence are mainly quality and resolution of the presented space, consistent presentation of the information space, possibilities for interaction with objects in the artificial space, correct responses to the users movement in consistence with the users expectations and simple connections between the user's actions and their effects.

- **Apparent Reality**
  Human perceptual experiences are not only controlled by external factors. The immersion of people into virtualities like books, films or in this case artificial environments is based on the peoples willingness to accept the illusion as real. This willingness to suspend disbelief is highly dependent on the credibility of the illusion. It has to be so realistic that the users are able to create an image in mind about the affordances of objects, the locations of places, and their inter-relationships. Metaphors thus have to be created and implemented so realistically, that they allow the user to sustain the illusion they create. However they do not have to be realistic and credible as compared to the real world, but only within the frame of the created virtual environment. In this way even things like magic features that have no counterpart in the real world can be made credible. Moreover, the more aesthetic and pleasing the environment is presented, the more users are willing to suspend their disbelief.

- **Affordances**
  People perceive an environment in terms of its potential for action. Participants in a virtual environment are able to behave most naturally if the virtual objects and events convey the same affordances as their real-life counterparts. A metaphor is useful if it has a visible affordance for the goal for which it has been designed. The visible affordance must be generally understandable, and preferably unconnected to cultural meaning. Because, if the metaphor suggests obscure or incorrect affordances for tasks connected with it, the transfer of everyday knowledge will be severely disturbed. So it

is important that metaphors and their suggested affordances take the users' abilities and needs into account.

- **Consistency**
  The conditions in artificial spatial environments must be extremely reliable so that experience is always consistent. If metaphors are carried too far or not far enough the illusion will be shattered. A useful interface simply has to be effective and easy to use. There should not be many different ways to perform a simple task. However for complicated tasks it is a good idea to have both, a short but maybe cryptic way for experienced users and a long but transparent way for average users or beginners. An interface that is effective and easy to use has to have a high level of internal consistency. If the commands are too different for each object or action, or too cryptic or too alien the illusion will quickly be disturbed by it.

- **Goodness of Fit**
  Some lessons for the design of good spatial metaphors might also be learned from the insights of environmental psychology. An effort should thus be made to achieve fitness between two entities: the form in question and its context. Form is that part of the metaphor that can be controlled while context is that part of the information that puts demands on this form. Good fit between form and context can be found by clarifying the context, and by neutralising the incongruities which cause misfit. When there are no misfits left, a condition of good fit will occur. So the process of finding a good metaphor needs research of the implemententation of different metaphors, their evaluation and improvement of their effectiveness and finally their integration into an optimal result.

3.5 When to Use Spatial User Interfaces

Even though spatial metaphors are supposed to alleviate problems of disorientation within the vast expanses of cyberspace, spatial user interfaces are not the universal solution to navigation problems and it is important to decide when to use them and when not to do so. The result of this decision greatly depends on the user's and the application's goal. To illustrate this, I will cite Ray McAleese's [McA93] comparison of different browsing strategies and the effectiveness of textual and graphical interfaces for all of them. He first lists the five types of searching hyperspaces according to Canter et al. [CRS85]:

- Scanning: covering a large area without depth.

- Browsing: following a path until a goal is achieved.
- Searching: striving to find an explicit goal.
- Exploring: finding out the extent of the information given.
- Wandering: purposeless and unstructured globe trotting.

Ray McAleese then states that users of textual interfaces have a very high tendency to use browsing, often use scanning and browsing, and only rarely explore and wander. As opposed to that, users of graphical interfaces have a high tendency to use wandering, often scan and explore, and only rarely use browse and search.

One problem that also has to be considered when building spatial user interfaces is that some people are not particularly adept at spatial reasoning. For them it is quite hard to navigate through real-life cities or road-systems. So the use of spatial information systems with often even more complicated structures might be a too difficult for them and thus not very helpful.

3.6 Implementations of Spatial Information Systems

This section introduces some projects that use various spatial metaphors for visualising information. Most of them are experimental implementations that incorporate some visualisation concepts to investigate the usefulness of spatial metaphors for information systems.

3.6.1 Pad++

Pad++ [Mey96] is a collaborative work by groups at the University of New Mexico and NYU Media Research Laboratory. It is a graphical interface which uses a bulletin board metaphor for two-dimensional information visualisation. The bulletin board is an information plane that is shared among users, much as a network file system is shared. Every object on the Pad++ surface occupies a well defined region. Objects render themselves differently when viewed at different sizes (see Figure 3.1). Users can infinitely and smoothly zoom into the surface to

see more details of an object. Through this *semantic zooming*, many of the mode switches required in traditional windowing systems can be avoided. No icons have to be clicked to switch into a mode where the icon's contents are visible. Instead the user simply looks more closely to see more detail. At any zooming level, the user can write or draw on the bulletin board. Pad++ can be used to visualise any kind of hierarchical data, e.g file systems, calendars, web pages, hypermedia presentations, etc.

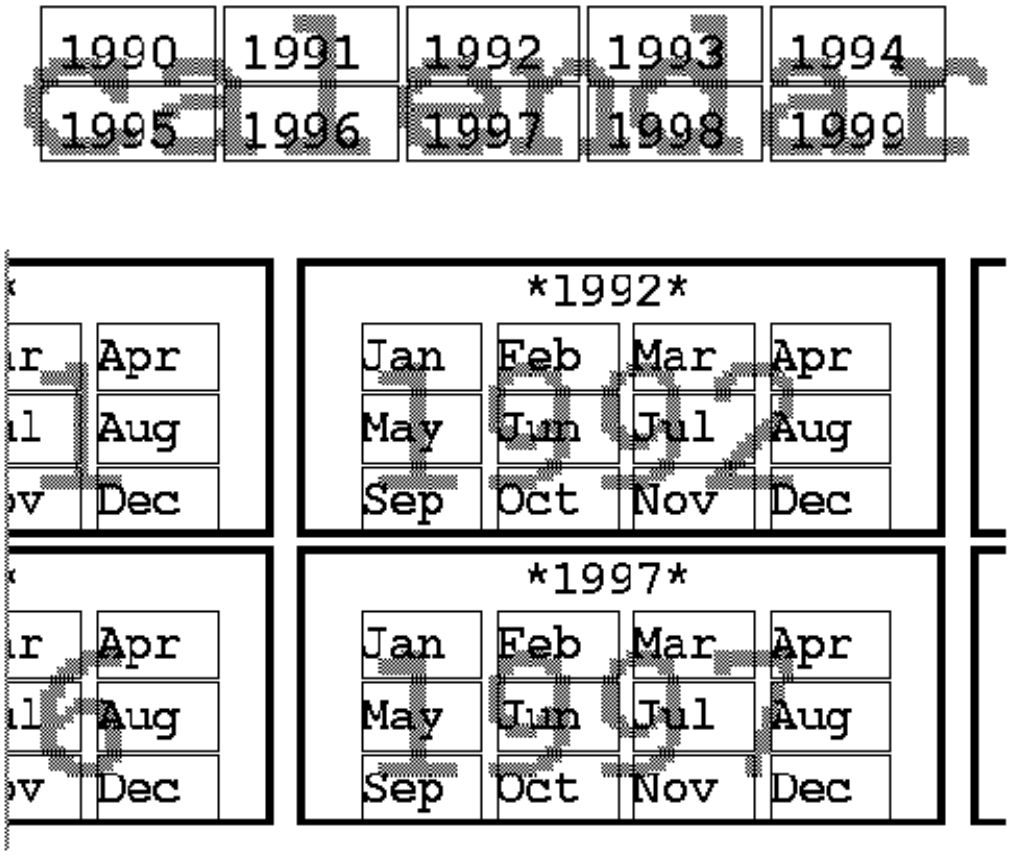

Figure 3.1: Two zooming levels of a Pad++ calendar

3.6.2 IVEE

The Information Visualization and Exploration Environment (IVEE) [Ahl96] was developed at Chalmers University of Technology. It is a system for automatic creation of dynamic query applications which can be used to explore data sets and

to interactively update visualisations. IVEE imports database relations and automatically generates an environment containing data visualisations and query devices. The IVEE architecture is based on visualising database objects by attaching graphical objects to them. These graphical objects can be simple points or squares, pre-defined glyphs, or arbitrary two- or three-dimensional icons. IVEE's visualisation techniques include starfields (see Figure 3.2) and maps. Multiple visualisations can be used at the same time. All visualisations can be interactively filtered, zoomed, panned, and changed by queries. To retrieve *details-on-demand* of an object, the visualisation of the object has to be clicked. Multimedia information can be presented by using an HTML file which specifies how details-on-demand information should visualised. Multiple IVEE clients running on one network can communicate and let one user's actions affect the visualisation of other clients. For short response times, IVEE adapts its rendering strategies to the current performance of the computer and to the user's behaviour.

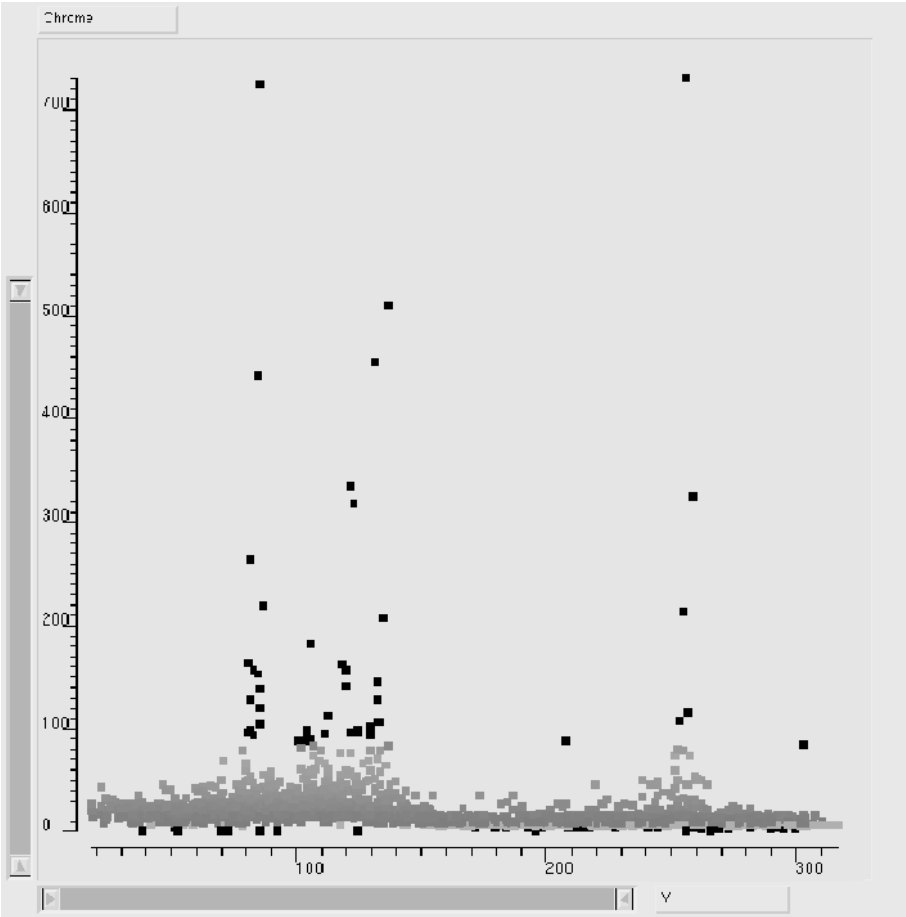

Figure 3.2: An IVEE starfield visualisation

3.6.3 urlHouse

The urlHouse [uH95] is a simple but nice looking example for spatially structuring a directory of various web-pages using a hand-crafted house metaphor. In the various rooms of the urlHouse there are pieces of furniture like a shelf with books, a desk, a piano, a TV set, etc. Those items are anchors connected to web-pages with contents that match the item's metaphor. So the piano is linked to documents related to music, books contain manuals, the TV set is an anchor to recreational sites and so on. The urlHouse is implemented in VRML and can be viewed with any VRML-browser. Figure 3.3 shows a bookshelf in the library of the urlHouse.

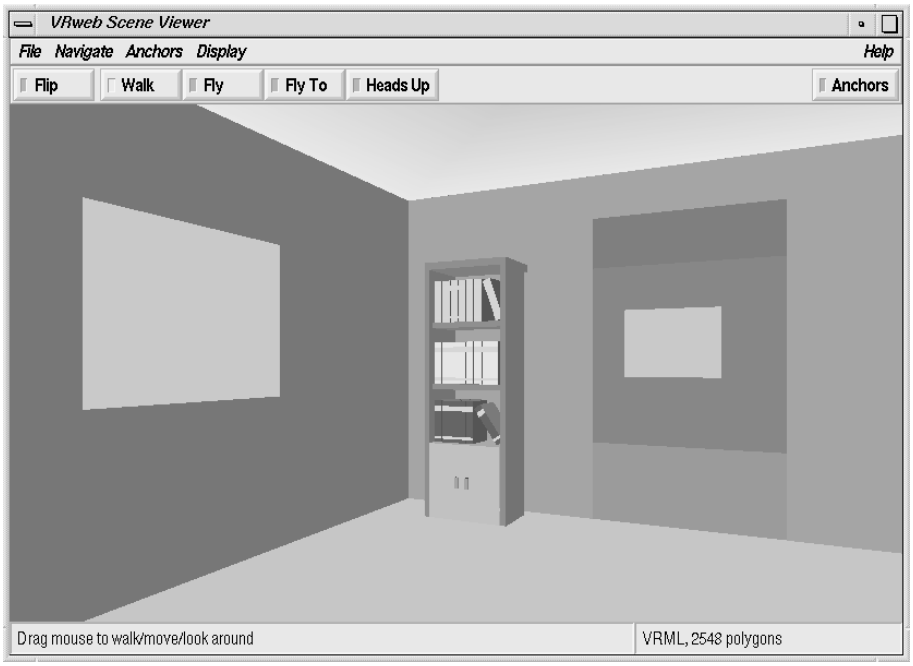

Figure 3.3: A room in the urlHouse

3.6.4 Web World

Ron Britwich's Web World used to be a system that visualised parts of the World Wide Web using a landscape metaphor with buildings representing web-pages and roads that connected them. The user looked from a fixed height and angle onto the landscape and could only move left, right, forward and back. So Web World

was no real three-dimensional system, but never the less very nice looking and thus very popular. Every user could enter pages from his server into the landscape by getting a house and activating a link to his pages. So the contents of the Web World were dependent of the people who were willing to be represented in it. However sometime in 1995 the company running the Web World server did not want to spend any more resources on the project and, to the grief of many users, removed it from the Internet.

3.6.5 LyberWorld

LyberWorld [WHK94] is a three-dimensional user interface to an information retrieval system. The retrieval system can locate documents relevant for a particular search term. Users start their search with a word and can then refine the search from the results they get by entering a new search term. This creates a tree structure containing documents with search terms as their children. These search terms again have documents which are relevant for them as children. The developers of LyberWorld visualised this alternating term-document tree in a three-dimensional structure called *cone tree* where each connection between a term and a document is represented by a cone. The terms and documents are shown as cylinders with a label describing its contents. Users can open or close any of those cones to add or remove subtrees and they can rotate each level of the tree. So they can easily focus on the important parts of the tree. To view a document, users can zoom into its cylinder which brings them into a room where the document is projected on a wall.

3.6.6 SemNet

SemNet [FPF88] was designed to help users to explore and recognise the organisation, structure and contents of large, arbitrary knowledge bases. Elements of the knowledge base are represented as labeled rectangles in three-dimensional space. Relationships between those elements are indicated by arcs that connect them. The position of the elements can be used in a few different ways. Similar or related elements can be located close to each other while unrelated elements are far apart. Another possibility is to map an element's properties to its position in space. Finally also users themselves can place elements according to their preferences.

To pack as much information as possible into the screen space, SemNet not only uses the natural effects of three-dimensional perspective, but also an algorithm

that joins elements that are far away from the users position to element clusters which are represented by a single rectangle of a different colour. For orientation SemNet provides overview maps in x/y- and x/z- planes which show the users current position in the knowledge base. SemNet's navigational features include absolute and relative movement and magic navigation like teleportation to already visited elements or *hyperspace movement*, i.e. movement along the arcs between related elements.

3.6.7 Information Islands

Information Islands [WS94] is an interaction model that enables users to navigate the world of distributed and networked computers without loosing their orientation. The net-world is represented as a set of archipelagoes, where each archipelago symbolises one major class of service or application, like communication services, information services, educational, medical or recreational services. Each archipelago consists of a collection of information islands. Each information island contains one subclass of the archipelago's information service. On the islands there are buildings, where each building contains a set of information sources on one particular theme, e.g. lectures, sports, newspapers, etc. Moreover, the buildings also have common features that give an overview of the buildings contents.

This world of archipelagoes can either be explored from a public or a private view or from both views at the same time. The public view shows all information that is available, while the private view only shows those pieces of information which are interesting for the user.

The Information Islands model also contains a number of different agents that tell the user what information is available and where it can be found. To reach a certain location, users can directly navigate the three-dimensional world, they can move around in a history list and they can zoom between the various levels of the world, i.e. from an archipelago via an island to the interior of buildings and vice versa.

3.6.8 The Xerox Information Visualizer

The Information Visualizer, designed by Card et al. [RCM93] at Xerox PARC, was implemented to visualise large amounts of hierarchical data and linear information. It is designed to keep the time it takes to access information low.

It wants to enable the user to handle more data at one time. The heart of the Information Visualizer's architecture is the so called *Cognitive Coprocessor* which helps the user to work effectively with the provided information. The Cognitive Coprocessor is a resource scheduler which tries to guarantee continuous animation and short response times of about one second to user actions.

To enlarge the virtual workspace, a room system was implemented. Each room is a working space on its own and users can shift to the room that is most convenient for the task they are currently working on. All rooms at the same time can be viewed on an overview map. For effective screen space usage and high density of the displayed information the Information Visualizer also uses cone trees for showing hierarchical structures. For the visualisation of linear information structures with spanning properties the Information Visualizer provides a *perspective wall* which presents the information using a fisheye view. Users can move items of interest to the central part of the wall, where they can view them in full detail. Items farther away from these items of interest can only be seen perspectively distorted.

The Information Visualizer supports five techniques for navigation and manipulation: the walking metaphor, point of interest logarithmic flight, object of interest logarithmic manipulation, doors between rooms and overview maps.

3.6.9 Bead

Bead [Cha93] defines itself as a prototype system for graphical information search and retrieval. It shows documents as icons on the groundplane of a three-dimensional landscape. It uses a metric based on word co-occurrence to place icons of documents that have approximately the same title, abstract or keyword sections close to each other. Thus the user can get an overview of the similarity and dissimilarity of documents in the information corpus. Clusters of icons and the gaps between them generate good landmarks for orientation and navigation. When the user enters a search term, Bead marks all objects that contain this term. When looking at the marked objects, the user can also check other unmarked objects which are very close to a marked one. As they are similar to the marked document there is a chance that they are relevant ones even though they do not contain the search term.

The view of the landscape created by Bead can be shared with other users on the network as other viewers with the same landscape can be run on other machines. Moreover, Bead is a multi-user system and every user can see all other people moving around and searching in his current information space.

3.6.10 n-Vision

n-Vision [FB90] is an example for hyper-dimensional information visualisation. It approaches the visualisation of multivariate functions in nested heterogeneous coordinate systems, by using a worlds-within-worlds metaphor. Figure 3.4 shows six 3-D worlds nested within one 'larger' world and thus creating a six-dimensional visualisation.

n-Vision allows users to explore virtual worlds populated by objects, which represent functions of large numbers of variables. A three-dimensional window system partitions the space in which users interact into a volume containing objects which are viewed in stereo and manipulated using a data-glove. Currently n-Vision is used for the visualisation of financial data.

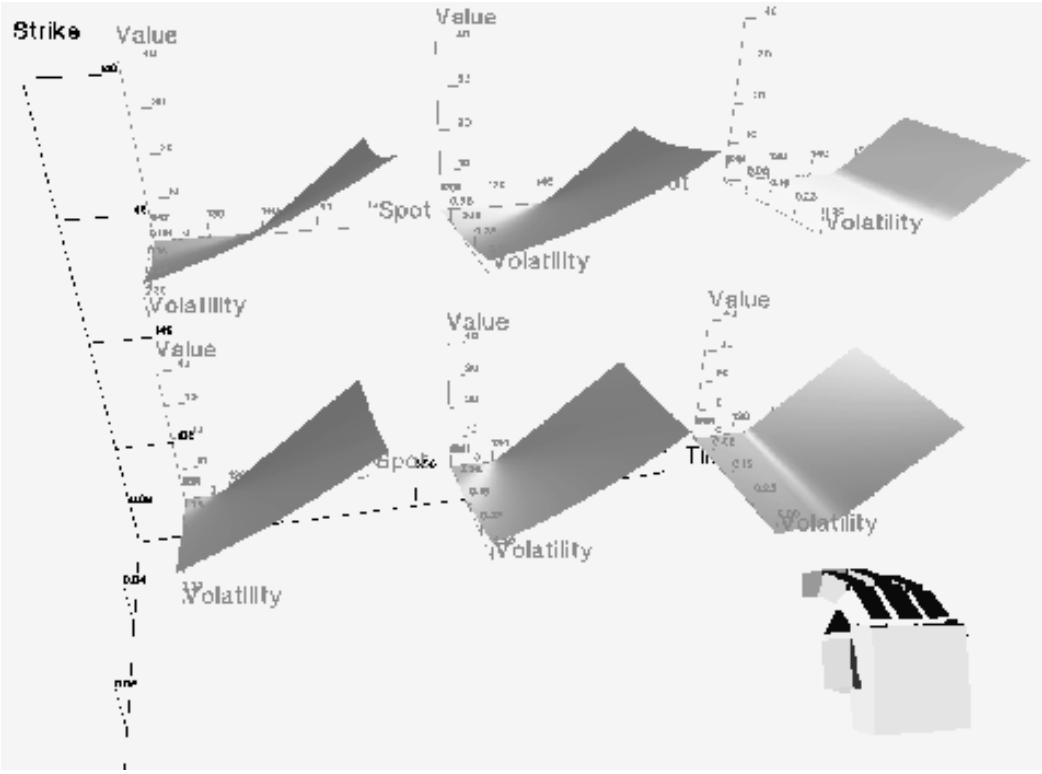

Figure 3.4: n-Vision: worlds-within-worlds

3.6.11 Gopher VR

Gopher VR [ME95], created at the University of Minnesota, is a three-dimensional interface enabling intuitive and comfortable browsing of the hierarchical Gopher menu structure. If a Gopher VR client is not told by the server how to render a scene the client will render the scene using pre-defined rules. The icons for each item of a menu are circularly arranged around the central 3-D *kiosk* icon (Figure 3.5). Results of searches are presented in a spiral around the kiosk. Icons of high scoring documents are located near the center and icons of low scoring documents are located at the distant end of the spiral. By default all icons are rectangular boxes with the name of the item written on them and a colour that indicates their type. On *Gopher+* servers a *3D block* can override the default rendering of the scene. By using a Gopher+ attribute any Gopher+ server can be used to create custom Gopher VR scenes. Server administrators can assign shapes, position, colour and orientation to each item on their server.

To view a document or to open a submenu in Gopher VR, users have to click the desired icon. This gets them into a new space with the selected document as kiosk and the new items arranged around it. To get back to a parent menu, the kiosk has to be clicked. For navigation and orientation there are a few interesting animations like jumping forward or jumping up to get a birdseye view.

Gopher VR helps to solve the lost-in-hyperspace problem that often occurs during browsing with conventional gopher clients.

3.6.12 File System Navigator

The File System Navigator (FSN or Fusion) [TS92], developed by Silicon Graphics, generates a three-dimensional representation of a UNIX file system's tree-structure by using a landscape metaphor (Figure 3.6). This tree-structure is stored in a special file and updated during each session. Directories are represented by platforms. Wires connect those platforms with the platforms of subdirectories. Blocks representing the files in a directory are placed on top of the directory's platform. The size of each of these blocks is determined by the size of the file it represents. The type of the file (text, executable, etc.) is indicated by an icon on the top-face of its block, and the age of a file is shown by the colour of the block. By dragging the mouse across the screen, the block under the mouse cursor can be highlighted. Clicking a block selects it, which is indicated by a spotlight on the block. Double-clicking opens the file that is represented by the block. For orientation, users can open an overview window that shows their position in the landscape. By using the mouse, users can fly across the landscape

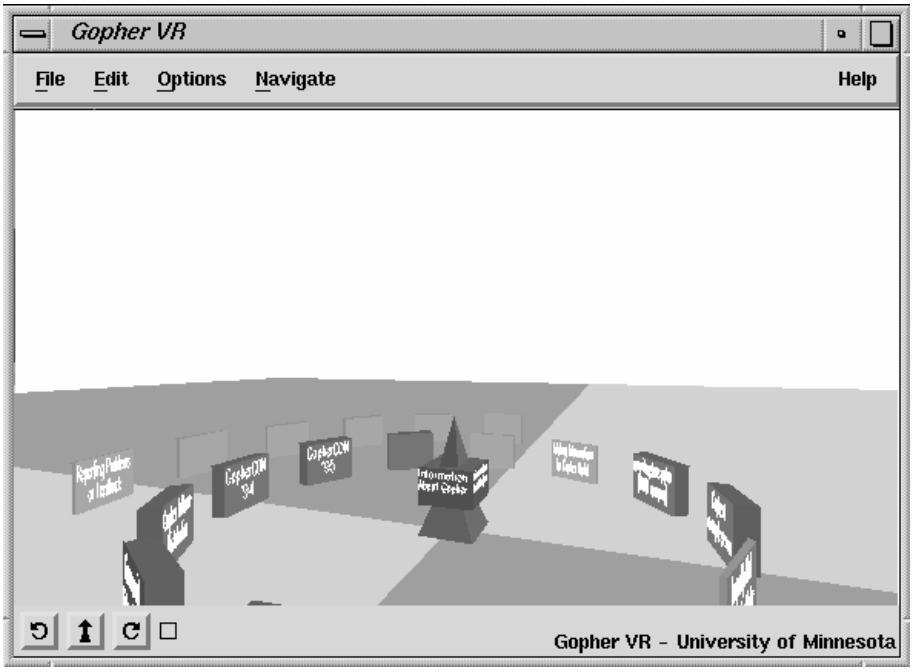

Figure 3.5: Gopher VR

and with the help of a scrollbar they can also change his altitude.

The File System Navigator provided design and layout ideas for the Harmony Information Landscape described in the following chapters.

3.6.13 Tecate

Tecate [KW95], developed at the San Diego Supercomputer Center in collaboration with the Digital Equipment Corporation, is a software platform for doing exploratory visualisation of data collected from networked data sources. It provides the infrastructure for applications that allow its users to browse the contents of data sources as well as inspect, measure, compare, and identify patterns in selected data-sets.

Tecate provides interfaces to the World-Wide Web, and to databases managed by the database management systems Postgres or Illustra. Interfaces to other data spaces can be implemented easily with tools provided by Tecate. In addition, Tecate dynamically crafts user-interfaces and interactive visualisations of data-sets with the aid of the Intelligent Visualization System (IVS), which is an expert

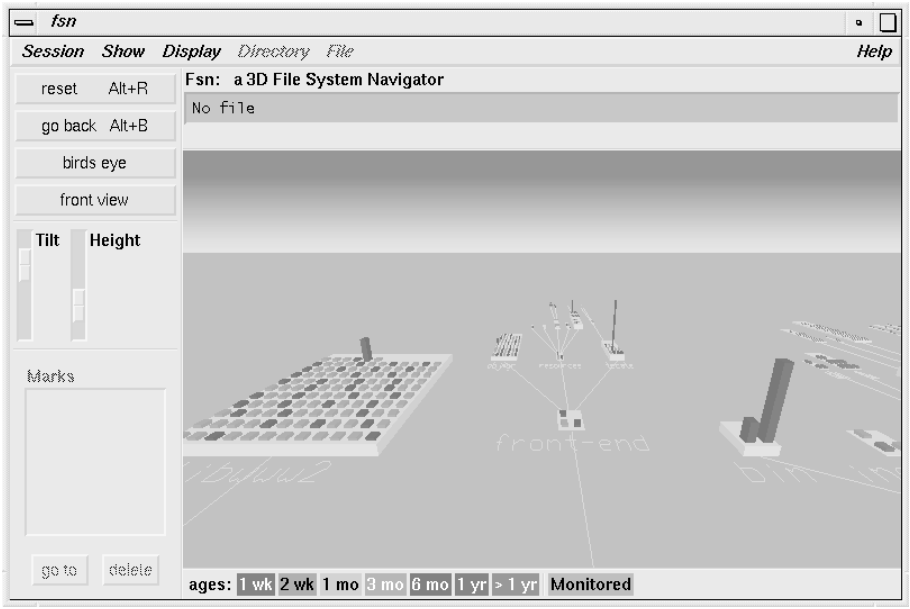

Figure 3.6: The File System Navigator

system with knowledge of visualisation techniques that transform data into three-dimensional virtual spaces. In this way many kinds of data-sets can be mapped into virtual worlds that can be explored by Tecate's users. For describing these worlds, Tecate uses the Abstract Visualization Language (AVL), an interpreted, object-oriented programming language that is capable of arbitrary computations, and the mediation of communication among different processes.

For browsing the World Wide Web, Tecate uses an information landscape concept. This landscape consists of a planar map of the Earth placed in three-dimensional space. Web servers are represented as three-dimensional icons positioned on this world map. Those icons are either located where the associated Web servers physically reside, or where the data within them refer. Figure 3.7 shows a screenshot of Tecate's visualisation of WWW sites.

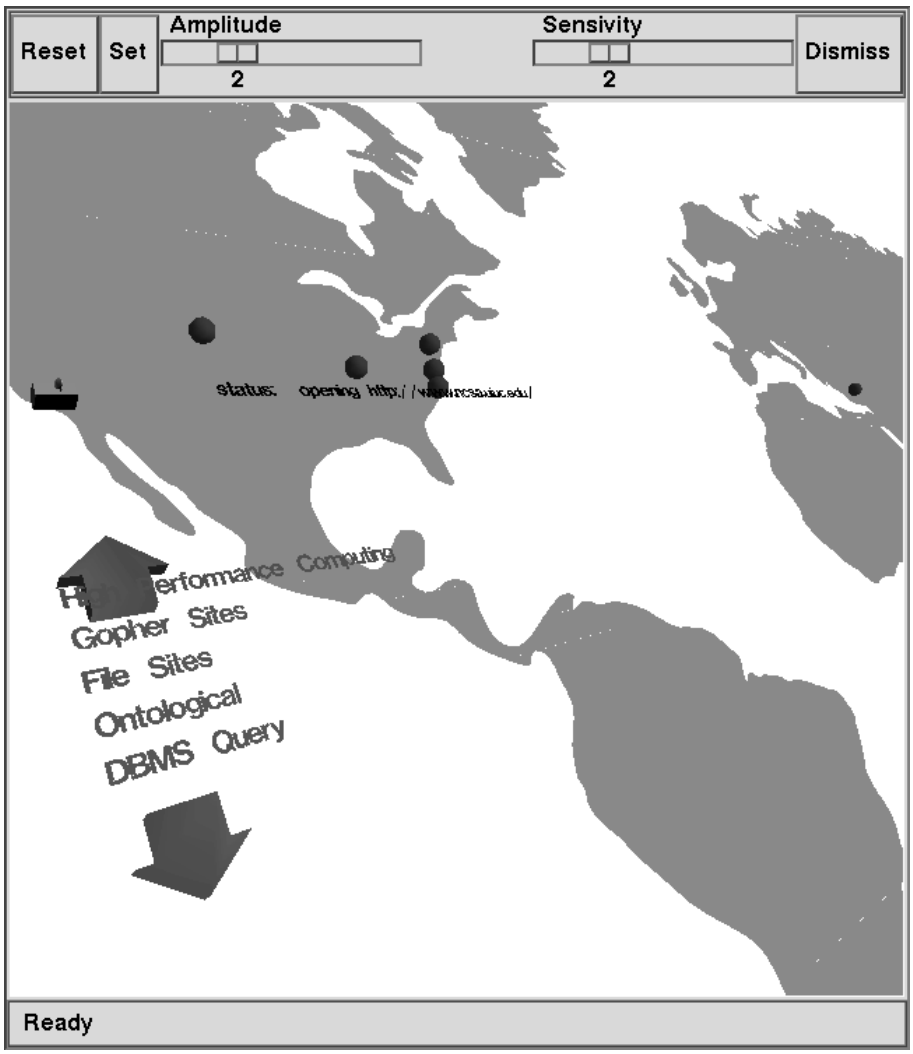

Figure 3.7: Tecate: WWW sites as 3-D icons on a world map

Chapter 4

The Harmony Information Landscape

The Harmony Information Landscape is a tool for browsing Hyper–G servers by navigating in an automatically generated, three-dimensional environment representing structure and content of a Hyper–G server. It encodes size, age and types of documents on the server and shows hierarchical and hyperlink relationships among the documents. The resulting information space can be explored interactively by the user.

The Landscape's basic functionality was implemented by Martin Eyl in the course of his masters thesis [Eyl95]. My work was to further extend and refine this basic implementation. This chapter describes the Landscape's new functionality and new features that I have implemented.

4.1 Link Visualisation and Layout

The link visualisation turns the Harmony Landscape from a 2.5- to a real 3-dimensional information visualisation system. Incoming and outgoing links of selected documents are drawn under and above the document respectively, and thus represent the link tree. Links are visualised by a 3-D icon representing the source or destination node type of the link. A wire connects this icon with the document to which the link belongs. If a node in the link tree is also located somewhere in the collection tree, a wire optionally connects these two representations of the same document. All wires in the link tree can be followed to their

source or destination by double-clicking them with the middle or left mouse button respectively. The depth of the link tree can be changed using the dialog *Set Local Map (3D)* (see 4.8.2). This dialog is also used to determine which link types are included in the link tree. The user can select referential links, annotations, inlines, textures, parents and children for both incoming and outgoing links.

As the collection hierarchy is no tree, but really an acyclic graph, one node can have multiple parents. But the collection tree can only contain one parent per node. To visualise multiple parents, a link tree, with parent links as the only selected type, can be used.

Any document in the link tree can be selected, and further link levels of this document can be displayed. The only limitation is that for a node in the incoming tree only further incoming links can be visualised. The same applies to outgoing links. The reason for this limitation is that without it no reasonable layout of the link tree would be possible.

The advantage of this three-dimensional link visualisation compared to the two-dimensional local map is, that on one hand links belonging to a document can be presented together with the document's position in the collection tree, and on the other hand that the limited screen space available can be used more effectively. Figure 4.1 shows one link level of an object visualised in 2-D and 3-D. Both visualisations use the same screen space. One can clearly see how much more information can be packed into the screen using the three-dimensional metaphor.

4.2 Navigation

Navigational features of the Landscape include the walking metaphor, point of interest logarithmic flight, moving in the overview map, magic features like 'home' and hyperspace movements along the wires connecting objects in link and collection trees. In the first version of the Landscape, however, there were two navigational problems: The movement could be very slow if complex scenes had to be rendered and the user could only look straight ahead. Both problems have been solved in the course of this thesis.

4.2.1 Arbitrary Direction of View

So far the users freedom of movement was very limited because, due to technical restrictions, users could not turn around to look at the Landscape from different

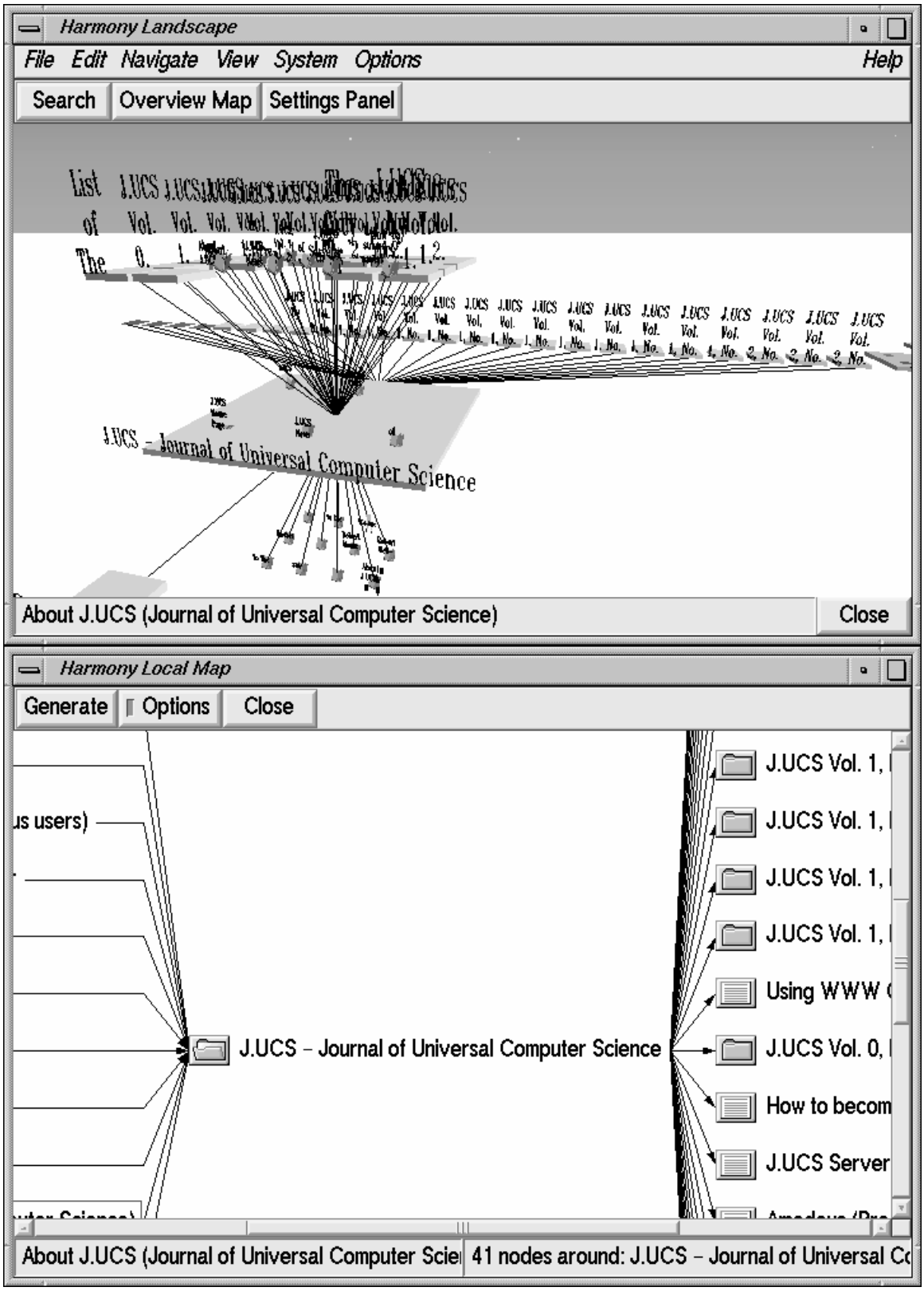

Figure 4.1: 2D and 3D Local Map

angles. The viewing direction always had to be straight ahead, away from the root. This also forced users who wanted to move right or left to tilt sideways in a quite unnatural way.

Now this problem has been solved and the viewing direction in the Landscape is no longer limited to straight ahead. Users can freely wander in any direction. If they want to move right or left, the camera turns to that direction and the users can move towards their new viewing direction. If the users turn around completely and look back towards the root of the collection tree, all object titles are moved to the backside of their icon and thus remain readable.

Generally this allows a more natural and intuitive way of navigation and also provides new and different views of the information structure. In some cases however, users might still prefer the old way of moving right and left without turning the camera. For example when they want to move sideways along a certain level in the collection tree , always looking straight ahead towards the titles of the collections. One way of accomplishing this is by pressing the middle mouse button and dragging the mouse left or right which causes a left/right movement without turning the camera. The other way is by activating *Lock Camera* in the *View* menu which locks the camera to the current direction of view and restores the old way of navigation heading towards the current viewing direction.

Due to this additional measure of freedom, users might get lost at some point, because they do not know in which direction they are looking. A little adaption of the overview map helps them to find their way again. The cross indicating the users position in the overview map has been replaced by an arrow (Figures 4.2 and 4.3). The tip of the arrow indicates the user's position while its direction indicates the user's viewing direction.

The camera's movement between horizontal and vertical is now no longer limited to the downward direction. It is now possible to look up and down within an angular of +/–90 degrees. This is necessary because the Landscape is no longer 2.5 dimensional with all nodes placed on the ground plane below the camera, but really uses the whole three-dimensional space for its link visualisation. Interesting documents can as well be placed above the camera as they can be underneath it.

Nodes of documents' incoming links are visualised under the documents. As the collection tree is located on the ground plane, this means that they are placed under the surface of the Landscape. A good way to visualise this would be to draw a transparent ground plane and solid nodes underneath it. Unfortunately, as GE3D (see Section 5.1) does not support transparency at the moment, the implementation of this feature has to be postponed to a later point of time.

However, if the user is located below the surface, this is indicated by drawing the ground plane above the user's position and the rest of the background in an 'underground' colour.

4.2.2 User Defined Frame Rate

Movement in former versions of the Landscape used to get slower and slower as more documents were added to the collection tree . With a great amount of documents in the tree the number of rendered frames per second could drop to one or even less. Of course, this made interactive navigation in a large information space often impossible or at least very tiresome.

One approach to improve the situation was to optimise all drawing algorithms and to extend the scenes preprocessing as much as possible. But still, the more documents in the Landscape, the lower dropped the frame rate. So the general solution to this problem was to provide an algorithm that assures a certain minimum frame rate. On the *Settings Panel* this frame rate can be adjusted between 1 and 25. Of course increasing frame rates imply decreasing image quality due to the reduced level of detail of the rendered scenes. But low-quality frames are only rendered as long as the user moves. When the movement stops, the last frame is drawn in normal quality and full details. The precondition for achieving the desired frame rate is that the X-server is fast enough. If the X-server limits the frame rate, the algorithm controlling the level of detail recognises it and does not reduce the image quality any further.

4.3 3-D Icons

In the first version of the Landscape all documents were represented by cubes. The type of the documents was indicated by their icons' colour. Textures could only be applied to collection and cluster icons, to the ground plane and to the sky. Now for every document type a 3-D icon can be defined. Using the *Icon Chooser* (see 4.8.5) one of 25 icons listed in Figure B.1 can be selected. Moreover, if the textured Landscape is used, the texture for each of the icons can be defined using the *Texture Chooser* (see 4.8.5).

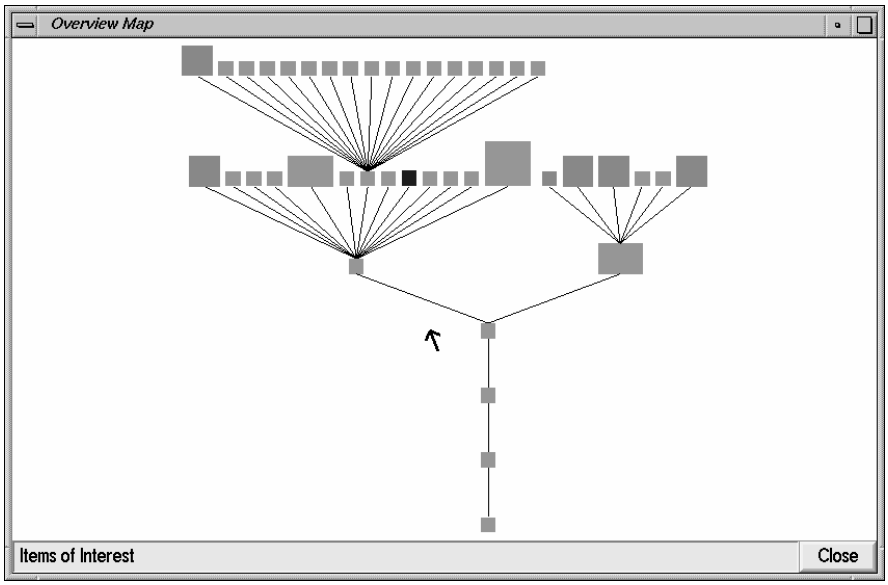

Figure 4.2: The Overview Map

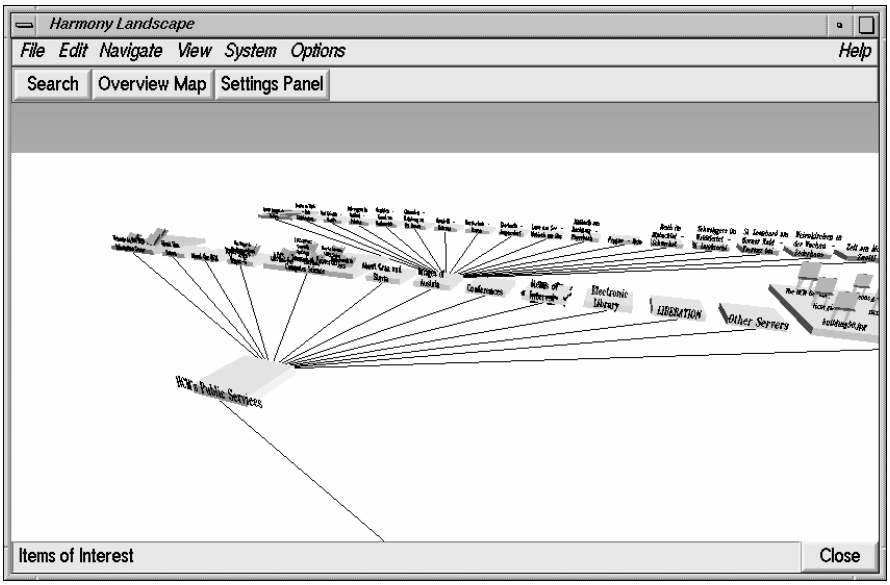

Figure 4.3: Landscape corresponding to Figure 4.2

4.4 Aging

The age of documents can now be encoded as the brightness of their icons. The older a document is, the darker its icon gets. The relation between age and brightness is logarithmic, such that newer documents fade faster than older ones. This takes into account that, e.g. for documents that are only one day old an age difference of a few hours usually is significant, while for documents that are some years old, only differences in the range of months will be important. To enable users to view the age of documents they are interested in with maximum contrast, the speed of the fading process can be adjusted on the *Settings Panel* (Figure 4.9). Another approach for visualising the documents' age by 'yellowing' (i.e. making documents more yellowish and brighter, the older they get), did not proof intuitive enough to be further supported.

Figure C.1 illustrates the effects of the aging algorithm. It shows editions of a monthly electronic journal with the latest one on the right hand side.

4.5 Lettering

Titles of documents are now drawn centered and using correct line-breaking with a maximum of three lines per document. The title is usually drawn in front of a document. If the camera turns around and looks back towards the root of the collection tree , the titles are moved to the backside of the document for better readability. The size of the letters in collection and cluster titles on one hand, and in document titles on the other can be changed on the Settings Panel.

4.6 Fisheye

This option applies a cartesian fisheye transformation as described by Furnas [Fur86] on the nodes' x- and y-coordinates. The idea behind the usage of this function is that objects located near the user are of the greatest interest and importance and thus have to be drawn in full size. At the same time, the farther objects are away from the user, the less they are interesting as individual documents. But they still can help to provide a context for the really important objects. So also objects which are farther away are drawn, but their size decreases with their distance from the user's position.

Fisheye does not calculate the view of real fisheye camera, but simply provides a smart method of packing more nodes and information into the visible part of the Landscape. Parameters controlling borders and distortion of the transformation can be modified on the *Settings Pannel* (see 4.8.3). Figures 4.4 and 4.5 show two views of a collection containing 25 audio clusters. Figure 4.4 shows how many of the clusters can be seen without the fisheye transformation from a position close to the clusters' parent collection. Figure 4.5 shows a view from the same camera position but using the fisheye transformation. Here one can clearly see the approximate number of clusters in the collection as well as a full view of the cluster in the center of the user's view.

Another possibility for better usage of the available screen space is the visualisation of the nodes in hyperbolic space described by Munzner et al.[MB95]. But this method would influence the appearance and layout of the Landscape in a degree that would more or less mean giving up the concept of the real-life landscape metaphor.

4.7 Save As VRML

It is possible to save a snapshot of the Landscape's nodes and the wires connecting them as a VRML file [BPP94]. The ground plane around the collection tree is stored in transparent mode. A VRML browser supporting transparency can give an impression how objects under the surface could be visualised as soon as GE3D supports it too. Figure 4.6 shows VRweb [Pic93, POA+95] rendering a VRML scene exported from the Landscape.

4.8 User Interface

Due to the number of new features that have been added, the Landscape's user interface had to be changed and extended quite considerably. This section describes all the menus and dialogs that have been added and the modifications of previously existing ones.

4.8.1 The View Menu

All the following dialogs can be accessed from the *View* (Figure 4.7) pull-down menu. Apart from that, *Local Map (3D)* in this menu can be used to switch on

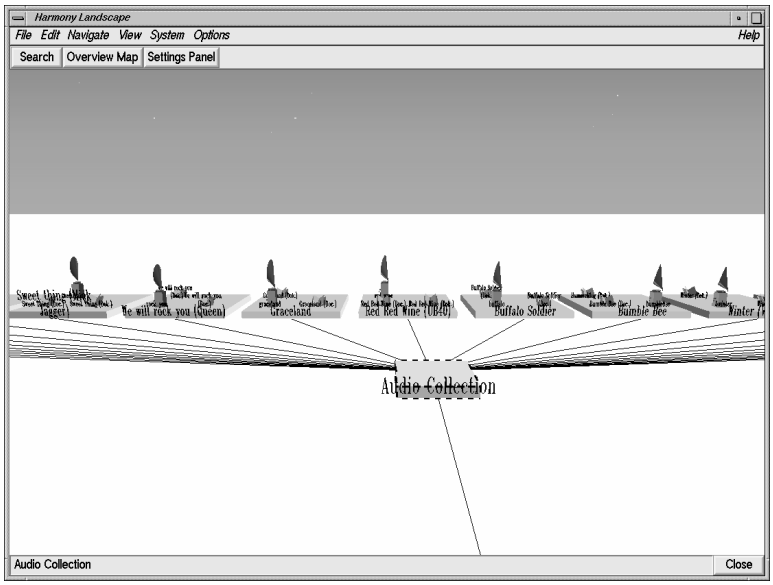

Figure 4.4: View of a collection without fisheye transformation

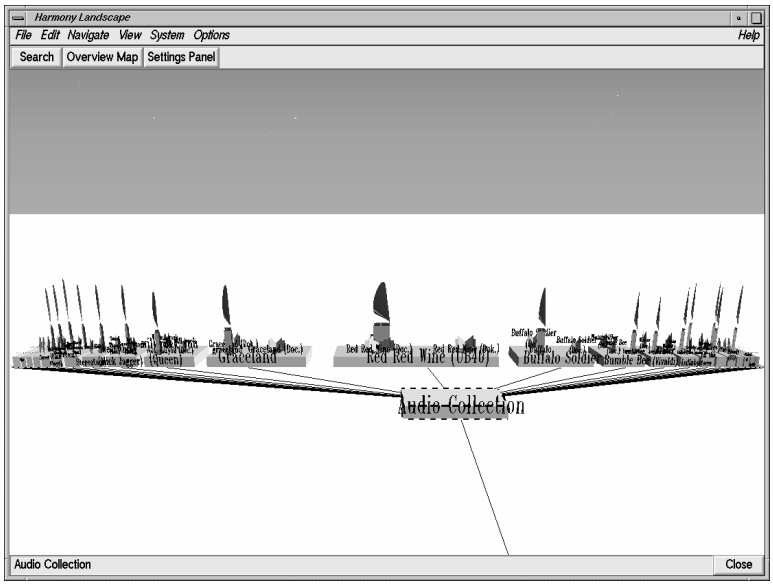

Figure 4.5: View of a collection with fisheye transformation

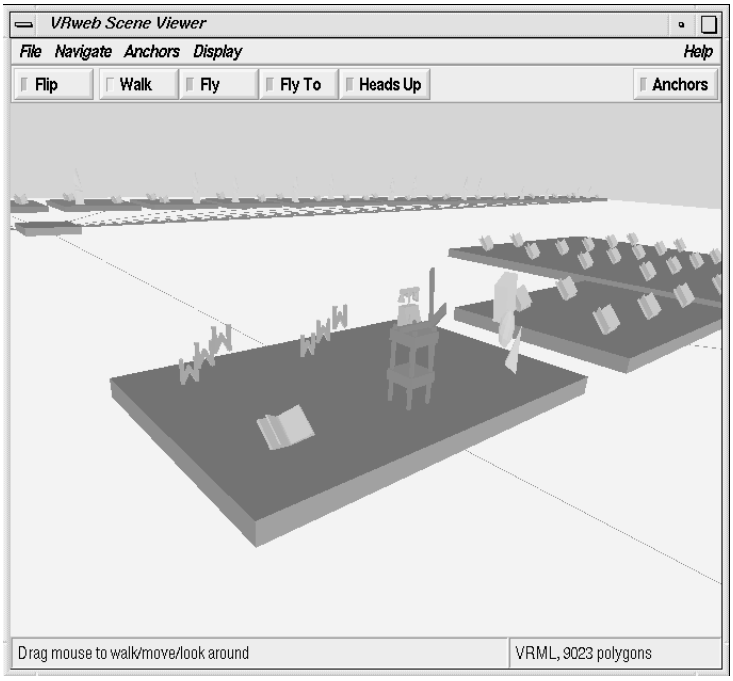

Figure 4.6: VRweb rendering a Landscape scene

and off the three-dimensional local map of the currently selected document.

4.8.2 Set Local Map (3D)

Set Local Map (3D) (Figure 4.8) is quite similar to *Options* in the *Harmony Local Map* [DH95]. The user can set the depth of incoming and outgoing links, as well as the types of links to be visualised. As the number of nodes in the link tree generally grows exponentially with the depth of the tree, the depth should only be increased in steps of one.

If *Open all links* is activated, every node found when building the link tree is also inserted into the collection tree . If *Display link tree only* is activated, there are no wires drawn that connect nodes in link and collection tree when they are the same. This can avoid having confusingly many lines when a lot of links are visualised.

Max. Nodes determines the maximum number of links per document that are visualised with an icon. All other links are only indicated by the wires that usually

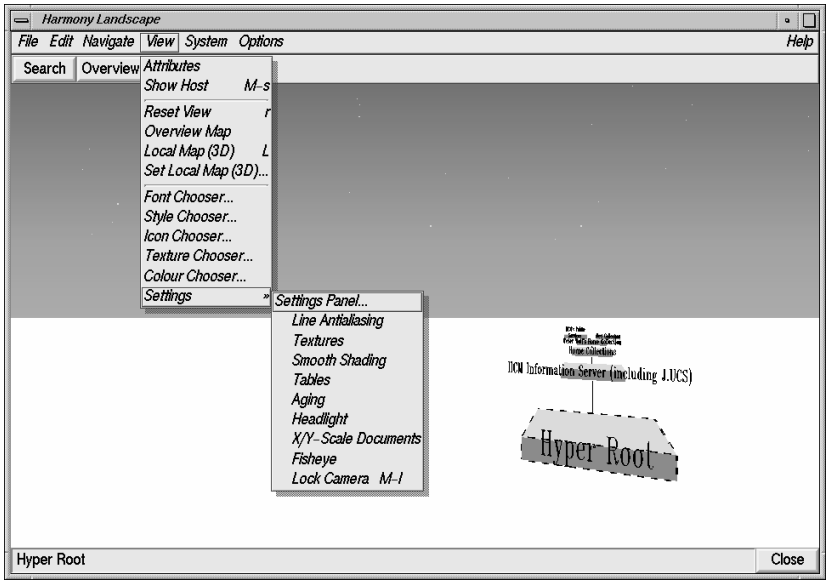

Figure 4.7: The View Menu

connect the icons with the collection tree. If this value is set to zero, all links are only visualised by a wire. This can sometimes provide a better overview, especially when a huge link tree is visualised. The icon of the selected node is always drawn.

4.8.3 Settings Panel

The former *Scale Dialog* is now called *Settings Panel* (Figure 4.9) as a number of new parameters can be modified by it. The panel now contains the following scrollbars:

- *Collection Title Size* scales collection and cluster titles.
- *Document Title Size* scales all other titles.
- *Object Height* scales the document icons.
- *Speed* changes the speed of movement.
- *Frame Rate* sets the minimum number of frames per second.
- *Lighting* changes the intensity of the lightsources.

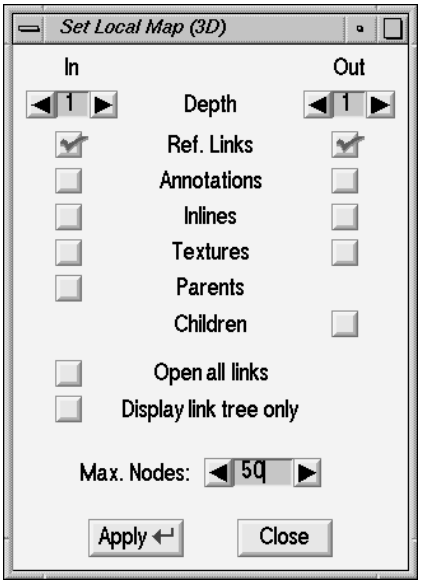

Figure 4.8: The Set Local Map (3D) Menu

- *Aging* determines how fast documents darken according to their age.
- *Link Height* scales the height of one link level.
- *Fisheye Border* sets the maximum distance that objects may have from the camera in *Fisheye* mode.
- *Fisheye Distortion* sets the distortion parameter in *Fisheye* mode.

4.8.4 Style Chooser

The Style Chooser (Figure 4.10) helps the user to quickly change the appearance of the Landscape by assigning predefined sets of textures and icons to all node types and to the background. At the moment seven different styles have been defined:

- Block
  The icons for all document types simply are cubes. This style is primarily intended for slow machines as it is only necessary to render six faces per node.

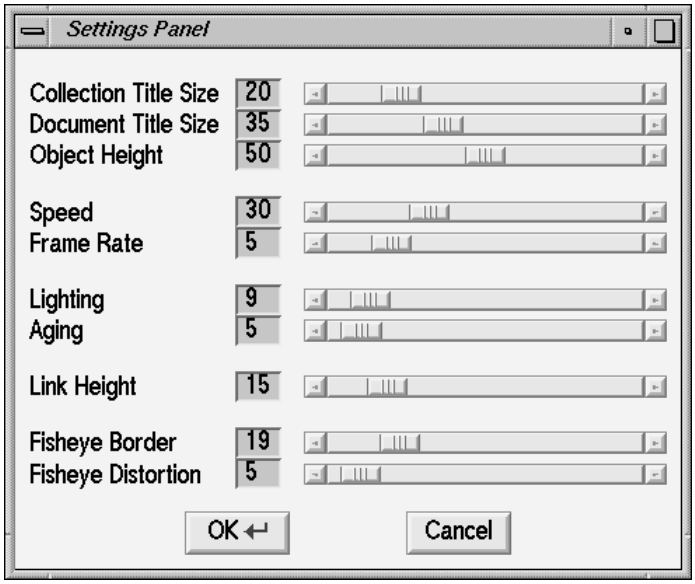

Figure 4.9: The Settings Panel

- Abstract
  Different abstract geometric icons like pyramids, cones, spheres, etc. are used to represent different node types. Using this style, the user can distinguish documents of different types. As those abstract shapes are no real-life metaphors for the document types first-time users need some practicing time to intuitively connect a certain shape with a certain node type. The rendering of those abstract icons is still quite fast as most icons only contain a few faces.

- Nice
  This style assigns rather complex icons, mostly real-life metaphors, to the document nodes. A camera for example symbolises a movie node, a gramophone represents a sound node, and a book marks a text node. As many of these icons take quite some time for rendering, this style should only be used on fast machines.

- Signs
  Is a style that uses icons similar to traffic signs. Attached to the signs are textures that represent the document type.

- Manhattan
  Selecting this style applies a street-grid texture to the ground plane and various 'skyscraper' textures to document nodes (Figure C.2).

- Meadow
  Is a second textured style with leaves on the ground plane. The icons used for the documents are the same as those of the *nice* style, but in addition to that, various textures fitting the shapes of the icons are applied to them (Figure C.3).

- Default
  Changes textures and shapes back to their default settings. Those settings can be defined as X Resources in the *Harmony* file (see Appendix B).

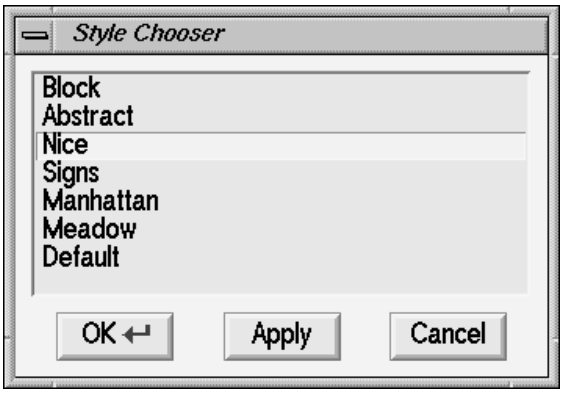

Figure 4.10: The Style Chooser

4.8.5 Icon Chooser and Texture Chooser

The *Icon Chooser* and the *Texture Chooser* both have been designed consistently with the *Colour Chooser* [Eyl95]. Texture or icon changes can be applied using the Apply button. If Cancel is pressed to end the Icon or Texture Chooser those changes are undone again. To commit the current settings, the OK button has to be pressed.

Icon Chooser

The shape of every document type in the Landscape can be changed by using the *Icon Chooser* (Figure 4.11). The user selects a node type from the list on the left-hand side of the chooser and then selects the desired new icon by dragging the slider underneath the preview window.

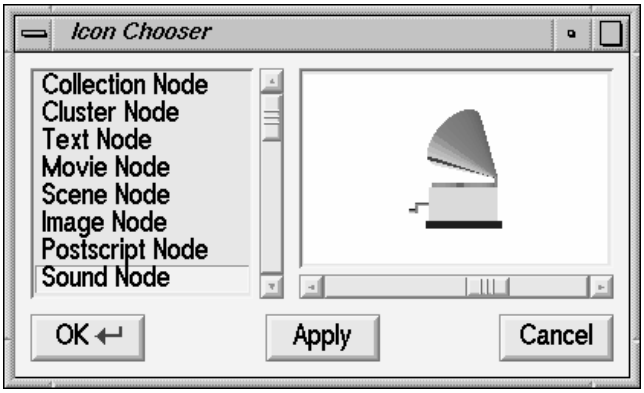

Figure 4.11: The Icon Chooser

Texture Chooser

When using the textured Landscape, the texture for each document type can be chosen in the fileselector of the *Texture Chooser* (Figure 4.12). The selected texture is shown in a preview window. The Landscape currently only supports TIFF images to be used as textures.

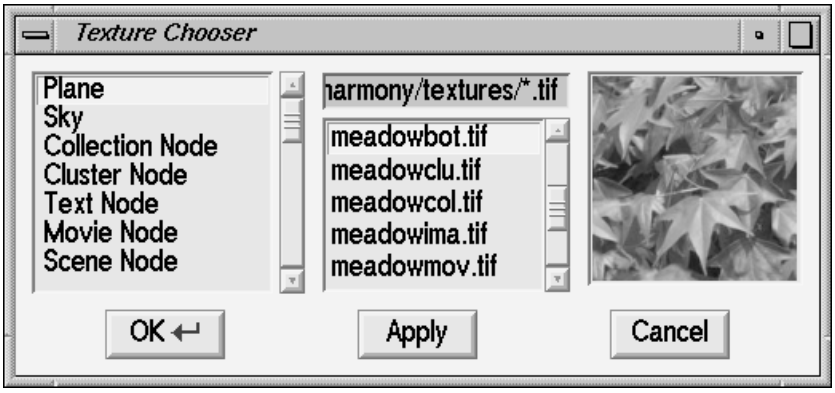

Figure 4.12: The Texture Chooser

Settings

In the Settings submenu (Figure 4.7) the following functions can be toggled:

- *Line Antialiasing* is very time-consuming and should only be used on fast machines.
- *Textures* switches texturing on and off. Rendering the textured Landscape can take a very long time on slow machines. It has, however, been speeded up compared to previous releases and may even achieve reasonable frame rates on average machines, if only a small window is used for the Landscape.
- *Smooth Shading* switches smooth shading on and off.
- *Tables* places icons of a constant size on tables. The tables grow with the size of the document represented by the icon. This is one of the options designed to prevent distortion of the predefined icons.
- *Aging* switches the encoding of the documents age as their icons' brightness on and off.
- *Headlight* switches an additional spotlight on and off. The light is located at the camera's position and directed at the users point of interest.
- *X/Y-Scale Documents* lets icons not only grow in height but also in width, depending on the size of the document they represent. This is the second layout concept that prevents distortion of the icons. It looks nicer than the tables-concept, but obviously consumes more space.
- *Fisheye* applies the cartesian fisheye transformation described in Section 4.6 to the icons x- and y- coordinates.
- *Lock Camera* toggles between the old navigational method with a fixed viewing direction and the new one where the user can turn to all directions.

Chapter 5

Implementation

My work further extends and refines Martin Eyl's implementation of the first version of the Harmony Landscape. So this chapter mainly describes what has changed from the state of the program introduced in Chapter 6 of his thesis [Eyl95]. The last section of this chapter contains a some ideas, about how the Landscape could further be improved.

5.1 System Requirements

The Harmony Information Landscape is implemented tightly coupled with the Harmony Session Manager. It is programmed in C++ [Str91] under UNIX, using the InterViews [LC+92] X11 user interface toolkit for building the menus, and IICM's device independent graphics library GE3D [Pic93, Eyl95] for the actual graphical output. GE3D is used as an interface between the Landscape and the graphic library of any platform supported by the Landscape. GE3D is available for OpenGL, the standardised 3-D graphics interface, as well as for Mesa [Pau96], a software graphics library that provides most of OpenGL's functionality without needing any special hardware. Thus the basic requirement for running the Landscape is a UNIX workstation with X11 window system. If however a system supports OpenGL, the Landscape can take advantage of this platform's higher graphics performance.

5.2 View3D and Node3D

View3D is derived from the Session Manager's *Graph* class. It is the main class of the Landscape that uses all other classes to accomplish its tasks. The most important tasks of *View3D* are to produce the 3-D layout of a Landscape scene, to initiate the scene's rendering, and to control the interaction with the user. Every time nodes or links are added to or deleted from the Landscape or when the scaling of the documents is changed, *View3D* has to calculate the new layout. When the scene is rendered, *View3D* draws the background (i.e. groundplane and sky) and ensures that all nodes are drawn in the right order, from the back to the front. It also is responsible for controlling the frame rate and adjusting it to the users preferences.

Node3D, derived from the Session Manager's *IvNode* and *Node* classes, is basically responsible for the handling of single nodes of all types. For each node type it has to get the right icon and texture. *Node3D* also contains the routines for rendering the icons with or without textures. It stores each node's position and it determines the scaling of the icons according to the size of their corresponding documents. Moreover *Node3D* calculates the colour of each icon from the age of the document it represents.

5.2.1 Reducing the Cost of Drawing

Based on ideas of the *i3D* VRML-viewer [i3D95], a time-dependent rendering algorithm for reducing the cost of drawing in various steps has been implemented in the Landscape. This algorithm allows the user to define a desired minimum frame rate. This frame rate is achieved by controlling the level of detail used for drawing each icon. Depending on the difference between the desired and the last achieved frame rate the level of detail is increased or decreased.

Basically, the level of detail for drawing any icon is determined by the node type it represents, and by the distance between icon and camera. The highest level of detail is used for the selected node, fewer details are drawn for collections and clusters and even fewer for document nodes. Additionally, the greater the distance between a given icon and the camera, and the greater the difference between the previously achieved and the desired frame rate, the lower the level of detail for this icon. The various steps for drawing a document from the highest to the lowest level of detail are:

- The icon is drawn smooth shaded, document titles are drawn with letters in line-width 2.
- The icon is drawn smooth shaded, document titles are drawn with letters in line-width 1.
- The icon is drawn flat shaded, document titles are drawn with letters in line-width 1.
- The icon is drawn flat shaded, no document titles are drawn.
- Only the wireframe of the icon is drawn (not used for clusters and collections).
- Instead of an icon only a flat square is drawn.
- Nothing is drawn at all (usually only used for very distant icons).

As long as the desired frame rate can not be achieved, the distances for the various levels of detail are reduced. Thus the quality of the rendered image decreases as the frame rate increases. When the user stops moving, the last frame is drawn in full detail again.

The relative distances for switching between the levels of detail are defined at the beginning of *node3d.C* and can easily be modified there:

```
// Document limits
const int distitledocwidth  = 90;    // limit for titles in linewidth 2
const int disflatdoc        = 1000;  // limit for flatshadeing
const int distitledoc       = 1450;  // limit for drawing the title
const int diswiredoc        = 1600;  // limit for wireframe
const int disdoc            = 1750;  // limit for drawing a square only
const int disnodoc          = 3000;  // limit for drawing nothing
// Collection limits
const int distitlecollwidth = 150;   // limit for titles in linewidth 2
const int disflatcoll       = 1750;  // limit for flatshadeing
const int distitlecoll      = 2000;  // limit for drawing the title
const int discoll           = 2200;  // limit for drawing a cube
const int disnocoll         = 4000;  // limit for drawing nothing
// Cluster limits
const int distitlecluswidth = 150;   // limit for titles in linewidth 2
const int disflatcluster    = 1750;  // limit for flatshadeing
const int distitlecluster   = 2000;  // limit for drawing the title
const int disclus           = 2200;  // limit for drawing a cube
const int disnoclus         = 4000;  // limit for drawing nothing
```

5.2.2 The Link Tree

For the layout of the link tree two problems had to be considered. On the one hand a lot of space is necessary for this tree, especially when more link levels have to be drawn. Within one link level, the size of nodes can be very different. So sophisticated placement algorithms could save a lot of space. On the other hand the relative position of nodes within one link level should not change, even if the size of one node changes (e.g. because its links are switched on or off). Otherwise the user might become disoriented and confused.

The layout for a link level that resulted from these considerations is a square shaped array containing the icons of this level's nodes (Figure 5.1). Each node is assigned a certain relative position in this array. This position never changes during one session. Each row and column in the array demands just as much space as the biggest icon in it.

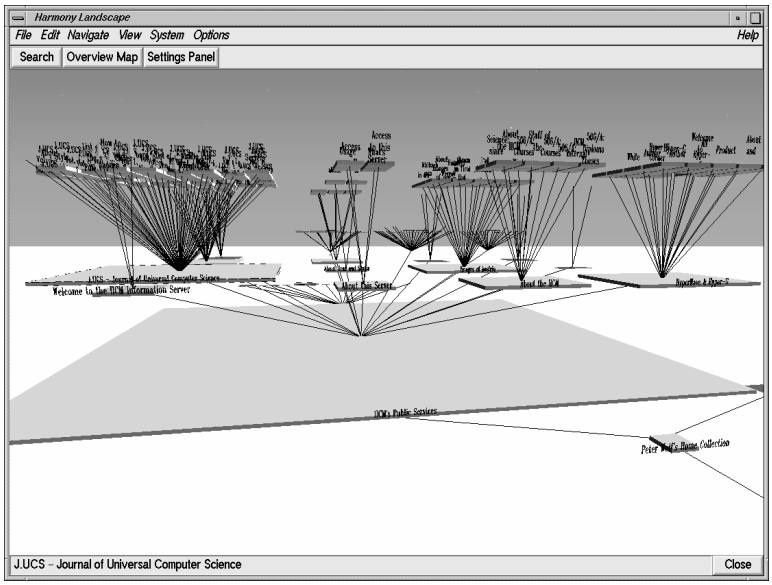

Figure 5.1: A link tree with two levels and more than 100 nodes

5.2.3 Picking 3-D Objects

Picking in the Landscape works for both lines and nodes. To keep the performance high, the picking-algorithm for icons only checks their bounding box, and not their possibly very complex shape for hits. The algorithm for picking wires had to be

modified to be able to pick arbitrarily oriented lines. In the old version of the Landscape wires only connected icons on the groundplane, so the old algorithm only supported picking of lines on the groundplane. The new algorithm also needs to pick wires of the link tree, which can have any orientation.

5.2.4 Drawing the Landscape without a Z-Buffer

The Landscape now allows the user to look in any direction, even though it still does not use a z-buffer for performance reasons. Without a z-buffer, the correct rendering of a three-dimensional scene can only be achieved by an algorithm which ensures that all graphic primitives are always drawn from the back to the front of the scene. Due to the predictable structure of the Landscape an algorithm like this, which is much faster than software z-buffering, could be implemented.

Figure 5.2 illustrates the sequence in which nodes in the groundplane have to be drawn. When rendering the scene, the collection tree has to be drawn level by level. If the user looks away from the root towards the leaves of the collection tree, collections and clusters are drawn beginning with the leaves and ending at the root. If the user looks into the opposite direction, from the leaves to the root, the tree-levels too have to be drawn in the opposite direction, from root to leaves.

Within one tree-level, the level's collections and clusters are drawn from left to right until the camera's x-position is reached, the remaining collections and clusters in this level are then drawn from right to left. The same mechanism is used for documents within collections or clusters, and also for all nodes within one level of the link tree. For each node in the collection tree, the link tree is drawn recursively from the deepest link level up to the cameras z-position, and then from the highest link level down to the same z-position. Special care has to be taken to draw the lines connecting different link levels in the right order. Of course, objects behind the camera do not have to be drawn at all.

5.3 Shapes

Shapes is a small program that includes predefined 3-D icons into the Landscape. The process of adding new 3-D icons to the Landscape is quite complicated, but still might be necessary as long as reading VRML-scenes is not supported. The following is a brief description of how to proceed using the Wavefront Modeller:

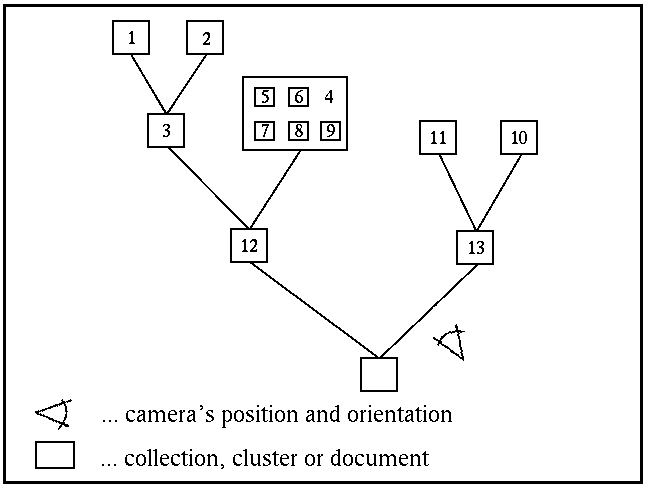

Figure 5.2: Drawing sequence of nodes in the Landscape

1. Design the icon using the *Wavefront Modeller* [Tec91] and save it using the obj-format.
2. Convert the obj-file into a C++ header-file by using the program *obj2inc* with option *normalize1*.
3. Copy the generated header file into the Session Manager's *3dicons* directory.
4. Include the new icon into *shapes.h* and *shapes.C* as it has been done for the other icons.

If other modellers are used to create the icons, conversion tools for step 2 of this procedure, with output as described in Appendix E, have to be implemented.

5.4 Camera3D, GlText and Overview3D

- Camera3D
  The camera class is basically responsible for positioning the camera and defining its reference point. This reference point can either be set explicitly or it can be calculated from the two angles α and β. α describes the angle that the camera looks up and down and can be in the range between 0 (vertically down) and π (vertically up). β is used to define the angle that the camera looks right or left. $\beta = 0$ makes the camera look straight ahead, positive values turn the camera right, negative ones turn it left.

- GlText
  This class now performs a preprocessing step for correct linebreaking and centering of the document titles. Since this way of printing text is more space consuming than simply cutting the words at the end of a line, document titles now can have up to three lines. New start points and positions of the line-feeds for an icon's title are recalculated in a preprocessing step when the size of the letters or the width of the icon changes or when nodes limiting the available space for the title are inserted into the collection tree.

- Overview3D
  Slight modifications have only been made to the overview cursor. The cursor now is an arrow indicating the direction in which the user looks. Moreover, the drawing of the cursor is no longer dependent on the existence of overlay planes. Without overlay planes the whole overview has to be redrawn every time the user's and thus also the cursor's position changes. With overlay planes, which are currently not supported by OpenGL, only the cursor would have to be redrawn.

5.5 Outlook

Most of Martin Eyl's suggestions for further work on the Harmony Landscape have been implemented in the course this thesis. However, there are still a few things left, which have not been implemented so far:

- Collision detection could be implemented so that the user can not fly through objects in the Landscape.

- At the moment the Landscape only presents a view of the information space. But it could also be used to modify the information on the server. For example by taking documents and moving them to other collections.

- Mountains on the horizon could be used to make orientation in the Landscape easier. But probably the drop in rendering-performance caused by additional objects could not be compensated by relatively small orientation advantages.

In addition to these suggestions, I would propose the following ideas for further improvement of the Landscape:

- The current version of the Landscape uses Hershey fonts for text output. It is rather difficult to add new fonts and very complicated to modify existing ones. It is also very difficult to add new 3D icons because the source code has to be changed.

  To solve those problems, a basic VRML parser and renderer, probably extracted from VRweb [POA+95, Pic93], could be included in the Landscape. This viewer should be able to render colours, textures, lines and faces of a VRML file. Now it would be possible to avoid the Hershey fonts, because fonts, even real three-dimensional ones, could easily be built and edited in VRML and then loaded into the Landscape during runtime. At the same time all kinds of 3-D icons could be designed and either be loaded from disk or, via links for single documents, even from VRML-documents on the server.

- As soon as GE3D supports transparency, a semi-transparent groundplane can be used to give a more realistic impression of objects that are underneath the surface.

- The user could be allowed to set special bookmark icons, for example looking like flags or pins, on certain documents to be able to find them again more easily. This would also solve the problem that the Landscape does not provide any real landmarks at the moment. To be good landmarks however, bookmark icons have to be quite big and must be drawn even in great distances from the camera, which would in turn be a drawback for the rendering time. By numbering such bookmarks it would also be possible to create guided tours through the information space.

- Thumbnails could be stored as an additional attribute for image and movie documents. Those thumbnails could be used as textures for their document's icon in the Landscape and thus give the user an idea of the contents of the document.

- A graph connecting previously visited documents could optionally be drawn to create a three-dimensional representation of the history list.

- Similar to the Session Manager, icons of previously viewed documents could be ticked with some 3-D symbol.

- The Landscape could be integrated in Hyper–G's WWW gateway. Users of WWW browsers on any platform could then receive VRML scenes with a three-dimensional representation of the information they are currently accessing. Thus the Landscape could be explored independently of Harmony, with a VRML viewer or Java as the only prerequisite.

5.6 Hints for Further Implementations

When I began my work on the Landscape, it took me some time to become acquainted with the relatively large source code and discover, how new features have to be inserted. Since then the code has grown considerably and so I want to add a few hints that are intended to help future Landscape programmers find out where to begin their work.

- The background of the Landscape is drawn in `View3D::sunSetBackground`, so features like mountains on the horizon have to be inserted here.
- Fonts are loaded and rendered in `text3d.C`, so this file has to be adapted for using fonts which are loaded from VRML files.
- All 3-D icons are actually rendered by `Node3D::drawCube`. If VRML scenes are used as icons, their rendering should also be done in this procedure.
- `Node3D::draw`, `CollectionNode3D::draw`, and `ClusterNode3D::draw` determine the level of detail for a single node. Accordingly they call `drawCube` and the functions for rendering the document titles. The marking of viewed documents, landmarks that have to be drawn even in great distances from the camera and also bookmarks could be implemented here.
- `View3D::saveVRML` and `Node3D::saveVRML` are used for creating VRML scenes of the Landscape's current layout. They can be used as a starting-point for integrating the Landscape into the WWW gateway.

Chapter 6

Conclusions

Ever increasing computer performance made it possible to create huge, networked databases containing interlinked multimedia information. Hyper–G is an information system that helps to build, structure and maintain such a database. For effective interaction between a user and the vast amount of accessible information, powerful tools are necessary. The Harmony Information Landscape provides such a tool using the rich underlying data model defined by Hyper–G. For the purpose of intuitive information access, it uses spatial metaphors to create a three-dimensional interactive environment representing the structure and content of Hyper–G databases. The basic functionality for visualising the hierarchical information structure on a Hyper–G server has already been implemented in the first version of the Harmony Information Landscape. The new version that has been developed in the course of this thesis adds a whole range of new features to the basic implementation.

These features include the further extension of the visualisation of document attributes. The type of a document can now be encoded through a corresponding 3-D icon, a texture, or a combination of both. The colour of untextured icons is used to reinforce the document type and the brightness of the colour can optionally be used to visualise the age of a document. Users can either apply a set of predefined combinations of icons and textures or they can manually define shape, colour and texture of each document type. For selected documents, incoming and outgoing links can now be visualised. Users can define which types of links shall be shown in a link tree. They can determine the depth of the link tree and can decide whether or not documents in the link tree also shall be inserted into the collection tree.

The navigational facilities in the Landscape have been extended and now al-

low the user to turn around freely and view the Landscape from all directions and perspectives. For orientation, the overview map now shows both the location and viewing direction of the user. To guarantee short response times during movement, all rendering procedures have been optimised and an algorithm ensuring a minimum frame rate has been developed and implemented.

Even the work in the course of the second thesis on the Landscape could not exploit all possibilities of this concept of three-dimensional information visualisation. There are still a number of good ideas for extensions and improvements. The most important one probably is the integration of the Landscape into Hyper–G's WWW gateway. This would make it possible to use the Landscape on any machine and platform. Until then the Landscape is only available as a part of Harmony using the X11 window system on UNIX workstations.

Appendix A

Harmony Landscape User's Guide

A.1 Basics

The Harmony Information Landscape visualises the contents of a Hyper–G server in a three-dimensional, navigable environment. Every object (document, collection or cluster) is mapped to a three-dimensional icon which is placed in the Landscape according to the objects location in the Hyper–G collection hierarchy.

A.1.1 The Selected Object

The selected object is indicated by a dashed bounding box drawn around it. If a node is represented more than once in the collection and link trees, other representations of the selected node are indicated by a darker dashed bounding box around them. A new object can be selected using the mouse (see Section A.2) or various keyboard commands (see Section A.3).

A.1.2 Activating an Object

Activating a collection means that a closed collection is opened and an open one is closed. An activated cluster displays the documents that are part of the cluster and presents its documents in the desired language. If a document is activated, it is displayed in the corresponding viewer.

A.1.3 Landscape and Session Manager

The Landscape is a part of Harmony and tightly coupled with the Harmony Session Manager. If objects are inserted, deleted, selected or activated this always happens simultaneously in the Landscape and in the Session Manager. Thus search dialog, history list, and all navigation aids of the Session Manager can be used for the Landscape too.

A.2 Mouse Handling

If the mouse is moved across the Landscape window without any buttons pressed, the status line shows the name of the currently selected node. If there is another node or a wire located under the mouse pointer, the name of this node or the destination of the wire is shown.

A single click on a node or wire selects the node or the wire's destination. Clicking an object with the right button displays its attributes. Double-clicking an object with the left button activates it. Double-clicking an object with the right mouse button starts an animation towards it, while double-clicking it with the middle mouse button starts an animation towards the object's parent.

Double-clicking a wire with the left or middle button starts an animation towards its destination or source respectively.

Moving, turning and tilting of the camera can be achieved by dragging the mouse with one mouse button pressed. All possible modes of movement are listed in Table A.1.

A.3 Keyboard Commands

All available key commands and their resulting actions are listed in Table A.2.

State	*Button*	*Movement*	*Resulting Action*
	left	up/down	move forward/back
	left	left/right	turn around left/right
camera locked	left	left/right	move left/right
	middle	up/down	move up/down
	middle	left/right	move left/right
	right	up/down	tilt camera up/down
	right	up/down	tilt camera left/right
Ctrl	middle		fly towards POI
Ctrl	right		fly away from POI
animation	middle		stop animation
animation	right		jump to final destination

Table A.1: Mouse actions for movement

Key	*Resulting Action*
SPACE	fly to selected object
RETURN	activate selected object
+	open collection / display document or cluster
–	close selected collection
PageUp	select first object one link level higher
PageDown	select first object one link level lower
↑	select first object in next collection tree level
↓	select parent collection
←	select left neighbour
→	select right neighbour
Shift-↑	select first child of collection or cluster
Shift-↓	select parent collection or cluster of document
c	activate current collection or cluster
Ctrl-c	close all
f	show frame rate
g	go to
h	history list
H	home
l	show 2D local map for the selected node
L	insert 3D local map for the selected node
Meta-l	lock camera
r	reset view
s	search-dialog
S	save current scene as VRML-file
t	toggles 'Link Tree Only' for the selected object

Table A.2: Keyboard commands

A.4 Dialogs, Menus, and Buttons

Nearly all Landscape dialogs can be found in the *View* menu. Only *Save As VRML* is located in the *File* menu. Most of the dialogs contain the three buttons *OK*, *Apply* and *Cancel*. Pressing the Apply button temporarily assigns the dialogs' current settings to the Landscape. Pressing the Cancel button ends the dialog and undoes all changes that have been made with it. By pressing OK the dialog is closed and all changes are applied permanently.

A.4.1 Reset View

Reset View in the *View* menu moves the camera back to its initial position at the root of the collection tree and turns the camera's viewing direction towards the leaves of the tree.

A.4.2 Overview Map

The *Overview Map* can either be opened by clicking the *Overview Map* button in the Landscape window or by selecting *Overview Map* in the *View* menu. The map shows a birdseye view of the current Landscape. View point and viewing direction of the main window are indicated by an arrow. The tip of the arrow shows the view point while its direction indicates the viewing direction. Clicking any location on the map with the left mouse button makes the view point in the main window and the arrow in the map move towards this location. Apart from that, the mouse handling in the Overview Map is similar to the mouse handling in Landscape's main window (see section A.2).

A.4.3 Local Map (3D)

Local Map (3D) generates a link tree for the selected object using the settings defined in *Set Local Map (3D)* (see below). For objects in the collection tree both incoming and outgoing links are visualised. For objects in an incoming link tree only further incoming links can be visualised. For objects in an outgoing link tree only outgoing links can be visualised.

A.4.4 Set Local Map (3D)

Set Local Map (3D) (Figure A.1) can be used to set the depth of incoming and outgoing links, as well as the types of links to be visualised in the link tree. As the number of nodes in the link tree generally grows exponentially with the depth of the tree, the depth should only be increased in steps of one.

If *Open all links* is activated, every node found when building the link tree is also inserted into the collection tree. Usually nodes in link and collection tree are connected by a wire when they are the same. These wires are not drawn if *Display link tree only* is activated. For the selected object's link tree, the drawing of these wires can also be toggled by pressing 't' (see Table A.2).

Max. Nodes determines the maximum number of links per document that are visualised with an icon. All other links are only indicated by the wires that usually connect the icons with the collection tree. If this value is set to zero, all links are only visualised by a wire.

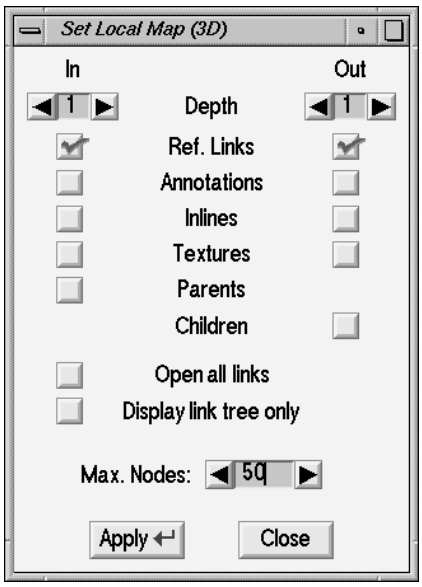

Figure A.1: The Set Local Map (3D) Menu

A.4.5 The Font Chooser

For drawing the titles of documents, clusters, and collections one out of twelve fonts can be selected in the Font Chooser (Figure A.2). On the right-hand side of the Font Chooser a preview of the currently selected font is drawn.

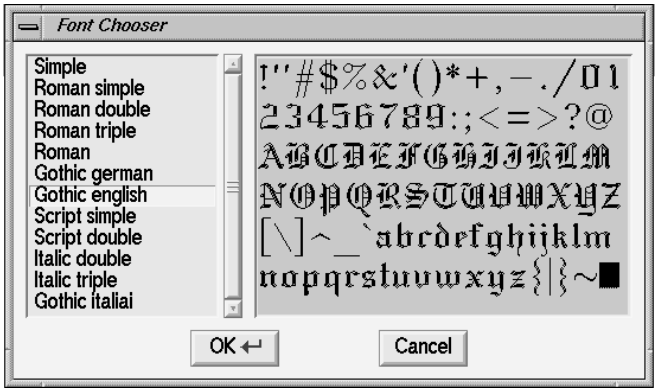

Figure A.2: The Font Chooser

A.4.6 The Style Chooser

The Style Chooser (Figure A.3) can be used to quickly apply a predefined set of icons and textures to all objects and to the background of the Landscape. Currently seven different styles are available:

- *Block* uses simple cubes as icons and is intended for slow machines.
- *Abstract* represents different node types by different geometric icons. The rendering of these simple icons is still quite fast on average machines.
- *Nice* assigns complex icons, mostly real-life metaphors, to the document nodes. As the rendering of these icons is very time consuming, this style should only be used on fast machines.
- *Signs, Manhattan* and *Meadow* are textured styles and should only be used on fast machines.
- *Default* changes all icons back to their default settings defined in the *Harmony* file (see Section A.5).

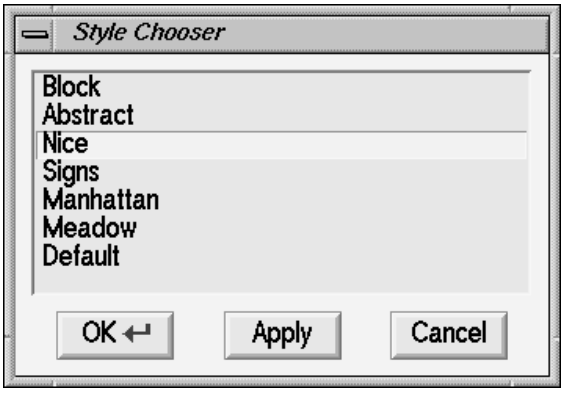

Figure A.3: The Style Chooser

A.4.7 The Icon Chooser

The shape of every document type in the Landscape can be changed by using the Icon Chooser (Figure A.4). First a node type has to be selected on the left-hand side of the chooser. Then desired new icon for this node is selected by dragging the slider underneath the preview window.

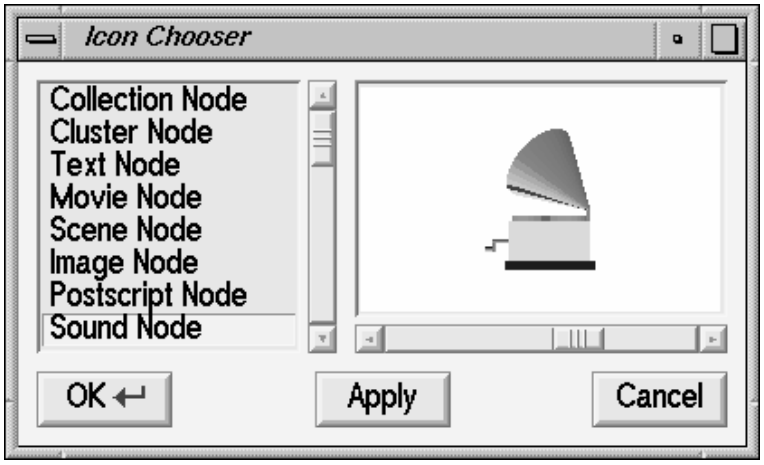

Figure A.4: The Icon Chooser

A.4.8 The Texture Chooser

In the textured Landscape, the texture for the background and each document type can be chosen in the fileselector of the Texture Chooser (Figure A.5). The selected texture is shown in a preview window.

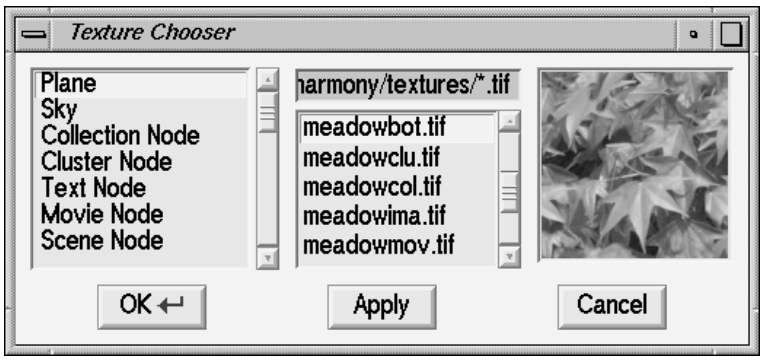

Figure A.5: The Texture Chooser

A.4.9 The Colour Chooser

The Colour Chooser (Figure A.6) can be used to change the colour of collections, clusters, wires, sky, ground plane, and all document types.

A.4.10 The Settings Submenu

The Settings submenu can be used for toggling:

- *Line Antialiasing*: should only be used on fast machines.
- *Textures*: should only be used on fast machines.
- *Smoot Shading*
- *Tables*: puts all document icons on tables. Only the tables grow with the document size.
- *Aging*: encodes the age of objects as their icon's brightness.

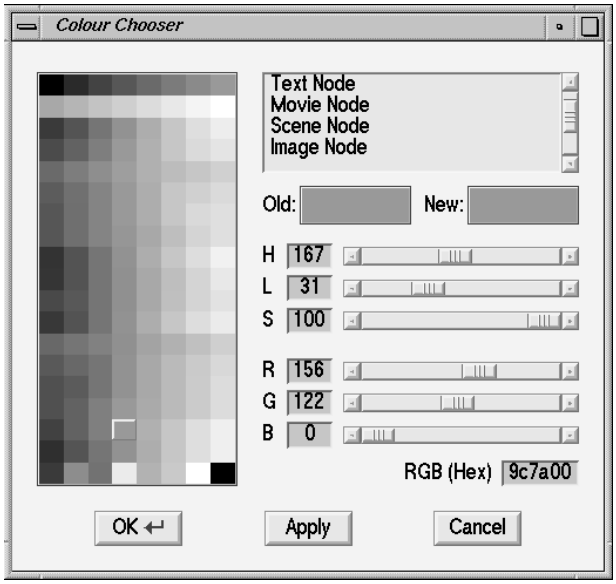

Figure A.6: The Colour Chooser

- *Headlight*
- *X/Y-Scale Documents*: lets icons not only grow in height but also in width, depending on the size of the document they represent.
- *Fisheye*: switches a fisheye transformation on. This transformation helps to pack more information into the available screen space. Parameters for the fisheye transformation can be changed on the *Settings Panel* (see below).
- *Lock Camera*: locks the camera's current viewing direction. If the camera is locked and the user moves left or right, the camera does not turn to the moving direction. It only tilts sidewise left or right.

A.4.11 The Settings Panel

The *Settings Panel* A.7 can be opened in the Settings submenu or by pressing the *Settings Panel* button in the Landscape window. The panel contains the following scrollbars:

- *Collection Title Size* scales collection and cluster titles.
- *Document Title Size* scales all other titles.

- *Object Height* scales the size of document icons.
- *Speed* changes the speed of movement.
- *Frame Rate* sets the minimum number of frames per second.
- *Lighting* changes the intensity of the lightsources.
- *Aging* determines how fast documents darken according to their age, if *Aging* in the *Settings* submenu is activated.
- *Link Height* scales the height of one link level.
- *Fisheye Border* sets the maximum distance that objects may have from the camera in *Fisheye* mode.
- *Fisheye Distortion* sets the distortion parameter in *Fisheye* mode.

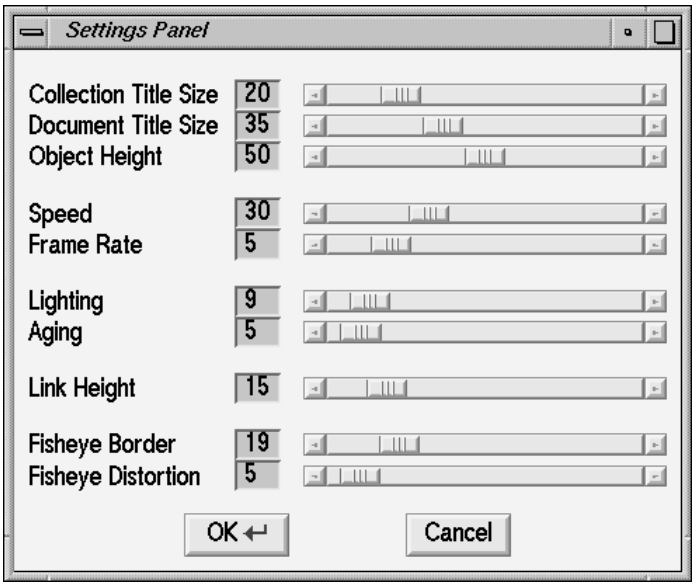

Figure A.7: The Settings Panel

A.4.12 Save as VRML

Save As VRML in the *File* menu is used to save a snapshot of the Landscape's nodes and the wires connecting them as a VRML file. In the Save As VRML dialog box only the desired filename has to be specified.

A.4.13 The Close Button

On the lower right corner of the Landscape and Overview window there is a *Close* button which terminates the window.

A.4.14 The Stop Button

If the Landscape handles a very time consuming request like opening a large collection or generating a complex link tree, there is a *Stop* button in the lower right corner of the Landscape window. Pressing this button terminates the request.

A.5 X defaults

Landscape settings which can be defined as X defaults in the *Harmony* file include:

- The colour of the background and of each node type.
- The shape of each node type.
- The texture for each node type.
- The documents' height.
- The fonts used for document titles and the titles' height.
- The minimum frame rate and the speed of movement.
- The Landscape window's position and size.

Appendix B

X Resources

Several settings of the Landscape can be changed by editing the X attributes in the *Harmony* file. The following listing contains all parts of this file that are relevant for the Harmony Landscape. For defining shapes, a number has to be used. Which number corresponds to which shape can be seen in Figure B.1.

```
!! position and size of the Landscape window.
Harmony.Session.Landscape.geometry:             1100x800+40+170

!! scaling factor for document blocks [1..100]
Harmony.Session.Landscape.DocumentHeight:            50

!! scaling factor for text [1..100]
Harmony.Session.Landscape.TitleSizeBig:              10
Harmony.Session.Landscape.TitleSizeSmall:            7

!! scaling factor for movement speed (use a large number on
!! a fast machine)
Harmony.Session.Landscape.Speed: 30

!! minimum number of rendered frames per second
Harmony.Session.Landscape.Framerate: 10

!! main font file
Harmony.Session.Landscape.fontdeffile:               3dfonts/hersh.oc

!! file name of the current font
```

```
Harmony.Session.Landscape.fontfile:                    3dfonts/romanc.hmp

!! font names and file names of all existing fonts
Harmony.Session.Landscape.fontinfofile:                3dfonts/fontinfo.dat

!!
!! color settings for the landscape
!!
Harmony.Session.Landscape.planecolour:                 #83d177
Harmony.Session.Landscape.edgecolour:                  white
Harmony.Session.Landscape.highlightedcolour:           white
Harmony.Session.Landscape.selectedcolour:              #69FFB7
Harmony.Session.Landscape.textcolour:                  red

!!
!! colors for the different document types
!!
Harmony.Session.Landscape.Collection.colour:           #222222
Harmony.Session.Landscape.Cluster.colour:              #001b48
Harmony.Session.Landscape.Text.colour:                 #00ffff
Harmony.Session.Landscape.Image.colour:                #3c0048
Harmony.Session.Landscape.Movie.colour:                #7c5a00
Harmony.Session.Landscape.Scene.colour:                #9c7a00
Harmony.Session.Landscape.Drawing.colour:              #dc78ff
Harmony.Session.Landscape.Remote.colour:               #006968
Harmony.Session.Landscape.Telnet.colour:               #006968
Harmony.Session.Landscape.Sound.colour:                #bc8457
Harmony.Session.Landscape.Generic.colour:              #ff8c05
Harmony.Session.Landscape.PostScript.colour:           #f63637
Harmony.Session.Landscape.GopherColl.colour:           #3c0400
Harmony.Session.Landscape.GopherText.colour:           #777777
Harmony.Session.Landscape.GopherImage.colour:          #3c0048
Harmony.Session.Landscape.GopherMovie.colour:          #7c18fa
Harmony.Session.Landscape.GopherSearch.colour:         #909000
Harmony.Session.Landscape.GopherSound.colour:          #bc8457
Harmony.Session.Landscape.GopherTelnet:                #006968
Harmony.Session.Landscape.WWWNode:                     #ff0000
Harmony.Session.Landscape.WWWText.colour:              #00ff00
Harmony.Session.Landscape.WWWImage.colour:             #3c0048
Harmony.Session.Landscape.WWWMovie.colour:             #7c18fa
Harmony.Session.Landscape.WWWSound.colour:             #bc8457
Harmony.Session.Landscape.WWWPostScript.colour:        #a015c0
Harmony.Session.Landscape.WWWScene:                    #a01510
```

```
!!
!! shapes for the different document types. Number is number of
!! icon in the icon chooser
!!
Harmony.Session.Landscape.Text.shape:                12
Harmony.Session.Landscape.Image.shape:               14
Harmony.Session.Landscape.Movie.shape:               15
Harmony.Session.Landscape.Scene.shape:               18
Harmony.Session.Landscape.Drawing.shape:             14
Harmony.Session.Landscape.Remote.shape:              20
Harmony.Session.Landscape.Telnet.shape:              20
Harmony.Session.Landscape.Sound.shape:               14
Harmony.Session.Landscape.Generic.shape:             7
Harmony.Session.Landscape.PostScript.shape:          17
Harmony.Session.Landscape.WWWText.shape:             23

!!
!! default textures when the textured Landscape is switched on
!! can be loaded by selecting 'default' in the style chooser
!!
Harmony.Session.Landscape.UpperBackTexture:    skytxc.tif
Harmony.Session.Landscape.LowerBackTexture:    grasstxc.tif
Harmony.Session.Landscape.Collection.texture:  marmor1.tif
Harmony.Session.Landscape.Cluster.texture:     marmor2.tif
Harmony.Session.Landscape.Image.texture:       mona.tif
Harmony.Session.Landscape.Movie.texture:       movie.tif
Harmony.Session.Landscape.Sound.texture:       music.tif
Harmony.Session.Landscape.WWW.texture:         text.tif
Harmony.Session.Landscape.WWWText.texture:     www.tif
Harmony.Session.Landscape.PostScript.texture:  text.tif
Harmony.Session.Landscape.Remote.texture:      remote.tif

!! antialiasing  (should be off for slow machines and Mesa binaries)
Harmony.Session.Landscape.lineantialiasing : off

!! texturing (should be off for slow machines and Mesa binaries)
Harmony.Session.Landscape.texturing : off
```

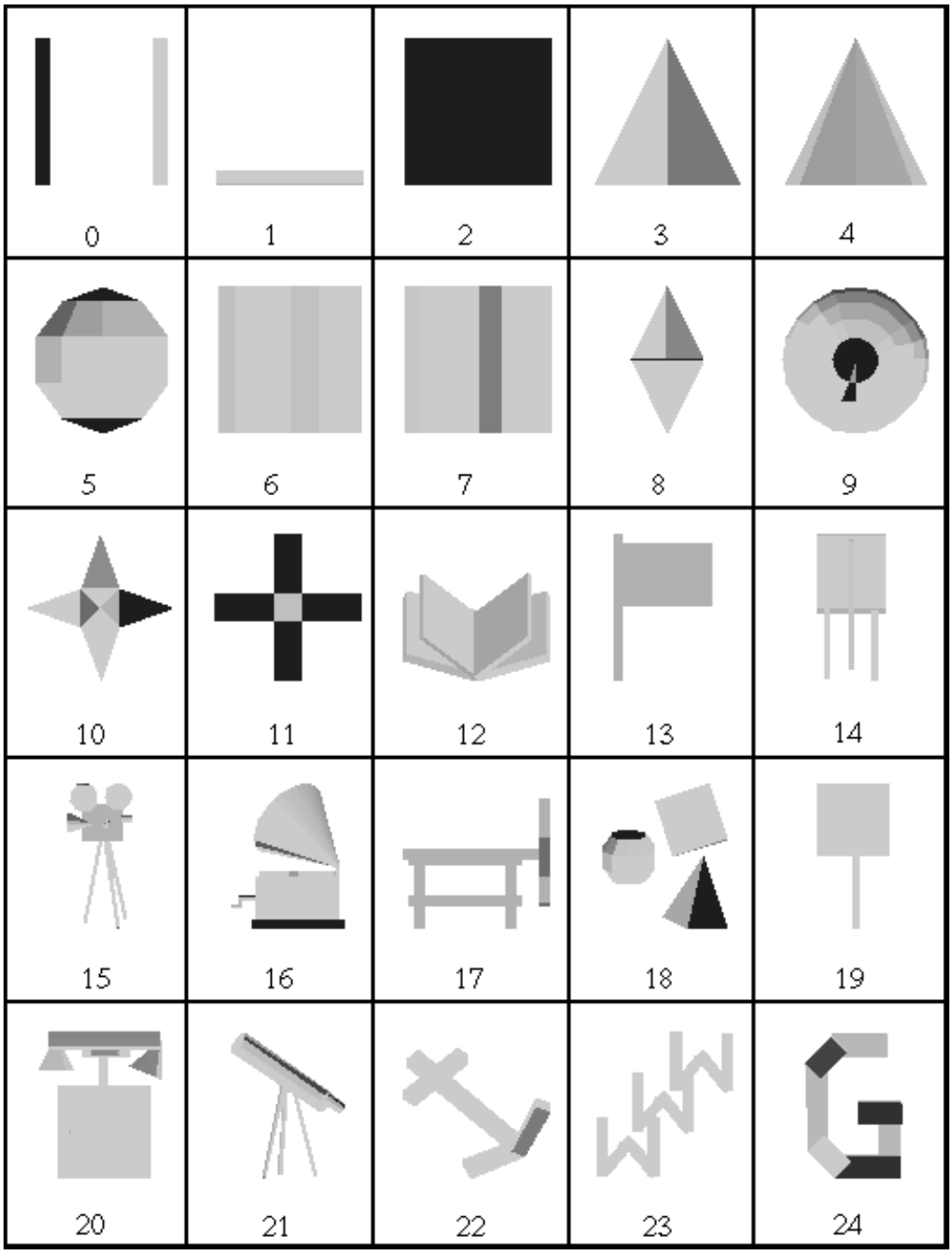

Figure B.1: All currently available 3-D icons

Appendix C

Colourplates

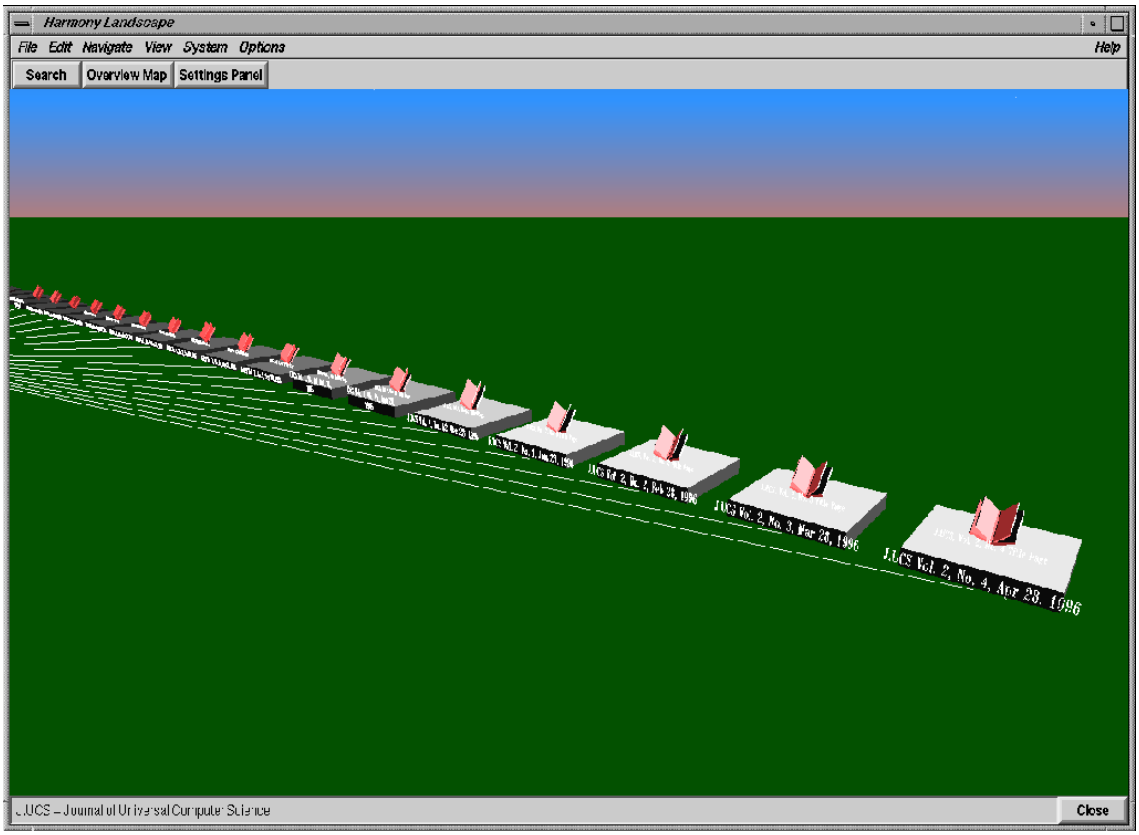

Figure C.1: Visualisation of the age of documents

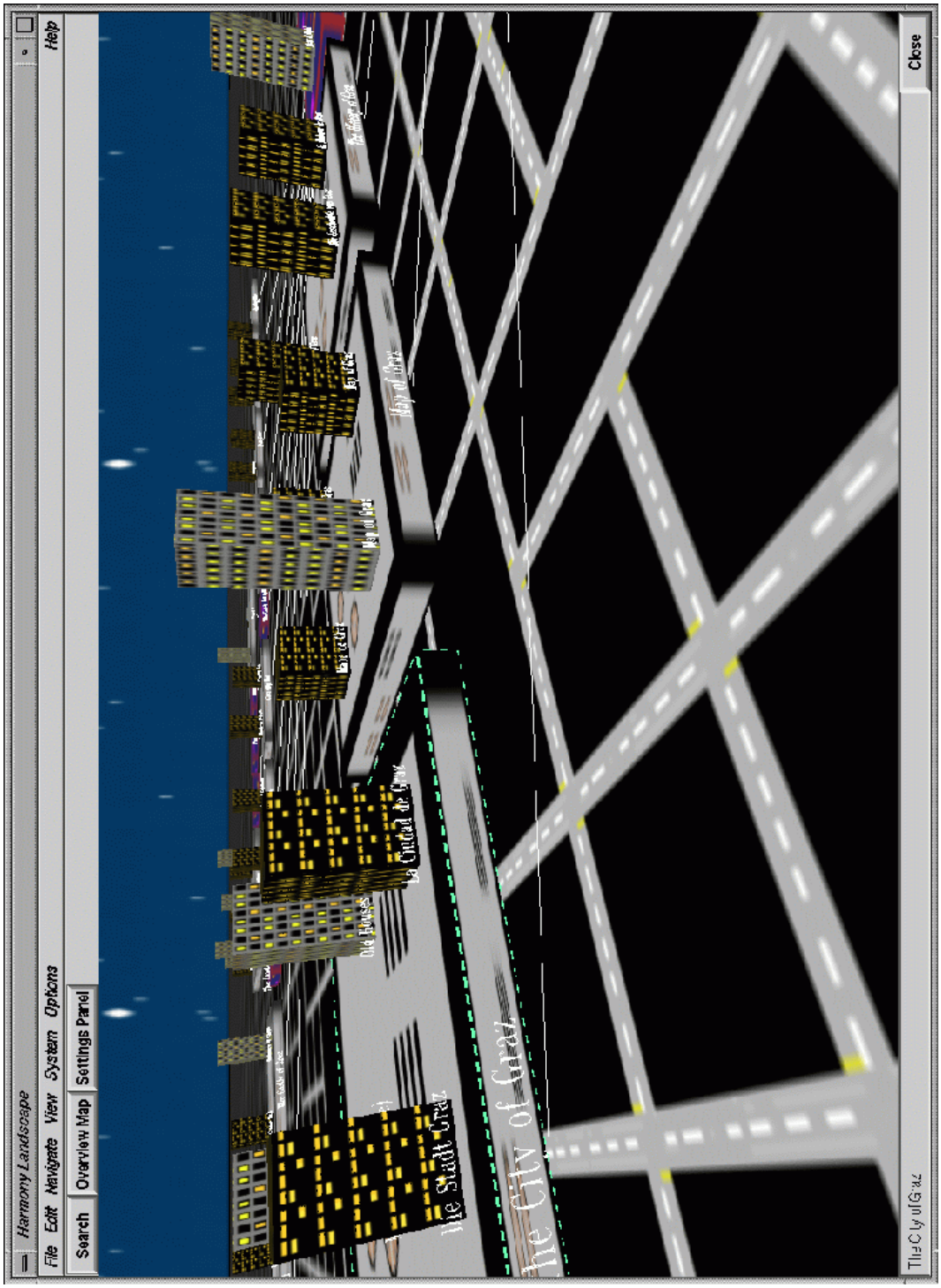

Figure C.2: Landscape with 'Manhattan' texture

Figure C.3: Landscape with 'Meadow' texture

Appendix D

Landscape Files

The Landscape is part of the Harmony Session Manager. The following Session Manager files contain the Landscape's source code:

```
camera3d.C      camera3d.h
dialog3d.C      dialog3d.h
icons.C         icons.h
menus.C         menus.h
node3d.C        node3d.h
overview3d.C    overview3d.h
shapes.C        shapes.h
text3d.C        text3d.h
view_3d.C       view_3d.h
```

The following header files defining the Landscape's 3D icons are stored in the `3dicons` subdirectory:

```
anchor.h        g.h          sphere.h
book.h          movie.h      sphere20.h
cone7.h         phone.h      star.h
cross.h         picture.h    tablelegs.h
cube.h          ps.h         tabletop.h
cylinder.h      pyramid.h    tele.h
cylinder20.h    scene.h      www.h
diamond.h       sign.h
flag.h          sound.h
```

Appendix E

Defining 3-D Icons

The following example shows how a file defining a 3D icon, in this case a cube, has to look like. Usually these files are created by *obj2inc* out of Wavefront object files. They contain vertices for shape and texture with normalised coordinates, indices for building shape and textures, face normals for every face defined by those indices and the number of shape and texture coordinates.

```
/* header file */
/* created with obj2inc (small modification of obj2wrl) */

const point3D verts_cube [] = {
        { 0.000000, 1.000000, 1.000000 },
        { 0.000000, 0.000000, 1.000000 },
        { 1.000000, 0.000000, 1.000000 },
        { 1.000000, 1.000000, 1.000000 },
        { 0.000000, 1.000000, 0.000000 },
        { 0.000000, 0.000000, 0.000000 },
        { 1.000000, 0.000000, 0.000000 },
        { 1.000000, 1.000000, 0.000000 }
};

const point2D texverts_cube [] = {
        { 0.000000, 1.000000 },
        { 0.000000, 0.000000 },
        { 1.000000, 0.000000 },
        { 1.000000, 1.000000 },
        { 0.000000, 1.000000 },
```

```
        { 0.000000, 0.000000 },
        { 1.000000, 0.000000 },
        { 1.000000, 1.000000 },
        { 0.000000, 1.000000 },
        { 0.000000, 0.000000 },
        { 1.000000, 0.000000 },
        { 1.000000, 1.000000 },
        { 0.000000, 1.000000 },
        { 0.000000, 0.000000 },
        { 1.000000, 0.000000 },
        { 1.000000, 1.000000 },
        { 0.000000, 1.000000 },
        { 0.000000, 0.000000 },
        { 1.000000, 0.000000 },
        { 1.000000, 1.000000 },
        { 0.000000, 1.000000 },
        { 0.000000, 0.000000 },
        { 1.000000, 0.000000 },
        { 1.000000, 1.000000 }
};

const int coordinds_cube [] = {
        0, 1, 2, 3, -1,
        7, 6, 5, 4, -1,
        3, 2, 6, 7, -1,
        4, 0, 3, 7, -1,
        4, 5, 1, 0, -1,
        1, 5, 6, 2, -1
};

const int texcoordinds_cube [] = {
        0, 1, 2, 3, -1,
        4, 5, 6, 7, -1,
        8, 9, 10, 11, -1,
        12, 13, 14, 15, -1,
        16, 17, 18, 19, -1,
        20, 21, 22, 23, -1
};

const vector3D facenormals_cube [] = {
        { 0, 0, 1 },
        { 0, 0, -1 },
        { 1, -0, 0 },
```

```
        { -0, 1, 0 },
        { -1, 0, 0 },
        { 0, -1, 0 }
};

int numcoords_cube = sizeof (coordinds_cube) /
sizeof (*coordinds_cube);
int numtexcoords_cube = sizeof (texcoordinds_cube) /
sizeof (*texcoordinds_cube);
```

Bibliography

[Ahl96] Christopher Ahlberg. Information visualization & exploration, April 1996. Available at URL `http://www.cs.chalmers.se/SSKKII/ivee.html`.

[AKM95] Keith Andrews, Frank Kappe, and Hermann Maurer. The Hyper-G network information system. *Journal of Universal Computer Science*, 1(4):206–220, April 1995. Available at URL `http://hyperg.iicm.tu-graz.ac.at/jucs`.

[AKMSr94] Keith Andrews, Frank Kappe, Hermann Maurer, and Klaus Schmaranz. On second generation hypermedia systems. *Journal of Universal Computer Science (Pilot Issue)*, 0(0):127–135, November 1994. Available at URL `http://hyperg.iicm.tu-graz.ac.at/jucs`.

[And94] Keith Andrews. Spatial metaphors for information systems. In *ECHT94 Workshop at the European Conference on Hypermedia Technology*, Edinburgh, Scotland, September 1994. Available at `http://www.gatech.edu/lcc/idt/Faculty/andreas_dieberger/ECHT94.WS.Andrews.html`.

[Ber94] Mark Bernstein. Reading with friends: Actors and agents in reading space. In *ECHT94 Workshop at the European Conference on Hypermedia Technology*, Edinburgh, Scotland, September 1994. Available at URL `http://www.gatech.edu/lcc/idt/Faculty/andreas_dieberger/ECHT94.WS.Bernstein.html`.

[BJ94] Heiner Benking and Anthony J.N. Judge. Design considerations for spatial metaphors. In *ECHT94 Workshop at the European Conference on Hypermedia Technology*, Edinburgh, Scotland, September 1994. Available at URL `http://www.gatech.edu/lcc/idt/Faculty/andreas_dieberger/ECHT94.WS.Benking.html`.

[BL96] Tim Berners-Lee. Hypertext Markup Language (HTML), 1996. Available at URL `http://www.w3.org/hypertext/WWW/MarkUp/`.

[BPP94] Gavin Bell, Anthony Parisi, and Mark Pesce. The Virtual Reality Modeling Language, version 1.0 specification (draft), November 1994. Available at URL `http://vrml.wired.com/vrml.tech/vrmlspec.html`.

[Bus45] Vannevar Bush. As we may think. *The Atlantic Monthly*, 176(1): 101–108, July 1945. Available at URL `http://www.isg.sfu.ca/~duchier/misc/vbush/`.

[Cha93] Mathew Chalmers. Using a landscape metaphor to represent a corpus of documents. *Spatial Information Theory, A Theoretical Basis for GIS*, pages 377–390, 1993.

[CRS85] D. Canter, R. Rivers, and G. Storrs. Characterizing user navigation through complex data structures. *Behaviour and Information Technology*, 4(2):93–102, 1985.

[DA94] Andreas Dieberger and Keith Andrews. Spatial user interface metaphors in hypermedia systems. In *Workshop at the European Conference on Hypermedia Technology*, Edinburgh, Scotland, September 1994. Available at URL `http://www.gatech.edu/lcc/idt/Faculty/andreas_dieberger/Workshop.ECHT94.html`.

[DH95] Wolfgang Dalitz and Gernot Heyer. *Hyper–G Das Internet-Informationssystem der 2. Generation.* dpunkt Verlag für digitale Technologie GmbH, first edition, 1995. Available at URL `http://hyperg.iicm.tu-graz.ac.at/books`.

[Eyl95] Martin Eyl. The harmony information landscape: Interactive, three-dimensional navigation through an information space. Master's thesis, Graz University of Technology, Austria, October 1995.

[FB90] S. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In *Proceedings of UIST '90 (ACM Symp. on User Interface Software and Technology)*, pages 76–83, Snowbird, UT, October 1990. Available at URL `http://www.cs.columbia.edu/graphics/publications/publications.html`.

[Flo95] Udo Flohr. Hyper–G organizes the web. *Byte Magazine*, November 1995. Available at URL `http://www.byte.com/art/9511/sec5/art4.htm`.

[FPF88] Kim M. Fairchild, Steven E. Poltrock, and George W. Furnas. Semnet: Three-dimensional representations of large knowledge bases.

Cognitive Science and its Applications for Human-Computer Interaction, pages 201–233, 1988. Information also available at URL `http://apple.sdsu.edu/People/SemNet_About_SemNet.html`.

[Fur86] George W. Furnas. Generalized fisheye views. In *Proceedings of CHI '86*, pages 16–23, Boston, Massachusetts, April 1986. ACM.

[Hed87] Charles L. Hedrick. Introduction to the Internet Protocols, July 1987. Available at URL `http://mph124.rh.psu.edu/~hunt/tcpip/`.

[i3D95] i3d – A New Dimension to Hypermedia, July 1995. Available at URL `http://www.crs4.it/~3diadm/`.

[KP94] Frank Kappe and Gerald Pani. Hyper–G client/server protocol (HG-CSP), December 1994. Available at URL `http://hyperg.iicm.tu-graz.ac.at/hg-prot-intro`.

[Kro94] Ed Krol. *The Whole Internet: User's Guide and Catalog.* O'Reilly & Associates, second edition, April 1994.

[KW95] Peter D. Kochevar and Leonard R. Wanger. Tecate: A software platform for browsing and visualizing data from networked data sources. *Digital Technical Journal*, 7(3), December 1995. Available at URL `http://www.digital.com:80/.i/info/DTJJ00/`.

[LC+92] M. A. Linton, P. R. Calder, et al. *InterViews Reference Manual, Version 3.1.* The Board of Trustees of the Leland Stanford Junior University, December 1992. Available via anonymous ftp from `interviews.stanford.edu` in directory `man`.

[M+96] Hermann Maurer et al. *HyperWave: The Next Generation Web Solution.* Addison-Wesley, April 1996. Available at `http://hyperg.iicm.tu-graz.ac.at/hgbook`.

[MB95] Tamara Munzner and Paul Burchard. Visualizing the structure of the world wide web in 3d hyperbolic space. In *Proceedings of VRML '95; First Annual Symposium on the Virtual Reality Modeling Language*, pages 33–37, San Diego, California, December 1995.

[McA93] Ray McAleese. *Hypertext: theory into practice.* Intellect Books, Oxford, England, 1993.

[ME95] Mark P. McCahill and Thomas Erickson. Design for a 3d spatial user interface for internet gopher. In *Proceedings of ED-MEDIA 95*, pages 39–44, Graz, Austria, June 1995. Available at URL `http://hyperg.iicm.tu-graz.ac.at/edmedia_papers_ps`.

[Mey96] Jonathan Meyer. The Pad++ home page, 1996. Available at URL `http://www.cs.unm.edu/pad%2b%2b/begin.html`.

[Pau96] Brian Paul. The Mesa 3-D graphics library, March 1996. Available at URL `http://www.ssec.wisc.edu/~brianp/Mesa.html`.

[Pic93] Michael Pichler. Interactive browsing of 3d scenes in hypermedia: The Hyper-G 3D viewer. Master's thesis, Graz University of Technology, Austria, October 1993.

[POA+95] Michael Pichler, Gerbert Orasche, Keith Andrews, et al. Vrweb: A multi-system vrml viewer. In *Proceedings of VRML '95; First Annual Symposium on the Virtual Reality Modeling Language*, pages 77–85, San Diego, California, December 1995.

[RCM93] George G. Robertson, Stuart K. Card, and Jock D. Mackinlay. Information visualisation using 3d interactive animation. *Communications of the ACM*, 36(4):57–71, April 1993.

[Shu90] Simon Buckingham Shum. Real and virtual spaces: Mapping from spatial cognition to hypertext. *Hypermedia*, 2(2):133–185, 1990.

[Str91] Bjarne Stroustrup. *The C++ Programming Language*. Addison-Wesley, Reading, Massachusetts, second edition, 1991.

[Tec91] Wavefront Technologies. Advanced visualizer user's guide, 1991.

[Tro94] Jolanda G. Tromp. Of metaphors and magic. In *ECHT94 Workshop at the European Conference on Hypermedia Technology*, Edinburgh, Scotland, September 1994. Available at `http://www.gatech.edu/lcc/idt/Faculty/andreas_dieberger/ECHT94.WS.Tromp.html`.

[TS92] Joel Tesler and Steve Strasnick. FSN: The 3D File System Navigator, 1992. Available by anonymous ftp from `sgi.sgi.com` in directory `sgi/fsn`.

[uH95] urlHouse, 1995. Available by anonymous ftp from `isy.liu.se` in directory `pub/bb/klas/VRML`.

[WHK94] Alexander Willet, Matthias Hemmje, and Clemens Kunkel. Lyberworld – a visualization user interface supporting fulltext retrieval. In W. B. Croft and C. J. van Rijsbergen, editors, *Proceedings of SIGIR '94*. Springer-Verlag, July 1994.

[Wiz96] Network Wizards. Internet domain survey, January 1996. Available at URL `http://www.nw.com/zone/WWW/report.html`.

[WS94] John A. Waterworth and Gurminder Singh. Information islands: Private views of public places. In *Proceedings of East-West International Conference on Multimedia, Hypermedia and Virtual Reality (MHVR '94)*, pages 201–206, Moscow, Russia, September 1994.