Aspects of a Modern Multi-Media Information System

Dissertation
for the Award of the Academic Degree
Doctor of Technical Sciences
at the
Graz University of Technology
submitted by

Frank M. Kappe

Institute for
Foundations of Information Processing and
Computer Supported New Media (IICM),
Graz University of Technology

Graz, June 1991

First Reader: O. Univ.-Prof. Dr. Hermann A. Maurer
Second Reader: A. O. Univ. Prof. Dr. Volkmar Haase

Aspects of a Modern Multi-Media Information System

Frank M. Kappe

Abstract

This thesis describes a new, large-scale Hypermedia project ("Hyper-G") currently being developed at the Institute for Foundations of Information Processing and Computer Supported New Media (Head: Professor Hermann Maurer) of the Technical University of Graz. Experience gathered from modern Hypermedia systems, large-scale information systems, computer aided instruction and user interface design considerations led to a number of ideas, features, and examples of applications of Hyper-G.

They were condensed, put into a logical relationship, and used to formulate a set of requirements. The requirements, additional design decisions, and a discussion of implementation-related issues are part of this thesis.

Also, a new concept for the creation of real-time, interactive animation is presented. It is essentially a combination of Computer Animation and Hypermedia technologies, therefore it is called "Hyper-Animation".

This concept is also the basis of some of the more advanced applications of Hyper-G that are described in this thesis. Applications range from information systems and electronic publishing to exhibits that may be found in a virtual museum or exhibition environment.

Acknowledgments

The support of parts of the research reported in this thesis by the Austrian Federal Ministry for Science and Research is gratefully acknowledged. In addition, a number of individuals supported me in research and preparation of the thesis:

Prof. Hermann Maurer, thesis supervisor, introduced me to Hypermedia and suggested the topic(s) of the thesis. Numerous fruitful discussions with him and others, including Prof. Ivan Tomek from Acadia University, Nova Scotia, Prof. Dieter Fellner from the Memorial University of Newfoundland and institute members Helmut Mülner, Gerald Pani, Peter Sammer, and Robert Stubenrauch led to ideas contained herein. Prof. Haase's fast reading allowed last-minute changes and completion within schedule.

Last, but not least, I want to thank my wife Sigrid for her love and patience during the preparation of this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

What is this thesis about?

To put is simple: This thesis is about Hypermedia.

1.1 What is Hypermedia?

This question cannot be answered trivially. The origin of the Hypermedia idea can be traced back to 1945, although it was not called Hypermedia at that time. The idea had no big impact then, because it could not be implemented in a satisfactory way with the available hardware and software.

However, Hypermedia has more than one root. The term itself is derived from Hypertext and Multimedia, but it seems that a number of different disciplines of computer science (and others) – including Information Retrieval, Computer Based Learning, Computer Graphics, Human-Computer Interactions, User Interface Design, Electronic Publishing, Communication Systems, and Cognitive Psychology – have evolved in such a way that eventually they met at Hypermedia.

This is why now, after almost 50 years, there is so much interest in Hypermedia. The time of Hypermedia has come. And it is also the reason why Hypermedia means different things to different people:

- For some people (typically those concerned with information retrieval), Hypertext is a way of organizing and retrieving information. The typical Hypertext application for this group is encyclopedia-like information, with no other structure other than cross-references (links) imposed on the data. Hypermedia is the obvious extension to this concept (pictures, sound, etc.).

- Computer Aided Instruction (CAI) experts use Hypermedia as a medium for presentation of courseware. However, the structure of lessons is often "pseudo-linear", i.e. linear or strictly hierarchical, in order to not confuse the users (students) and guarantee that they find their way through the material, see all the material, etc.

- Computer Graphics experts tired of defining 2D and 3D graphics standards and producing realistic images have found a new toy: Multimedia. For graphics people, Multimedia means basically digital video and/or real-time animation ("virtual reality"). To them, Hypermedia is basically Multimedia plus links.

- User interface designers may argue that Hypermedia is just a user interface technique. At last, there is a concept for software that lets them forget about keyboards and makes use of new user interface devices like touch screens, position sensors, eye trackers etc., and heavily relies on graphical user interfaces.

- Cognitive Psychology argues that the concept is more suited to the associative nature of human thinking than the conventional referential access to electronic data (e.g. having to know in what directory and file the information is in order to find it).

- Hypertext and Hypermedia may also be seen as a concept of electronic publishing. When compared to a conventional book, electronic Hypertext and Hypermedia versions offer much greater functionality. Electronic publishers would probably say "Hypermedia is what you will find on a typical CD ROM".

- Hypermedia should enhance communication and collaboration of users. In fact, early systems emphasized this aspect more than today's modern implementations. "Groupware", "Computer Supported Collaborative Work", "Electronic Classroom", and "Electronic Meeting Systems" are new challenges for Hypermedia systems.

- The casual computer users who are no expert in any computer science field will probably already have seen one incarnation of Hypertext: The context-sensitive on-line help that is built into most modern personal computer software. So they also (think to) know what is Hypertext.

Now what is Hypermedia, again?

My opinion is that Hypermedia is all of the above, or at least has the potential to be. This thesis describes the design and some applications of a new Hypermedia system called Hyper-G. The system is developed at the Technical University of Graz, and is designed in such a way that it serves as a solid foundation on which to build diverse Hypermedia applications.

1.2 Previous Work

This work relies on experience gathered at the Institutes of Information Processing of the Graz University of Technology under the leadership of Professor Hermann Maurer over the last decade or so, as briefly described below.

First, there is rich experience with Bildschirmtext – the Austrian videotex system, that can be considered a simple Hypertext system. Information is structured in "pages" that are organized hierarchically; however, arbitrary links between pages can be defined. The pages can also contain (simple) graphics, so it can almost be considered a Hypermedia system.

However, the user interface is cumbersome: Links are activated by pressing the keys "0" to "9", there is no full-text search, transmission is slow, and so on. Users report the typical problems with Hypertext: It takes them too long to find information, they have difficulty finding it again in a later session, get lost in the information, etc.

Encyclopedias have been prepared in Hypertext form and are now available to the general public. On-line communication of videotex users and off-line bulletin boards, a sort of E-mail, and access to other databases (e.g. Homebanking, phone directory) are the communication facilities already implemented in the Austrian videotex system. It is a large-scale, multi-user information system and many the problems of such systems apply and have been investigated.

Second, experience with Computer Aided Instruction (CAI) – more than 700 lessons on various topics are available – has shown us some of the user interface related problems of presentation-type CAI systems. Also, the system employed can be regarded as simple Hypertext (Hypermedia) system. There is some experience with question-answer dialogs not found in similar systems, and this will be incorporated into Hyper-G.

Third, my personal experience is in the field of computer graphics, graphical user interfaces, and computer animation. This is why chapters 3 and 4 are a bit graphics- and user-interface-biased.

1.3 Overview of Thesis

Many of the ideas that can now be found in the design of Hyper-G are condensed from discussions with Hermann Maurer, Ivan Tomek, Robert Stubenrauch, Helmut Mülner, and others. There have been previous attempts to come up with a clean design of the overall system, based on (sometimes vague) ideas of what might be possible applications of Hyper-G, but none was entirely successful in separating requirement specification and implementation details.

I tried to grasp those ideas, structure them into a set of requirements, and to strictly separate the specification of requirements from additional design decisions and the description of the implementation model. This can be found in chapter 2, which certainly constitutes the main part of the thesis. In its clear separation of the various aspects and its elegant design using a relatively small number of orthogonal concepts this can be considered an important and novel contribution to Hypermedia research.

Chapter 3 introduces a new concept for the creation of real-time, interactive animation. It is essentially a combination of Computer Animation and Hypermedia technologies, therefore I call it *Hyper-Animation*. A key point is that it can be naturally integrated within the Hyper-G environment.

This concept is also the basis of some of the more advanced applications of Hyper-G described in chapter 4. Typical applications will be information systems, electronic publishing, and exhibits that may be found in a museum or exhibition environment. Chapter 4 also closes the circle of this thesis, as the applications described there led to the original requirements contained in chapter 2.

Finally, a word on the structure of the text. In order to make this introductionary chapter as readable as possible, I have deliberately avoided cross-references, references to figures and tables, citations, and footnotes.

However, technical texts like this thesis are difficult to write as linear text. In the forthcoming chapters, you will see a lot of constructs like " *(section 4.3.1.8, page 136)*", "*see figure 2.1 on page 17*", "[1]", and "*[136]*", which are typical elements of Hypertext ("links"). Because the text contains also some figures, you might regard it as a (poor) implementation of Hypermedia.

As a consequence, the text is difficult to read on paper. I have included page numbers in the "long jump" references to reduce "access time" when following the link. On some occasions, I have avoided cross-references and duplicated information (e.g. where it is only one sentence). This introduces a bit of redundancy, but eases reading.

The text was written using LaTeX[90, 91], which lets you specify the structure of the text instead of the appearance. It is easy to decompose the text into its individual sections, subsections, and paragraphs and then create overview documents that contain just links to those pieces of text [57]. References are not explicit but use symbolic names of the target (e.g. Referencing figure 2.1 is done by specifying `\ref{figCoreSystem}`; at the figure there is a `\label{figCoreSystem}`). Such

[1]This is a footnote (or annotation in Hypertext terminology).

references can easily be converted to Hypertext links to the actual figure. The same is true for cross-references in the text, footnotes and citations (if the cited literature is also available as Hypertext).

As a conclusion, I hope that soon documents like this one can be automatically converted to Hyper-G documents and links, and be easier to read and use as a reference.

Chapter 2

Hypermedia

This chapter constitutes the main part of this thesis. It contains both the specification of requirements and an overall system design of Hyper-G. Great care has been taken to strictly separate the specification of requirements from additional design decisions and the description of the implementation model.

While most of today's Hypermedia systems are designed for a specific purpose, Hyper-G is designed as a general-purpose Hypermedia system that can be used for a number of applications. Only relatively few 'orthogonal' concepts are combined in a clear design to achieve different features found in different special-purpose Hypermedia systems.

E.g., the concept of '*Associative Link Following*' of static links resembles the idea of bi-directional links found in the Intermedia system. '*Dynamic Links*' integrate database queries within the Hypermedia metaphor. '*Associative Link Following*' and '*Dynamic Links*' combined yield the '*full text search*' facility found in some Hypermedia systems related to electronic publishing, without the need for another independent concept. The '*Remote Database*' concept combined with '*Dynamic Links*' allows to impose Hypermedia structure on non-Hypermedia, remote databases, which is a step towards the global Hypermedia and electronic publishing system envisioned by Ted Nelson [136]. In contrast, almost all current Hypermedia systems are limited to working with local data [139] that is under control of the Hypermedia system.

The fairly clear separation between the user interface and the underlying Hypermedia engine which I have achieved allows to run Hyper-G with a number of user interface metaphors, which is desirable for a general-purpose Hypermedia system that is to be used in diverse environments by different kinds of users. To my knowledge, all current Hypermedia systems stick to a specific user interface metaphor, and can therefore be considered more or less special-purpose systems.

I specifically designed Hyper-G as a large-scale Hypermedia system. High-level navigation tools and search strategies, a multi-lingual Hypermedia concept, new user identification notions and communication facilities ease the use of Hyper-G in an international, multi-user environment. My implementation model allows to implement Hyper-G in a network of standard UNIX workstations, and is specifically designed for extensibility of the system (e.g., new document types, new user interface metaphors).

2.1 Overview

Generally, the term *Hypermedia* refers to a combination of two other terms that have been used quite loosely in the past 20 years for many different collections of features: *Hypertext* and *Multimedia*.

The basic idea of Hypertext is not exactly a new one: The term, coined by Ted Nelson in the 1960's [135], describes a vast network of text fragments linked together, an electronic writing and reading system that uses the power of the computer for more than editing and display. However, the idea can be traced back to the 1940's. Vannevar Bush, President Roosevelt's director of the Office of Scientific Research of Development, is usually credited with first describing a Hypertext system in his 1945 article "As we may think" [16][1]. He envisaged a machine for browsing and making notes, and linking information by "associative trails", modeled after the associative way of human thinking. In general, Hypertext is not so much a new idea as an evolving concept of the possible applications of the computer.

In europe, Sam Fedida – the "father" of videotex – had similar visions concerning large-scale, publicly available information systems [46].

Implementations of such large-scale information systems have been pioneered at institutions like Brown University (*Intermedia* [129, 189]), the University of Maryland (*Hyperties* [163]), Xerox PARC (*NoteCards* [60]) and by Ted Nelson (*Xanadu* [136]). More complete surveys can be found in [30], [52], [139] and [176].

Multimedia adds new facets to Hypertext. Although the term Multimedia is used by a number of persons (and companies) with a number of different meanings, the following definition seems to be the greatest common denominator:

> A Multimedia system supports at least three out of this list of data types:
> Text, still image, video, and audio information.

[1] You probably won't be able to get the original. However, the article was reprinted in [15] and [136].

Hypermedia can be seen as the Hypertext paradigm plus Multimedia features. In the last years, significant research has been undertaken in this field. E.g., [123] lists some 1400 articles and books on related topics. Hypermedia research integrates methods and ideas from information retrieval, user interface design, computer graphics, and cognitive science. Although significant progress has been made, a number of open problems still remain.

Today's Hypermedia systems can roughly be divided into two groups: The large, multi-user information retrieval systems (like the ones mentioned above), that rely on special hardware and are not commercially available. And there are small but popular single-user "Hypermedia" systems (e.g. *HyperCard* [185]), that lack a number of essential features, but are commercially available. As pointed out in [116], these systems can not truly be regarded as Hypermedia systems, due to the lack of communication features and the limitations on database size. Also, the user interface metaphor of such systems is more targeted towards browsing knowledge in a way predetermined by an author than to finding facts [112].

2.2 The Design of Hyper-G

Hyper-G is the name of an ambitious Hypermedia project currently being carried out as a joint effort by a number of institutes of the IIG (Institutes for Information-Processing Graz) of the Technical University of Graz[2] and the Austrian Computer Society. Part of this research is carried out in cooperation with other research institutions, including Joanneum Research (Graz), the Memorial University of Newfoundland (St. John's), Acadia University of Nova Scotia (Wolfville), and the University of Colorado (Boulder).

Implementation of the full system is scheduled for 1995, but a prototype will be implemented by end of 1992[3]. A very simple prototype called *Hypermedia Base System (HBS)* has been implemented on both PC and UNIX platforms.

Also, data acquisition is performed parallel to the design and implementation stages, in order to fill the system with real-world data from the beginning on, as one of the aims of the Hyper-G development is to test user interface metaphors and search strategies in very large Hypermedia systems.

[2] Yes, the 'G' in Hyper-G stands for Graz.

[3] Funding of this project by the Austrian Ministry of Science, grant 613.531/5-26/90, is gratefully acknowledged.

2.2.1 Design Strategy

The forthcoming sections describe the Hyper-G system as well as the prototype system in some detail. But they do not contain just a descriptive specification, but also the requirements and design decisions that led to certain system features. A great effort has been undertaken to not mix up requirement and specification.

The rest of this chapter is structured as follows:

1. Section 2.2.2 lists 'axiomatic' system requirements. These requirements were gathered by comparing other Hypermedia systems and analysis (or requirement engineering) of their strengths and shortcomings [116]. From the beginning on, requirements are classified in five groups and hierarchically structured.

2. This set of requirements is then refined in section 2.2.4 on the basis of a number of additional design decisions made in section 2.2.3.

3. This large set of requirements logically leads to a specification of Hyper-G. The specification itself is split into:

   (a) Functional descriptions of the processes involved and their intercommunication (section 2.2.5).

   (b) Definitions of the data items the system will handle and the databases needed to find them (section 2.2.6).

   (c) The user's view of the system (section 2.2.7).

   (d) The author's view (section 2.2.8).

   (e) The system administration's view (section 2.2.9).

4. Eventually, a specification of a Hyper-G prototype, based on priorities given to the requirements, may be found in section 2.3.

However, it is not necessary to read this document in the order presented. For example, if you are interested in the features of the prototype, but not in the requirements and design decisions that led to them, you may skip the corresponding sections.

Of course, this strict division of requirements, design decisions, and descriptive specification introduces some redundancy to the text. E.g., you may read about access rights in sections 2.2.2.4, 2.2.3.16, 2.2.4.3, 2.2.6.1, and more.

Part of the information is duplicated (e.g., where it is just one sentence), and a large number of cross-references is contained within the text. Unfortunately, paper does not offer the same functionality as Hypertext (just click on the references), so page numbers have been added to the cross-references where it seemed reasonable.

Still the document remains difficult to read. One personal experience I got from writing this text is that ordinary text just isn't good enough to write technical specifications. With all these cross-references, figures, and tables it would have been better to write is as a Hypertext from the beginning on. If I have not convinced you by now, I hope that the forthcoming specification of a fine Hypermedia system will do so. Let's start!

2.2.1.1 Definition of Terms

Before we begin, however, table 2.1 defines some terms that will be commonly used throughout this chapter, so we can rely on understood basic definitions. Other definitions are supplied where needed.

System	Will almost always mean the Hyper-G system. Includes all the hardware and software of the run-time system (i.e. the system the user sees), plus additional hard- and software for authors and system administrations (tools).
User	User of the system is the person that utilizes the run-time system, e.g. to retrieve information.
Author	Author is the person that creates information that is to be included and may be accessed by the run-time system. It is the 'electronic' author, the 'owner' of an object (e.g. in the case of a picture of the 'Mona Lisa', the author would be the person that inserted that picture into the System, rather than Leonardo da Vinci).
Editing	Will mean preparation of information by the author. Information may include textual, pictorial, sound, and other data.
Terminal	The hardware the user is working on directly.
Session	The period of time the user works with the system on a terminal. Usually, the users have to log in (identify themselves to the system) when starting a session, and log out to stop a session.
Document	An item of information that the user may interact with (e.g. look at it). Throughout the system, information is structured in documents.
User State	Conceptually, the system sees the users' work as changes of their user states. The user state records such information on what documents the user is looking at, etc.
Database	A storage mechanism for documents and other data items the system needs.
Attribute	A piece of information attached to documents. Used to find documents with certain attributes in the database.
Collection	A set of documents that have the same set of attributes attached to them. In some way, these documents belong together.
Link	A connection between two documents. Links play an important role in Hypermedia systems.
Annotation	Sometimes, a user wants to (temporarily) become an author and add information (private or public) to the system, attached to other documents.
Tour	A concept that allows to visit documents in a context supplied by a tour author.

Table 2.1: Definition of Terms

2.2.2 Basic Requirements

This section contains a list of 'first-generation' requirements for Hyper-G. They are structured in five groups: *General, User, Author, System* and *Implementation*. For ease of reference, a unique label is assigned to each requirement. This set of basic requirements will be refined in section 2.2.4 on the basis of additional design decisions made in section 2.2.3.

2.2.2.1 General Requirements

Req. G.1:	The system should allow access to all kinds of information one can think of (*Multimedia Axiom*).

Let us refine this to a set of more specific requirements:

Req. G.1.1:	Information is organized in *documents* of a specific *document type*. Document types include (but are not limited to):
Req. G.1.1.1:	Textual Information
Req. G.1.1.2:	Technical Drawings
Req. G.1.1.3:	Raster Images
Req. G.1.1.4:	Digitized Sound
Req. G.1.1.5:	Digital Movies
Req. G.1.1.6:	Animation
Req. G.1.1.7:	Maps
Req. G.1.1.8:	Electronic Mail, Bulletin Boards, Computer Conferencing
Req. G.1.1.9:	Dialogs (e.g. Menus, Forms)

A system able to handle above document types is generally referred to as a *multimedia system*. To put it simple, above requirements state that Hyper-G is such a multimedia system. Above list is by no means complete; e.g., document types like synthesized voice and music may be added in the future.

Req. G.2:	Documents (**Req. G.1.1**) may be cross-referenced and accessed by *links* (*Hypertext Axiom*).

This means that Hyper-G adopts the well-known Hypertext paradigm. Requirements **Req. G.1** together with **Req. G.2** state that Hyper-G is what is known as a *Hypermedia system*. More specific versions of **Req. G.2** will be given in section 2.2.4.1 (page 35).

Req. G.3:	Hyper-G provides access to very large amounts of data.

As stated in [35], Hyper-G is targeted to mega-quantities of documents and terra-quantities of bytes. This leads to interesting problems not only for system design, but – more important – for the users and the authors of such a large system [86]. An immediate consequence is

Req. G.3.1:	Documents and links are maintained automatically.

Link maintenance involves such problems as deleting all links from or to a document that is to be deleted, in order to avoid dangling references and to ensure database integrity. Also, the system should assist the system administration in protecting the database against obsolete or outdated documents and/or links.

Req. G.3.2:	The mayority of the links is generated automatically.

Especially for text documents, a number of links can be generated automatically either when inserting the document into the database, or at runtime. More on this in section 2.2.3.4 (page 22).

A user related consequence (**Req. U.3**) of **Req. G.3** is given in section 2.2.2.2 (page 13).

An author related consequence (**Req. A.2**) of **Req. G.3** is found in section 2.2.2.3 (page 15).

A system related consequence (**Req. S.3**) of **Req. G.3** is found in section 2.2.2.4 (page 15).

Req. G.4:	The system should be easily extensible.

This is especially true for the set of supported document types and user interfaces (**Req. U.1.1**) and has some effect on the implementation model (sections 2.2.5, 2.2.6).

2.2.2.2 User-related Requirements

Req. U.1:	The system should be usable by untrained users as well as expert users.

In general, Hyper-G is likely to see a number of scenarios [87, 35, 121, 126]. One is a university information system (section 4.1) based on Hyper-G where users are expected to communicate with computers on a regular basis, another one is a so-called *viewseum* (section 4.3) available to the general public. Hyper-G should adapt to the needs of all user groups. Implications of this requirement are discussed in section 2.2.4.2 (page 39).

Req. U.2:	The system should offer a choice of user languages.

This not only requires the user interface to work with a number of languages, but also means support of multi-lingual documents (e.g. translations of the same document provided by the author, or simple computer translations).

Req. U.3:	The user is supported by high-level navigation tools.

Above requirement may also be seen as a consequence of **Req. G.3** (large number of documents). High-level navigation tools include use of user interface metaphors, database queries, collections, tours and scripts. These concepts are discussed in detail in sections 2.2.3 (page 17 and 2.2.4.2 (page 39).

Req. U.4:	The system is configurable on a per-user basis. It remembers a user's preferences between sessions.

This means that users may configure the system to adapt to their special needs (e.g., language preferences (see **Req. U.2**).). Moreover, this configuration is retained by the system until the next session. Of course, this implies that the system is able to distinguish between users by some login procedure and keeps configuration files. It also suggests use of user groups, access rights and so on. However, these are not basic requirements and discussion is therefore postponed until section 2.2.4.2.

Req. U.5:	At any time, the user may return to a previous user state.

This applies to all the previous states of an active session, which yields sort of an *undo* function. As a special case, when entering a new session, the user may return to where the previous session was left. **Req. U.5** requires the system to remember user states, keep user logs etc. Again, the details are related to the actual implementation and are discussed in section 2.2.4.2.

Req. U.6:	The system can be used anonymously.

An information system that is to be used by a large number of users on a regular basis raises a data security problem. If users are known by name, the system might be used to find out who looks for certain information, monitoring users working time, etc. To avoid such problems from the beginning on, users may login anonymously, as in the Austrian videotex system [118]. The modes *semi-identified* and *anonymously identified* (section 2.2.3.15, page 32) still allow to perform most operations of the system.

Req. U.7:	Certain system features and information items are chargeable.

For example, printing a screen dump or saving it to disk may cost some (small) amount of money. In addition, information providers may charge something for the access to certain pieces of information, remote databases, etc. In conjunction with **Req. U.6** this implies that a rather tricky charging mechanism has to be implemented.

2.2.2.3 Author-related Requirements

Req. A.1:	Any user is a potential author.

The database may be manipulated by potentially every user, either by manipulating documents or links (*annotation*). Of course, access rights control the user's ability to change existing data items or create new ones (section 2.2.3.16, page 34).

Req. A.2:	The author is supported by high-level information editing tools.

These information editing tools divide into document preparation tools and link editing tools. They are not part of the Hyper-G runtime system, so the details are postponed to section 2.2.8.

2.2.2.4 System Requirements

This section describes basic requirements related to system administration.

Req. S.1:	A number of *terminals* is attached to the system. On any terminal a Hyper-G *session* may be started. The capabilities of the terminals may differ.

That is, the terminals are not all the same. There may be terminals with special user interfaces, communication features, hard/softcopy capabilities etc. As a consequence, Hyper-G must not rely on the presence of specific terminal facilities.

Req. S.2:	Access rights control whether a given user on a given terminal may perform a certain function on an object.

After describing some design decisions related to access rights (section 2.2.3.16), this concept is elaborated in more detail in section 2.2.4.3.

Req. S.3:	The system should avoid uncontrolled growth of unimportant or outdated information.

The system administration should be supported in finding such information and deleting it at regular intervals.

2.2.2.5 Implementation Requirements

Req. I.1:	Access to information should be reasonably fast.

This requirement is somehow contradictory to **Req. G.3** (large amount of data). However, it is well-known that a response time larger than about 250 milliseconds confuses users [19]. A lower response-time variance seems to be more important than absolute response times [23]. Also, users are willing to accept, e.g., to wait a few seconds for a database query, if the system supplies them with immediate feedback of acceptance of the query. We have seen systems (e.g. videotex [46, 114, 115]) whose acceptance by the public has been dampened by less than optimal response time.

Above requirement has significant impact on the design of the databases (speed vs. space tradeoffs) and the computers used. More on the database design may be found in sections 2.2.4.4 (page 50) and 2.2.6.2.1 (page 70).

2.2.3 Design Decisions

2.2.3.1 Distributed System

Requirements **Req. G.3** (large amount of data) and **Req. S.1** (number of terminals) suggest to implement Hyper-G as a distributed system running concurrently on a heterogeneous network of computers. To achieve high performance, Hyper-G is split into concurrent processes (section 2.2.5, page 52) that may execute on a number of machines simultaneously.

Also, the data is distributed among the machines (section 2.2.6, page 60). Data includes documents of various types, links, log files etc. The use of object-oriented design eases the task of distribution of data and processes.

2.2.3.1.1 Core System

The *Core System* of Hyper-G will consist of a high-speed local area network (LAN) of Hyper-G Servers and Workstations (see figure 2.1).

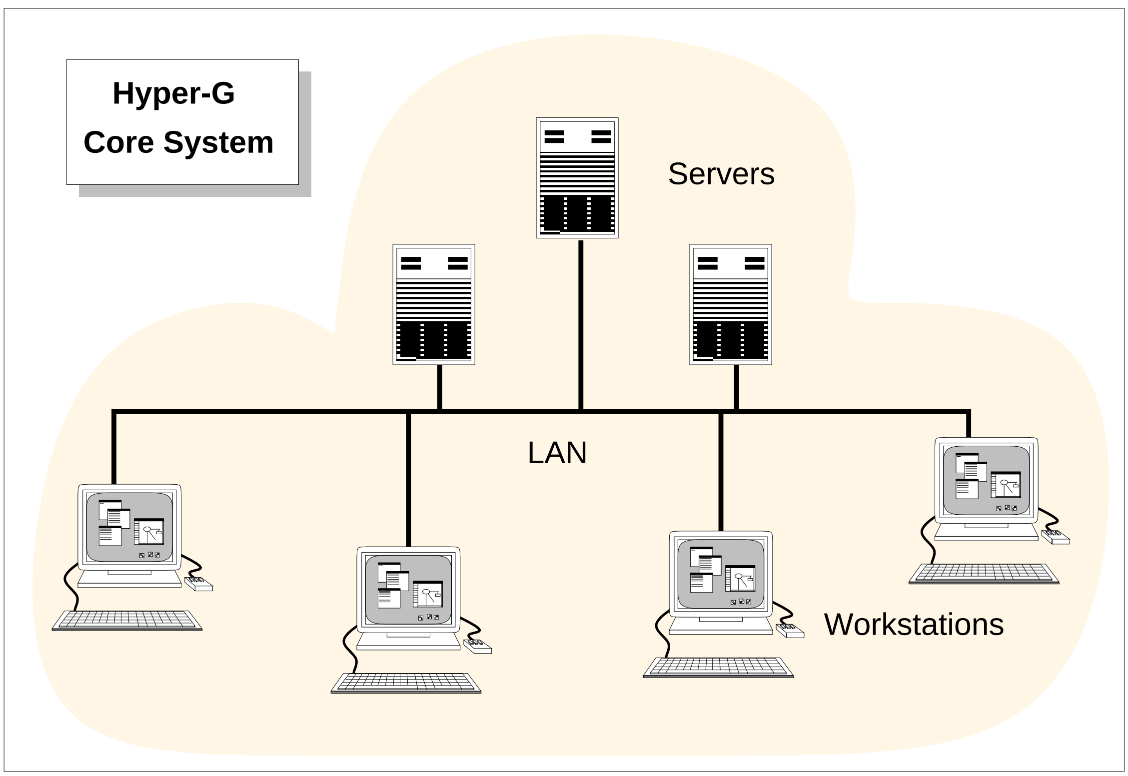

Figure 2.1: The Hyper-G Core System

Hyper-G Servers will typically be supplied with gigabytes of fast mass storage (harddisks), but there may be special-purpose Hyper-G servers equipped with, e.g, CD-ROMs, Video or Audio Discs or Tapes.

Workstations will not be all the same. Most of them will be standard graphic workstations, but some of them may have additional audio/video capabilities or special user interfaces.

2.2.3.1.2 Front-Ends

In addition, Hyper-G will be accessible from remote terminals over a wide area network (*WAN*, see figure 2.2). In most cases, this WAN will be slow (typically dial-up phone lines or telematic services networks; up to 64 kbits/s), and the facilities of the terminals will vary, so the access to certain types of documents will be limited.

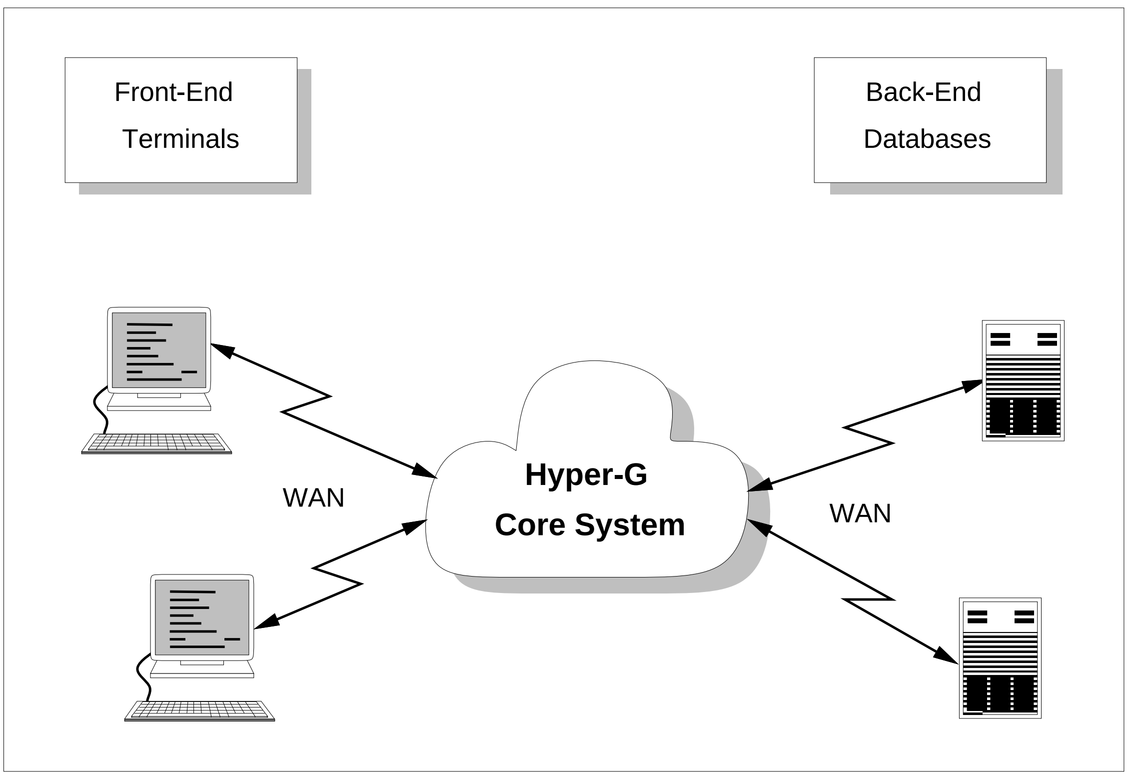

Figure 2.2: Hyper-G and the World

2.2.3.1.3 Back-Ends

Also, it will not be feasible to keep all the data in the core system. Therefore, Hyper-G will support access to remote databases (e.g. official airline guide, phone books, any kind of data that keeps changing), but this access will be hidden from the user. In particular, that kind of data will be embedded into the user interface metaphor (section 2.2.3.10, page 27) and will be accessible by links.

2.2.3.2 Software Environment

We decided to implement the Hyper-G core system within the following Software Environment:

1. **Operating System:** UNIX[4].
2. **Graphical I/O:** X Window System [141, 142, 143, 161], CGI [51, 73], InterViews [97, 98].
3. **Programming Language:** C++ [169].

There are more standards that are used implicitly (e.g. Ethernet, TCP/IP, NFS, SCSI) but handled inside UNIX, and therefore not visible to Hyper-G. Hyper-G should not rely on these underlying standards; they can be changed (e.g. FDDI instead of Ethernet) without effect on Hyper-G.

It should be noted that Hyper-G is not aimed at a single type of computer, nor at a homogeneous network of computers, but at a heterogeneous network of different types of computers, possibly with different user interface devices attached etc., as implied by requirement **Req. S.1**.

2.2.3.3 Multimedia Documents vs. Document Clusters

A *multimedia document* is a single document consisting of information items of various types; for example, an image glued into explanatory text (figure 2.3). The author of such a document controls the exact layout of the document. This paradigm is used by most of the small Hypermedia systems (e.g. HyperCard [185]).

In contrast, a *document cluster* is actually a number of documents of possibly different document types that are semantically related (in our previous example the image document and the explanatory text document). The individual documents are linked by so-called *cluster links* that are provided by the author, thus creating a document cluster (figure 2.4). This approach is adopted by some existing large Hypermedia systems, like the *InterMedia* system [129, 189]), NoteCards [60] and Hyperties [163], and also by Hyper-G.

[4] UNIX is a trademark of AT&T Bell Laboratories

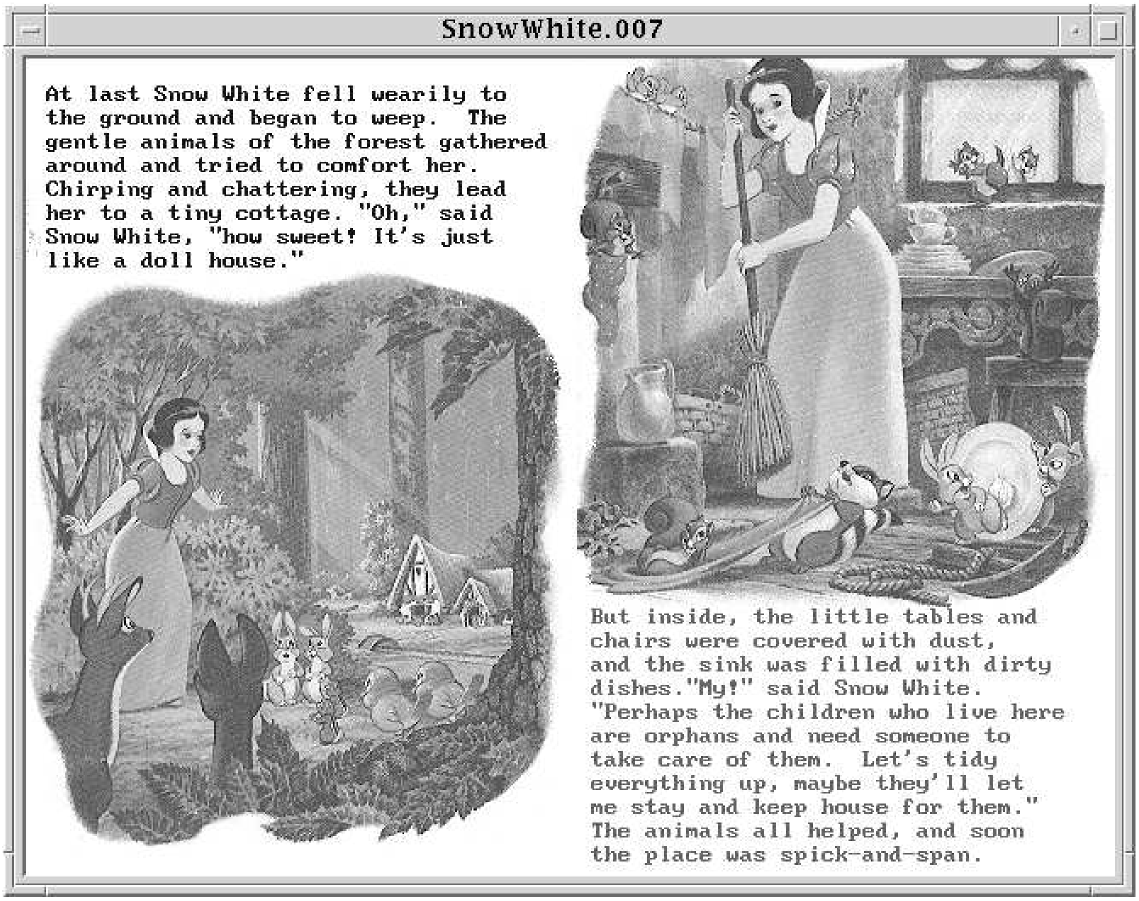

Figure 2.3: A Multimedia Document (Window Dump)

With document clusters, the author's control of layout is limited (for example, scrolling the text document does not affect the image document). On the other hand, document clusters offer a number of advantages:

- Document clusters may be managed by distributed processes (one kind of *document manager* for every document type), possibly on a number of machines, increasing performance.
- This concept (a document manager for every document type) makes the system very extensible. Adding a new document type means adding a new document manager, without affecting the rest of the system, as opposed to a single multimedia document type with all the knowledge about all document types.
- The concept makes Hyper-G implementable by a number of individual programmers, programming and identifying with their 'personal' document manager.
- There is no need for a single, complicated document standard (like ISO's Office Document Architecture (ODA) [78], Standardized General Markup Language (SGML) [79], Abstract Syntax Notation One (ASN.1) [76, 77] standards or

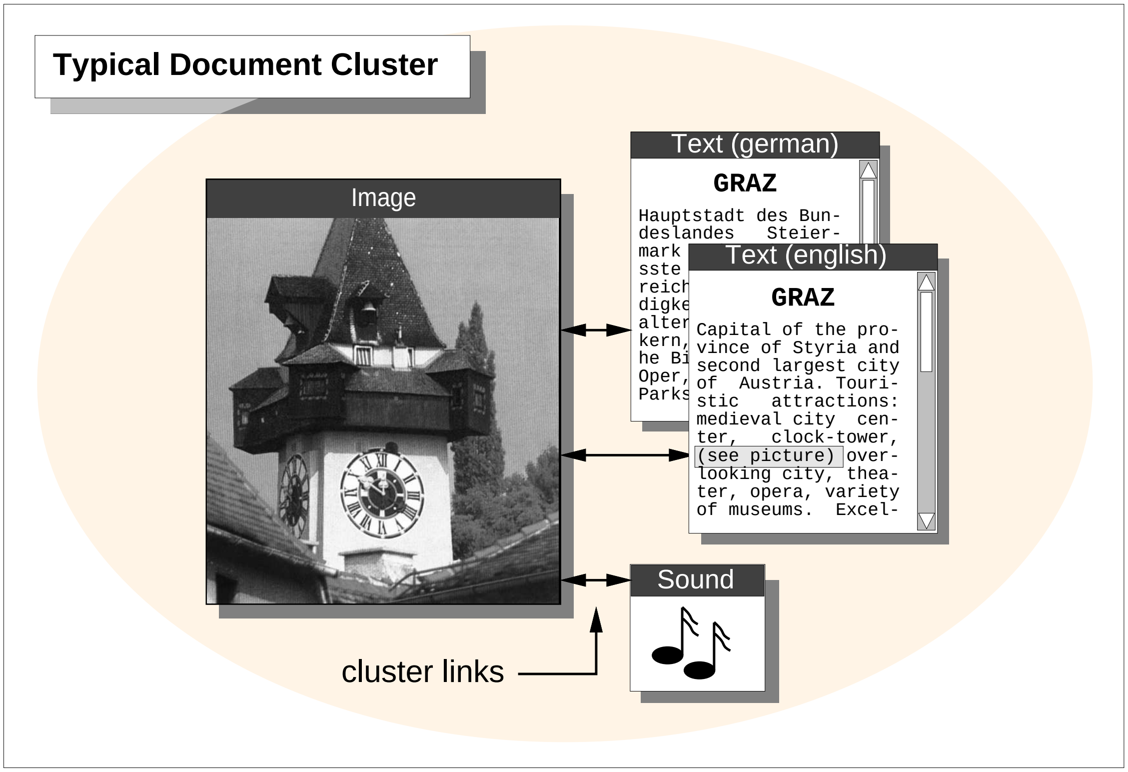

Figure 2.4: A Typical Document Cluster

DEC's Compound Document Architecture (CDA) [36]) that covers all the document types and hinders extension [13].

- This concept eases implementation of multilingual documents. An example application of a document cluster may be a picture with associated descriptions of the picture in different languages, either as text documents or as voice documents.

Of course, this approach has some impact on the user interface, too. As opposed to multimedia documents, where figures are directly glued into the text by the author, in the case of document clusters the user would find a *'see figure x'* in the text (like in most books, by the way). Depending on the particular user interface, the user would have to click on this message, or the figure would be displayed automatically beneath the text. Scrolling or reformatting the text would not affect the figure. A certain user interface metaphor may give the user the opportunity to move the figure away or may explicitly forbid to do so, so that document clusters will look like multimedia documents.

2.2.3.4 Dynamic Links

As required by **Req. G.3.2**, Hyper-G will support automatic link generation. Automatic generation of Hypertext links has already been demonstrated in the case of encyclopedias [133] and structured text like technical books [57].

In a large Hypermedia system with a number of general-purpose and specialized encyclopedias and dictionaries (such as the Oxford English Dictionary [12, 154] or the *Duden* for German), it is pretty likely that, for any word the user sees on the screen, there is a document explaining at least the meaning of that word. Because of automatic link generation, there will be a link attached to that word.

However, if there is a link from (almost) every word, it does not make too much sense to mark all these words as 'clickable' (exact meaning depends on user interface). The users are simply allowed to click on any word on the screen. In most cases, they will get information on that word by doing so.

If we assume fast database access, we can also postpone the generation of these links to runtime, i.e. the moment of the user's click. The system would respond by performing a database query, looking for entries of the clicked word. If found, the associated document is presented. This *dynamic link generation* offers the following advantages:

- It reduces the number of (static) links, reducing not only storage requirements, but also link maintenance. For example, when a document is deleted, the document is just no longer found in the database, so that no dynamic links to the document are possible any more. Vice versa, no words in the document just deleted can be clicked any more, so the 'outgoing' links are also destroyed automatically. Also, when a new document is inserted, it is automatically linked with all the other documents already in the database or to be added in the future.
- The user may define an *active list* of databases to be searched for links, reducing the number of unwanted links (*link filtering*).
- Access to remote databases in a way consistent with the Hypertext axiom (**Req. G.2**), i.e. by links, is only possible with dynamic links. Because the information in the remote database may change at any time, no maintenance of static links to or from a remote database is possible.

Hyper-G supports automatic, dynamic generation and maintenance of links. Of course there will be user-defined links, typically attached to graphical, video and audio documents. In addition, Hyper-G supports automatic, static links that may be found using algorithms too complex for runtime execution.

2.2.3.5 Associative Link Following

Vannevar Bush is usually credited with first describing a Hypertext-like system called *memex* in his 1945 article "As we may think" [15, 16]. He speculated to model an information system by mimicking the operation of the human mind (see also [30]):

> "The human mind ...operates by association. [...] One cannot hope to equal the speed and flexibility with which the mind follows an associative trail, but it should be possible to beat the mind decisively in regard to the permanence and clarity of the items resurrected from storage."

And indeed, *associative searching* seems to be a major issue in Hypertext and Hypermedia systems, and links to be the implementation of Bush's 'associative trail'. However, the uni-directional links found in most of today's Hypermedia systems (e.g. HyperCard [185]), cannot be called fully associative. Consider the following example:

In a document cluster 'Austria', there is a link to document cluster 'Vienna', e.g. a static link originating from the appropriate location in the map of Austria, or a dynamic link originating from the word 'Vienna' in a text on Austria. Now let us assume that the user got to cluster 'Vienna' somehow, perhaps by a database query (section 2.2.3.13, page 30). Typical questions the user might ask are:

- *Where is Vienna?* Answer: Show cluster 'Austria' (with map highlighting Vienna).
- *What is Vienna?* Answer: Show cluster 'Austria' (with text stating that "Vienna is the capital of Austria".
- *What other documents (document clusters) refer to 'Vienna'?*

In other words, the user would like to have the ability to trace links backward to the source when being at the target. Above questions may be generalized to "Show me the documents with links to the current document." This 'associative search' may be considered the opposite of conventional (forward) link following.

Hyper-G will support both forward and backward (associative) link following. Note that there are no new links or link types involved; rather the user is given the choice of how to use them. For the implementation, this has the following consequences:

- **Static links** are kept in a database that can be searched equally fast for link source and destination. Associative static links allow to find all documents that (statically) refer to the current document.

- To support associative **dynamic links**, appropriate data structures that allow the fast full-text search of text documents have to be implemented (section 2.2.6.2.1, page 70). Associative dynamic links generate answers to the question "What other documents contain the word I am just clicking on?"

Things get more complicated when considering *anchors* (section 2.2.3.7, page 24).

2.2.3.6 Link Priority

Given the different link generation methods described above, the relevance of links will vary. A high link relevance means that with a high probability the user may get to relevant information following that link. We get the following hierarchy of link relevance (ordered by decreasing relevance):

1. Cluster link
2. User defined link
    (a) Defined by the same user (e.g. annotation)
    (b) Defined by the author of the document
    (c) Defined by system administration
    (d) Defined by some other user
3. Automatically generated link
    (a) Static link
    (b) Dynamic link

When there are a number of links to choose from, the system offers the user the link with the highest relevance, so link relevance may also be regarded as *link priority*. Alternatively, the user may get a (sorted by priority) list of links to choose from. Also, it may only be necessary to look for dynamic links when there are no links of higher priority or the user is not satisfied with them. Note that choosing forward or backward link following (section 2.2.3.5, page 23) does not affect link relevance.

2.2.3.7 Links and Anchors

Conceptually, a link is a connection between two documents. Links are directed, i.e. there is a *source* document and a *destination* document. Note that this distinction does not prohibit backward (associative) link following (section 2.2.3.5, page 23).

However, we do not always want to link whole documents, but rather a part of a source document (e.g. a word, an area of a picture) to a destination document or a part of it (e.g. a paragraph, a point in a map). This document part is described by a so-called *anchor*. There are source and destination anchors.

When linking to a destination anchor, the system should try to position the destination document in such a way that the destination anchor is visible. E.g., in the case of a destination paragraph in a text document, the document should be scrolled so that the beginning of the paragraph is visible. In the case of a destination point in a map the map should be positioned so that the point is in the center of the visible portion of the map[5].

Conceptually, anchors in Hyper-G are parts of the links rather than part of the documents, so that even users without the right to modify the document may tie links to them (annotation), if they have the right to do that. However, the anchor description may be used to present a visual representation of the link on top of the document (e.g. an area to click on). As anchors define a certain part of a document, the anchor description can become rather complex:

- In text documents, an anchor will most likely be a single word, a sequence of words, or a paragraph.
- In image documents, it might be a rectangular area of the image, a point in the image, or a generic polygon.
- In an object-oriented graphic document, it might be a group of objects (i.e. a segment).
- An anchor of an audio document could be defined by start and end time. These marks are especially useful for playing only a certain section of the audio document (when used as destination anchor), and for synchronizing sound with other documents (when used as a source anchor; see section 3.3.4 on page 114).
- Anchors in animation documents could be far more complicated, as they could be moving with respect to time, following a complex path.

Also, the visual appearence of the anchor may vary with the type of the destination document. This feature is used to lead the user to 'rare information', such as video clips, animation, sound, and images, and is most important in the early phase of Hyper-G. At a later stage, e.g., when high-quality images are not so rare any more, it may become partially obsolete.

[5] In the case of backward link following, the word 'destination' in this paragraph is to be replaced by 'source'.

An anchor that links to a document (cluster) already seen during the current session is shown in a special way (e.g. color) to avoid unwanted re-visits of the same document.

In addition, anchors may be *active anchors*, i.e. there is a piece of code associated with them that is executed when the anchor is activated. An example of Hyper-G's pre-defined active anchors is the *branch anchor*: Of course, it is possible to tie a number of links to an anchor. The branch anchor takes an integer as argument and uses it to choose a link to follow from the set of attached links (see figure 2.5).

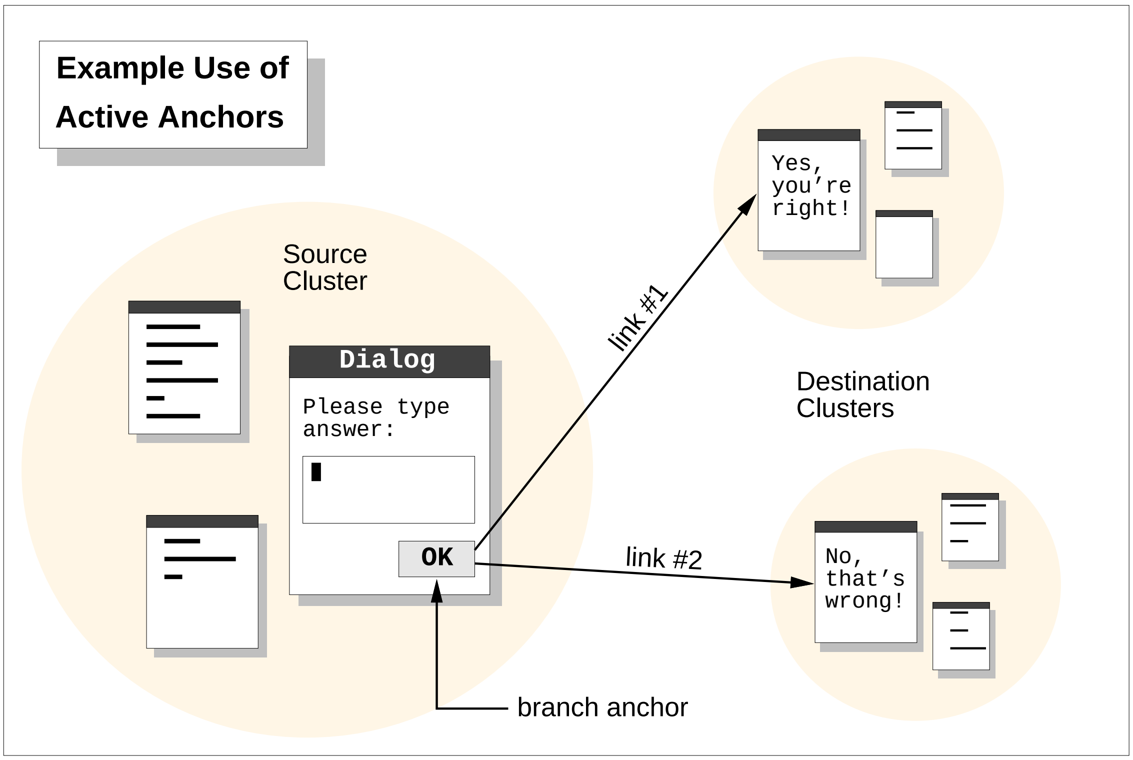

Figure 2.5: Active Anchors: The Branch Anchor

Observe the following:

- Dialogs (e.g. Menus, Answer Judging, Database queries) are most naturally implemented using active anchors.
- Anchors work with all link types (section 2.2.3.6, page 24). As a special case, active anchors that generate dynamic links (section 2.2.3.4, page 22) can be used to implement forms for database queries (section 2.2.3.13, page 30).
- As active anchors require the execution of code, their definition is not easy. However, there are a number of pre-defined active anchors supplied by Hyper-G. These include anchors for dialogs and special effects (active destination anchors) for the display of documents.

2.2.3.8 Annotation

We may distinguish between two types of annotations:

1. The user creates a new document, and links it to an existing document.
2. The user only creates a new link (including anchors) between existing documents.

Note that in Hyper-G there is no special document type 'annotation'. Rather, the links (like all Hyper-G objects) carry information on who created the link (*link author*). The annotations of user *xyz* are simply the links *xyz* created. Because there is no special document type, annotations may be of any type, including graphical, audio or video annotations.

Truly anonymous users may not create annotations; this feature is reserved to fully identified, semi-identified and anonymously identified users (section 2.2.3.15, page 32).

2.2.3.9 Navigation Tools

Users of large Hypermedia systems experience navigation problems, including: getting lost in "Hyper-space"; having difficulty gaining an overview; not being able to find information that is known to exist; determining how much information on a given topic exists; how much of it has been seen and how much is left [139, 138]. Typical solutions that work well on small systems fail completely when applied to large systems. Figure 2.6 (from [30]) shows an experimental implementation of a *graphical browser* displaying a global map in the *InterMedia* [129, 189] system. The display shows only 80 documents out of some 1000 total documents, so one can easily imagine what happens in a mega-document system...

Hyper-G will use other methods to aid the user's navigation, namely the use of user interface metaphors, collections, tours, and database queries.

2.2.3.10 User Interface Metaphors

Hyper-G will not adopt one specific user interface metaphor. Rather, it will serve as experimental playground to test a variety of Hypermedia presentation metaphors [35]. It should be even possible to configure the preferred presentation metaphor on a per-user basis or even switch metaphors in the middle of a session or to look at the same set of documents using different metaphors at the same time.

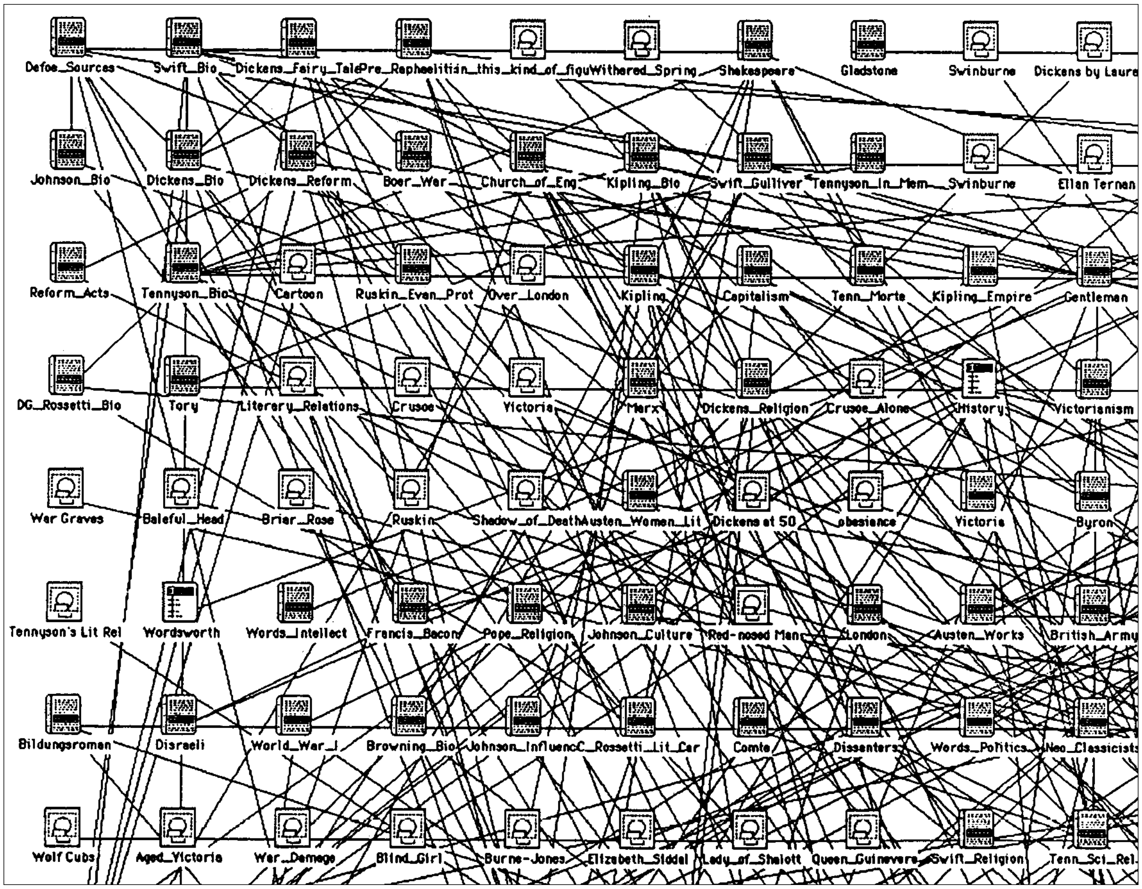

Figure 2.6: Graphical Browser of the *InterMedia* System

Therefore, the system has to be designed in a way totally independent of the user interface. The internal functions of Hyper-G have to be separated clearly from user interface functions. Also, this allows to support very different and unorthodox kinds of user interfaces to extract information from the system, e.g. HOTACT [120].

2.2.3.11 Collections

In order to ease the use of the system by untrained users (**Req. U.1**) the documents of Hyper-G are arranged in so-called *collections*. Collections group together documents or collections of related content (note the recursive definition). This yields a hierarchical structure of collections (with 'subcollections'). However, this structure is not a tree, but an acyclic directed graph, i.e. a collection may be a subcollection of a number of collections. For example, the collection 'Biochemistry' might be a subcollection of collection 'Chemistry' as well as of 'Biology' (see figure 2.7).

This collection – subcollection relation eases the user's navigation. E.g., at any time the current position in the collection hierarchy can be displayed, the user may 'descend' or 'ascend' in the hierarchy, finding related collections.

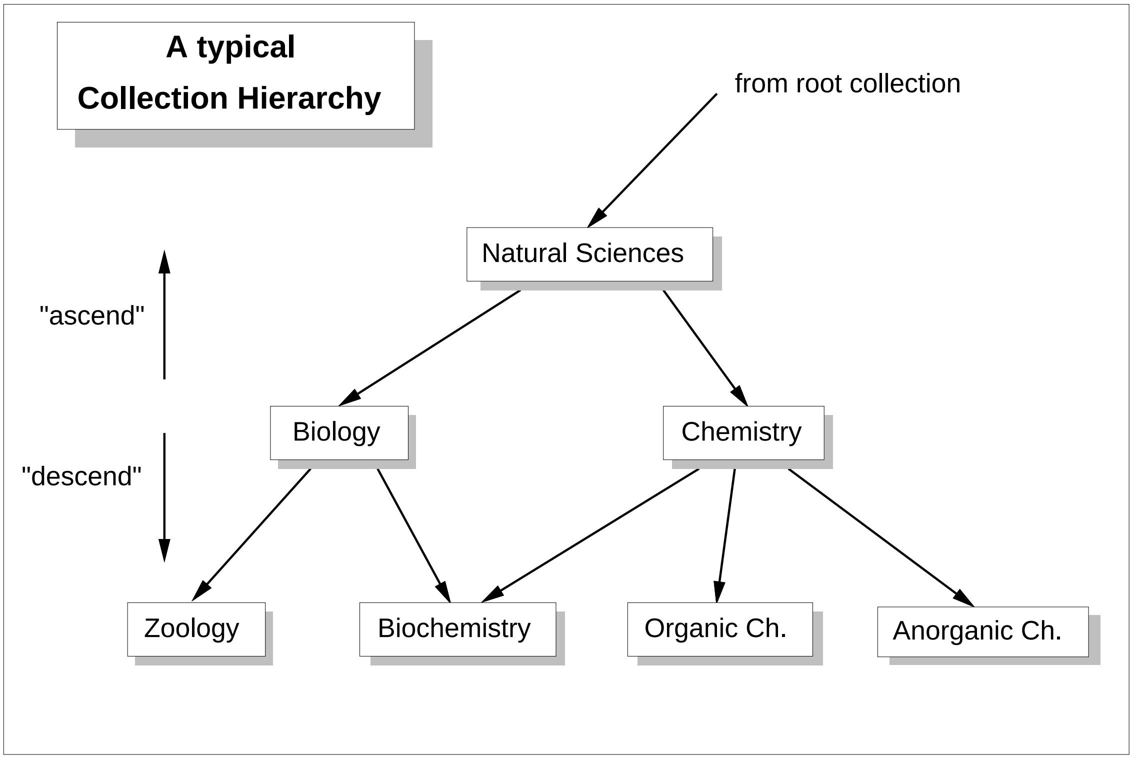

Figure 2.7: A Typical Collection Hierarchy

2.2.3.12 Tours

In addition to the collection concept, the user has the possibility to go on 'guided tours' through the information network that have been provided by some author (potentially any user). Tours do not care about the hierarchical collection structure, but link documents together in an arbitrary way. Tours do not have to be linear lists of documents, but may use the full flexibility of a Hypermedia system. Computer aided instruction is an example application of the tour concept [124, 170, 171].

Two useful applications of the tour concept would be:

- Tours that might also be called 'lessons' in CAI terms. This kind of tours will typically be authored by some expert in the field, logically structured and contain a number of cross-references to other tours (lessons) and Hyper-G databases. In fact, the over 700 lessons of the COSTOC project [110] covering topics ranging from medicine, ecology, and natural sciences to computer science, will be transformed (to Hyper-COSTOC [70]) and become an integral part of the Hyper-G database.
- Other tours may be made by any user, linking together document clusters of interest, so the user can easily find them again in future session, very similar to the 'associative trail' of Bush [16]. These tours may be intended for private

use only, but may also be made available to the public or specific user groups. Typical tours of this class might be "my favorite impressionist paintings" or "some interesting features of UNIX".

Tours may be implemented by links (very much like annotations) and by script languages, and may 'overlap', i.e. documents may be part of a number of tours, 'sub-tours' may be used by a number of tours. At such documents, the user may be given the opportunity to 'switch' tours. Within tours, the user is aided by navigation facilities that are not available at the global Hypermedia level (e.g. graphical representation of the tour that is supplied by the author of the tour, special highlighting of links to information yet unseen, etc.).

2.2.3.13 Database Queries

Database queries allow for rapid access to specific information. Hyper-G supports two kinds of Database queries.

- **Key queries** are most valuable at the beginning of a session. Key queries are simple, possibly AND-connective prefix queries that are passed to a user-defined list of active databases. Using key queries, the user may find out very fast about existing information on a given topic.
- **Form queries** are collection-specific. For example, the collection 'fine art', subcollection 'paintings' may allow to fill out forms with fields *painter, name of picture, year, style, theme,* and *museum*, so that the user would easily find all paintings of Leonardo da Vinci that are on display in the Louvre. Form queries may be very complex dialogs, including incremental form-filling (*flex-forms*), i.e. the answer to the query dynamically appears as the form is filled out.

2.2.3.14 Multilingual Hypermedia

According to **Req. U.2** Hyper-G is a multilingual system. This implies that:

- The documents themselves may be multilingual, i.e. *document clusters* (section 2.2.3.3, page 19) that consist of several documents, storing approximately the same information in different languages. This obviously applies to text and voice documents, but may also be required for pictures, films, animation etc. in special situations.

- Special links between these documents (so-called *cluster links*) allow the runtime system to choose from available languages based on the user's preferences (**Req. U.4**).
- The user interface of the runtime system is available in a number of languages, too.
- The user may specify an ordered list of language preferences (e.g. 'Hungarian' – 'German' – 'English'). This means that if a Hungarian version of a document is available, show it, else show the German version, else the English one. If none of the user's list is available, the system may display an appropriate error message, or display whatever it has.
- The mechanism described above may be temporarily overridden, e.g. to see the document in its original language, or to see more than one version at the same time.

Further complications arise with the use of dynamic links. Suppose the user has chosen language preference 'Hungarian' – 'German' – 'English'. Now, for some reason, a German text is on the screen. The user might click on any word, which will most probably be a German word, e.g. 'Rechner' (computer), so a simple dynamic link (as described in section 2.2.3.4, page 22) will most probably find only German documents that contain the word 'Rechner' as keyword. In order to get existing Hungarian documents on 'komputer' (the Hungarian word for computer) as well, there are two solutions:

- We could assume that all documents on 'Rechner' of all languages are linked by cluster links, creating a document cluster (section 2.2.3.3, page 19) holding equivalent documents, but in different languages. When, in our example, the runtime system gets to the German document on 'Rechner', it would see an equivalent Hungarian document on 'komputer' (if it exists) and display that instead. Although this solution may be applicable to small amounts of data (e.g. a single tour), where the author creates the cluster links 'by hand', this approach does not generalize well, because of two reasons:
  - For large amounts of data (e.g. a number of encyclopedias in a number of languages), it is too much work to look for equivalent documents manually. On the other hand, the decision whether two documents in different languages are 'equivalent' is very hard to do for machines.
  - If in our example a German document on 'Rechner' does not exist, the runtime system would never find the cluster with documents on 'komputer' and 'computer'. Still the user would want to see the 'komputer' or else the 'computer' document, if one exists.

- In order to solve this problem, the Hyper-G database will contain some translation mechanisms (translation programs, dictionaries) to translate the word 'Rechner' to Hungarian before searching the database. Of course, this translation process introduces further 'blur' to the database query. For example, 'Rechner' translates to 'computer', but also 'calculator', 'arithmetician' and 'reckoner'. Without regard to the context and background knowledge, the system is not able to decide which translation is the most 'equivalent' one. This is a general problem of automatic translators.

  To make things even more complicated, in order to support n languages and to be able to translate from any language to any other language, we would need $n(n-1)$ translators or dictionaries. A possibility would be to translate first from the source language to an intermediate language (e.g. English) and then from that language to the target language. This way we would need only $2(n-1)$ translators. In our example, we would first translate 'Rechner' to English, and the resulting words ('computer', 'calculator', 'arithmetician' and 'reckoner') to Hungarian. Of course, this translation would be even more inaccurate.

  Vice versa, it is possible that two German words (e.g. "Taste" – the kind of key that you press on a keyboard – and "Schlüssel" – the kind of key that you use to lock doors) translate to the same English word (*key*). As a result, to the surprise of a German-speaking user looking for information on "Taste", information on "Schlüssel" may be returned instead. The usual solution is to have additional attributes in the intermediate language (e.g., key (computer) and key (door)).

With Hyper-G we will have the opportunity to experiment with both approaches. To our knowledge, Hyper-G will be the first Hypermedia system supporting multilingual documents.

Of course, languages like chinese, japanese, arabic etc. impose even more problems. For example, a large number of fonts has to be supported, writing is not restricted to the usual "from left to right, from top to bottom" way, and more.

2.2.3.15 User Identification Modes

As stated by **Req. U.1**, the system can be use anonymously. This is to avoid the "Big Brother" scenario possibly associated with large information systems [118]. On the other hand, certain system features (e.g. user-specific system configuration) are impossible or undesirable (e.g. anonymous authoring) if the user remains truly anonymous. In order to combine advantages of both identified and anonymous user modes, Hyper-G supports four identification modes:

1. **Identified:** In this mode, the identity (name, address, etc.) of users is known not only to the system, but also to other users. E.g., when you receive mail or read annotations or documents from identified users, you know who they are. Users have to log in supplying user name and passowrd. Obviously, monitoring of users' actions is possible in this mode.

2. **Semi-Identified:** The user may select a pseudonym (provided that no other user has already chosen the same name) and a password in identified mode. Subsequently, the user may use semi-identified mode by supplying the pseudonym plus password. This means that the user is still known to the system by full name, however, other users see only the pseudonym when reading annotations, mail, etc. A user may also use more than one pseudonym.

   The fact that the user is known to the system makes user monitoring possible. However, it retains users from misuse of the communication facilities of the system (as shown by our experience with a public mailbox accessible by the austrian videotex system), and requires only minimal software extensions. Also, accounting may be done in the same way as for identified users.

3. **Anonymously Identified:** This mode is simlilar to the semi-identified mode, i.e. users are identified by pseudonyms and passwords. However, the identity of anonymously identified users is not known to the system. E.g., when logging in at a terminal, the user may either enter a valid pseudonym and password or select (or let the system generate) a new combination[6].

   The advantage of this mode compared to anonymous mode is that the system is not able to tell who the user really is (no "Big Brother"), but recognize the user in forthcoming sessions, which allows to store user preferences and configurations and allows *private write* operations (i.e. modifications of the database visible only to the same user, e.g. private annotations). However, *public writes* (i.e. changes to the database visible by other users) are disabled in this identification mode.

   Accounting is still possible in this mode by use of anonymous accounts, i.e. the user pays a certain amount to be added to a certain pseudonym's account.

4. **Anonymous:** This mode will be used where a user identification is too complicated and time-consuming for the job to be done (e.g. at public terminals, museums...). No user preferences or configurations nor any write operations are possible in this mode. The user's and configuration and rights are determined by the terminal profile. However, accounting is still possible at terminals equipped with special hardware to accept tokens, coins, IC-cards and the like.

   Also, system monitoring for system administration and tuning purposes is possible in all identification modes.

[6]Of course, passwords may be changed by the user at any time, supplying the old password.

Identification Mode	System Monitoring	Accounting	Private Write	User Monitoring	Public Write
Identified	YES	YES	YES	YES	YES
Semi-Identified	YES	YES	YES	YES	YES
Anonymously Identified	YES	YES	YES	NO	NO
Anonymous	YES	YES[a]	NO	NO	NO

[a]Special hardware required.

Table 2.2: Supported Functions of User Identification Modes

Table 2.2 summarizes the possibilities of the four identification modes. Note that users may access the system in any desired mode. This enables users to use anonymous or anonymously identified mode most of the time, and switch to other modes only when necessary (e.g. for a public write operation).

2.2.3.16 Access Rights

As mentioned earlier, every user is also a potential author. The extent to which a user is allowed to modify the database is controlled by access rights. A user may belong to a number of user groups, each group being granted particular rights to do something with some portion of the database.

In addition, there are rights associated with the individual terminals and the pieces of information (documents, links etc.) The right to perform a particular function on a particular data item is deducted from the rights associated with the user, the terminal, and the data item itself. A more specific description may be found in section 2.2.4.3 on page 48.

2.2.4 Implied Requirements

This section contains additional requirements that are implied by the basic requirements of section 2.2.2 and the design decisions made in section 2.2.3. Again, they are structured in five groups: *General, User, Author, System* and *Implementation*. Unique labels are assigned to the following requirements in such a way that, if requirement b is implied by basic requirement a, the label of a is a prefix of the label of b.

2.2.4.1 General Requirements

Basic Requirement **Req. G.1** introduces the notion of *documents*. The design decisions made in section 2.2.3.3 and figure 2.4, page 21) suggest the following definition:

Req. G.1.2:	*Document Clusters* are groups of *documents* that are semantically equivalent or related.

Examples of "semantically equivalent or related" documents are:

- A text document in German and the English translation.
- A text document and a sound document containing the voice of someone reading the text document.
- A picture and the description of the picture.

A special type of link (cluster link, see **Req. G.2.4**) is used to link the individual documents of a document cluster.

Req. G.1.3:	Documents, as well as links, collections, tours etc. are subclasses of the class *Hyper-G object*. All Hyper-G objects (and its subclasses) store attributes in addition to the information. These attributes include:
Req. G.1.3.1:	Author and User Group.
Req. G.1.3.2:	Creation/Modification/Access/Expiration time.
Req. G.1.3.3:	Language.
Req. G.1.3.4:	Rights for object's functions, for *author, group* and *world*.
Req. G.1.3.5:	Price for object's functions, for *group* and *world*.

Basic Requirement **Req. G.2** (*Hypertext Axiom*) introduces *links*. The design decisions made in sections 2.2.3.7 (page 24) are used to refine this requirement to:

Req. G.2.1:	The user is offered a visual representation of the link, called *anchor*. The user may 'activate' the link by activating the corresponding anchor.

Of course, the actual meaning of 'activate' depends on the user interface. Typically, the user will point to the anchor using some pointing device and then activate it by pressing a button. However, other means of activation are possible with unorthodox user interfaces (section 4.3.1, page 129).

Req. G.2.1.1:	Anchors belong to links rather than documents. Creation, modification or deletion of anchors is not regarded as document modification.

This implies that even users that have no right to modify a document may create links (with anchors) from/to that document, if they are granted the right to create links (annotation).

Req. G.2.1.2:	The visual appearance of an anchor depends on the destination document type, and on whether the destination document has already been visited during the active session.

This significantly reduces the number of unwanted re-visits of the same document in a session. Also, the special appearence of anchors leading to not-so-common document types actually leads the user to such 'rare information' (e.g. video clips, animation, sound, and images). This feature is most important in the early phase of Hyper-G, where we want to make it easy for users to find such document types. At a later stage, e.g., when high-quality images are not so rare any more, it may become partially obsolete.

The concept of forward and backward (associative) link following described in section 2.2.3.5 (page 23) motivates the following requirements.

Req. G.2.2:	Links are directed, i.e. there are *source* and *destination* anchors. However, Hyper-G supports forward and backward link following.

Req. G.2.2.1:	When forward following a link (i.e. source to destination), on activating the source anchor, the system will activate the destination document in such a way that the destination anchor is visible (if possible).

An example would be a paragraph in a destination text document that is scrolled so that at least the beginning of the paragraph is visible.

Req. G.2.2.2:	When backward following a link (i.e. destination to source), on activating the destination anchor, the system will activate the source document in such a way that the source anchor is visible (if possible).

An example of this would be a part of a map that is scrolled such that the source anchor linking to the destination document (a more detailed map) is centered in the display window.

Req. G.2.3:	Every document has a *default anchor* associated with it. The default anchor covers the whole document and can be used for both the source and the destination of links.

The default anchor is used to link whole documents, e.g. as destination or when using cluster links (see **Req. G.2.4**).

Links can be classified according to a number of classification schemes. The following requirements specify the types of links distinguished by Hyper-G.

Req. G.2.4:	Hyper-G supports *cluster links* and *referential links*. Cluster links are used to tightly couple the individual documents of document clusters.

Cluster links are always user defined, as the semantic relation between two documents can not reasonably be detected automatically. However, referential links may be further subdivided:

Req. G.2.4.1:	Hyper-G supports *user defined* links as well as *automatically generated* links.

Req. G.2.4.1.1:	Automatically generated links may be pre-generated (*static*) or generated on demand (*dynamic*).

Dynamic links are generated using hidden database key queries (section 2.2.4.2, page 39). The user may specify a list of *active databases* that are searched (*link filtering*), as required by **Req. U.3.4.1.1**.

Note that according to above definition user-defined links as well as cluster links are always static. Figure 2.8 gives an overview of the basic link types supported by Hyper-G. The set of link types is extensible, which may be necessary for certain applications such as *Hyper-Animation* (section 3.5.1, page 117).

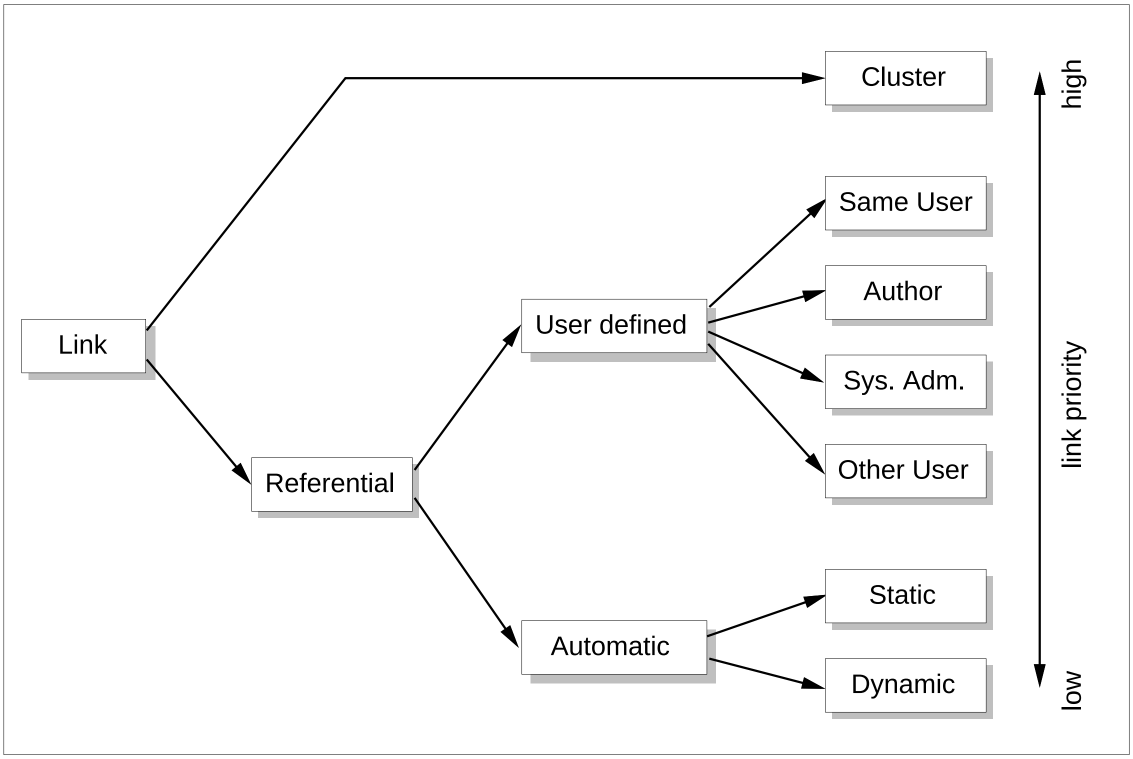

Figure 2.8: Basic Link Types and their Relative Priority

Req. G.2.5:	A *Link Priority* is associated with any Hyper-G link. The link priority is used to reduce the number of links presented to the user.

Of course, the actual meaning of "priority" will vary with the user interface. In particular, a user interface may choose to always follow cluster links (highest priority) without user interference, i.e. to display a whole document cluster at a time. Also, a user interface may choose not to generate dynamic links (lowest priority) when there are any higher-priority links available. Figure 2.8 shows the relative priority assigned to the basic link types of Hyper-G.

Req. G.2.6:	*Active Anchors* have an associated piece of code that is executed when the anchor is activated. Hyper-G supports a number of pre-defined active anchors for dialog documents and special effects.

Active anchors have been discussed in section 2.2.3.7 (page 24). They are most useful with dialog documents (e.g. answer judging) and for special effects (e.g. a special kind of destination anchor might 'fade out' the source document and 'fade in' the destination document.).

Requirement **G.3.1** (Documents and links are maintained automatically) motivates the following:

Req. G.3.1.1:	All Hyper-G objects have an *Expiration Date* associated with them. Hyper-G objects age. When the expiration date is reached, the object may be deleted.

This is to protect the database from outdated or obsolete information. A typical expiration time of a user-supplied document might be six months. Two months before expiration, the user (i.e. the author of the document) is notified (by means of e-mail) and may extend this period. If no response is received, the object may be deleted (and with it all associated links).

Of course, some objects may be given a very long life time by the system administration. Also, additional factors such as use count and time of last access may be used to decide on whether an object is to be deleted (section 2.2.9, page 81).

Req. G.3.2.1:	Whenever a new document is inserted, (static) links are generated automatically.

This is especially useful for annotations: A new annotation of a user is automatically linked to other documents, possibly annotations of other users. An off-line communication between users and authors starts.

2.2.4.2 User-related Requirements

Basic Requirement **Req. U.1** states that Hyper-G shall be usable by untrained users as well as expert users. While expert users may be expected to be familiar with the user interfaces of today's computers, the casual user must not. Therefore, Hyper-G can be operated using a number of different user interfaces (see also section 4.3.1, page 129).

Req. U.1.1:	Users control the system with user interface devices suited to their background knowledge and the application.

Interface Device	User Type	Typical Application	section	page
Keyboard	Home User	Remote terminal access	2.2.3.1.2	18
Keyboard & Mouse	Expert	Univ. Information System	4.1	121
Touch-Screen	Visitor	Viewseum	4.3	128
Home-Trainer	Visitor	HOTACT	4.3.1.3	130
3D input/output	Visitor	Advanced 3D-Applications	4.3.4	140

Table 2.3: Typical User Interfaces, Users and Applications

Table 2.3 shows some typical combinations of user interface device, user type, and application.

Requirement **Req. U.2** ("Hyper-G is a multilingual system") may be refined using the design decisions of section 2.2.3.3 (page 19 and 2.2.3.14 (page 30):

Req. U.2.1:	The user may specify an ordered list of language preferences.

Req. U.2.2:	A document cluster may include semantically equivalent documents that differ only in the *Language* attribute. A user interface may use this information and the user's preferences to select which document to show.

Req. U.2.3:	The user interface of the runtime system is available in a number of languages.

Req. U.2.4:	The language strategy of the user interface can be temporarily overridden.

The last requirement is important in situations where the user wants to see the original version of a document. E.g, when looking at the English translation of a German poem, the user may wish to see the original German version even if the perferred language is English, or even see both versions at the same time.

The Hyper-G user is supported by high-level navigation tools (**Req. G.3**). These tools include user interface metaphors (section 2.2.3.10, page 27), database queries (section 2.2.3.13, page 30), collections (section 2.2.3.11, page 28), and tours (section 2.2.3.12, page 29). Let's deal with user interface metaphors first:

Req. U.3.1:	A variety of user interface metaphors is offered, including:
Req. U.3.1.1:	*Simple Hypermedia* (section 2.3.3)
Req. U.3.1.2:	*Desk-Top* (Xerox' *Star* [81], Apple Macintosh)
Req. U.3.1.3:	*Stack* (HyperCard [185])
Req. U.3.1.4:	*Book* [10]
Req. U.3.1.5:	*Holiday Travel* [61]
Req. U.3.1.6:	*Library* [35, 138]

The list of supported user interface metaphors is extensible, as the user interface is built upon an underlying 'Hyper-G Kernel' that supports common functions (section 2.2.5, page 52).

Req. U.3.1.7:	Most User Interface Metaphors support simple navigation functions, such as *undo, redo, help, drop/goto book-mark, show copyright, show links, show annotations, show collection, show tours...*

Other means to aid the user's navigation are collections:

Req. U.3.2:	Information is structured in *Collections*. A collection is a set of documents or other collections ('subcollections').

This recursive definition yields a hierarchical structure of collections:

Req. U.3.2.1:	The collection hierarchy is an acyclic directed graph. There are a number of 'root' collections.

The collection structure has been presented in section 2.2.3.11 (page 28). To put it simple, 'acyclic directed graph' means that the terms 'ascending' and 'descending' the structure are well defined like in a tree, however, a collection may be a subcollection of a number of collections.

Req. U.3.2.2:	Any document in the system is a member of at least one collection. The user may find the collection(s) a document belongs to, get a graphical representation of the collection hierarchy at that point, 'ascend' or 'descend' the hierarchy, and find other documents belonging to the collection.

Note that because of the logical definition of the collection hierarchy, and because of the relatively small number of collections, graphical browsing of the collection hierarchy seems feasible.

Tours are described in section 2.2.3.12 (page 29) and motivate the following requirements:

Req. U.3.3:	*Guided Tours* allow to traverse the information network in a way specified by the author (of the tour). However, tours do not have to be a linear list of document clusters to be visited, but may utilize the full flexibility of the Hypermedia system.

Req. U.3.3.1:	Document clusters may be visited by a number of tours. When the user follows tour *a* and visits a document cluster that is also visited by tour *b*, the user may switch tours at this point.

That is, the user can get a list of the tours visiting that document cluster, and may choose a tour to follow. However, at some later point, the user should have the opportunity to continue the original tour. It turns out that requirement **Req. U.3.3.1** leads to the concept of labeled links, i.e. links that are labeled with the tour name (section 2.2.6.1.5, page 67).

Especially in the case of educational tours the user should be supported in making sure that nothing has been overlooked:

Req. U.3.3.2:	The user is offered information on what parts of the tour have already be seen and what parts have not.

In addition to the database queries that are made behind the scenes when generating dynamic links, Hyper-G offers user controlled database queries (section 2.2.3.13, page 30):

Req. U.3.4:	Hyper-G supports *Database Queries* to allow fast access to relevant information.

Two types of queries may be distinguished:

Req. U.3.4.1:	*Key Queries* allow for prefix search in the database(s).

This type of query is most useful at the beginning of a session, when the user wants to gain an overview of what is in the database(s) related to a given topic (keyword). Key queries may be combined by 'and', 'or' and 'not' operators.

Req. U.3.4.1.1:	The user may specify a list of *active databases*. Only these databases are searched.

This not only increases speed, but also reduces the number of unwanted links (*link filtering*). Note that dynamic links are done by key queries, too, so that the list of active databases also applies to the generation of dynamic links.

The databases correspond to the *collections* of **Req. U.3.2**. Thus, a 'meta-collection' containing all collections allows to search for whole collections instead of individual documents, if this meta-collection is the only active database. Similarly, a collection containing all the tours lets the user search for tours with a certain associated keyword instead of individual documents which may be part of the tours.

Req. U.3.4.1.2:	The system allows for *inexact match* key queries.

This may be done in two ways:

- Either the user may specify a kind of regular expression to be searched for, or
- The system itself tolerates a certain amount of spelling errors (e.g. "naboleon" should find "Napoleon").

Hyper-G will offer both approaches. In addition, the use of synonym databases and hierarchical synonym classification schemes will be investigated.

Req. U.3.4.2:	*Form Queries* are more flexible, but collection-specific queries. The user is allowed to fill out a form with a number of attributes that may be combined to perform complex queries. Forms may be filled out iteratively, showing after each incremental step the quantity of information found and hints on how to continue.

As stated in section 2.2.3.7 (page 24), the user interface for database queries (especially form queries) is most easily made by dialog documents with active anchors.

Basic requirement **Req U.4** ("The system is configurable on a per-user basis...") motivates:

Req. U.4.1:	The system maintains a user profile, describing:
Req. U.4.1.1:	Security Data (User name, user group(s), account number, password, access rights
Req. U.4.1.2:	The user's system entry point (login menu, starting collection)
Req. U.4.1.3:	The user's language preferences
Req. U.4.1.4:	Preferred user interface metaphor
Req. U.4.1.5:	Preferred user interface metaphor's options
Req. U.4.1.6:	User logging (full/system/off)
Req. U.4.1.7:	Default active databases for queries
Req. U.4.1.8:	Whether user may change profile

A few explanations are necessary:

The *security data* is used to identify the user. However, the system may also be used anonymously (**Req. U.6**), in which case the default user profile of the user's terminal is applied. Of course, anonymous users cannot alter the configuration[7].

In principle, the *user's system entry point* might be any Hyper-G object. However, to define a root-level collection (remember, the collection hierarchy may have more than one root) or a dialog document that performs a database query would be reasonable.

The *user's language preference* is an ordered list of languages (**Req. U.2.1**). The selection 'Hungarian' – 'German' – 'English', for example, means that if a Hungarian version of a document is available, show it, else show the German version, else the English one. If none of the user's list is available, the system may display an appropriate error message, or display whatever it has. Also, the user interface language is to be set to the language of the first choice, if possible.

The *preferred user interface metaphor's options* are further options specific to the *preferred user interface metaphor*. For example, the user may specify to prefer short versions instead of long versions of text or voice documents, if semantically equivalent versions are available in a document cluster; or to prefer hearing a voice document rather than reading a semantically equivalent text document. In addition, preferred document managers (section 2.2.5.2, page 56) that should be used to

[7]However, semi-identified and anonymously identified users potentially can (section 2.2.3.15, page 32).

display certain document types can be specified. An interface metaphor might even allow to configure the exact layout (position, size) of the various windows.

User logging (**Req. U.5.1**) can be turned to 'full' (full user state logged), 'system' (only the items necessary for **Req. U.5.1.2** logged) or 'off'.

As we shall see, the list of *active databases* is actually a list of active collections, to which queries are to be passed.

Basic Requirement **Req. U.5** ("... user may return to a previous user state") requires the system to keep user logs:

Req. U.5.1:	For every user, the system maintains a *user log*. The log stores a sequence of *user states*.

The log enables the system to perform a sort of *Undo* function, i.e. to return to a previous state (**Req. U.5**). The user state records all information necessary to do this. In addition, some graphical representation of the user's path through the information network may be displayed (*history*).

Req. U.5.1.1:	Identified, semi-identified and anonymously identified users may also navigate backwards to the user states of previous sessions.

This is done by having *persistent* user logs, i.e. by saving the user log across sessions. Of course, this feature is not available for anonymous users.

Req. U.5.1.2:	User logs may be used by authors and/or system administrators to maintain the database.

Sometimes, users get stuck because of misleading or missing links, etc. The user logs may be used to find such weaknesses of the database. Not all the information of the user state is required to be logged ('User logging' set to 'system' would be sufficient (**Req. U.4.1.6**).). For example, only unsuccessful database queries need to be logged for this purpose. In particular, anonymous users may be traced also.

In addition to the identified and anonymous user mode (**Req. U.6**), Hyper-G supports two more identification modes:

Req. U.6.1:	The system may be used in *Semi-Identified Mode*. In this mode, the real identity of the user is known to the system, but not to other users.

Req. U.6.2:	The system may be used in *Anonymously Identified Mode*. In this mode, the real identity of the user is not known to the system. However, the user's pseudo-identity persists across sessions.

Advantages and disadvantages of the four user identification modes have been discussed in section 2.2.3.15 (page 32).

Requirement **Req. U.7** states that certain system features and/or information items are chargeable.

Req. U.7.1:	Any Hyper-G object may be priced. The price is specified by the author of the object (**Req. G.1.3.5**). The system automatically tells the user in advance the price of the object to be accessed, if its greater than zero.

In order to charge both identified and anonymous users, Hyper-G supports two charging schemes:

Req. U.7.2:	Hyper-G maintains user accounts for identified, semi-identified and anonymously identified users. Whenever a priced object is shown, the object's costs are deducted from the user's account.

In theory, objects may also have a negative price, in which case the amount is added to the user's account. Identified and semi-identified users may be allowed to overdraw their account, and will receive an invoice at regular intervals.

Req. U.7.3:	At special terminals, anonymous users may be charged also.

This is done by letting the users (anonymously) buy tokens of a certain value, and then deduct from that value until the remaining value on the token reaches zero. Tokens will typically be credit card-like magnetic cards or IC-cards. Of course, anonymous and anonymously identified users are not allowed to overdraw their account.

2.2.4.3 System Requirements

Req. S.1.1:	For each terminal, there is a terminal profile, describing:
Req. S.1.1.1:	Terminal facilities, such as:
Req. S.1.1.1.1:	Input Devices
Req. S.1.1.1.2:	Voice I/O capability
Req. S.1.1.1.3:	Digital Sound I/O capability
Req. S.1.1.1.4:	Digital Video I/O capability
Req. S.1.1.1.5:	Hardcopy capability
Req. S.1.1.1.6:	Softcopy (diskette) capability
Req. S.1.1.1.7:	Communication capabilities (E-Mail, conferencing)
Req. S.1.1.1.8:	Charging of anonymous users
Req. S.1.1.2:	Security information, such as:
Req. S.1.1.2.1:	The terminal's access rights (operations allowed on this terminal)
Req. S.1.1.2.2:	The rights of anonymous users at this terminal
Req. S.1.1.3:	The prices of specific terminal facilities

Again, a few explanations are in order:

Useful combinations of *Input Devices* are listed in table 2.3 (page 40). Some terminals may be equipped with hardcopy or softcopy devices ('softcopy' means that the user/visitor can take a diskette back home with selected documents on it).

Hard- and Softcopy facilities are usually priced (**Req. S.1.1.3**). The charging of anonymous users may be done using tokens, coins, or terminals equipped with magnetic or IC-Card readers or similar devices (**Req. U.7.3**). No such hardware is neccessary for anonymously identified users (section 2.2.3.15, page 32), as they can be charged by prepayment.

Requirement **Req. S.2** states that access rights control whether a certain user on a certain terminal may access a certain object. Considering the design decisions made (section 2.2.3.16, page 34) and implied requirements **Req. G.1.3.4**, **Req. U.4.1.1**, and **Req. S.1.1.2**, we can further refine this concept:

Req. S.2.1:	The right of a certain user on a certain terminal to perform a specific function of certain Hyper-G object is deduced from the object's protection bits (for *author, group* and *world*, **Req. G.1.3.4**), the user's rights **Req. U.4.1.1** and the terminal's access rights **Req. S.1.1.2.1**. Anonymous users get the terminal's default right (**Req. S.1.1.2.2**). Users of user group *system* may override rights.

To be specific, the user's right to perform a certain operation is calculated in the following way: For any operation (e.g. read, write, print, copy, link to, link from; described in section 2.2.6.1, page 60) there is an associated bit, say bit number i. If it's set to '1', this means "allowed", a '0' means "not allowed". These *protection bits* are stored together with the following entities:

- Any Hyper-G object holds protection bits for:
  - Author (*object.author.bits*)
  - Group (*object.group.bits*)
  - World (*object.world.bits*)
- Protection bits are associated with every user (*user.bits*).
- Access rights are also associated with terminals (*terminal.bits*).

In addition, a user may be a member of a number of user groups ($user \in G_i$). Members of group *system* may override any protection bits with exception of *terminal.bits*, i.e. system administration can only be done from a subset of the terminals as an additional security measure. Also, if for example a terminal does not support printing, it does not make too much sense to insist on your right to print something. With these prerequisites in mind, the protection algorithm looks like shown in figure 2.9.

This protection scheme is somewhat similar to the UNIX approach, but with the following modifications:

- There are terminal protection bits. This serves for two reasons:
  - The terminal's protection bits are taken into account for all users, including the system administration. This means that the system administrators may login at any terminal, however, the specific rights needed to perform system administration (e.g. the right to delete objects) are only granted at specific terminals.

if *user* $\in$ *system*
  then *bits* := *terminal.bits*
  else if *user* = *object.author*
    then *bits* := *object.author.bits* **and** *user.bits* **and** *terminal.bits*
    else if *user* $\in$ *object.group*
      then *bits* := *object.group.bits* **and** *user.bits* **and** *terminal.bits*
      else *bits* := *object.world.bits* **and** *user.bits* **and** *terminal.bits*

if (*bits* **and** 2^i) $\neq 0$
  then /* *function #i allowed* */
  else /* *function #i not allowed* */

Figure 2.9: Hyper-G Protection Scheme

  - The protection bits also specify certain functions to do with the document that are not available at all terminals, e.g. hardcopying. A terminal that does not support that specific function would set the according protection bit to "not allowed", overriding any different settings of user and/or document protection bits.

- The users themselves have protection bits associated with them. This may be used to specifically deny certain rights from a user. For example, for some reason a user may be (temporarily) denied the right to annotate something, without the need to change the rights of the objects or modify user groups.

- The system may be used anonymously, in which case users get their rights from the terminal profile (**Req. S.1.1.2.2**).

As stated in **Req. G.1.3.5**, a Hyper-G object has a price associated with a particular function (e.g. read, hardcopy, softcopy, link to, link from) for group and world (of course, if the corresponding function is prohibited by the associated protection bit, the 'price tag' is meaningless; also, the price for the author is always zero). Calculation of the price to be paid by a specific user is done similar to the protection scheme described above. If the user is a member of the specified user group, the 'group' price tag is used, else the 'world' price is to be paid.

A typical application would be to specify that linking-to an object (e.g. using it in a tour) costs amount w for anybody except users of group *mybestfriends*, which pay amount g.

It should be noted that variable-length lists have been avoided in the design of Hyper-G object's attributes, because of their implementation overhead associated with database design. As a consequence, in order to distinguish between a number of user groups, more database objects pointing to the same piece of information (e.g. document) have to be established. In our above example, in order to let users of user group *mysecondbestfriends* pay amount *g*1, a second Hyper-G object with 'group' set to *mysecondbestfriends* has to be created.

Also, the terminal has prices associated with certain functions, which are added to the price determined by the object (e.g., the price for a hardcopy is calculated by the price the author specified plus the price the terminal's price for a copy). However, the amount collected for using the Hyper-G object is transferred to the author, while the amount collected by the terminal is for the system (administration).

So, as a conclusion:

Req. S.2.2:	The price to be paid for a certain function on a certain object at a certain terminal is determined by the object's price tags (for *group* or *world*, **Req. G.1.3.5**) and the terminal's price for that particular function.

2.2.4.4 Implementation Requirements

From the design decisions and requirements of previous sections it is quite obvious that Hyper-G will use a set of databases to store its documents, links, anchors etc. Because Hyper-G's data items are structured in an object-oriented way (e.g. they are all derived from a class 'Hyper-G object') it would be best to use object-oriented databases to do the job. Unfortunately, only experimental object-oriented database systems suitable for our purposes exist at this time. Also, a standard relational database system would turn out to be too slow to handle the huge number of objects Hyper-G developed for. In addition, there are legal problems associated with the use of commercial database systems. Hence we decided to build our own (simple, but speedy) database management system customized to the special needs of Hyper-G.

In fact, Hyper-G will use two kinds of databases (some arguments for this split are discussed in sections 2.2.5 (page 52) and 2.2.6 (page 60)):

Req. I.1.1:	The *Keys & Attributes Database (KAD)* holds the keys and attributes of all Hyper-G objects. In order to gain speed, the database is split into specialized databases holding:
Req. I.1.1.1:	documents
Req. I.1.1.2:	links
Req. I.1.1.3:	anchors
Req. I.1.1.4:	collections
Req. I.1.1.5:	tours

The attributes stored with each object are listed under **Req. G.1.3**. The keys are used to search for the object using the simple key queries described under **Req. U.3.1.4** as well as to link the links to the anchors and the anchors to the documents. The *KAD* will be implemented using simple text files with inverted indices. It does not contain documents, only pointers to the documents are stored.

The documents themselves are stored as part of a collection database. While sometimes this collection database will just contain pointers to the actual documents (e.g. to raster images), text documents (e.g. the entries of an encyclopedia) might be grouped together in one file, as this allows a collection-wide full text search that is needed for associative dynamic links (section 2.2.3.5, page 23) and form queries (section 2.2.3.13, page 30). Also, the graphical appearance of anchors is stored together with the documents, while their attributes are stored in the KAD.

Req. I.1.2:	The documents and anchors are stored in collection databases, while their keys and attribute are held in the KAD. The actual layout of the databases will vary with the type of collection. The collection supports collection-wide full-text search (if applicable) and/or form queries.

Note that the notion that the collection itself performs this actions and the exact layout of the database is not to be known outside conforms to the object-oriented design principles of encapsulation and data hiding.

Req. I.1.3:	The databases themselves ensure database integrity.

For example, if a document is to be deleted, also the anchors of the document as well as the links to/from that anchors are deleted. This facilitates database maintenance as required by **Req. G.3.1** and **Req. S.3**.

As stated in section 2.2.3.1.3 and shown in figure 2.2 (page 18, Hyper-G allows access to remote databases.

Req. I.2:	Hyper-G supports access to remote databases. Access to such databases shall be hidden from the user.

This means that users areconfronted with a consistent user interface metaphor throughout the system, so that they would not be able to tell whether an information item is retrieved from a local or remote database (except, perhaps, because of a delay when retrieving remote information). More on this in section 2.2.6.2.3 (page 74).

2.2.5 Process Distribution

As discussed in section 2.2.3.1 (page 17) Hyper-G will be developed as a distributed system, running concurrently on a heterogeneous network of computers. For the sake of simplicity we will distinguish between workstations and Hyper-G servers, although the distinction is somewhat arbitrary, as modern UNIX workstations may also accomplish the tasks that we will associate with Hyper-G servers.

Hyper-G servers and workstations are interconnected by a high-speed local area network (LAN), with optional front-ends (terminals) and back-ends (remote databases). In this section, we will concentrate on the core system (see figure 2.1 on page 17), and forget about front- and back-ends.

Communication between Hyper-G servers and workstations is done via the X protocol [141], i.e. the workstations run *X Servers*[8] and the Hyper-G servers run X clients. On the Hyper-G server side, in most cases we do not access the *Xlib* [142, 143] directly, but use InterViews [97, 98] – an object-oriented user interface toolkit written in C++ [169] – instead.

The Hyper-G server itself consists of a number of *Server Processes* (figure 2.10) that communicate via message passing and may be distributed among a number of server machines. This split into different processes offers a number of advantages:

- The processes may run concurrently on different machines, increasing performance.
- Certain processes may run on specialized server machines (e.g. database server, 'video server' equipped with real-time digitizing hardware).
- It makes Hyper-G implementable by a number of individual programmers that implement a single process without needing a lot of knowledge about other processes. This aspect is also due to object-oriented design principles (section 2.2.6.1.1, page 61).
- As we will see, each document type is handled by a specialized process. This allows for painless extension of the set of supported objects (**Req. G.4**) and the implementation of document clusters.

As shown in figure 2.10, Hyper-G consists of the following groups of processes:

1. The **Session/Interface Manager (S/IM)** controls the session and user interface.

[8] Sorry about this terminology, but thats the way it is...

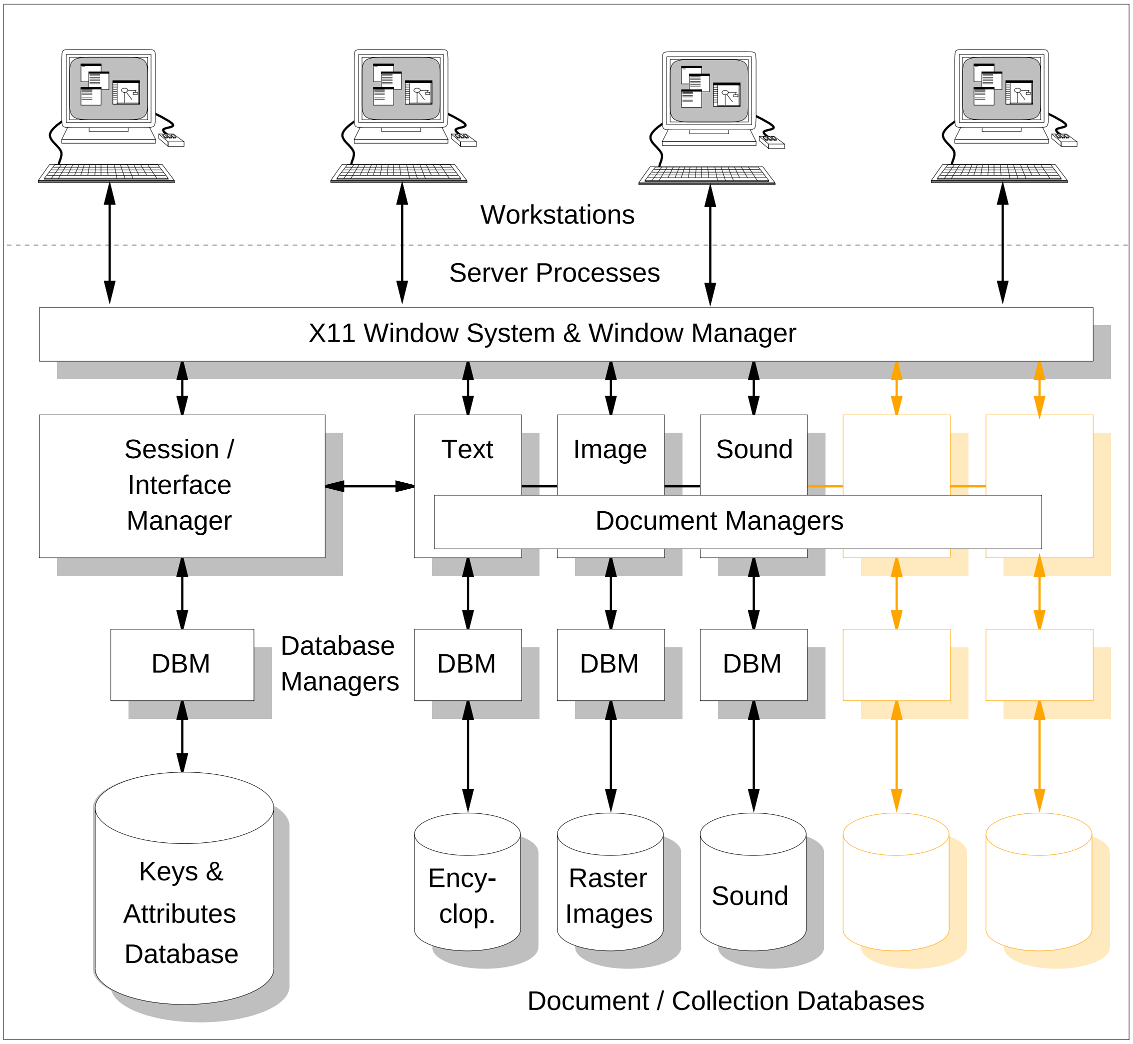

Figure 2.10: Server Processes

2. **Document Managers** (one for each document type) control the display and manipulation (e.g. scrolling) of the documents of corresponding type as well as anchors tied to that document.

3. A number of **Database Managers (DBM)** implement the interface to the corresponding databases.

The following sections discuss these modules of Hyper-G in more detail.

2.2.5.1 Session/Interface Manager

The *Session/Interface Manager (S/IM)* accomplishes two closely related tasks:

1. It controls the entire Hyper-G session, involving the following sub-tasks:

(a) It starts and controls the other processes (document managers and database managers) via remote procedure calls (RPC) [31].

(b) It decides on the documents of a document cluster that are to be displayed. This decision is based on the user's *preferred user interface metaphor* (**Req. U.4.1.4** and *preferred user interface metaphor's options* (**Req. U.4.1.5**) of the *user profile* (**Req. U.4.1**, page 44). For example, the user may specify to prefer short versions instead of long versions of text or voice documents, if possible, or to prefer hearing a voice document rather than having to read text document. The cluster links stored in the KAD are used to find the individual documents of a document cluster.

(c) When a document is to be 'executed' (e.g. displayed), the S/IM may either start a new document manager or re-use an existing one (again depending on the user interface metaphor). Which document manager is to be used is determined in the following way:

The KAD (**Req. I.1.1**) stores the document type of each document. This document type is taken as an index into the *Document Manager Table (DMT)* to select an appropriate document manager for that document. Depending on the user interface metaphor and the user's and terminal's rights, the DMT may be user-configurable (**Req. U.4.1.5**), i.e. the user may tell the S/IM which document manager to use for documents of a specific document type. Also, the document managers may be started with configurable parameters.

(d) It implements link following, using the information provided by the KAD (section 2.2.6.2.1, page 70). To be specific, the link and attribute concepts of Hyper-G are realized as follows:

i. **Static Links:** When the user clicks on a *source anchor* (section 2.2.3.7, page 24) of a document, the corresponding document manager returns the *anchor ID* to the S/IM, which in turn uses the *link-anchor relation* of the KAD (via a DBM) to find the links attached to that anchor. If there is more than one link, the *link priority* (section 2.2.3.6, page 24) or user interaction may be used to select a single link to follow.

Once the link is identified, the corresponding *destination anchor* is known and used to find the destination document using the *document-anchor relation* of the KAD. The new document is part af a document cluster, which is then displayed as described under 1b.

ii. **Associative (backward) link following** (section 2.2.3.5, page 23) of static links: Just exchange the words 'source' and 'destination' in the preceding paragraphs.

iii. **Dynamic Links** (forward and backward following): When the user clicks on a word, a database query (described under 1e) is performed. The result is a document ID, which is then used to branch to the corresponding document cluster. There are no links and anchors involved.

iv. **Active Anchors:** When a document (e.g. a dialog document) contains active anchors, the code associated with them is executed by the corresponding document manager. The result is always an anchor ID, which is returned to the S/IM. The S/IM sees no difference to an ordinary static link, so it is treated as described under 1(d)i.

(e) It performs Database Queries both as a part of dynamic link generation and under user control (section 2.2.3.13, page 30) . Depending on the desired operation, two kinds of low-level database queries have to be performed:

i. **Forward dynamic link generation** and **key queries** (**Req. U.3.4.1**) require the *document database* (**Req. I.1.1.1**) of the KAD to be searched for a document that has the search string in its key list. The user may restrict the search to documents of specific collections (**Req. U.3.4.1.1**).

ii. **Backward dynamic link generation** and **form queries** require collection-specific database queries, and are therefore passed to the collections, which in turn use the corresponding collection database to perform the search. In general, this kind of query will be slower than a search in the KAD (in the case of a collection of text documents – e.g. an encyclopedia – a full-text search is required), but because of process distribution can be performed in parallel on a number of collections, if the corresponding DBMs are executing on different machines.

(f) It checks the access rights of a user to perform specific functions on an object (**Req. S.2.1**), based on the algorithm shown in figure 2.9 on page 49.

(g) It maintains the user accounts (**Req. S.2.2**).

2. It implements the user interface. In particular,

(a) It implements the user interface metaphor (**Req. U.3.1**), based on the user's preferences (**Req. U.4.1.4**).

(b) It controls the various input and output devices (**Req. U.1.1**).

(c) It allows simple navigation functions, such as *undo, redo, help, drop/goto book-mark, show copyright* etc.

(d) It allows to display (a subset of) the collection hierarchy, relative to the 'current' collection, and to navigate within the collection hierarchy.

(e) It allows the user to go on guided tours (section 2.2.3.12, page 29) and switch between tours.

(f) It lets users configure their user profiles (**Req. U.4.1**).

(g) It maintains the user logs, which may be used for navigation (*history*) and system administration.

2.2.5.2 Document Managers

The document managers are responsible for the display and manipulation (e.g. scrolling, zooming etc.) of the individual documents. In addition, they handle the appearance and activation of anchors tied to the documents. The important point is that for the S/IM the document managers look all the same, i.e. the private data of the documents and anchors is hidden from the S/IM. This allows to extend and/or change the set of supported document types, the way the documents and anchors are stored etc. by changing only the document manager or corresponding database manager without impact on the rest of the system.

In Hyper-G, every document is a member of at least one collection (**Req. U.3.2.2**). Every collection has a specific database manager (DBM) associated with it. For example, an encyclopedia collection consisting of thousands of text documents will usually be stored as a single text file with associated index files to speed up searching. However, the corresponding DBM will present only one document at a time to the text document manager. Again, changing the underlying data representation would only affect the DBM and leave the rest of the system unchanged.

By now it should be obvious that the design of Hyper-G relies heavily on the object-oriented design principles of *abstraction* and *encapsulation* [32]. In addition, the implementation of document managers utilizes the concepts of *inheritance* and *polymorphism*. To put it simple, from outside the document managers look all the same (i.e. they implement the same set of methods), but internally they are inherited from a generic class *DocumentManager*. The *virtual methods* of this class get called from outside and are passed on to the real document manager that actually implements the method. Table 2.4 lists the most important and public virtual methods of class *DocumentManager*, corresponding to the functions that can be performed with a document.

Note that the base class DocumentManager contains no methods to manipulate the document itself or the attached anchors. Because of the great differences of the document types, these methods are incorporated into the individual document managers and are used by editing tools that are specialized for specific document types. The functions *copy, link to, link from* etc. can be performed in the KAD without interference with the document manager.

Also, there is no method to poll the state of attached anchors. Rather, the document manager communicates asynchronously with the S/IM in case of an anchor being activated. It calls S/IM's method

```
void ActivateAnchor(AnchorID, int mode)
```

to tell the S/IM that an anchor was activated. The 'mode' flag can be used to distinguish between modes of activation, e.g. single or double click, left or right mouse button, etc.

Method	Comments
int ReadDocument(Path)	Identify the 'current document'. If a current document is already read in, it is replaced by the new one. Depending on document type, suffixes may be used to specify portions of a file (e.g. file pointer offset).
int Execute(AnchorID)	'Executes' the current document. With standard document types, this will just display the document so that the given anchor is visible. E.g., with sound documents, this will (re-)start the sound at the given anchor. If AnchorID is nil, the whole document is to be executed.
int SetState(AnchorID,state)	Sets the state of an anchor. State bits include visible/invisible, clickable, color, etc.
int SetGeometry(char*)	Allows to move and/or resize the window of the document manager from outside. E.g. the string "640x480+20-10" will resize the window to 640×480 and position it at 20 pixels from the left and 10 pixels from the bottom.
char* GetGeometry()	Returns the position and size of the document manager's window.
int Iconify()	Tells the document manager to iconify itself.
int DeIconify()	Tells the document manager to deiconify itself.
char* HardCopy(char*)	Returns PostScript [1] code that contains a printable representation of the document. Options may be passed.
char* SoftCopy(char*)	Returns an encoding of the document suited for saving on user disks. Options may control the target format.
int Terminate()	Terminates the document manager.

Table 2.4: Public Virtual Methods of Class DocumentManager (C++ -like notation)

2.2.5.3 Database Managers

Database Managers allow to access databases without too much knowledge about the actual data layout. There are a number of database managers, namely the *Keys & Attributes Database Manager (KADBM)* and the *Collection Database Managers (CDBMs)*. They all are inherited from a generic database manager class *Database-Manager*.

Method	Comments
Links* GetLinkBySrc(AnchorID)	Uses the *anchor-link relation* of the KAD to find the link corresponding to the passed source anchor. Returns a list of link objects (section 2.2.6.1.4, page 65).
Links* GetLinkByDest(AnchorID)	Same as above, but the passed anchor is interpreted as destination anchor (used for backward link following).
Document* GetDocBySrc(AnchorID)	Uses the *document-anchor relation* of the KAD to find the document that contains the passed anchor as source anchor. Returns pointer to document object (section 2.2.6.1.2, page 62).
Document* GetDocByDest(AnchorID)	Same as above, but the passed anchor is interpreted as destination anchor.
DocList* GetDocByKey(char*)	Searches the document database for documents with the passed key in their key list. Returns a list of document objects.
DocList* GetDocCluster(DocumentID)	Returns a list of all document objects that are semantically equivalent with the passed document by examining the link database.
AnchorList* GetAnchors(DocumentID)	Uses the *document-anchor relation* of the KAD to find all the anchors attached to a given document. Returns a list of anchor objects (section 2.2.6.1.3, page 65).

Table 2.5: Public Methods of Class KADBM used by S/IM

Table 2.5 lists the methods offered by the KADBM that are used by the S/IM to perform link following and database queries. Of course, there are more functions that are used to create and modify Hyper-G objects.

The methods of the individual collection database managers vary according to the type of collection. However, this is not a serious drawback because the only other processes that need to communicate with the collection database are the corresponding document manager and dialog documents performing collection database queries. The S/IM requires only one method of the collection database managers (table 2.6).

Method	Comments
DocIDList* FindDocument(char*)	Searches the collection database for documents that contain the passed string. Returns a list of document IDs (used for backward following dynamic links).

Table 2.6: Public Virtual Methods of Class CDBM used by S/IM

Care has to be taken in the implementation that the database manager code is reentrant, so that it can be used by a number of other processes simultaneously. This is most easily done by offering only atomic functions, i.e. no information (including state information) is to be preserved by the document manager between function calls.

2.2.6 Data Distribution

Figure 2.10 (page 53) shows not only the distribution of Hyper-G processes, but also the distribution of Hyper-G's databases. In general, three types of data are stored in separate databases:

1. Keys and attributes common to all Hyper-G objects are stored in the KAD (section 2.2.6.2.1, page 70). Only the S/IM is allowed to access the KAD by using the KADBM (section 2.2.5.3, page 57). This way, structural changes in the KAD (e.g. changing data structures and algorithms) require changes in the KADBM only. Changes in the contents of the KAD (e.g. additional attributes) are reflected by changes of the S/IM. No other processes are concerned.

2. Collection-specific data may be stored in collection-specific databases, that are accessed via CDBMs (section 2.2.5.3, page 57). These CDBMs are accessed by the S/IM in order to perform full-text searches (table 2.6 on page 59) and by collection-specific dialog documents that may perform more complicated queries on the collection (e.g. in a geographical collection, the distance between two locations may be computed and used to query the database; in order to do so special attributes – degrees of latitude and longitude – have to be contained in the collection-specific database). Again, changes in the collection-specific database affect only the corresponding CDBM and the dialog object.

3. The document itself is passed to the document manager without interpretation by the CDBM. Conceptually, it is contained in the collection's database as a *binary large object* (BLOB), while in reality the database will just contain a pointer (e.g. a filename) to the document and pass it to the document manager. Only the document manager needs to know about the format of the document, restricting changes in the document format to affect the document manager's code.

2.2.6.1 Hyper-G Objects

Documents, as well as links, anchors, collections, tours etc. are subclasses of a base class called *Hyper-G object* (**Req. G.1.3**). The keys and attributes of Hyper-G objects are stored in the KAD. Table 2.7 lists the data items a Hyper-G object consists of (and hence are stored with any instance of a derived object class).

The actual representation of the individual data items of table 2.7 (e.g., whether times are stored as seconds past midnight on January 1^{st} 1970, or in a specific string format) affects only the KADBM and is of no significance to the rest of the system. In a first version, all databases are implemented as simple text files, so the data items are all represented as strings. Figure 2.12 (page 64) lists an example entry of

Data Item	Comments
ObjectID	A unique identification of the object. When referencing to an object of a derived class, e.g. document, we will call this ID *DocumentID* to emphasize that it identifies a document.
Type	Type of the object. Currently one of *document, anchor, link, collection* and *tour*.
Author	The electronic author (owner) of the object.
Group	A user group this object is assigned to. Users of this group may perform certain functions on the object, as specified by the rights field.
TimeCreated	The time when the object was created.
TimeModified	The time when the object was modified for the last time.
TimeAccess	The time of the last access to the object.
TimeExpire	Expiration date of the object (**Req. G.3.1.1**)
Accessed	The number of accesses (read or write) to the object.
Language	The language of the object (if applicable).
BitsAuthor	The access rights for the author of the object (see **Req.S.2.1**, page 48).
BitsGroup	The access rights for the users of user group *Group* (see above).
BitsWorld	The access rights for the users that are neither the author of the object nor members of user group *Group*.
PriceGroup	The prices of individual functions for members of user group *Group*.
PriceWorld	The prices of individual functions for users not member of user group *Group*.

Table 2.7: Keys and Attributes of Class *Hyper-G Object*

the document database. Note that the file format can be changed at any time to increase speed or reduce storage requirements by just changing the KADBM.

2.2.6.1.1 Object-Oriented Design Principles

To the reader familiar with the paradigm of object-oriented programming it must be quite obvious that an object-oriented approach is ideally suited to the task of implementing a Hypermedia system like Hyper-G [9]. The design principles of *abstraction* and *encapsulation* have already been discussed in previous sections (section 2.2.5.2, page 56). This and the following section are concerned with *inheritance*.

To put it simple, inheritance means that a class can be derived from another class (the so-called base class) and automatically inherits the variables and methods of the base class. In addition, more variables and methods may be defined, and methods of the the base class may be redefined in the derived class. By applying this concept recursively, a *class hierarchy* grows. Figure 2.11 shows part of Hyper-G's class hierarchy derived from the base class *Hyper-G Object*.

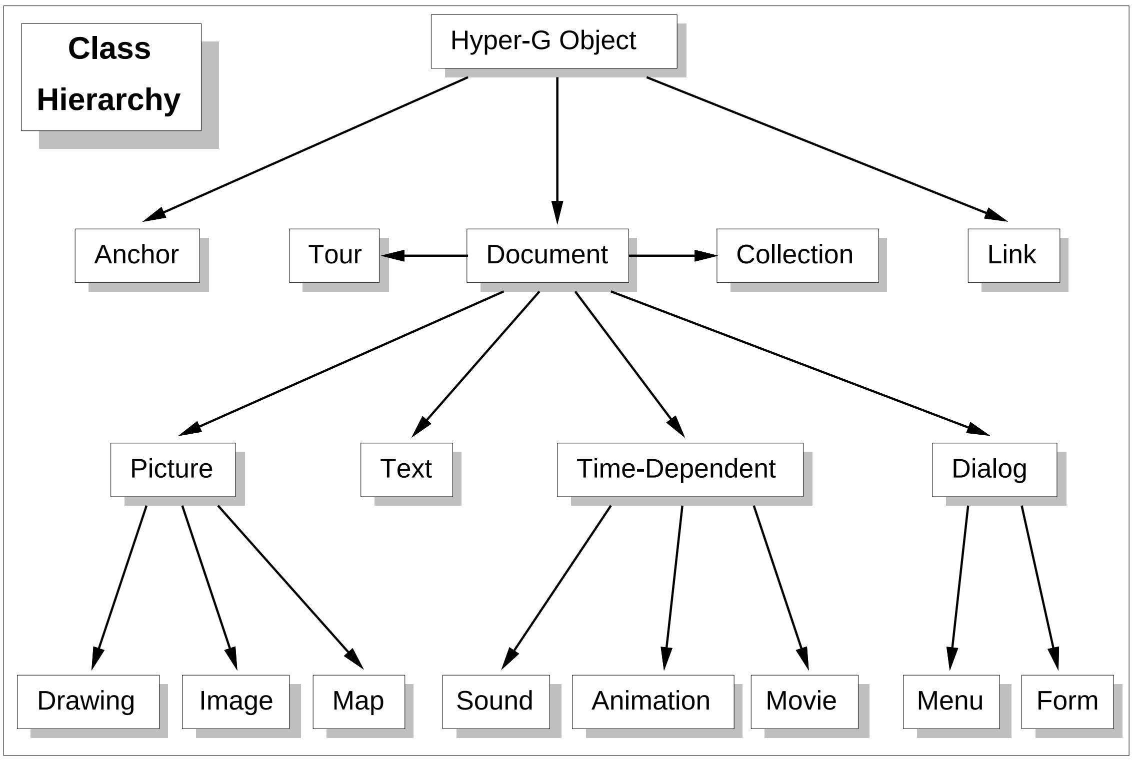

Figure 2.11: Part of the Hyper-G Object Class Hierarchy

As mentioned earlier, Hyper-G will be implemented in C++. C++ supports *multiple inheritance*, which allows to inherit from a number of base classes. However, this feature will be used rarely in order to keep the class hierarchy simple. The class hierarchy as shown in figure 2.11 applies to both the data stored in the KAD, as well as to the code in the S/IM and document managers.

2.2.6.1.2 Documents

[9]A comprehensive discussion of the advantages of object-oriented design and object-oriented programming languages is beyond the scope of this document. For in-depth information the reader is referred to [32, 169, 181] and [182].

As shown in figure 2.11, the class *Document* is inherited from the base class *Hyper-G Object*, i.e. in incorporates all the attributes of Hyper-G objects (table 2.7). Also, class *Document* is a base class for the individual document types, i.e. only information common to all document types is part of class *Document*.

Data Item	Comments
DocumentType	Used by the S/IM to determine the corresponding document manager.
Name	Name of the document; This field is treated as a *primary key* by the KADBM and S/IM. Names of documents need not be unique; the field should be set to an expression that most accurately describes the contents of the document.
Keywords	Additional keywords used by the KADBM to perform key queries (together with the *Name* field.
Path	A file name that tells the document manager where to find the document. Depending on document type, suffixes may be used to specify portions of a file (e.g. file pointer offset).
Title	A short title of the document (1 line) that may be used by the S/IM to give the user an overview of documents found as a result of a database query, etc.
Rights	Legal information (e.g. copyright, if applicable).

Table 2.8: Keys and Attributes of Class *Document*

Table 2.8 shows attributes common to all document type. Only those and the attributes of table 2.7 are stored in the KAD and can be used by the S/IM to perform system-wide searches for document, with the exception of full-text searches that are handled by the CDBMs (section 2.2.5.3, page 57). Figure 2.12 lists an example entry of the document database of the KAD, describing an entry of an encyclopedia on the city of Graz. Observe the following:

- For the sake of simplicity, and according to the UNIX philosophy, Hyper-G databases will be stored as simple text files. However, it may turn out in a later stage of implementation that a change in the format is necessary (e.g. due to speed or storage requirements). Fortunately, changes in the database formats are reflected only in changes in the corresponding database managers and do not influence the rest of the system.

- Attributes are stored as "<attributename>=<attributevalue>" pairs, allowing a variable number of attributes for each data set. To illustrate this, in figure 2.12 attribute *TimeModified* is missing (we assume the object stores has never been modified).

```
ObjectID=0x30127880
Type=document
Author=hmuelner
Group=iicm
TimeCreated=91/01/10 09:30:01
TimeAccess=91/02/25 22:01:17
Accessed=215
Language=German
BitsAuthor=RWTFHS
BitsGroup=RTFHS
BitsWorld=RS
PriceGroup=TF:0.1,HS:1
PriceWorld=S:2
DocumentType=text
Name=Graz
Keywords=Styria,Austria,City
Path=/usr/iicm/Hyper_g/data/encyclopedias/meyer:0x00020450
Title=Graz (Meyer's Encyclopedia)
Rights=Meyer Verlag
```

Figure 2.12: An Example Document Object

- In figure 2.12, the attribute *BitsAuthor:RWTFHS* specifies that the author has the right to read, write, link to, link from, hardcopy and softcopy the document, respectively. Again, the clear-text format allows for easy extension of the set of rights.
- The attribute *PriceGroup=TF:0.1,HS:1* specify that members of user group *iicm* pay 0.1 units for linking to and from the object (e.g. by incorporating it into a tour), and 1 unit for hard- and softcopying. Other users pay 2 units for softcopying. Of course, it makes no sense to specify prices for operations that are not allowed by the corresponding rights attribute.

In addition documents contain optional collection-specific data (additional attributes) that is used by the CDMS to perform collection-specific form queries (**Req. U.3.4.2**) and data specific to the document type that is interpreted by the document manager. These items are stored in the collection databases (figure 2.10). Note that storage format of a specific document type needs to be known only by the corresponding document manager and is of no significance to the rest of the system. Therefore, the format may be changed and/or extended without too much effort. Just to give the reader an idea, table 2.9 suggests some file formats for the individual document types.

Document Type	File Format
Text	7-bit format described in [132].
Drawing	Picture Interchange Coding (PIC) as described in [50, 85], and generated by EDEN-based graphics editors [49].
Image	PIC raster profile ('RAST') [134, 89]. Perhaps in a later stage JPEG format [25, 26, 179].
Sound	Audio Interchange File Format (AIFF) [7], based on the IFF format [28] used by Commodore and Apple computers.
Digital Video	Extended PIC RAST format. Perhaps in the future MPEG [96] format or DVI [100].
Animation	May be implemented as digital video (see above) or real-time animation (see chapter 3). For the latter the PHIGS [6, 75] Archive file format is an obvious choice.
Map	May be stored either like drawings or images (see above).

Table 2.9: Suggested Document File Formats

2.2.6.1.3 Anchors

Anchor-related data is split into attributes, which are stored in the KAD, and additional data (e.g. positional information) stored together with the document the anchor is tied to. No attributes beyond the generic Hyper-G Object attributes of table 2.7 (page 61) are required to be stored in the KAD. The data that gets stored together with the document and is to be interpreted by the corresponding document manager, however, strongly depends on the type of document the anchor is tied to. Table 2.10 gives an overview of the data required to define an anchor depending on the underlying document type.

In addition, every document has a default anchor associated with it (**Req. G.2.3**). The default anchor covers the whole document and can be used as both a source and destination anchor. It is used to link whole documents (e.g. with cluster links).

2.2.6.1.4 Links

Other than documents and anchors, links are handled entirely by the S/IM and KADBM. Table 2.11 lists the link attributes that have to be stored in the KAD in addition to the generic Hyper-G object's attributes (table 2.7 on page 61).

Document Type	Anchor Data	Comments
Text	2 indices	The indices indicate the positions of the first and last character of the text to be marked as (clickable) anchor (e.g. a word, a paragraph...).
Drawing	Segment ID	The name of a segment within the PIC file [85, 50]. Clicking on this segment will activate the anchor, if used as a source anchor. The picture will be scaled and/or scrolled so that the segment is visible, if used as a target anchor.
Image	Rectangle	A rectangle (2 coordinate pairs) defines the anchor. Source and destination behavior like drawings.
Sound	2 Markers	Markers can be stored and named using AIFF Marker Chunks [7]. When used as destination anchor, the section defined by the begin and end markers is played. When used as a source anchor, the anchor is activated when playback reaches the begin marker, which can be use to synchronize playback of sound to display of documents.
Digital Video	n Rectangles	Similar to images, rectangles describe the position of anchors in digital videos. However, a rectangle per frame may be defined, which allows the rectangle to move during playback. Also, beginning and ending time marks limit the area of an anchor.
Animation	Structure ID	Depending on the implementation (table 2.9), anchors are stored like with digital video, or as PHIGS *structure IDs* [6, 75].
Map	like drawings or images	
Dialog	Function	The name of a function within the *Dialog Manager* that is to be called when the anchor is activated (*active anchor*; see section 2.2.3.7 on page 24).

Table 2.10: Anchor Data Depending on Document Type

The interaction of links, anchors and documents is described in section 2.2.6.2.1 (page 70).

Data Item	Comments
Label	Sort of *link name* that serves a number of purposes: It may be displayed to the user when a selection of links is to be done, it is used to follow tours (section 2.2.6.1.5, page 67), and may be used to implement semantic (typed) links, e.g. so that all links referring to history are labeled "history", etc.
LinkType	One of C (cluster), U (user defined) or A (automatic) (see section 2.2.3.6 (page 24) and figure 2.8 (page 38).
TargetSize	One of S (small), M (medium) or L (large). Used (like *Language*) to select between equivalent documents in a document cluster (e.g. a long text versus a short text on the same subject), based on the user's preferences.
UseCount	The number of times this link was followed since *TimeCreated*. Used in conjunction with *LinkType* and *Author* to calculate link priority (oftenly used links are granted a slight increase in priority).
UndoCount	The number of times the user followed the link and stepped back immediately. May be used in conjunction with *UseCount* by the system administration to find incorrect and/or outdated links (**Req. G.3.1**).

Table 2.11: Attributes of Class *Link*

2.2.6.1.5 Tours

As shown in figure 2.11 (page 62), the *Tour* class is derived from the *Document Class*. By doing so, tours inherit names, keywords, titles and copyright info from documents, and the S/IM is able to select tours and documents in response to key queries. However, there are substantial differences in the execution of tours and documents:

- Tours are implemented using labeled links and scripts (implementation details are discussed below) . No attributes or data other than the attributes listed in tables 2.7, 2.8 and 2.11 are needed.
- A *DocumentType* of "tour" indicates that the object is a tour rather than an ordinary document. In this case the *Path* attribute points to the script.
- A document manager called *TourGuide* interprets the tour document script (section 2.3.5.10, page 96). The script language used depends on the *TourGuide*. Therefore, Hyper-G is not restricted to a single script language, as the document manager for a document (and therefore a tour) is chosen dynamically (section 2.2.5.1, page 53).

- The *TourGuide* provides a comfortable user interface that supports the user in tour navigation and execution. Within a tour, the following functions are available:
  - Any tour has two special document clusters: The *start cluster* and the *index cluster*. At any time, the user may "jump" to one of these clusters. The start cluster is supposed to be the entry point of the tour, while the index cluster is supposed to contain a graphical representation of the tour, with links to every frame of the tour. If the tour is too complex to fit on a single index cluster, sub-indexes may be attached.
  - A "next cluster" is associated with any cluster. By activating the function "go to next cluster" (e.g. by pressing a special button), the user may proceed to the next cluster.
  - Similarly, the user may step backwards to the previous cluster. However, unlike the "next" cluster, which is defined by the author of the tour, the function "go to previous cluster" is dynamic, i.e. a sort of "undo" function.
  - The user may also follow other links that are provided by the author, labeled as being part of the tour, and tied to individual documents. Thus, a tour is not necessarily a linear list of document clusters, but may also incorporate branches.
  - The anchors of such "branch links" are highlighted in such a way (e.g. by the use of colors) that the user receives a feedback on whether the destination cluster of the link has already been visited. This is especially useful with index clusters, as the user gets graphical feedback on what parts of the tour have not been seen yet.
  - When a document cluster is executed that is also part of another tour, the user is given the opportunity to change tours. At a later time, the original tour may be continued.
  - In addition, the user may select tour-specific help documents (if provided by the author), information on the author, copyright information, visit a cluster not yet visited, etc.

As has been stated above, simple tours may be implemented using links. This is done in the following way (see also figure 2.13):

- Tours are given a unique name (attribute *Name* in table 2.8). Because the author can not be expected to know the names of all tours in the system, the *TourGuide* also uses the *Author* attribute to generate a tour name. This way, only the tours of a single author are required to have a unique name.
- Links that store this tour name in their *Label* attribute (table 2.11) are considered part of this tour, i.e. the tour is made up by links labeled with the tour name. Only links with this label are followed within a tour.

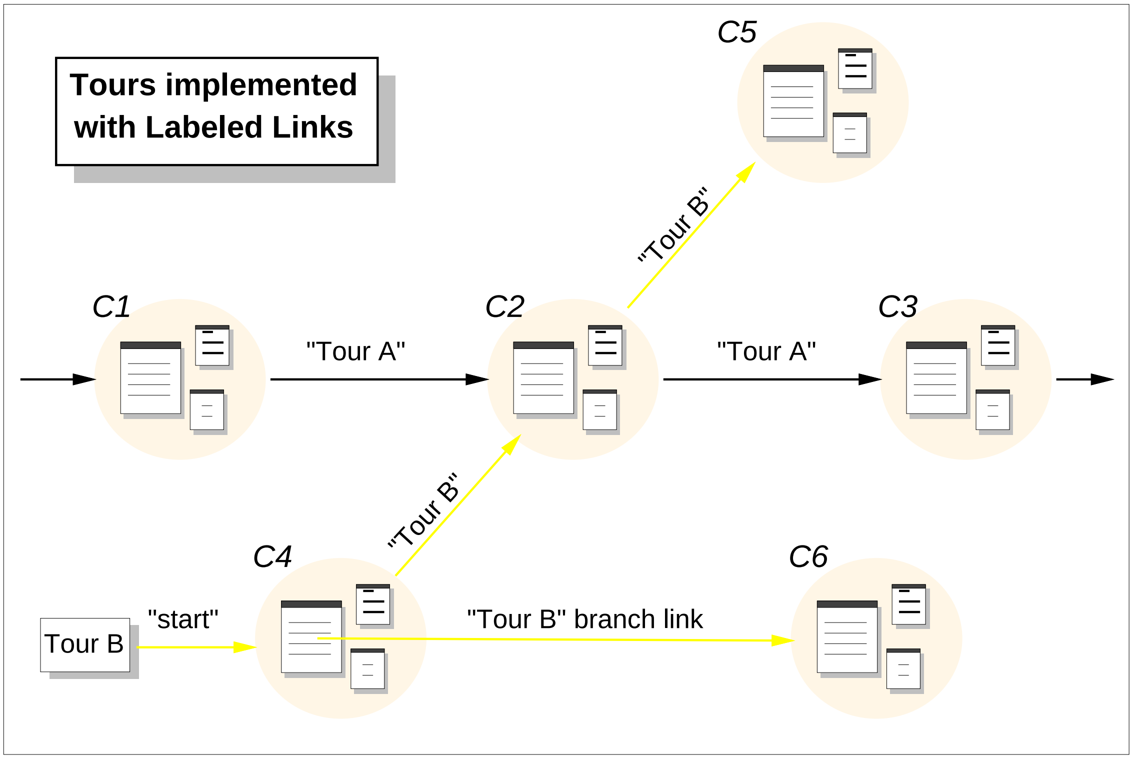

Figure 2.13: Tours Implemented with Labeled Links

- A link that references the default anchor of a document (i.e. the whole document) as a source anchor is treated as the "next cluster" link (links $C1 \rightarrow C2$, $C2 \rightarrow C3$, $C4 \rightarrow C2$ and $C2 \rightarrow C5$ in figure 2.13). Links tied to other source anchors are the "branch links" described above (link $C4 \rightarrow C6$ in figure 2.13). Remember, only links that are labeled with the tour name are considered.

- There exist two special links from the tour object to two document clusters that are labeled "start" and "index". These links (that are stored in the link database like all other links) are used by the TourGuide to find the start and index clusters. Optional "help" links reference author-supplied, tour-specific help documents.

- When a document (cluster) has an additional link attached with another tour name in its label attribute (such as cluster $C2$ in figure 2.13), the user may switch to the new tour and follow the other link. He may also select functions "go to start/index cluster" to gain an overview of the new tour, or return to the old tour by selecting "go to previous cluster" (remember, this is a dynamic function, i.e. its meaning is generated at runtime).

The concept of labeled links allows authors to build tours easily (using the link editor). Also, it enables the user to "switch tours". However, certain advanced features call for the script approach:

- It may be useful to modify the tour dynamically (e.g. in educational tours), based on decisions of the student or TourGuide (e.g. to guess whether the student understood the previous section). These modifications can in general not be implemented with branch anchors (figure 2.5 on page 26), as they lack the memory that is necessary to remember states during a tour.
- A very simple script language allows to associate priorities with the labels of the labeled links. This feature can be used to incorporate large portions of other people's tours into one's own tour (see section 2.3.5.10 and figure 2.15 on page 97 for an example).

While in principle it is possible to implement tours with scripts only (given a powerful script language) this should be avoided, because the "tour switching" feature of Hyper-G relies on labeled links. Scripts should be used only where the same result can not be achieved with links. In these cases, a user may be required to start the tour at the beginning, whereas with the link approach, the tour may be followed from any point. For the sake of completeness, Hyper-G supports both concepts.

2.2.6.1.6 Collections

Like tours, collections are derived from the *Document* class, i.e. inherit names, keywords, titles and copyright info from documents. Also, the S/IM may return a whole collection in response to a key query. A *DocumentType* (table 2.8) of *collection* indicates that the object is a collection rather than an ordinary document.

In this case the *Path* attribute is used by the S/IM to determine a collection-specific database manager (CDBM) that is responsible for that collection (section 2.2.6.2.2, page 73), i.e. may be used to perform collection-wide key queries if the collection is in the user's list of active databases (by methods found in table 2.6 on page 59).

Usually, one or more links attached to the collection object and labeled "query" point to dialog objects that may be used to perform collection-specific form queries (**Req. U.3.4.2**), including menus. Additional referential links point to sub-collections. If a collection is in the user's list of active databases, the sub-collections are searched also.

2.2.6.2 Databases

2.2.6.2.1 The Keys & Attributes Database (KAD)

In general, the KAD may be implemented by an ordinary relational database. Although it would be possible to store all Hyper-G objects (documents, links, anchors etc.) in one database, because of performance considerations partitioning the KAD into a number of smaller databases and relations is preferred.

So, the KAD will in fact be split into the following databases containing Hyper-G objects:

1. Documents (and tours) will be stored in a number of databases (one per collection) of *Document* objects (section 2.2.6.1.2, page 62), in order to be able to restrict searches to collections in the user's list of active databases.
2. The *Collection* objects themselves get stored in an extra database.
3. Anchors (section 2.2.6.1.3, page 65) are contained in an anchor database.
4. Links (section 2.2.6.1.4, page 65) will be stored in one of two databases:
   (a) Cluster links are contained in an extra database to speed up common searches for cluster links only.
   (b) Referential links (user-defined and automatic) are held in another database.

As has been pointed out earlier (section 2.2.6.1, page 60), Hyper-G objects will be stored as simple (editable) text files (for an example see figure 2.12 on page 64). A number of indices and/or search trees may be used internally to increase performance of common searches, e.g. to find Hyper-G objects by *ObjectID* or to find links by source anchor ID.

In addition to the Hyper-G objects databases, the KAD uses the following relations to link objects[10]:

1. The **link-anchor relation** stores the data shown in table 2.12 in each record. Its main use is to find the destination anchor and link data once the source anchor is known[11], i.e. the link-anchor relation links two anchors.
2. The **anchor-document relation** links anchors to documents (table 2.13). Its main use is to find a document once the anchor ID is known.
3. The **document-collection relation** links documents to collections (table 2.14). Its main purpose is to find the collection(s) a certain document is a member of. It may also be used to find all the documents of a collection.

[10]A relational database model has been preferred to a network database model, mostly because of the flexibility and simplicity of the relational concept. Note that no Hyper-G object contains references to other Hyper-G objects (as in a network database), e.g. documents do not contain a list of attached links, as with this approach it would not be possible to backward-follow links (i.e. find all documents linking to a given document), unless information is duplicated.

[11]And vice versa, in the case of backward link following.

Data Item	Comments
LinkID	ObjectID of the corresponding *Link* Object. Used to find additional information (table 2.11) on the link (label, type, author, language, size etc.).
SourceID	ObjectID of the source anchor of the link. Used to find corresponding document with anchor-document relation and additional information on the anchor.
DestID	ObjectID of the destination anchor of the link (see above).
TextID	This optional *DocumentID* points to a document with a description of the link's purpose (preferably a small text document; but may be any type). Some user interfaces may display this information before actually linking to the target document.

Table 2.12: The Link-Anchor Relation

Data Item	Comments
AnchorID	ObjectID of the anchor. If the *AnchorID* equals the *DocumentID*, it is a default anchor.
DocumentID	ObjectID of the document.

Table 2.13: The Anchor-Document Relation

Let us once again illustrate the use of the two relations in the case of forward following a static link, which is the main application of these relations (figure 2.14):

1. When the user activates (e.g. clicks on) a source anchor of a document, the document manager informs the S/IM (i.e. sends the anchor ID). The link-anchor relation is searched in order to find links that originate at this anchor.

2. Additional link attributes and the link description may be used to select a link to follow.

3. The destination anchor is looked up in the anchor-document relation in order to find the destination document.

Data Item	Comments
DocumentID	ObjectID of the document.
CollectionID	ObjectID of the collection. The document is a member of this collection.

Table 2.14: The Document-Collection Relation

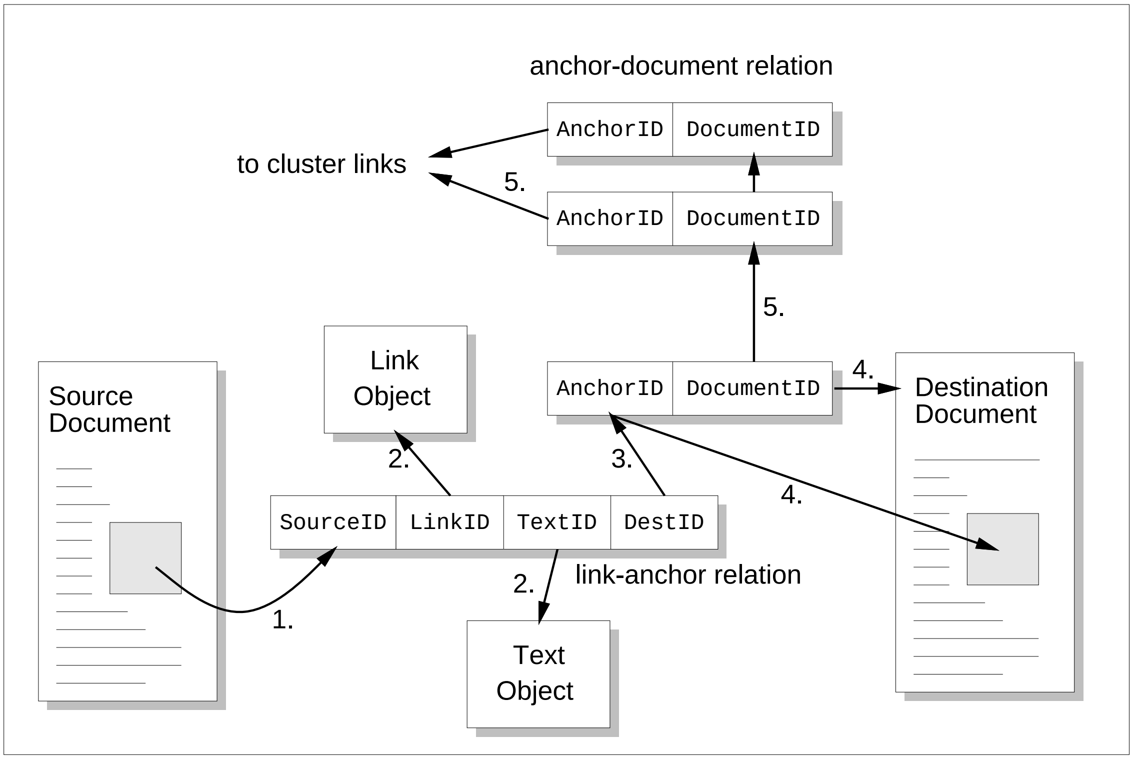

Figure 2.14: Using the KAD for Forward Link Following

4. The destination document is displayed so that the destination anchor is visible.

5. The *DocumentID* (which is also the default anchor ID) may be used to find and select between cluster links originating or ending at the document, in order to find the other documents of the destination document cluster.

For the time being, relations will be stored as simple text files, like Hyper-G objects. Special indices and trees are used to reduce access time.

2.2.6.2.2 Collection-Specific Databases

As stated in **Req. U.3.2**, Hyper-G's information database is structured in a collections. Every document is a member of at least one collection (**Req. U.3.2.2**). The hierarchy is an acyclic directed graph with a number of 'root' collections (**Req. U.3.2.1**). This allows a number of classification schemes for Hyper-G's collections, e.g.:

- According to the *area of knowledge*, e.g. as shown in figure 2.7 on page 29. In this case, the collection-specific database would contain additional attributes and search criteria that are only significant in this field. For example, the database of collection 'paintings' would store attributes like *painter, name of picture, year, style, theme,* and *museum* (section 2.2.3.13, page 30).

- According to the *document type* of the object. E.g., all raster images may be grouped in one collection, allowing database queries specific to this data type, e.g. to find images by choosing between small versions (icons) of the images. In this example, the collection-specific database would contain information related to the physical representation of the images, such as *size, number of colors* and *file format*.

- Other *"Stand-alone collections"* may be used to allow for form queries of information that does not find into above classification schemes.

Note that because a document may be a member of a number of collections, and because a number of 'root' collections (i.e. a number of collection hierarchies) are allowed, Hyper-G can support all the classifications schemes described above simultaneously. The user has the choice.

Also, the attributes and layout of the collection-specific databases need not be known by the S/IM or the document managers. Only the collection-specific database managers (CDBMs, figure 2.10 on page 53) and collection-specific dialog documents that perform form queries need to know about these attributes.

2.2.6.2.3 Remote Databases

As specified by **Req. I.2**, Hyper-G supports access to remote databases (section 2.2.3.1.3, page 18). However, in order to confront the user with a consistent user interface, access to such a database shall be hidden from the user, i.e. the remote database looks like an integral part of Hyper-G.

This can be achieved most elegantly by presenting the remote database to the user as an ordinary Hyper-G collection. The dialog document usually attached to collections handles the user's form queries and passes them on to a CDBM, as in the case of a standard collection. This CDBM, however, communicates with the remote database, i.e. translates the queries into a form the remote database understands, retrieves the information, and passes it to a document manager that displays the information. Either the information of the database can be easily translated to an existing document type (e.g. text), or a special document manager is needed.

This way the remote database is fully integrated within the Hyper-G environment, i.e. the user would not notice any difference to other collections[12]. Also, the col-

[12] Of course, the time to retrieve and transfer the information could be significantly longer than with local collections (databases).

lection can be searched using key queries by implementing the CDBM methods of table 2.6 (page 59).

2.2.7 The User's View

As has been pointed out in previous sections, the user interface of Hyper-G will be configurable to a large extent. A number of user interface metaphors can be supported by Hyper-G (**Req. U.3.1.1** - **Req. U.3.1.6**). Also, the user interface devices used to communicate with the system will vary depending on the application and environment (see table 2.3 on page 40). Clearly, an in-depth specification of the various user interfaces is beyond the scope of this document. Also, user interfaces metaphors will be tested, evaluated and modified on the basis of a first version of Hyper-G. During this evaluation process, it may very likely turn out that some metaphors are not suited for use with large systems like Hyper-G, as suspected in [86]. So, let us concentrate on some general features of Hyper-G's user interface(s) in this section.

2.2.7.1 Search Strategies

In general, the user looks for information. Hyper-G offers the user a number of possible strategies to find that information:

- If users know exactly what they are looking for (e.g. "I want information on UNIX file systems"), the fastest method is to issue a key query to search for keywords. Key queries are available in two flavors:
  - The fast version generates dynamic links to documents containing a combination (*and, or, not...*) of these keywords in their *Keywords* attribute (table 2.8 on page 63). The user may specify additional attributes (e.g. document type, author) in a query-by-example manner to reduce the number of links generated.
  - If too few documents are found with above method, the user may consider to use the slower version that actually performs full-text searches (e.g. "what documents contain the word UNIX?") on the documents. Here the users may specify a list of databases (collections) they are interested in, in order to reduce the time required and number of links generated.
- Alternatively, the user may explore the collection hierarchies. This strategy is best if you look for things like "some pretty pictures from painters" etc., and you just can't think of an appropriate keyword. Once a promising collection is found, the user may use form queries to reduce the number of possible documents.

  In above example, the user might find a collection *paintings*, and fill in a form containing *painter, name of picture, year, style, theme, museum* and the like, again in a query-by-example manner.

- If accurate information retrieval is not the primary goal, but users want to explore the system in a browsing manner, they might go on a guided tour, e.g. "my favorite paintings" by author *xyz*.

  An important feature of Hyper-G's tour concept is the ability to switch tours whenever a document cluster is reached that is part of another tour. Thus if another author has prepared a similar tour, it is likely that the tours share a few document clusters (unless the sets of favorite paintings of the two authors are completely disjunct), and the user may visit the union of the two tours.

  Also, the navigation aids available within a tour are quite comfortable (section 2.2.6.1.5, page 67).

- The ability of any user to (potentially) create private links – a sort of annotation facility; remember, there is no explicit annotation (section 2.2.3.8, page 27) – allows to create a private list (or tour) of document clusters that are considered interesting but were difficult to find. These document clusters can then easily be found in future sessions.

- With all of above strategies, once the users find an interesting document, they may follow the links attached to the documents and freely wander around the Hypermedia system. However, in this mode the navigation facilities are limited and restricted to rather simple functions like *undo, redo, drop/goto book-mark, show document info, show link info*, etc.

Note that a particular user interface metaphor does not need to support all of above strategies, but may offer a subset of them. Also, they may be hidden from the user and combined by the metaphor to offer the user a different method of information retrieval.

2.2.7.2 User Configurable Options

The user interface may be fine-tuned on a per-terminal and per-user basis by the terminal profile and user profile, respectively. A flag in the user profile specifies what parts of the profile may be changed by the users themselves (**Req. U.4.1.8**).

Most notably, the user may (potentially) change:

- The user's system entry point, which may be any Hyper-G object. However, to define a root-level collection or a dialog document containing some menu or form would be reasonable.

- The user's language preference (explained in section 2.2.3.14 starting page 30).

- The preferred user interface metaphor.

- Depending on the user interface metaphor, additional parameters may be configured, e.g.:
  - Preference of short, medium or long versions of semantically equivalent text documents.
  - Preference of digitized voice documents or semantically equivalent text documents.
  - A list of preferred document managers that is used by the S/IM to generate the *Document Manager Table (DMT)* (section 2.2.5.1, page 53).
  - Parameters describing the visual appearance of the user interface (e.g. window positions and sizes, colors, window manager behavior etc.). Configurations of these parameters is most easily managed using the *X Resource Manager* concept of the X window system.

2.2.8 The Author's View

As has been pointed out previously, any Hyper-G user is a potential author, i.e. is allowed to insert information into the database. Depending on the kind of information (documents, links, tours), a number of steps have to be performed by the author preparing that information. Each step is made using a specialized editing tool.

2.2.8.1 Document Editing

Documents are prepared using a number of document editing tools, depending on the document type. The philosophy of Hyper-G is not to supply a tool for every document type. Rather, documents are prepared outside the Hyper-G environment, using standard software where possible. This approach offers the following advantages:

- We do not have to write lots of editing software ourselves, saving manpower for the implementation of the Hyper-G runtime system and other utilities.
- There is a number of good commercial software packages available, each one suited for the generation of a certain document type. Table 2.15 lists packages that are used at our institute to prepare documents[13].
- Authors can incorporate their own documents (e.g., raster images in about any possible format, from whatever source) instead of having to generate documents on-line, with a single editor.

Document Type	Software Packages used
Text	Any text editor will do. Currently, we use GNU EMACS [165] on UNIX workstations, because of its availability on a large number of platforms. On PC's a large number of good text editors is available. Because of embedded control sequences for text formatting, text has to be converted to the format described in [132] to be incorporated into the Hyper-G database.
Drawings	EDEN-based graphics editors [49] are available on both workstations [4] and PC's [11, 84]. Because of the ability to convert CGM's [74] and GEM metafiles [38, 37, 153] to PIC format [50, 85], we can also use the PC-based editors *CorelDraw!*[a] and *Designer*[b] and other software capable of producing CGMs or GEM metafiles.
Images	Images are stored as PIC file that contain pixel arrays. Images can be converted from a variety of formats, including PPM [150], TIFF [34] and GIF [29].
Sound	Sound is currently digitized on a MacIntosh[c] equipped with the *AudioMedia* board and software.
Animation	For time being, we use the *Wavefront Advanced Visualizer*[d] running on a *Silicon Graphics Personal Iris 4D/35* [164] to generate the individual frames of a computer animation clip. The frames are then converted to digital videos for playback.
Digital Video	At this time the creation of digital videos other than computer animations is a tedious process using a single-frame digitizer (we currenlty use a product called *Screen Machine* for this purpose). In the future, we will probably use frame compression hardware (e.g. the *NeXTDimension* board [8],[17, page 159]) for real-time digitizing of analog video.

[a] ©Corel Systems Corp.

[b] ©Micrografx. Inc.

[c] ©Apple Computer Inc.

[d] ©Wavefront Technologies

Table 2.15: Software Used to Generate Documents

Of course, there are also some disadvantages:

- We need to write a number of converters to convert other people's document formats to Hyper-G's format. Alternatively, a document manager may be able

[13] Fortunately, software gets better all the time, so this list is subject to change.

to interpret a number of document formats. However, in most cases, writing such a converter is far easier than writing a good editing package.

- We need to know about commercial software packages that are commonly used and periodically update table 2.15.

After the document is prepared, in can be entered into the database. At this stage, the attributes of the document are determined. Some of them (e.g. author, creation date) are automatically assigned, other (e.g. keywords, title) have to be supplied by the author. Also, the author has to decide in what collections the document should be inserted, and supply additional collection-specific attributes. After this has been done, the document is accessible via database queries, but is not part of a document cluster (strictly speaking, it is a document cluster containing only one document), nor can it be reached via links or tours.

2.2.8.2 Link Editing

In order to group documents in document clusters, or to reference it from other documents and use it in tours, links have to be added to the document. These links may be generated automatically by the system (static automatic links), however, this approach works only for some document types (e.g. text) and the link priority is low (section 2.2.3.6, page 24). Therefore, higher-priority links may be generated "manually", following these steps:

1. **Anchor Placement** requires a special tool for each document type, as the appearance of anchors varies (see table 2.10 on page 66). If the source or destination of the link is the whole document, this step is skipped (e.g. the case with cluster links that group documents to document clusters).

2. **Link Definition** involves the specification of link attributes as well as the source and target anchors. This is done using the *link tool* that allows to see the two documents while defining the link and to test the link.

3. **Tour Generation** is done using the *tour editor*. Basically, links are defined with the link tool, but additional capabilities allow convenient editing of the tour. Most notably, a graphical representation of the tour is created that is then also incorporated into the tour as the *index cluster* (section 2.2.6.1.5, page 67). Also, the author needs to specify a *start cluster*, a *next cluster* for any cluster, and optional tour-specific help clusters.

2.2.9 The System Administrator's View

In addition to the tools described in section 2.2.8, the system administration is equipped with tools that allow:

- The bulk update of documents and whole collections (e.g. when adding a new encyclopedia).
- The automatic generation of static links [133].
- Checking the database to find and remove bad links.
- The deletion of links and documents (including the removal of referencing links, anchors, etc.).
- The automatic maintenance of objects (**Req. G.3.1**) is needed to get rid of obsolete or outdated information. Several methods are possible:
  - Use of the expiration date associated with each object (**Req. G.3.1.1**). A typical expiration time of a user-supplied document might be six months. Two months before expiration, the user (i.e. the author of the document) is notified (by means of e-mail) and may extend the lifetime of the object. If the case of response, the object may be deleted (and with it all associated links).
  - Use of the *time of last access* and *number of accesses* (table 2.7 on page 61) may be used to gather statistics of the object's usage and relevance. Obviously, it would be a bad idea to delete objects that are used frequently.
  - The *Undo Count* associated with each link records the number of times the user followed the link and stepped back immediately. If this happens very often (in relation to the link's use count), this indicates an incorrect or badly placed link. The system administration would then consider to remove the link.
  - The user logs also may be used to find weaknesses of the database **Req. U.5.1.2**). For example, unsuccessful database queries indicate missing information (especially if a number of users cannot find the same things). Also, the *undo* and *help* actions of users point to difficulties using the system.

2.3 Implementing a Prototype of Hyper-G

2.3.1 Purpose

In order to evaluate the ideas and specification presented in the previous sections, a prototype of Hyper-G is being developed. The aim of this project, which will be finished by end of 1992, is to have a "demonstratable" version of Hyper-G running on both PC's and UNIX-Workstations, with the following goals:

1. The UNIX version will serve as a base on which the "real" Hyper-G system can be built on, i.e. it should be possible to create Hyper-G by incremental changes to the prototype.
2. The PC version will be used to demonstrate some of the features of Hyper-G. Because it runs on standard PC's we will be able to send demo diskettes with small Hypermedia applications (e.g. a single tour) to interested organizations or persons.
3. In addition the prototype can be used to create some small "real" Hypermedia applications as spin-offs. Because of the widespread availability of PC's, the PC version will be adapted to publish Hypermedia material on diskettes (for small applications) or CD-ROM's (for larger applications). More on this electronic publishing aspect of Hyper-G can be found in section 4.2 (page 126).
4. By-products like HOTACT (section 4.3.1.3, page 130) and other demonstrations of unorthodox user interfaces may be used in a "Viewseum" (section 4.3, page 128).
5. From a more scientific point of view, the possibility to evaluate and modify certain features and options of Hyper-G (e.g. user interface, navigation aids, database design) at an early stage in the system development (*rapid prototyping*) is tempting and will contribute to the success of the real system.

However, the implementation of such a prototype is not sufficient. In order to test the ability to handle large amounts of data and to create real-world demos and applications, large amounts of real-world data has to be prepared and incorporated into the prototype system. The backbone of the prototype system will be formed by a number of general-purpose and specialized encyclopedias [132, 133], some 10,000 raster images (for a short discussion of the problems associated with the preparation of images see section 3.3.1 on page 109), and a few sound and digital video clips.

2.3.2 Priority of Requirements

In order to derive a reasonable subset of requirements for the Hyper-G prototype, tables 2.16, 2.17, 2.18, 2.19, 2.20, 2.21, and 2.22 list the requirements of sections 2.2.2 and 2.2.4, with associated priorities. As the system of priorities is organized in an hierarchical way, only the highest level with a unique priority is listed (e.g. the low priority of Requirement **Req. G.3.1** implies the low priority of **Req. G.3.1.1**, so it is not listed. Also, **Req. G.3** needs not to be listed, as all of its implied requirements are contained in table 2.17). Every requirement has one of three priorities associated with it:

H: Requirements of this priority are absolute musts. They are implemented in a very early stage named HBS (section 2.3.3, page 87) of prototype development.

M: These requirements are classified as medium-priority. They are implemented in the prototype, at least in a simplified way.

L: These are low-priority requirements that will be implemented in the "real" Hyper-G, but not in the prototype.

A few explanations of the choice of priorities seem to be necessary:

- The prototype should implement the Hyper-G Core System (section 2.2.3.1.1, page 17) as shown in figure 2.1, i.e. a system of servers and workstations in a reasonable-speed local area network. Access from remote terminals (section 2.2.3.1.2, page 18) and to remote databases (section 2.2.3.1.3, page 18) is postponed to the implementation of the "real" Hyper-G, except for demonstration of access to "in-house remote" databases.
- Only one user interface metaphor ("Desk-top") is available in the prototype. This metaphor is best used with standard mouse and keyboard user interface devices, so these are the only input devices supported, except for some special applications (section 4.3, page 128).
- As the prototype is not going to be used by the general public, no provisions concerning access rights have been made. Also, no truly anonymous use of the system is possible.
- There is no charging for specific functions in the prototype.
- Multilingual Hypermedia (with use of user-specific language preference) is already implemented in HBS and will be available with the Hyper-G prototype.
- The same is true for the *Tour* concept.

- Key queries will be implemented in the prototype; complex form queries and dialog documents for collection-specific database queries will not.
- No user logs will be maintained. As a consequence, the *Undo* function is available only within a session, and automatic maintenance of the database is limited.

Requirement		Priority		
ID	Short Description	H	M	L
G.1.1.1	Document Type: Text	H		
G.1.1.2	Document Type: Drawing		M	
G.1.1.3	Document Type: Raster Image	H		
G.1.1.4	Document Type: Digitized Sound		M	
G.1.1.5	Document Type: Digital Movie		M	
G.1.1.6	Document Type: Animation		M	
G.1.1.7	Document Type: Maps		M	
G.1.1.8	Document Type: Electronic Mail, Bulletin Board, Computer Conferencing		M	
G.1.1.9	Document Type: Dialog[a]			L
G.1.2	Document Clusters		M	
G.1.3.1	Object Attribute: Author and Group		M	
G.1.3.2	Object Attribute: Creation/Modification/Access/Expiration time			L
G.1.3.3	Object Attribute: Language	H		
G.1.3.4	Object Attribute: Access Rights			L
G.1.3.5	Object Attribute: Price			L
G.2.1	User may activate link by activating anchor.	H		
G.2.1.1	Anchors belong to links rather than documents		M	
G.2.1.2	Visual appearance depends on document type and already visited		M	
G.2.2.1	Forward Following of links & Anchor Positioning	H		
G.2.2.2	Backward Following of links & Anchor Positioning		M	
G.2.3	Default Anchor		M	
G.2.4	Cluster Links[b] and Referential Links	H		
G.2.4.1	Referential Links: User Defined and Automatic		M	
G.2.4.1.1	Automatic Links: Static or Dynamic		M	
G.2.5	Link Priority		M	
G.2.6	Active Anchors			L

[a] This applies to complex dialog objects. Simple ones like menus are implemented in the prototype.

[b] Cluster links are implicitly defined between text and image documents in HBS.

Table 2.16: Priority of General Requirements (Part 1)

Requirement ID	Short Description	Priority H	M	L
G.3.1	Documents and links are maintained automatically			L
G.3.2	The majority of the links is generated automatically		M	
G.3.2.1	Automatic generation of static links for new document			L
G.4	The system should be easily extensible		M	

Table 2.17: Priority of General Requirements (Part 2)

Requirement ID	Short Description	Priority H	M	L
U.1.1	Application-specific user interfaces (table 2.3 on page 40)			L
U.2.1	User may specify language preferences	H		
U.2.2	System selects documents by language	H		
U.2.3	Multilingual user interface[a]	H		
U.2.4	Temporarily override language strategy			L
U.3.1.1	User Interface Metaphor: Simple Hypermedia (HBS)	H		
U.3.1.2	User Interface Metaphor: Desk-Top		M	
U.3.1.3	User Interface Metaphor: Stack			L
U.3.1.4	User Interface Metaphor: Book			L
U.3.1.5	User Interface Metaphor: Holiday Travel			L
U.3.1.6	User Interface Metaphor: Library			L
U.3.1.7	Simple Navigation Facilities		M	
U.3.2.1	Collection Hierarchy		M	
U.3.2.2	Advanced navigation in the collection hierarchy			L
U.3.3	Guided Tours	H		
U.3.3.1	Tour switching		M	
U.3.3.2	Advanced tour navigation functions	H		
U.3.4.1	Key Queries		M	
U.3.4.1.1	List of active databases (collections)		M	
U.3.4.1.2	Inexact match key queries			L
U.3.4.2	Form Queries			L

[a]Implemented in the PC Version of HBS by use of pictorial icons.

Table 2.18: Priority of User-Related Requirements (Part 1)

Requirement ID	Short Description	Priority H	M	L
U.4.1	The system maintains a user profile, containing:			
U.4.1.1	Security Information			L
U.4.1.2	The user's system entry point		M	
U.4.1.3	The user's language preferences	H		
U.4.1.4	Preferred user interface metaphor[a]			L
U.4.1.5	Preferred user interface metaphor's options[b]		M	
U.4.1.6	User logging (full/system/off)			L
U.4.1.7	Default active databases for queries		M	
U.4.1.8	Whether user may change profile			L
U.5	User may return to previous state[b]	H		
U.5.1	The system maintains user logs			L
U.6	The system may be used anonymously[c]			L
U.7	Some system features and information items are chargeable			L

[a]Only one user interface metaphor ("Desk-Top") is available in the prototype. However, options of this metaphor are configurable.

[b]Supported only within a session (no user logs needed).

[c]Not applicable in the prototype (Core System only).

Table 2.19: Priority of User-Related Requirements (Part 2)

Requirement ID	Short Description	Priority H	M	L
A.1	Any user is a potential author[a]		M	
A.2	Editing tools for authors (section 2.2.8, page 78)		M	

[a]In the prototype, only text documents (annotations) and links may be created by the casual user.

Table 2.20: Priority of Author-Related Requirements

2.3.3 The Hypermedia Base System (HBS)

The *Hypermedia Base System (HBS)* was implemented as an attempt to demonstrate some ideas of Hyper-G, and to create tools, libraries and platforms on which the Hyper-G prototype can be built. This section contains a brief overview of the

Requirement ID	Short Description	Priority H	M	L
S.1.1	For each terminal, there is a terminal profile, describing:		M	
S.1.1.1	Terminal facilities, such as:		M	
S.1.1.1.1	Input Devices		M	
S.1.1.1.2	Voice I/O capability			L
S.1.1.1.3	Digital Sound I/O capability		M	
S.1.1.1.4	Digital Video I/O capability		M	
S.1.1.1.5	Hardcopy capability			L
S.1.1.1.6	Softcopy (diskette) capability			L
S.1.1.1.7	Communication capabilities (E-Mail, conferencing)		M	
S.1.1.1.8	Charging of anonymous users			L
S.1.1.2	Security information			L
S.1.1.3	The prices of specific terminal facilities			L
S.2	Access rights control whether a certain user on a certain terminal may perform a certain function on a certain object.			L
S.2.2	A similar mechanism is used to determine the price for a certain function performed by a certain user on a certain terminal with a certain object.			L
S.3	The system avoids uncontrolled growth of information			L

Table 2.21: Priority of System Requirements

Requirement ID	Short Description	Priority H	M	L
I.1.1	Keys & attributes Database (KAD)		M	
I.1.2	Collection-specific Databases (CDBs)			L
I.1.3	The databases themselves ensure database integrity			L
I.2	Hyper-G allows access to remote databases[a]			L

[a] May be demonstrated for some cases by the prototype.

Table 2.22: Priority of Implementation Requirements

functionality of HBS, compared with the functionality of the Hyper-G prototype. For a complete description of HBS the reader is referred to [71].

HBS utilizes the so-called "Simple Hypermedia" Metaphor. The user may follow predefined tours, but because tours do not have to be organized as a linear succession of scenes, the user has the possibility to select from the offered information by activation of referential links. Despite the limited functionality, HBS may already be used to realize some real-world applications, such as electronic publishing of instructional or advertising material.

HBS is implemented in both a PC (MS-DOS) and Workstation (UNIX) version. Although the both operate on the same data, there are minor differences in functionality and major differences in the user interface. For example, the PC versions always shows full-screen raster images and a single text window, while the UNIX version imposes no restriction on the number of windows opened simultaneously and allows individual placement, resizing and iconification of windows by the user. However, we do not see these differences as a disadvantage; rather, it allows us to evaluate the different user interface metaphors and to demonstrate access to the same material via different interfaces.

Concepts of the Hyper-G prototype that are already extensively implemented in HBS are:

- The *Tour* concept. In fact, tours are the only way of navigation supported by HBS. The tour description is contained in a control file that specifies the *next* links, as well as referential links. All of the advanced navigation features (except tour switching) described in section 2.2.6.1.5 (page 67) are available.
- The idea of *Multilingual Hypermedia*. Text documents are chosen according to a list of language preferences supplied by the user. While the UNIX version also changes the labels of buttons and other user interface widgets, the PC version circumvents the problem by extensive use of pictorial icons.

Concepts that are in part found in HBS are:

- The *Document Clusters* are simplified to *Scenes*. Tours are made out of scenes; scenes contain exactly one raster image and a number of equivalent text documents that differ only in their language. HBS selects one of the text documents based on the user's languages preferences, i.e. the user sees one image and one text document at a time. Display of the text document may be switched off by the user.
- *Text* and *Raster Image* are the only document types supported. In the UNIX version, the *Raster Image Document Manager* allows to resize and scroll images. Text can be scrolled and is automatically reformatted and hyphenized.
- *Cluster links* are contained implicitly in the scene description. Only author-defined, static *referential links* are supported in HBS.

- Only forward following of links is possible (as there is no link database).
- *Anchors* are contained within text documents, but are specified together with the link for raster images.

Concepts that are missing in HBS, but will be implemented in the Hyper-G prototype are discussed in the next section.

2.3.4 From HBS to the Hyper-G Prototype

The Hyper-G prototype is created in an evolutionary process by incremental changes and extensions of HBS[14]. In particular, things that have to be changed or added are:

- The following additional document types are supported: Object-Oriented Drawing, Digitized Sound, Digital Movie, Animation, Map and Communication. Animation may be simulated by digital movies, maps will be implemented on top of raster or object-oriented images. Communication will be demonstrated by electronic mail and on-line communication between Hyper-G users.
- The *Scene* concept will be generalized to the *Document Cluster* concept. Such a cluster will contain an arbitrary set of documents of the document type listed above. Cluster links are used to configure document clusters.
- A strict separation of documents, links, and anchors will be enforced.
- All Hyper-G objects will be stored in the KAD. This serves to implement the following features:
  - A prefix key query for documents will be possible.
  - All link types and priorities (section 2.2.3.6, page 24) are supported, including static and dynamic automatically-generated links.
  - Forward and backward following of links is supported.
- The process structure of figure 2.10 is partially implemented, making the system very extensible.
- The user interface metaphor evolves from "simple Hypermedia" to "Desk-Top" (section 2.3.6, page 97), with simple navigation functions (e.g. Undo, Drop Bookmark) available throughout the system.

[14] An intermediate stage – called HBS-II – is currently being developed at the Institute for Multi-Media Information Systems of Joanneum Research, Graz, Austria.

- Information will be organized in a collection hierarchy. The user may specify a list of active collections for key queries.
- *Labeled links* are used to implement tours (section 2.2.6.1.5, page 67), so tour switching becomes possible. In addition, a simple script language that allows to incorporate sub-tours by associating priorities with labels, is implemented.
- The user will be able to extensively configure the user interface of the desktop metaphor (implemented by means of the X resource concept).
- The user may also become an author. However, in the prototype authoring of standard users is restricted to creating text documents and links (annotations). Creation of document types other than text is still more complex and reserved to "real authors".

Although not fully implemented, demonstration of hidden access to remote databases will be possible in special cases, such as access to the institute's in-house videotex database. This computer can also be reached from middle-european public videotex networks, so this connection can be used to demonstrate a number of the communication aspects of Hyper-G.

2.3.5 Document Managers

Document managers will be implemented as individual processes (section 2.2.5, page 52). A lot of the functionality and appearance of the system is therefore in the hands of the document managers. This section briefly describes the features of the document managers that will be used in the prototype system.

2.3.5.1 Text Document Manager

Text is stored in a format similar to TEX[90, 91] as defined in [132], which means that the text is not entered in a manner that directly implies document layout (e.g. where to break a line); rather, control sequences specify layout information (e.g. paragraphs, tabs, headings), and the document is formatted (i.e. line breaks and hyphenations are determined) on the fly by the document manager. This approach has the obvious disadvantage of consuming more computation time than formatted entry or pre-formatting, but offers the following advantages:

- The authors need not care about the size of the target text display. In fact, this size (i.e. characters per line) may vary on a per-terminal or per-user basis, and is difficult to tell with proportionally spaced fonts.

- The user may at any time resize the text window, which initiates a re-formatting of the text by the text document manager.
- The user may configure a certain font family and size to be used to display the text. Obviously, changing the size requires re-formatting.

The text document manager is able to highlight anchors, determine the user's click on an anchor and pass the anchor ID to the S/IM. In addition, when the user clicks on a word that is not part of an anchor, this word is passed to the S/IM, which in turn initiates a key query with this word. Other interactions with the user are handled by the text document manager itself, such as:

- Vertical scrolling (no horizontal scrolling is needed because the text is always formatted to fit into the width of the text document manager's window).
- Moving and resizing the window. Moving is handled by the display manager, but resizing requires reformatting, which is the task of text document manager.
- Selecting font family and size. Requires re-display of the text; if the font size is changed, the text has to be re-formatted also.
- Portions of the text may be marked and put into the system-wide past buffer of the X window system, so that it can be copied to other windows (like terminals or text editors).

When the text document is displayed and the target anchor is not the default anchor (i.e. the whole document), the text is scrolled vertically so that the beginning of the anchor is near the top of the display. No text editing is supported in the document manager.

2.3.5.2 Drawing Document Manager

Drawings are stored as PIC files [50, 85]. Anchors are represented by local segments within the picture. Because of the object-oriented nature of the picture, it may be scrolled and/or rescaled so that the target anchor is visible. If the target anchor is the default anchor (i.e. the whole picture), the picture is scaled to fit into the window of the drawing document manager.

The user may click on a segment (by clicking in the interior of the segment's bounding box), in which case the drawing document manager informs the S/IM by sending the anchor ID. The S/IM will then activate the corresponding link.

The drawing document manager is built on top of the EDEN kernel [4], which allows the user to perform a number of interactive operations, such as:

- Horizontal and vertical scrolling.
- Zooming.
- Rescaling.
- Changing colors.

Potentially, the user could even edit the drawing, even while other users watch the same document. Doing so, a graphical communication is possible.

2.3.5.3 Raster Image Document Manager

The raster image document manager is able to display raster images of various kinds. Depending on the kind of terminal and the user's preferences the following image types can be displayed:

- Full-color (24 bits/pixel) images offer the best quality, but need much storage space and can only be viewed on workstations with 24-bits (or more) per pixel frame buffers.
- 8 bits/pixel images with variable color table offer good quality, can be viewed on most workstations, and require less storage. However, only one image of this kind can be shown at a time, as the 'optimal' colors are chosen image-dependently using a modified *Median-Cut* algorithm [64].
- 8 bits/pixel images with 216 fixed colors still offer reasonable quality (if the resolutions of the image and display are high enough). Basically, the RGB cube is split into 6 slices in each of the three directions, resulting in $6^3 = 216$ colors. Then, each image is dithered to this set of colors using the Floyd-Steinberg algorithm[54]. The advantage of this method is that a number of images may be displayed simultaneously. On a workstation with 8 bits/pixel display memory, this leaves $2^8 - 216 = 40$ colors for other applications.
- Also, the images may vary in size. For performance reasons, a user might choose to look at the smallest available image ("icon") first, before displaying the full-size image. This is especially valuable when connected with Hyper-G via a slow-speed network (e.g. phone line).

Anchors of raster images are defined as rectangular areas with optional icons (small raster images). When the user clicks on such a rectangle, the anchor ID is sent to the S/IM. Other user interactions handled by the raster image document manager include:

- Horizontal and vertical scrolling.
- Integer Zooming (i.e. every pixel is displayed as a rectangle of $n \times n$ pixels).
- Integer Reducing (i.e. every n^{th} pixel in every n^{th} row is displayed).
- Color changes (e.g. gamma correction).

When the image is first displayed, the window automatically adapts to the image's size. If the window cannot be made large enough, the image is scaled down to fit or the user may scroll the image (this option is user configurable). When the target anchor is not the whole document, the image is scrolled so that the target anchor is visible.

2.3.5.4 Digitized Sound Document Manager

The digitized sound document manager is able to replay sound that was recorded and digitized in advance. As with raster images, sound may be stored in different qualities:

- CD-quality sound means stereo, 44.1 KHz sampling rate, 16 bits/sample, and gives the best results. However, much storage space (176 KB per second) is required.
- For most cases, mono hi-fi quality sound (22 KHz, 8 bits/sample) will do. Obviously, only an eighth of the storage of CD quality sound is needed (22 KB/sec.).
- For ordinary speech the quality of an ISDN phone line (8 KB/sec.) may be considered. If even less space is required, a number of compression algorithms are available [187].

Anchors of sound documents are beginning and ending time marks. When the document is "displayed" the portion between the two marks is played back (the whole document in case of the default anchor). When used as a source anchor, the users are visually informed during playback when an anchor is reached. They may then press a button, in which case the sound document manager sends the anchor ID to the S/IM.

The user is given a visual representation of the document (a sort of a scrollbar that shows how much has been played), and may:

- Increase/Decrease the volume (if the hardware supports it).
- Terminate the playback.
- Move the scrollbar back and forth, in order to re-play parts of the document.

2.3.5.5 Digital Movie Document Manager

Digital movies are stored as a sequence of differential raster images. Because of limitations of storage and I/O bandwidth, the actually moving parts of the movie are limited (up to 400×300 pixels) in order to achieve reasonable frame rates. However, this approach allows to move relatively small objects over a large but stable background.

Anchors in movies are a combination of the rectangular areas of raster images (one per image, so the anchor may move also) and the beginning and ending time marks of sound documents. When used as a destination anchor, playback of the movie is started at the beginning mark. During playback, anchors are highlighted and may be clicked by the user, in which case playback stops and the anchor id is passed to the S/IM.

Like with sound playback, the user is shown a sort of scrollbar that shows how much has been seen. The user may interactively:

- Move the scrollbar back and forth, in order to re-play parts of the document.
- Additional buttons (similar to a standard video player) allow the following functions: Play forward, play backward[15], fast forward, fast backward, slower/faster, still image.
- When a still image is displayed, the standard image operations (section 2.3.5.3, page 93) may be performed on the image.
- Of course, the user may at any time terminate the playback of the movie.

Synchronization of audio and video is a rather complex task (section 3.3.4, page 114). In the Hyper-G prototype, both sound and digital movie documents are synchronized with respect to time, i.e. played back with correct speed and therefore remain synchronized as long as no user intervention takes place. However, e.g., stopping the movie does not stop the sound.

2.3.5.6 Animation Document Manager

For the time being, animations are implemented as digital movies, so the same document manager is used for playback. In the future, animation may be performed by a document manager implemented on top of PHIGS. However, this approach requires high-performance 3D graphics hardware and will not be available on all workstations of the system.

[15]Because differential images are used to encode the movie, playing backward may result in strange effects if no backward channel is supplied with the movie (section 4.3.2, page 137).

2.3.5.7 Map Document Manager

In the prototype, maps are displayed as either raster images or object-oriented drawings, so the standard document managers for the appropriate document type may be used. However, in the "real" Hyper-G system it is planned to supply additional functionality to maps (e.g. display of information depending on the zooming level).

2.3.5.8 Communications Document Manager

Communication will be possible via electronic mail and on-line communication between Hyper-G users. The Communications Document manager is therefore just a layer on top of the corresponding UNIX tools ('mail' and 'talk') that is compatible to Hyper-G's user interface metaphor.

In addition, communication with users of middle-european videotex systems is demonstrated by using a layer on top a a videotex decoder running under UNIX.

2.3.5.9 Dialog Document Manager

While in Hyper-G, dialog objects may become very complex (e.g. *flex-forms*), in the prototype only very simple dialogs (like menus) are implemented. In addition, a special *answer judging* document manager is needed to port some 700 existing lessons of the COSTOC project [110] to Hyper-G tours [70], as well as advanced features described in [69].

2.3.5.10 Tour Guide

As stated in section 2.2.6.1.5 (page 67), a special document manager called *TourGuide* is used to interpret the tour (i.e. the tour's script), and to present a suitable user interface that provides the advanced navigation facilities possible within tours (**Req. U.3.3.2**). While in HBS the script implicitly contained the tour's links, the Hyper-G prototype relies on labeled links stored in the KAD.

In addition, one very simple script language is implemented that consists of an ordered list of labels. The meaning is that the order defines a priority for following labeled links. For example, the script

```
MyFavouritePictures, FineArtPictures
```

allows the author of the tour *MyFavouritePictures* to include large portions of the existing tour *FineArtPictures* by creating links to an entry cluster and from the exit cluster(s) of the desired subset of tour *FineArtPictures* (figure 2.15). When executing the tour (script), the *TourGuide* follows the links *MyFavouritePictures* where they exist, else the links *FineArtPictures*.

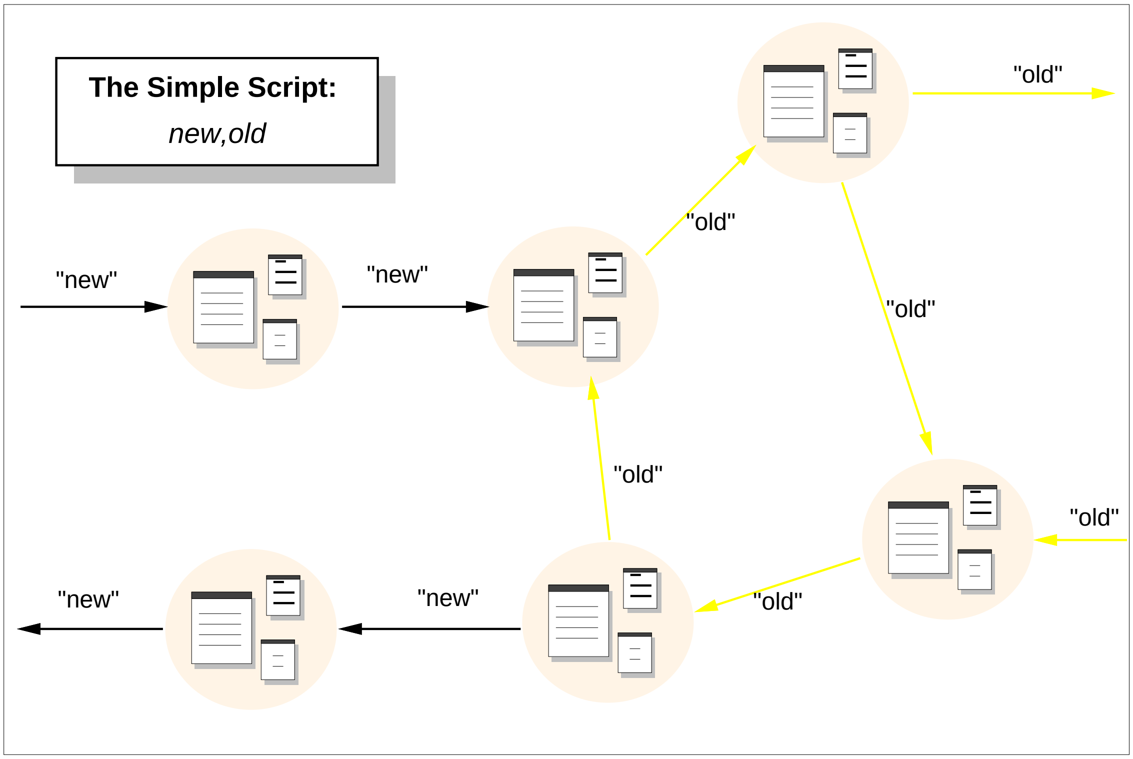

Figure 2.15: Sub-Tours Implemented with the Simple Script Language

Of course, the author could have achieved the almost the same by creating new links labeled *MyFavouritePictures* parallel to the existing *FineArtPictures* links. However, with the priorities approach, the tour *MyFavouritePictures* automatically changes when the sub-tour of *FineArtPictures* is modified (e.g. new clusters are inserted).

2.3.6 User Interface

The user interface of the prototype adopts the *Desk-Top* metaphor that has been around since Xerox' Star [81], was refined by the user interfaces of computers like Apple's MacIntosh, the Commodore Amiga, the NeXT, and – with the advent of the X Window System – made its way to today's UNIX workstations. To put it simple, this metaphor allows the user to simulate a desk on the computer's screen. Each application opens a window that usually can be moved, resized or iconified. Communication with the applications relies heavily on mouse, buttons, pull-down and pop-up menus, and the like. A major disadvantage of this metaphor is that it allows the user to cram the screen with lots of windows (like the piles of sheets found on some real desks). No matter how large the number of pixels, the screen is always too small. Figure 2.16 shows the typical working environment of the author.

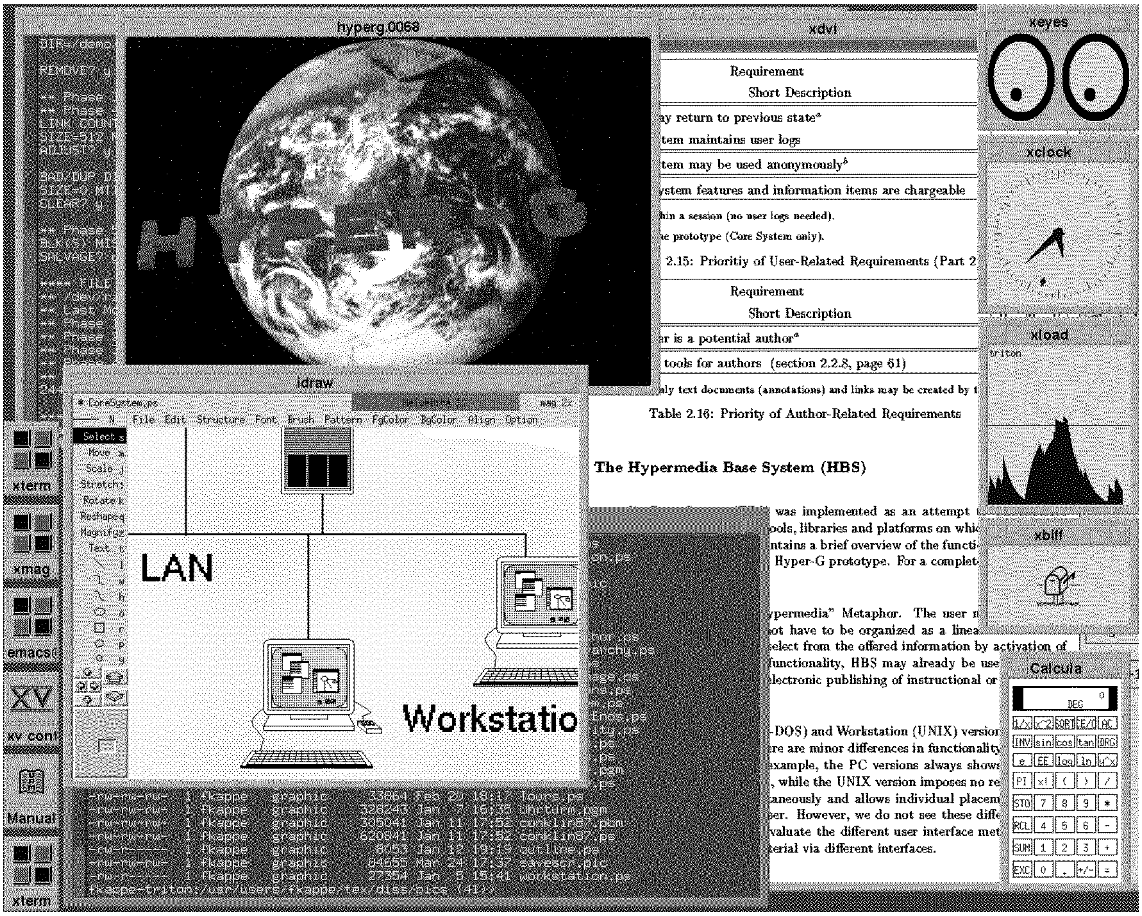

Figure 2.16: The Desk-Top User Interface Metaphor

Consequently, every document manager opens its own window. The window may be placed and sized according to the user's preferences. Also, the user may at any time move, resize or iconify the window. The S/IM uses its own window to communicate with the user.

In order to reduce communication overhead between the document managers and the S/IM, most user interface functions are handled locally within the document manager (section 2.3.5, page 91). This requires the document managers to be equipped with buttons, menus as well as keyboard and mouse bindings. To guarantee that the user is not confronted with inconsistent user interfaces of the document managers, some rules have to be obeyed in the design of them:

- The left mouse button is always reserved for the clicking on anchors, buttons and the like.
- The right mouse button may activate a menu specific to the document manager.
- The middle mouse button may be used for scrolling.

- There is always a 'Quit' and an 'Undo' button available. Alternatively, the user may use the keyboard accelerators 'Q' and 'U'.
- The labeling of the buttons is always in the user's preferred language (if available). Alternatively, icons may be used consistently throughout the system.

The local Undo function is interpreted by the document manager. In addition, there is a global Undo available in the S/IM's window.

2.3.7 Data

In order to evaluate the suitability of the proposed navigation tools (database queries, collections, tours), a large number of documents is needed already in the prototype phase. Particularly, we plan to integrate the following databases:

- A number of German encyclopedias[16]:
  - Meyer's encyclopedia in 10 volumes (51300 documents)
  - Duden Vol. 8 (German synonyms; 57900 documents)
  - Duden Vol. 10 (definitions of German words; 15600 documents)
  - Duden Informatics (600 documents)
  - Eulenspiegel (an archive for a science magazine in cooperation with ORF, the Austrian Broadcasting Organization; 1000 documents)
  - Human diseases (300 text documents plus 300 pictures)
  - Technology (300 text documents plus 300 pictures)
- We have not yet decided on what English encyclopedias to include in the prototype.
- A number (approximately 10000) raster images, from areas like:
  - Geography (e.g. illustrations for an electronic map of the world)
  - Science (e.g. space science, biology)
  - Art (e.g. museum-like exhibitions)
  - Illustrations of encyclopedias
- Some object-oriented drawings, mostly technical illustrations in encyclopedias.

[16]Meyer's encyclopedia, Duden Vol. 8 and 10, Duden Informatics, Eulenspiegel are already accessible from videotex, plus other similar databases.

- A small number of digital videos and animations.
- Some sound documents (e.g. CD quality samples of famous compositions, hi-fi quality voices of animals and famous persons, voices reading text documents).
- Maps of the world, all countries of the world and regions of interest.
- Some 700 existing COSTOC lessons [110] covering topics ranging from medicine, ecology, and natural sciences to computer science.

Chapter 3

Hyper-Animation

3.1 Introduction

In this chapter we introduce a new concept for the creation of real-time, interactive animation[1]. It is essentially a combination of Computer Animation and Hypermedia technologies, therefore we will call it *Hyper-Animation*. A key feature of Hyper-Animation is that it may be integrated within the Hyper-G environment.

Hyper-Animation is not limited to traditional computer animation, but combines still images, digital video, computer animation and sound together with non-standard input devices to achieve a high degree of interactivity. In Hyper-G, links from an animation sequence to all kinds of information, including other animation sequences can be defined. Such links may be followed by explicit choice of the user, or by some random or external mechanisms.

Hyper-Animation is a general approach, allowing complex animation sequences (in terms of complexity of objects, number of objects, complexity of motion), and – most important – can be edited conveniently. The *Hyper-Animation Editor* (section 3.6, page 118) acts like a cutting desk, where prefabricated animation sequences are linked together.

After a short overview on the state of the art of conventional computer animation (section 3.2, page 102) we will discuss some of the more complex data types (i.e. document types) supported by Hyper-Animation (section 3.3, page 108). Then, the Hyper-Animation functionality (section 3.4, page 114) and the integration within Hyper-G (section 3.5, page 116) are addressed. Example applications are only sketched (section 3.7, page 118). However, this chapter serves as a foundation for the more concise descriptions of applications that may be found in chapter 4.

[1]An early outline of the Hyper-Animation concept has appeared in [88].

3.2 Conventional Computer Animation

The term *Computer Animation* covers a number of systems, applications and concepts, ranging from computer generated movies, commercial graphics, video games, and education to flight simulators and scientific visualization. The greatest common denominator of animation systems is that computer generated images change with respect to time.

A number of possibilities exist to create such changes:

- Appearance and disappearance of objects.
- Motion of objects.
- Change of the shape of objects (metamorphosis).
- Change of an object's attributes (e.g. color, texture, transparency).
- Changes of the lighting (e.g. shadows).
- Change of camera parameters (e.g. position, direction, aperture angle, exposure time).

All these changes occur and must be viewed in a continuous way, but are really computed for a discrete number of instants. The visual effect of a continuous change is created by display of individual images *(frames)* at a rate high enough to fool the human eye (approximately 20 frames per second).

Computer animation is sometimes derived from traditional ("cartoon") animation, and some of the terms are borrowed from there. The production of a computer animation as well as traditional animation can be divided in the following phases:

1. **Pre-processing:** The *Story* is defined in various levels of detail. The *Storyboard* contains comic-strip-like snapshots of the film. The film consists of *sequences*, which in turn consist of *scenes*.
2. **Scene editing and object modeling:** The individual scenes and objects are entered. Depending on the system, objects may be program-generated, extracted from a database, scanned or edited manually. Also, an object's material (color coefficients, shininess, texture, dissolve, etc.) is defined at this stage.
3. **Animation:** Now the motion of objects, cameras and light sources are defined. As we shall see, there are a number of possibilities at this stage.

4. **Rendering:** The time domain is cut into discrete steps and the individual frames are rendered. The process may be very time consuming when realistic images have to be delivered. Therefore, improvement of algorithms and hardware used to generate the frames (*Image Synthesis*) are a main area of research.

5. **Post-processing:** This stage includes shooting on film or video, synchronization with sound, and copying.

6. **Analysis of the Results:** Is necessary in simulations and scientific visualization applications.

3.2.1 Classification of Computer Animation Systems

Computer animation systems can be classified in a number of different ways. Magnenat-Thalmann and Thalmann [105] define five levels of animation systems:

Level 1: are basically graphic editors that do not take time into account.

Level 2: can compute in-betweens and move an object along a trajectory. These systems do take time into account.

Level 3: provide operations which can be applied to objects (e.g. translation, zooming). These systems may also include virtual camera operations like zoom, pan or tilt [109].

Level 4: provide a means of defining actors; i.e. objects which possess their own animation. The motion of these objects may also be constrained.

Level 5: are extensible and can learn as they work. With each use, such a system becomes more powerful and "intelligent".

Pueyo and Tost [151, 178] propose a more complex classification scheme based on various criteria (table 3.1):

1. **Historical:**
   Because computer animation originally developed from traditional animation, stages of development can be distinguished:

   (a) Early systems were intended primarily to mechanize the long and tedious task of drawing so-called "in-betweens", i.e. the frames between the important ones ("key frames"), and the coloring of pictures. This kind of animation system is generally called *Computer Assisted Animation*.

Historical:	• Computer Assisted Animation • Computer Modeled Animation
Application:	• production of commercial or entertainment films • industrial or scientific simulation
Motion Control:	• guiding • program level • task level
Dimension:	• 2D • $2\frac{1}{2}$D • 3D
Scene Generation:	• by drawing • by instantiation of primitives • by reconstruction of a real model
Animation Model:	• based on kinematic data – interpolation of key-frames – definition of geometrical transformations • dynamic data (forces and torques)
Rendering:	more or less realism
Time Performance:	• real time animation • batch animation and shooting
Post-processing:	necessary or not

Table 3.1: Classification Criteria of Computer Animation

(b) In the newer *Computer Modeled Animation* instead, the human part is even more relieved. Models of the film's objects are stored within the computer, which is able to perform different actions on them.

2. **Application:**
   Two classical applications of computer animation can be distinguished.

   (a) Production of films for commercial purposes, education and presentation. Such films utilize highly realistic images, while the motion is not required to conform to physical laws.

   (b) Industrial or scientific visualization. Here the animation is not supposed to yield a realistic impression; rather, the interpretation of real-world data shall be facilitated. Obviously, only computer modeled animation (see above) is suited for this purpose.

3. **Motion Control:**
   The *animator* is the human being that defines th motion of objects (includ-

ing cameras and light sources). Depending on the effort necessary to specify motion, three classes of systems can be distinguished:

(a) In so-called *guiding systems*, the animator must know "a priori" the different positions the objects of the scene will have at each moment. Guiding motion control (also referred to as interactive motion control [183]) usually adopts the *kinematic* animation model, which produces motion from positions, speeds and accelerations and is used by most of today's animation systems. Guiding systems correspond to the "Level 2" systems of the hierarchical classification scheme mentioned above.

The earliest, but still one of the most common, guiding motion control technique is *keyframing*. In these systems, the user defines a set of frames and the times when they occur, and the computer interpolates intermediate frames, generally using some splining technique to provide first-derivative (velocity) and sometimes second derivative (acceleration) continuity to the motion. Keyframing has the advantage that the user sees the total scene at given times, easily noting collisions or undesirable interactions. The disadvantages are that it is low-level and requires a lot of user input.

Another guiding technique is path specification. In contrast to keyframing, where the user has to specify all objects at particular time points, path specification lets the user specify the entire path (usually some sort of spline) of a particular object over time. The disadvantages are that it is low-level and does not allow easy visualization of the configuration of all objects at a particular time instant, making it difficult to detect collisions.

(b) In the *program level systems*, the computer has sufficient knowledge in order to interpret a formal program (the *script*) and carry out all later stages of the animation independent of further user input. Some script languages, such as MIRANIM [108] and ASAS [156] can be considered high-level. ASAS is a complete LISP-based structured programming language including procedures, recursion and typed data structures.

The main drawback of program-level systems is that they require the user to have certain knowledge of computer science, thus their use by artists is limited. On the other hand, high-level animation languages can be interpreted to low-level languages by means of parameterization, finite state machines, command libraries or hierarchies, providing the user with higher level motion control. Program-level systems correspond to the "Level 3" systems of [105].

Much research has been carried out in controlling the motion of the human body [106, 104, 151, 178], especially the human face. It turns out that dance notations (Labanotation, Benesh notation) can be used to describe motion for computer graphics. Human body motion control languages are also of interest in robotics.

(c) In *task level systems* (also referred to as *motor control* systems [191] or "Level 4" animation systems [105]), the objects (also called *actors*) possess their own motion and "move themselves" based on global data, their internal state, and information about other objects. Motion may also be restricted by a set of constraints.

Task-level systems have the choice of adopting the *kinematic* or the *dynamic* animation model. In the kinematic model, motion is produced from positions, speeds, and accelerations, whereas dynamic models describe motion by masses, forces, and torques, from which kinematic data are derived using physical laws and constraints. Most of the currently operative computer animation systems use the kinematic model, which has the advantage of being more intuitive and requiring less computation time. On the other hand, dynamic models are useful to describe complex movements of interacting objects and result in realistic motion, making them suited for simulation purposes.

Task-level systems in general are convenient to use, because most of the work is done automatically by the actors, and complex animations can be specified with minimal user input. The major disadvantage is the lack of fine control over the movements of individual elements. Furthermore, it is impossible to define motions not matching laws and constraints known to the actors.

4. **Dimension:**

(a) 2D- and

(b) $2\frac{1}{2}$D-Systems are usually computer assisted animation systems.

(c) 3D-Systems are more complex and usually computer modeled animation systems.

5. **Scene Generation:**
Similar to motion control, three kinds of models may be distinguished, according to the effort on behalf of the animator:

(a) In 2-dimensional animation, a 2-dimensional graphics editor is used.

(b) In 3-dimensional animation, a geometric model of the scene is stored in the computer (lines, faces, surfaces, boolean operations, etc.), and used in the generation of images.

(c) Real images can be entered using scanners (section 3.3.1, page 108). 3D objects can be entered using 3D digitizers [148].

6. **Animation Model:**
As shown in table 3.1 two main methods of specifying action of a scene can be distinguished:

(a) Most of today's animation systems use *kinematic data* to specify motion, in one of two flavors:

i. In the *key frame* method, the animator must entirely edit some of the frames of the sequence to be displayed. The computer interpolates intermediate ones (*in-betweening*). This method is used in computer assisted animation systems.

ii. Another way is to edit only the objects of the scene, and to specify elemental actions to be performed by each object, such as translation, rotation etc. Most modern animation systems use this approach.

(b) In the *dynamic* model, the animator associates masses, forces and torques with objects. The objects then may generate their own motion, based on physical laws and other constraints (*constraint based systems*) or life laws (*stochastic particle system*). Although this approach is not adopted by most commercially available systems, a lot of research is undertaken in this field. Ideas include more 'intelligence' for the objects (which are then called *actors*; see e.g. [2, 106, 156, 66]), dynamic analysis and inverse kinematics (e.g. [183]), path planning and collision avoidance (e.g. [83, 101]), particle systems (e.g. [155]), visual programming (e.g. [173]), and learning of actors [113, 188].

7. **Rendering:**
Most of today's animation systems concentrate on creating realistic images. As the techniques used to render such images are very time-consuming, much research effort is spent on developing faster algorithms that allow even more realistic images of increasingly complex scenes. Various topics in image synthesis such as shading techniques, complex light sources, aliasing, shadows, ray-tracing and optimization, texture and bump mapping, fractals, particle systems and special natural phenomena have been discussed in this major research area of computer graphics ([107] lists 381 research papers plus some books and tutorials on image synthesis). Effects available in most animation systems are:

(a) Removal of hidden surfaces.

(b) Shading models (reflections, shadows, transparency).

(c) Texture mapping

(d) Anti-aliasing.

(e) Different light sources, atmospheres.

(f) Reduction of the *strobe effect* of fast moving objects using temporal anti-aliasing (*motion blur*).

(g) Simulation of a real camera (focal length, depth of focus).

8. **Time Performance:**

(a) Real-time animation delivers frames at a rate high enough to produce the sensation of continuity for the human eye (more than 20 frames per second). This is only possible with high-performance 3D Hardware and losses in image quality (section 4.3.4, page 140).

(b) "Realistic" movies need more rendering time, and are then copied on film (*shooting phase*) and played back with high speed (*real-time play-back*).

9. **Post-processing:**
Depending on the application, post-processing can be necessary. E.g, the movie may have to be copied on film or video. The process of synchronization of sound and pictures can become difficult with computer modeled animation systems.

3.3 Hyper-Animation Data Types

Animation in Hyper-G (*Hyper-Animation*) includes classical computer animation, but in addition

- Still images (either drawings or digitized images)
- Digital Video
- Computer Animation mixed with digital video
- Sound

will also be supported. Above data types will be explained in some detail in sections 3.3.1 to 3.3.4.

The document types listed above are treated in a special way when they are part of a Hyper-Animation document cluster (section 3.4, page 114). Note that other document types may also be part of a Hyper-Animation cluster, but are treated the usual way, i.e. displayed by their corresponding document manager (section 2.2.5.2, page 56).

In Hyper-G, links from an animation sequence to all kinds of information, including other animation sequences can be defined. Such links may be followed by explicit choice of the user, or by some random or external mechanisms. More on this may be found in section 3.4.

3.3.1 Still Images

Usually still images are not covered by the term 'animation'. However, they may certainly be treated as a special case. Vice versa, animation is mostly generated by just displaying a series of still images fast enough to fool the human eye. Therefore we will first discuss still images, in order to allow the description of true animation in terms of still images.

Hyper-G will support two types of still images (see also chapter 2, e.g. sections 2.3.5.2 and 2.3.5.3):

1. **Object-Oriented Drawings** are most often used for explanatory purposes together with text in encyclopedias and computer aided instruction. Typically, some kind of technical drawing is displayed. Such items will be stored as PIC files [50, 85] generated by EDEN-based graphics editors [48, 49]. The main advantage of object-oriented graphics is the size-invariance of the picture definition. In other words, such images can be displayed in any resolution on any graphical output device; the user may zoom in interactively, etc. Furthermore, the definition of object-oriented drawings requires significantly less storage than other methods.

2. **Digitized (Scanned) Images** will be used to store information not suited for object-oriented graphics. Typically, such images are the digital representation of a picture or photograph. There are four possibilities to obtain a digitized representation. One can use:

   (a) a scanner to digitize images on paper,

   (b) a special kind of scanner for scanning transparency slides,

   (c) a video digitizer for processing of video sources (VCR, CamCorder, CD-V),

   (d) or one can generate the image by computer

   Digitized images are also stored in PIC files [89]. This allows the combination with object-oriented drawings (background – foreground etc.).

While the storage of object-oriented drawings represents no difficulty, the storage of digitized images requires careful attention, because of their huge memory requirements and their inability to adapt to different resolutions without loss of information. If for instance an image was scanned at 3000×2000 pixels, each in one of 2^{24} colors, some processing has to take place before it can be displayed in a 640×480 window with only 256 colors.

The four image sources identified above yield images of different resolutions:

- A typical scanner (e.g. MICROTEK MSF-300 Z) scans at 300 dots per inch, i.e. the resolution depends on the size of the original. A full DIN A4 page (about 8" by 11") gives a maximum resolution of 2400×3300 pixels, out of 2^{24} (16,777,216) colors or 256 gray values. This is also the resolution of a typical laser printer.

- The NIKON LS-3500 slide scanner (on the upper end of the spectrum of slide scanners) delivers 4096×6144 dots, also with 2^{24} colors. This is incidently the resolution of professional color graphic recorders used to output computer graphics to slides or movie film.
- The resolution of video sources is substantially less. The european PAL standard uses about 600 horizontal lines, the north american NTSC TV standard even less than that. The number of pixels per line is not specified, because current TV standards are analog signals, but is about the same magnitude. Even worse, a standard VHS or Video-8 VCR records only about 250 lines of video information, better S-VHS or Hi-8 recorders make it up to 400 lines.
- Computer generated images may have arbitrary resolution and number of colors.

Thus, although it might be desirable to have a standard resolution for the storage of all Hyper-G images, it is not feasible. Another issue is image compression. Good compression techniques (in terms of compression rate), such as *Color Cell Compression* [18] or JPEG [24, 25, 179] lead to loss of information (let us call such techniques a *destructive compression*, which means that the original image can not be fully reconstructed from the compressed image). To cope with these problems, the following organization for storing images will be used in Hyper-G:

All images are stored in a so-called *long term archive* in their native (full) resolution, with full color, using a non-destructive compression (e.g. LZW [180]) or no compression at all. Because of the expected size and number of images, the long-term archive will typically reside on slow but large mass storage media, such as removable optical disks or magnetic tape (e.g. DAT [99]).

Whenever an image is to be inserted into the Hyper-G database, a program called the *Image Retriever* creates a version of the original image, which is inserted into the fast, random-access storage of the execution system. The Image Retriever offers the following features:

- Using image processing techniques, the image may be reduced (or enlarged) to any desired target resolution.
- Likewise, the number of colors may be reduced to a desired value (e.g., by the Median-Cut Algorithm [64] with Floyd-Steinberg [54] error propagation).
- The image may be compressed using a suited image compression method, even a destructive one, e.g. Color Cell Compression.

This strategy offers the following advantages:

- The amount of storage needed in the run-time system servers is kept to a minimum. E.g., a 640×480 frame with 256 colors, coded using Color Cell Compression needs less then 80KB run-time storage, while the original $4K \times 6K$ slide scan image with full color needs 75MB of long-term archive, if stored uncompressed.

- Whenever another run-time image of the same image is needed, with a different resolution or a different window, it can be generated from the long-term archive, without re-scanning the original slide.

- The high resolution of the long-term image allows to generate 'zoomed in' views of the original image. A typical Hypermedia application could allow the user to get detailed views of some (pre-determined) interesting areas of the whole image, and real-time zoom effects in some isolated and special cases. The number of such cases increases with the further development of storage capacities and processing speeds.

- The long term archive may eventually become part of Hyper-G if external storage devices with good access speeds and capacities in the tera-byte range become available. Thus, archiving at high resolution will avoid rapid obsolescence of the picture material.

The search for a specific image in the archive is eased by *galleries*, which act as directories of the archive. In a gallery, small versions of the original images are used to select the latter (like file names used in standard directories). Clever organization of these galleries helps to find the desired image. In addition, keywords and attributes assigned to the images allow database queries.

3.3.2 Digital Video

Digital videos are video (or film) sequences that may be shown in a window of Hyper-G. In order to allow their display and manipulation without specialized hardware (such as video mixing adapters), they are stored in digitized form. Typically, they are generated from standard video sources, such as VCRs, CamCorders or CD-V.

Such video sources have a comparably low resolution (see section 3.3.1). Thus, displaying them on a whole workstation screen with, say, 1280×1024 pixels would not make too much sense, because of a blurred picture. Also, the width-to-height ratio of video is constant $(4:3)$. A low-resolution digital video would use 320×240 pixels per frame, 640×480 pixels would define a high-resolution video image.

In contrast to the low resolution, video offers a high color capability (because of the analog transmission of color). In addition, 25 frames per second are displayed, which is not possible on currently used computers because of their limited I/O bandwidth. A state-of the art graphics workstation (the DEC 5000/200) allows to display only about 10 frames per second at a resolution of 320×240. However, the next generation of workstations is likely to eliminate these problems.

For the time being, digital videos in Hyper-G will be stored as a sequence of still images, using a compression technique called CCC [18, 160]. The refresh rate will be limited to 20 frames per second, and a maximum of 216 colors (out of 2^{24}) will be available in a single frame. A typical low-resolution frame (320×240) is stored in about 40 KB, a high-resolution frame (640×480) will need 160 KB. A one-minute sequence ($20 \times 60 = 1200$ frames) stored in differential run-length format [89] will need about 25 MB (low-resolution) or 100 MB (high-resolution), assuming that only an average of 50 % of the pixels change in comparison to the previous frame.

Recently, an *Extended Color Cell Compression (ECCC)* algorithm has been proposed [147] that achieves compression ratios from 0.6 % to 3 %, with real-time decompression on standard hardware. In the future, even better compression methods will be implemented in hardware (e.g. the JPEG/MPEG standards [24, 25, 96, 179] by C-Cube Microsystem's CL550 chip [26] that is built into the NeXTdimension board [8][17, page 159], or *Digital Video Interactive (DVI)* [100] implemented by Intel's i750 chipset [63]). When such hardware becomes a workstation standard, Hyper-G is prepared to utilize it.

With the advent of HDTV expected in the mid 90's, higher quality video sources will become available, allowing display of full-screen digital video. The continuing development of hardware makes it likely that the fast display and the archiving of such pictures will be, nevertheless, easier than today's handling of smaller resolution.

3.3.3 Computer Animation

Two kinds of computer animation will be available in Hyper-G:

- **Real-Time Animation** requires execution on powerful workstations with dedicated hardware (3D-accelerators, double buffering, special input devices etc.). Possible interactivity is the main advantage of this kind of animation. The main drawback of this kind of interactive computer animation is the difficulty of editing the animation in advance. Therefore, interactive real-time animation in Hyper-G will be carried out by executing some kind of 'custom' program, i.e. a special document manager.

- **Real-Time Playback** of computer animation allows the display of more complex scenes that were computed previously (e.g. by ray tracing) and recorded in digital form, very much like digital videos (see section 3.3.2). In addition, these pre-recorded sequences may be mixed with actually digitized videos, yielding high-quality results without too much effort on behalf of the editor. The penalty paid is a decrease in interactivity.

Because of its generality, we will now concentrate on the second approach. A piece of software, the *Hyper-Animation Editor* acts as a post-production tool. It allows the combination of animation sequences with digital video, still images, and sound, as well as the definition of some standardized navigation mechanisms to allow a sort of interactivity between the user (viewer) and the animation. The Hyper-Animation Editor will be discussed in more detail in section 3.6.

3.3.4 Sound

Animation in Hyper-G will include sound to make it more realistic. The synchronization of the sound with the pictures leads to some interesting challenges which we want to mention briefly.

Computer sound output may have different sources. The sound may be generated by the computer itself, behaving like a synthesizer. As a special case, a speech synthesizer (e.g., a phoneme generator) may generate speech at run-time.

However, the most flexible approach is to record ('sample') natural sound. To have sound available on computers, it has to be converted to digital form. This involves discretization both in time (*sampling*) and in amplitude (*quantizing*), called *pulse code modulation (PCM)*. The fundamental sampling theorem states that the original signal can only be reconstructed from a sampled version if it does not contain frequencies greater than half the sampling rate. Telephone quality speech does not contain frequencies above 4 kHz, so a sampling rate of 8 kHz is sufficient (and used in ISDN). However, to achieve high-quality sound suited for music, noise etc. a higher sampling rate is required (e.g., CD's are sampled at 44.1 kHz). Also, the number of available quantization levels varies from 8 to 16 bits (section 2.3.5.4, page 94).

When replaying sound on modern multi-user, multi-tasking workstations, the following problems arise:

- The digital signal has to be passed to a D/A converter exactly at the sampling rate. This is a real-time process, i.e. it may not be interrupted for a period of time longer than the sampling period (e.g. $23\mu s$ with 44.1 kHz sampling rate). This is not possible in a standard UNIX-based multi-tasking system. The solution is the use of special hardware for sound output (it is not enough to just have a workstation with a built-in loudspeaker). This hardware will have a small processor for itself, and is fed from the host computer via an input queue by a host program. The input queue has to be large enough to hold the digitized sound for the largest expected time that this host program gets interrupted by other processes, say n seconds.

- In general, sound will have to be synchronized with other events. In particular, we will want to synchronize sound with animation. Typically, it may be required that a certain frame of the animation corresponds with a portion of the sound replayed together with the images (the *sound track*). As the image display speed may vary with the workload imposed on the workstation, and because the sound cannot easily be slowed down or speeded up, the images have to be synchronized by the sound, i.e. the sound is the master of time (and the images are the slaves). This may be achieved by skipping a frame if the images are behind, and by extending the display period of an image if the images are ahead. Again, the increase of speed of graphic processors and communication channels should eventually ease this problem at least for "standard applications".
- Using the mentioned approach to above problems leads to a new problem. Whenever the host program gets active, it completely fills the sound output buffer. As a consequence, a delay of up to n seconds between the filling of the buffer and the actual output of the corresponding sound is introduced. This means that the sound to be synchronized with an image has to be fed into the sound output buffer up to n seconds before the corresponding image is displayed. This not only complicates the run-time synchronization of sound with images, but also the editing of sequences with sound attached to them, especially, if the order of sequences is determined at run-time (by user interaction, random choice etc.).

To make things even more complicated, Hyper-G supports multi-track sound assigned to a certain animation. This feature may be used for high-quality stereo sound, as well as for multi-lingual speech. E.g., a certain animation may be used simultaneously by a number of users in different languages.

3.4 Functionality

Looking back at section 3.2, we may distinguish between real-time animation and real-time playback of animation. While real-time animation allows virtually any type of user interaction (e.g. games), interaction is somewhat limited with real-time playback. However, real-time playback is a more general approach, allows arbitrarily complex animation (in terms of complexity of objects, number of objects, complexity of motion), and can be edited conveniently.

Hyper-Animation is based on the real-time playback approach, but tries to push the (user) interaction features to the limit. In addition, the concept combines interactive computer animation with Hypermedia paradigms. To be specific, the following document types may be combined in an Hyper-animation sequence:

1. **Still Images**: These are either digitized images or object-oriented drawings (section 3.3.1). A typical application of digitized still images is use as background, while drawings may be used as titles.

2. **Digital Video**: This is probably the easiest way to generate animation, although the features are nothing more than those of video film.

3. **Computer Animation**: Animation sequences generated by computer animation software (the *Animation Editor*) allow virtually everything one can think of. However, it turns out that realistic rendering of some objects (e.g. clouds) or realistic motions (e.g. of humans) are not easily handled by a pure computer animation system. In such cases it is more convenient to have a digitized image of the cloud or a digital video of the moving human, and combine them with the computer-generated animation.

4. **Sound**: Sound is recorded (sampled) from natural sources and digitally stored by another stand-alone program (the *Sound Editor*). It is the task of the Hyper-Animation Editor to link and synchronize these prerecorded sound tracks with the animation sequences.

5. **Custom Program:** A program may be used to generate graphics on the screen, based on some algorithm or under user control. At edit-time, the behavior of the program is not known to the Hyper-Animation Editor, except, that program has to obey certain rules. This feature is intended as a 'last recourse', if the desired effect cannot be achieved using other interaction features of Hyper-Animation (see section 3.4.1).

With the exception of sound, the data types may be combined in the following ways:

1. The most obvious combination is concatenation in the time dimension. I.e., individual animation sequences of type still image, digital video and computer animation are linked together and may be replayed in sequential order. However, the sequence does not have to be static (see section 3.4.1), hence allowing some interaction or non-determinism in the succession of sequences.

2. Animation sequences may also be combined in the z-dimension, assuming that the z-axis grows out of the computer screen (*layering*). Typical applications are background-foreground type animations (of course, an arbitrary number of layers is possible), superimposing of titles etc.

   Layering may be done directly by the Hyper-Animation Editor. A certain color or a number of colors of the foreground-layers is defined to be 'transparent'.

Then, the full image can be generated similar to the 'blue box' technique of TV. In some situations (e.g., the foreground is changed under algorithmic control), it may be necessary to postpone layering until execution at run-time, facing a possible decrease of performance.

3. We can also combine animation sequences in the x-y-dimension. I.e., two or more sequences may run simultaneously side by side. Again, this may be done at edit-time or at run-time, depending on the scene. If we have a small video sequence running in an environment with a large still image, performance will be better if the combination is done at run-time.

3.4.1 Styles of Interaction

The aim of Hyper-Animation is to allow convenient editing and integration within a Hypermedia environment, without sacrificing interactivity. In principle, the ability to include 'custom' programs allows any kind of interactivity; however, having to write programs is not exactly what we would call user-friendly animation editing. To aid the author of a Hyper-animation sequence, there are a number of 'built-in' possibilities to achieve interactivity without having to write programs.

- The 'standard' links and anchor concepts of Hyper-G allow the user to click on anchors, modifying the succession of sequences.
- In addition, the sequence (or appearance) may vary at random, based on external events (sensors), or on previous animation sequences.
- An animation is implemented based on the Hyper-G *Tour* concept, giving the user all of the advanced navigation facilities available in tours (section 2.2.6.1.5, page 67).

3.5 Integration within Hyper-G

After the preceding introductory sections we will now concentrate on the integration of Hyper-Animation within Hyper-G.

Animation is decomposed into so-called *sequences*. A sequence is a group of elementary data types (still image, digital video, computer animation, sound). The sequences are represented and stored as Hyper-G document clusters (section 2.2.3.3, page 19), i.e. documents of document types *HA_image, HA_drawing, HA_video, HA_animation*, and *HA_sound* grouped together by cluster links.

Unlike other document types, above ones are not interpreted by a document manager of their own, but by a single Hyper-Animation document manager. This is necessary to combine, overlay and synchronize the display (execution) of the documents.

Hyper-Animations are implemented on top of Hyper-G's *tour* concept, i.e. using labeled links and/or scripts. The navigation facilities possible in tours apply (section 2.2.6.1.5, page 67); also, sequences that are part of more than one Hyper-Animation may be used to 'switch' to another Hyper-Animation.

3.5.1 Hyper-Animation Links and Anchors

Interaction with – or, in Hypermedia terminology: Navigation in – the Hyper-Animation is performed using Hyper-G's *link* and *anchor* concepts:

- The most common operation is to proceed from one sequence to another. This may be achieved interactively by clicking on an anchor, which in turn activates a referential link, proceeding to another document cluster (animation sequence). Like with tours, the link is chosen based on the link's *label*, allowing to use a single animation sequence in more than one Hyper-Animation.
- In addition, active anchors (section 2.2.3.7, page 26) can activate links without user interaction, e.g. after the end of the animation sequence, after some (random) period of time, based on more complex user input, etc.
- The Hyper-Animation Document Manager is able to overlay still images, digital videos and animation sequences in the z or $x - y$ dimension (e.g. a digital video or animation of bird flying over a still image background). This function is also initiated by anchors (active or interactive), with links pointing to special destination anchors that contain overlay and coordinate information. Documents that (may) have to be played back together are in the same document cluster.
- A special kind of active anchor – the *Repeat Anchor* – allows to repeat a certain animation sequence a number of times specified by the *repeat count*, or until a user interrupt. Again, the type of the destination anchor determines whether (and where) the target is to be overlayed on the current animation, or replaces the current animation sequence. If the source sequence is only a single still image, the animation just stops and waits either a number of seconds specified by the repeat count, or until user input occurs. The repeat count may also be generated as a random number, or could be based on external data on a file or a set of special programs (e.g. using the time of day or unusual input devices to generate the number).

- The *Branch Anchor* (section 2.2.3.7, page 26) selects a target sequence from a number of possibilities, either by user input, or based on random numbers or external data (e.g. sensors).
- The *Asynchronous Anchor* is not attached to a specific sequence, but to a kind of event and is handled internally by the Hyper-Animation Document Manager. If the event occurs (e.g. user interrupt, timer, at random), a certain sequence is executed. Additional attributes control whether the new sequence replaces what was going on (like a 'goto'), or is executed like a subroutine and returns to the original sequence afterwards. Again, the link is not restricted to the time domain. For example, an asynchronous anchor may be used to let a plane fly over the screen every n seconds, no matter what is on the screen. To further enhance this concept, asynchronous links may be enabled and disabled for individual animation sequences.

3.6 The Hyper-Animation Editor

The Hyper-Animation Editor adopts the metaphor of a cutting desk. In terminology of conventional film or video production, this is the post-production stage: The sequences of film generated by the cameras have to be cut and glued together, the sound track is synchronized, and special effects and titles are added.

The same is true for the Hyper-Animation Editor: It does not generate the animation; instead, prefabricated animation sequences are linked together by defining link types. Therefore, Hyper-G's tour editing package (section 2.2.8.2, page 80), equipped with a few extensions (e.g. an animation previewer) will do the job.

3.7 Examples

This section only sketches some of the applications of the Hyper-Animation concept, as a detailed discussion is contained in chapter 4.

A simple examples might be a single animation sequence (i.e. document cluster) that may be used in instructional lessons (tours), e.g. a background drawing with a schematic diagram of some electrical circuit, stating "Press Button A to see what happens when the switch S_1 is closed", "Press Button B to see what happens when the switch S_1 is opened", and "Press Button C when you have seen enough". The document cluster looks as follows (see also figure 3.1):

It consists of 3 documents: One of type *HA_drawing* and two of type *HA_animation*. They are linked by cluster links. On activation only the drawing is displayed, because the the cluster links to the animation documents origin at the buttons A and

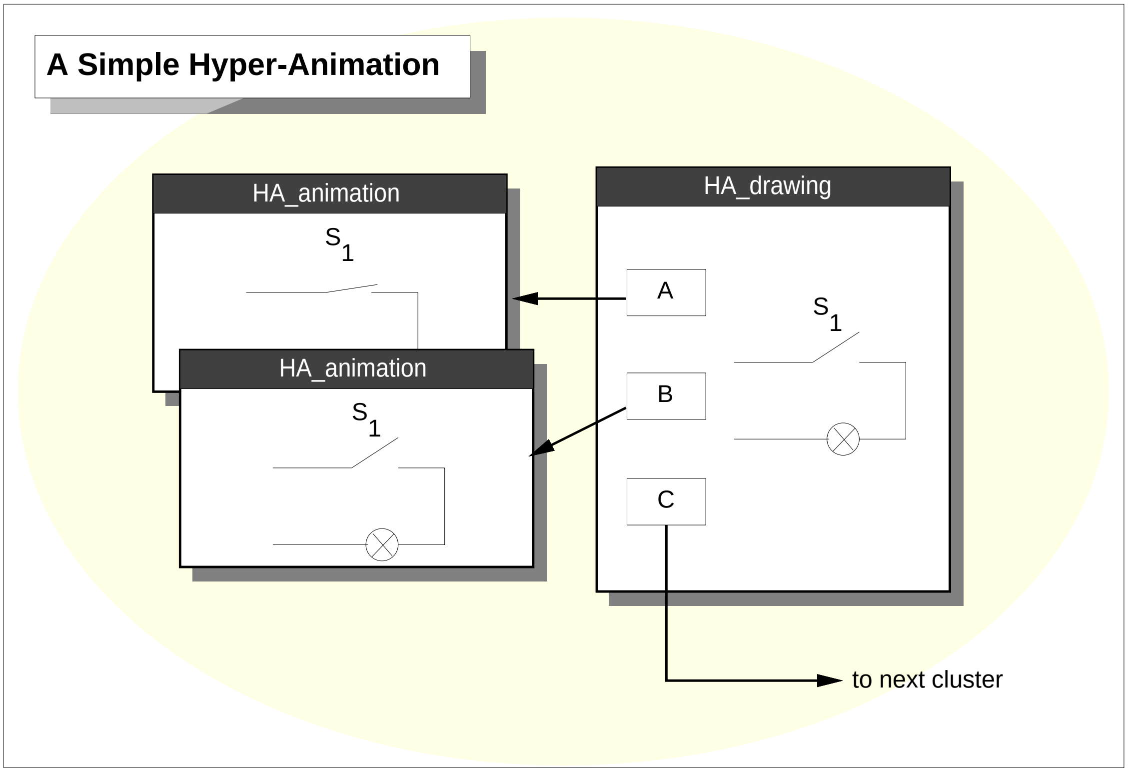

Figure 3.1: A Simple Hyper-Animation Example

B, rather than at the default anchor (the whole document). When the user presses buttons A or B, the corresponding computer animation is overlayed on top of the drawing, as defined by the destination anchor. Button C activates a standard labeled link to the next document cluster. Links to other document clusters can be added, allowing this animation sequence to be incorporated into a number of tours (animations, lessons).

Additional sound documents could have been attached to buttons A and B, resulting in playback of the sound while the animation is running). Also, an additional active anchor could have been attached to the drawing. This anchor could activate other documents (e.g. animations) at random, or responding to external events.

More complex examples of Hyper-Animations ranging from unorthodox user interfaces to virtual reality are discussed in chapter 4.

Chapter 4

Some Advanced Applications of Hyper-G

This chapter describes some of the more advanced applications of Hyper-G that led to the requirements formulated in chapter 2. Roughly, these applications can be divided into three groups:

1. **Information Systems** are a typical application of Hypertext and Hypermedia Systems that were envisioned by pioneers like Bush [16], Engelbart [44] and Nelson [135, 136]. Hyper-G is supposed to become the basis of a *University Information System* which combines concepts of information retrieval systems, documentation systems, communication and collaboration, and computer supported teaching and learning.

2. **Electronic Publishing** through CD-ROM hardware technology and Hypermedia software technology seems to become more and more important (section 4.2, page 126).

3. The **Viewseum** (section 4.3, page 128) covers a broad spectrum of applications that are to be used by the general public, typically in a museum or exhibition environment. Applications range from public information systems to interactive movies, utilizing unorthodox user interfaces from trackball to virtual reality (section 4.3.4, page 140). Especially tha applications "n-dimensional Movie" (section 4.3.2, page 137) and "Interactive Movie" (section 4.3.3, page 139) are new concepts within the Hypermedia environment.

4.1 University Information System

The creation of a modern University Information System is both one of the most important and most ambitious Hyper-G projects[1]. Such a system should help the scientist in the process of information retrieval (e.g. finding literature), ease the communication and collaboration of scientists, and support computer aided instruction of students. The motivation for such a tool is not exactly a new one. In his 1945 article “As we may think” [15, 16, 136] Vannevar Bush writes:

> “There is a growing mountain of research. But there is increased evidence that we are being bogged down today as specialization extends. The investigator is staggered by the findings and conclusions of thousands of other workers - conclusions which he cannot find time to grasp, much less to remember, as they appear. [...]
>
> Mendel’s concept of the laws of genetics was lost to the world for a generation because his publication did not reach the few who were capable of grasping and extending it; and this sort of catastrophe is undoubtedly being repeated all about us, as truly significant attainments become lost in the mass of the inconsequential.[...]
>
> Publication has been extended far beyond our present ability to make real use of the record.”

Since 1945, the situation has become even worse. We are now not only confronted with an increasing number of books and periodicals, but also with electronic media. On the other hand, the availability of knowledge in electronic form potentially allows us to look for specific information and browse through it more efficiently by use of computers and information retrieval systems.

4.1.1 Information Retrieval

Information retrieval is one of the most important aspects of a University Information System. Today, there are several on-line information services (e.g. DIALOG, BIX, CompuServe) available. These services are highly interactive, charging users on the basis of minutes spent on-line, and each has a unique user interface. Also, connectivity between such services is limited. These limitations will hopefully be reduced by the Wide-Area Information Server (WAIS) project currently being performed by Thinking Machines, Apple Computer, and Dow Jones News/Retrieval

[1] Of course, the Technical University of Graz would be the first one to use this system

[166]. By defining a public WAIS protocol which should then be adopted by information servers and end-user user interfaces, WAISes should become connected, usable in a consistent way, and cheaper.

The "remote database" concept of Hyper-G (see figure 2.2 on page 18 and section 2.2.6.2.3 on page 74) allows to integrate such information servers and to make them accessible using the Hypermedia paradigm (i.e. links). It is possible to create both links to and from such remote databases by means of *dynamic links* (section 2.2.3.4, page 22), even when the remote database does not adopt the Hypermedia paradigm. Moreover, it is possible to create links between Hyper-G-based University Information Systems installed at different locations (universities), if this should ever be the case.

While access to remote databases is possible, the University Information System at the Technical University of Graz will have stored information locally that is both accessed on a regular basis and not subject to frequent changes, including:

- A number of general-purpose, scientific, and technical encyclopedias in both English and German.
- Dictionaries for both English (Webster's Dictionary or Oxford English Dictionary [12, 154]) and German (Duden).
- Translators between English and German.
- Geographic information, including pictures from locations throughout the world, maps, etc.
- Access to the university's library.

If available, books and periodicals available on CD-ROM can be integrated also. More on-line data is described in the forthcoming sections.

4.1.2 Computer Mediated Communication (CMC)

As shown in figure 2.1 on page 17 the Hyper-G Core System includes a number of workstations attached to a high-speed LAN. Additional terminals (see figure 2.2 on page 18) may be connected via low-speed (up to 64 kbits/s) phone lines. While simple textual communication is available on any terminal, the workstations will allow voice and (in theory) even video communication between Hyper-G users. Users that are logged in simultaneously may communicate via an on-line communication system [119].

Hyper-G users may annotate any document (provided that access rights allow it). In particular, annotations may be annotated, starting an electronic discussion about some document; similar to bulletin boards. The users need not be logged in simultaneously.

In addition to this "local communication", Hyper-G users may communicate with computer users throughout the world using *Electronic Mail (E-mail)*. E-mail documents are incorporated into the Hyper-G database as private text documents, and may be linked (either static or dynamic) with other documents (including other E-mail, e.g. questions and answers). In order to find E-mail, keywords have to be attached or extracted (automatically), allowing links to be targeted to the piece of mail in question. Of course, E-mail may be annotated.

Today, cost and speed of hardware are no longer limiting the use of CMC systems. However, mechanisms with which the user can organize information to avoid *information overload* [67, 68] have to be provided [152, page 155]. It is believed that Hypertext is a promising mechanism for this purpose, and there are Hypertext implementations like NLS/Augment [45] and gIBIS [9] addressing the E-mail problem.

According to International Data Corp (Framingham, MA), the number of electronic mailboxes will soar from 17.4 million in 1990 to 64.7 million in 1995 [157], and so will increase the amount of mail to be handled. *Information Lens*, a research project at MIT's Sloan School of Management, addresses electronic information overload by semistructured message templates and rules that define what to do (automatically) with mail of certain type and content [157].

An even larger amount of data (approximately 20 MBytes a day) is distributed by a communication system known as *Usenet News* [152, page 235]. There are more than 2000 hierarchically structured newsgroups (discussion topics) available covering computer science, other sciences, social issues, recreational activities and other topics. Unfortunately, the sheer amount of data makes it very time-consuming to follow even only a few newsgroups regularly.

However, if the individual news (which, by the way, conform to certain standards) can be automatically supplied with keywords, stored, and cross-referenced in a Hypertext system, the user could later on search for news covering a specific topic, which would be a valuable tool for the scientist. A prototype of a system that automatically constructs hypertext links between news articles based on a similarity rating has already been implemented at the Technical University of Denmark [3]. We will try to incorporate a similar mechanism into the University Information System, i.e. integrate news articles into the Hyper-G database.

4.1.3 Computer Supported Collaborative Work (CSCW)

Computer Supported Collaborative Work (CSCW), sometimes also referred to as *Groupware* is one of the buzzwords of the beginning 90's. While some people use *Groupware* as a rather general term [59] that refers to anything from electronic mail to distributed databases, Ellis et al. [42] define *Groupware* as:

> "computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment"

and give an overview of concepts and commercial products. Engelbart and Lehtman [45] define CSCW and Groupware as:

> "Computer Supported Cooperative Work (CSCW) deals with the study and development of systems that encourage organizational collaboration. Most Groupware products fall under this classification."

They distinguish between three groups of CSCW projects:

1. Tools for augmenting collaboration and problem solving within a group geographically co-located in real time.
2. Real-time tools for collaboration among people who are geographically distributed.
3. Tools for asynchronous collaboration among teams distributed geographically.

The spectrum of Groupware products ranges from simple or "intelligent" [33, 157] E-mail systems [144, 186] to "joint editing" of documents (text or picture).

"Electronic Meeting Systems" can be used to prepare or even carry out meetings. The *semi-identified* user mode of Hyper-G supports a (semi-)anonymous discussion during the "brain-storming" phase of a meeting, which encourages discussion. A similar application – the "Electronic Classroom" – is discussed in the next section.

According to above definitions, Hyper-G is a piece of Groupware or CSCW, because of its communication facilities described in the previous section. In addition, we will evaluate the feasibility of distributed editing of technical drawings within the Hyper-G environment, made possible by use of the "Distributed EDEN" as described in [4] and [47].

4.1.4 Computer Aided Instruction

As teaching is one of the responsibilities of a university, a University Information System can be expected to also support the process of teaching and learning. Hyper-G addresses this problem in two ways:

- **Computer Aided Instruction** (CAI) is supported by over 700 COSTOC lessons [110] covering topics ranging from medicine, ecology, and natural sciences to computer science, that will be converted to Hyper-G tours (section 2.2.3.12, page 29). New lessons can be edited (as tours) with much greater flexibility. Also, students may at any time consult additional material from other tours, encyclopedias, books, etc. using automatically generated links (section 2.2.3.4, page 22).

- The **Electronic Classroom** is a more collaborative approach to computer assisted learning. Such a classroom is equipped with a number of Hyper-G workstations. A utility built into Hyper-G called “shared X” that was developed by Michael Altenhofen at CEC Karlsruhe allows “WYSIWIS” (What You See Is What I See) distribution of the output of an X Windows client to a number of X Windows servers (workstations) and has been tested in learning/training environments to provide online help from remote tutors. Two scenarios are possible:

  - The teacher shows something to the students by distributing his output to their terminals. Input is taken from the teacher's terminal. At some point, the teacher may ask questions. Then, students may grab input and perform some operation (e.g. answer a question, click on some anchor). Output is distributed to the other student's and the teacher's terminal. So the whole class works together to solve a problem.
  - Students work on their own. When they get stuck, they may ask other students or the teacher for help by sending them a copy of their screen's contents. The teacher or tutor may then grab the input in order to perform the necessary steps while the student watches. Student and teacher may switch control over the input as often as they like to complete the task.

  Note that it is not necessary that the terminals of the electronic classroom reside in the same room, or even building. Potentially, the electronic classroom may be distributed over the whole world.

Additional benefits related to teaching are that the students may call the system via phone lines from home, thus having access to, e.g. the university's library, exam results, e-mail exchange with professors and other students, etc.

4.1.5 Other Aspects

The user of a University Information System can be expected to have some experiences with computers and communication systems, and be familiar with keyboard and mouse. Therefore, the standard *Desktop User Interface Metaphor* (see figure 2.16 on page 98) seems feasible.

Additional features may include a "Hyper-Calendar" which is similar to the calendar found in most DeskTop environments. However, the Hyper-Calendar allows to tie links to certain dates and times which point to other documents. Also, a links may origin at other documents and point to the calendar.

4.2 Electronic Publishing

Electronic Publishing by means of Hypertext and Hypermedia has always been on the minds of Hypertext pioneers like Ted Nelson [135, 136, 137] and others [190]. Already in 1978, Maurer and Angstmann have implemented an electronic encyclopedia [5]. Basically, electronic publishing – i.e., distributing electronic Hypertexts or Hypermedia documents from "author" to "customer" – can be done in the following ways:

- By allowing customers to log into a Hypermedia system and to "read" the documents interactively.
- By sending the documents to customers electronically (like E-mail).
- By sending or selling storage media to the customers that contain the documents.

The first approach is easily implemented with public phone lines and modems. However, throughput is very low so that access to images, movies and sound documents takes prohibitively long and costs too much.

The same is true for the second solution. In addition, missing Hypertext and Hypermedia document standards hinder the distribution of such documents. Although some standardization attempts have been made (see, e.g. [130] and [80]), much work remains to be done in this area.

Distribution of (parts of) Hypermedia databases by means of storage media also suffer from the standardization problem. However, the Hypermedia system itself (or versions of it) can also be distributed on the same medium together with the data, so that no interface to other software is required.

With the exception of very small applications that could be distributed on floppy discs, the CD ROM seems to be the optimal medium for the (Hypermedia) message. Without getting too much into detail (see, e.g., [94]), the characteristics of the CD ROM medium are as follows:

- CD ROM technology is an offspring of audio CD technology, and as a consequence is very cheap: drives cost about $500, the production costs per disc are about $2 to $5. However, mastering costs are high.
- A CD ROM can store about 600 MB of user data. This is the equivalent of approximately 300 1,000-page books, 10,000 low-resolution (320×200 pixels, 256 colors) raster images, or 1,000 high-resolution (1000×600 pixels, 256 colors) raster images, without any kind of data compression involved.
- CD ROM data is practically error-free. A sophisticated scheme of Error Correction Codes (ECC) gives an overall byte error rate of one per 10^{13} bytes, corresponding to a single-byte error in every 20,000 discs [62].
- Throughput is approximately 150 KB per second, which is better than a floppy but less than a Winchester hard disk. Data compression is required to play live video off a CD ROM [100].
- Access time is modest: Radial access time is about 500 ms or more (compared to about 10 ms of a good Winchester), latency (the time to wait until the disc has rotated so that the desired sector is under the head) is between 60 and 150 ms average (compared with 8 ms for Winchesters). As a consequence, sophisticated indices are required to allow access to databases without too much random access to the CD ROM [12].
- CD ROMs are read-only, i.e. the end user cannot modify the data. This may be considered an advantage, however, it complicates annotations, tours, etc.

And the future for CD ROMs looks bright: In 1988, Bill Gates forecast that by Christmas 1991, the typical $1,000 PC would come equipped with a CD ROM drive [111]. Link Resources, a New York research firm estimated a total installed base of 85,000 CD ROM players (1988) and predicted a total of 220,000 for year-end 1989 and 1.3 million for year-end 1992 [111]. While these figure may be a bit optimistic, there is no doubt that CD ROM technology is becoming mature.

Also, Philips and Sony have announced *Compact Disc-Interactive (CD-I)*, that is basically a specification of a delivery system based on the CD-ROM standard, but provides a tightly specified software and hardware framework for developing and presenting data, images, and sound [93]. A CD-I player is a self-contained low-cost computer plus a CD-ROM drive intended to plug directly into home television and hi-fi sets. The big Japanese companies see CD-I as customer electronics more than a computer application.

For the huge amounts of storage typically associated with Hypermedia documents (encyclopedias, images, digital movies, sound ...), the CD ROM is the optimal distribution medium. We intend to distribute parts of the Hyper-G database on CD ROM, together with a CD ROM version of HBS (section 2.3.3, page 87). To be specific, the following collections of data are likely to be published this way:

- An interactive map of Austria with pictures of sights in Austria (to be presented at the World EXPO '92 in Sevilla, Spain).
- Trips around the Himalaya region (about 2,000 high-quality images).
- A hierarchical collection of maps of the world including pictures (similar to the EXPO project).
- A number of hiking trips (maps plus sights) in Austria.
- An electronic encyclopedia (modified Meyer's encyclopedia) with images.

4.3 The Viewseum

A *Viewseum* is a new kind of museum, a *virtual museum* where the exhibits are not on display as such, but are virtual objects that can be viewed and manipulated by the visitor by means of computers and audio-visual techniques. The term has been coined by Maurer [117], and it turns out that Hypermedia systems are well suited for the task [127].

With its ability to adapt to different user interface devices (see table 2.3 on page 40) and user interface metaphors (section 2.2.3.10, page 27), its *anonymous* and *anonymously identified* user modes (section 2.2.3.15, page 32), multilingual documents (section 2.2.3.14, page 30) and chargeable system functions (**Req. U.7**), Hyper-G can also serve as a solid foundation on which to build a Viewseum environment.

Exhibits in such a museum are pieces of audio and video information (video clips, computer animation, pictures) in combination with encyclopedias, communication facilities, specifically designed software and unorthodox user interfaces. A description of the kind of information that may be found in a Viewseum is contained in [127]. In the following sections, we will concentrate on advanced features of such a museum: First, we will discuss some of the not-so-common user interfaces, then we will describe ways of presenting complex (higher dimensional) objects by n-dimensional movies, interactive movies, and virtual reality.

4.3.1 Unorthodox User Interfaces

4.3.1.1 Track Balls and Thumb Wheels

A *Track Ball* is a device similar to an upside-down mouse: A ball is mounted so that it can be rotated freely in any direction. It is typically moved by drawing one's palm or thumb across it. In addition, there may be buttons mounted on the same device. Internally, the balls turns potentiometers, whose output is converted into digital form. Advantages in a Viewseum environment (when compared to a mouse) are:

- Trackballs can be made very resistant to user abuse.
- They can be fixed in place.
- They need less space than mice.
- They are more easily handled by novice users than mice.

Thumb Wheels are more simple devices: Here the potentiometer is rotated directly by one's thumb. Therefore, only unidirectional movement is possible.

4.3.1.2 Touch Screen

A *touch screen* is a device that usually is understood by people who have never touched a keyboard before. Although there are a number of technologies used to implement touch screens (infrared light, gold wires, inductive), the one considered most mature consists of a special kind of coating that is mounted on top of the CRT's glass, and is able to detect the changes in capacity created by a human finger touching the coating, to translate that information to 2-D coordinates of the user's finger, and to transmit the data to the computer (usually via a serial interface).

Depending on the application, the user may choose between menus, click on anchors, and even draw directly on the screen. Occasional text input may be handled by displaying a keyboard on the screen.

However, there is a small disadvantage: The screen is rapidly covered with finger prints and appears blurred. Still, experiences at other museums (e.g. EPCOT Center in Orlando, Florida) show that the touch screen can be operated by most novice users and therefore is an ideal candidate for a Viewseum environment.

4.3.1.3 HOTACT

HOTACT (an acronym for **HO**me**T**rainer **A**nd **C**omputer **T**echnology) is currently developed at the IMMIS (implementation details may be found in [120]) and consists of a stationary bicycle fitness machine connected to a MS-DOS PC with high-resolution graphics. The bicycle trainer has two switches attaches to each side of the handlebars. The user can pedal this combination on a virtual trip through a tour pre-determined by some author. It is actually just a special kind of user interface to Hyper-G tours. By pressing one of the two buttons, the user can navigate within the tour (e.g. turning left, entering a building).

The possible tours are not restricted to the obvious “Trip through Austria” likes, but also “A guided tour through the Louvre”, “A trip through the Solar System”, “A Journey into the Human Body”, “A Journey through Time”, etc. may be provided.

4.3.1.4 Sensors

A number of sensors installed in the Viewseum can detect positions of visitors in order to activate exhibits when visitors come close to the exhibit. More sophisticated systems are even able to recognize the visitor based on an anonymous ID card (such as the ELIS from the Austrian company ELIN or the Keywatch bi Skidata), thus allowing to fine-tune the exhibit’s user interface according to the user’s language, interests, preferences and whether the user already visited the exhibit.

Other sensors may detect temperature, light, the number of users watching and the noise level in the exhibit’s room, to let the exhibit to respond to that data. Pressure sensitive floors may be used to detect the positions of users in the exhibition room.

4.3.1.5 3D I/O Devices

To interact with exhibits like virtual objects (e.g. a statue that can be viewed from all directions; see also sections 4.3.2 and 4.3.4), 3D input and output devices may be attached to the system.

4.3.1.5.1 3D Input Devices

Digitizers that deliver 3D position information (x, y, z) of a point in space have been available for some time. They usually work on an ultrasonic or mechanical basis [82, 128].

More powerful are devices working with low frequency magnetic fields. The "3Space Tracker" of Polhemus [40, 148] delivers not only the position (x, y, z), but also th orientation (*Azimuth, Elevation, Roll*) of a sensor in relation to a source, so it is actually a 6D input device. The same holds true for the "Spaceball" of Spatial Systems. It is essentially a force sensitive device that relates the forces and torques applied to the ball mounted on top of the device, and sends them to the computer [103].

The "Dataglove" of VPL Research can also considered a 3D input device. It consists of a lightweight nylon glove with optical sensors mounted along the fingers that measure the bending angles of the joints of the thumb and the lower and middle knuckles of the other fingers (see figure 4.1; from [55]). In conjunction with the 3Space Tracker the absolute position and orientation of the hand can also be detected, allowing for a new kind of user interface [14, 55]. The users may directly manipulate virtual objects in a way they are used to in the "real world". This device is commonly used together with the "EyePhone" in virtual reality environments (section 4.3.4, page 140). There are cheaper variants (e.g. the "PowerGlove" of Mattel [41]) that work on an ultrasonic basis and are less accurate.

4.3.1.5.2 3D Output Devices

Output devices capable of producing depth perception in the eyes (more accurately the brains) of viewers by means of physiological depth cues are usually referred to as *true* 3D displays. Physiological depth cues are *Accomodation, Convergence*, and *Binocular Parallax* [145]:

- *Accomodation* is the muscular tension needed to adjust the focal length of the eye in order to focus on an object in space. This is a monocular depth cue and is effective only for near-situated objects.
- *Convergence* refers to the muscular tension necessary to rotate the eyes slightly so that the viewing axes of each eye cross at the point of interest.
- The retinal images in both eyes differ slightly due to slightly different viewpoints. They are fused by the brain, and the difference between them is used to produce depth perception. This effect is known as *Binocular Parallax*, and is the depth cue used by most true 3D output devices to produce depth perception.

We can distinguish between the following groups of 3D output devices [53, 145, 158]:

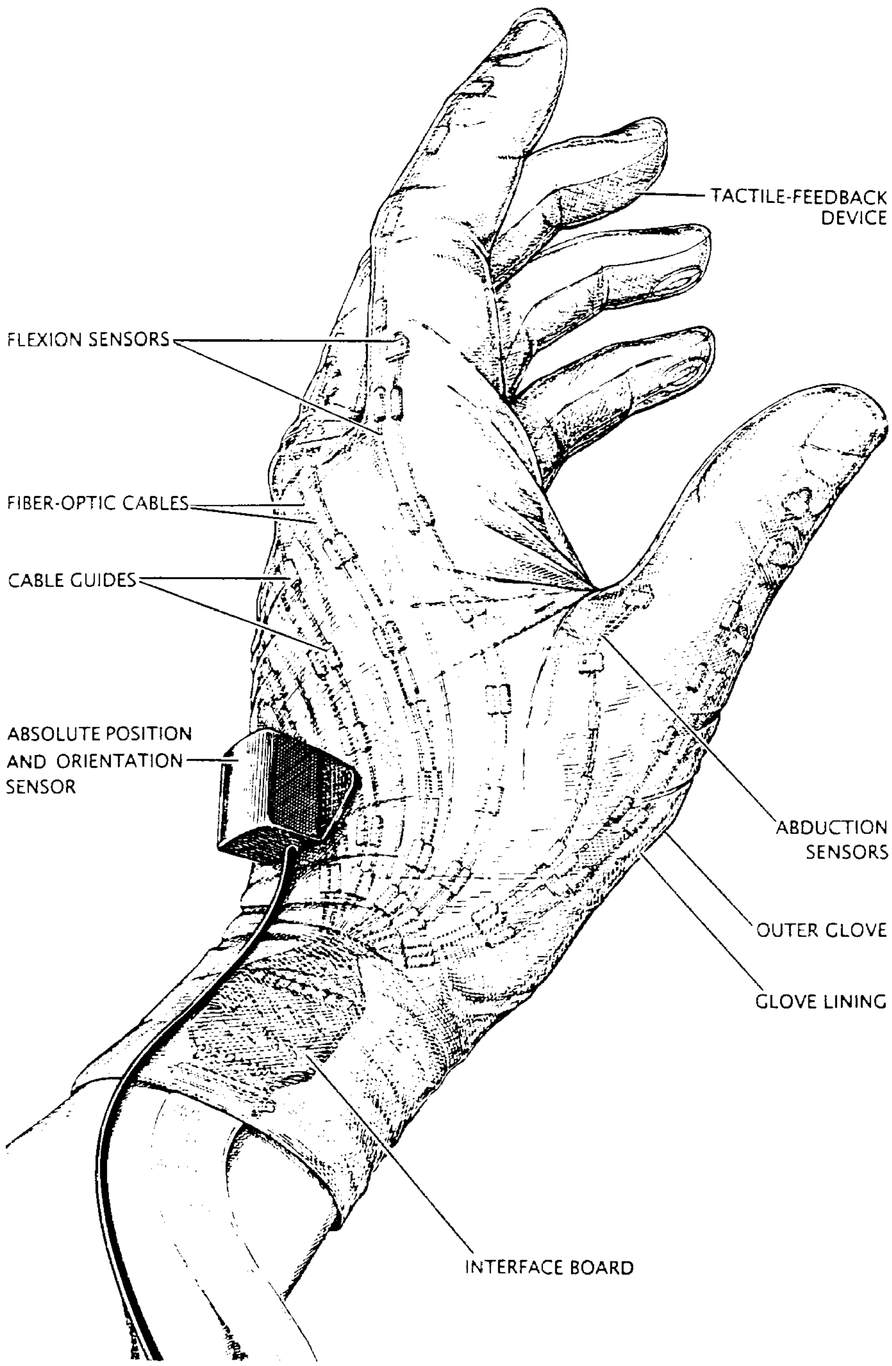

Figure 4.1: The DataGlove (VPL Research)

1. **Stereoscopic Systems:**
   In stereoscopic systems, the eyes of the viewer are directly provided with different views of the object or scene. Typically, they are perspective views taken with virtual cameras [109] with a rotational difference of about 6°. To ensure that each eye sees only the image that it is meant to see, a number of

techniques can be used:

(a) **Time-Parallel Systems:**
The easiest approach is to use different colors (e.g. red and green) for the two images, and to supply the viewer with colored glasses that filter out the "wrong" image for each eye. This is the solution with the lowest hardware costs, however, there are disadvantages: Pictures have to be monochrome, and the technique causes eye strain and fatigue due to conflict in our visual system when each eye is exposed to a different color. Also, color-blind persons cannot use the system.

A different technique uses polarizers mounted in front of two monitors. A half-silvered mirror combines the two images (figure 4.2; from [145]). The viewer wears polarized glasses that filter out the "wrong" image for each eye. A major disadvantage of this technique is the limited viewing position imposed by the mechanical arrangement, so that only one person can use the system at a time.

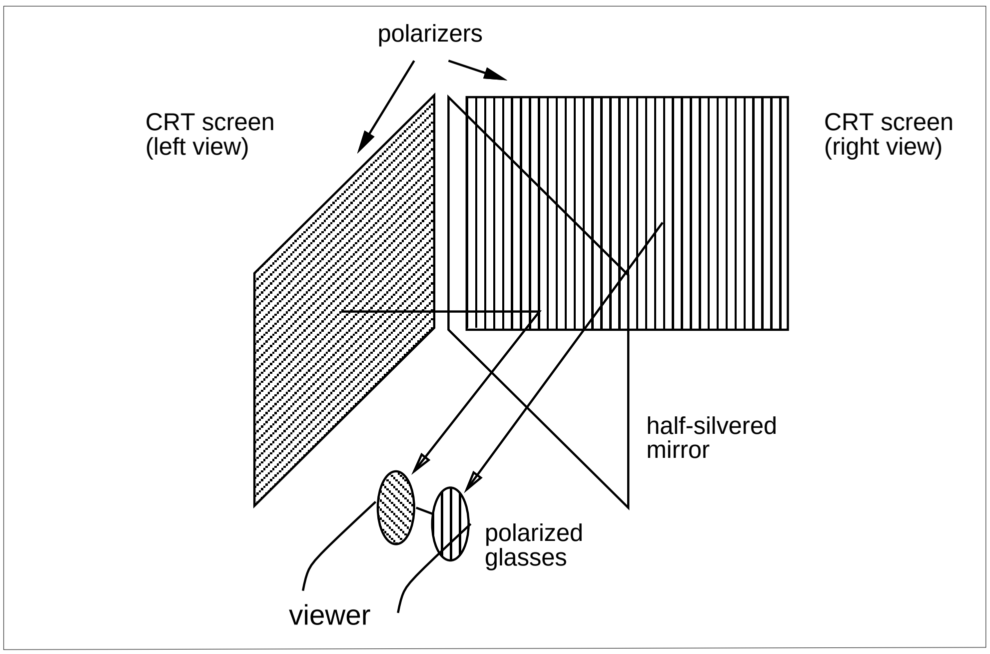

Figure 4.2: Dual-Monitor Stereoscopic Display

The "EyePhone" of VPL Research is a so-called "Head-Mounted-Display" [53]. Each eye looks into its own display, achieving perfect image separation. However, hardware costs are high and only one person can use the system.

(b) **Time-Multiplexed Systems**
In time-multiplexed systems, there is only one monitor, and the two views are presented one to each eye alternately (one field per eye on interlaced screens). Some type of shuttering mechanism is required to prevent the

left eye from viewing the monitor while the right eye image is being displayed, and vice versa.

The currently most promising technology is based on liquid crystal shutters that are incorporated into glasses the user has to wear. Switching images has to occur fast enough to avoid flickering, and it has to be synchronized with the monitor. While older systems used synchronization cables for this purpose, modern devices (e.g. "CrystalEyes" of Stereographics Corp. [158, 167]) use infrared receivers in the glasses and transmitters at the monitor for synchronization. A number of users can use the system simultaneously, provided that each one wears a pair of glasses, and watch the monitor from within a wide area.

The critical factors of this technology are brightness and contrast of the image, synchronization, flickering, image separation, and equal illumination of both images. Also, the viewer has to keep his head strictly vertical. According to [145] approximately 8 % of the population cannot fuse stereo pairs. However, this deficiency can be improved with practice.

2. **Auto-Stereoscopic Systems**
   Auto-stereoscopic systems are systems that do not require the viewer to wear glasses of any kind. Also, the object or scene can be watched from arbitrary viewing angles by a number of persons simultaneously. Obviously, this kind of 3D displays is best suited for the Viewseum environment. Unfortunately, the technology is currently still immature and very expensive. We can distinguish between three kinds of auto-stereoscopic devices:

   (a) **Varifocal-Mirror Displays**
   Varifocal mirror displays combine a vibrating circular mirror with a CRT. The mirror is vibrated 30 times a second between convex and concave by a hi-fi speaker driven by a sine wave generator. A CRT display can be seen in the mirror (figure 4.3; from [145]). Vibrations of the mirror are synchronized with the display of the object on the CRT so that each point on an object is reflected from the mirror at a position corresponding to the depth of that point. Thus, the image has to be redisplayed 30 times a second, limiting resolution. On the other hand, the observer can change position to see the image from a different viewpoint and no special glasses are necessary. The technique has been used successfully in medical imaging [145].

   (b) **Laser Systems**
   The *Omniview* of Texas Instruments [184] uses laser beams that "write" on a translucent double-helix disk that is mounted on a rotating shaft. When properly synchronized, any point within a cylinder volume can be highlighted by the lasers. The observer's eye fuses the discrete points of light to 3D objects. Only prototypes are available at this time, for prices of around $\$200,000$.

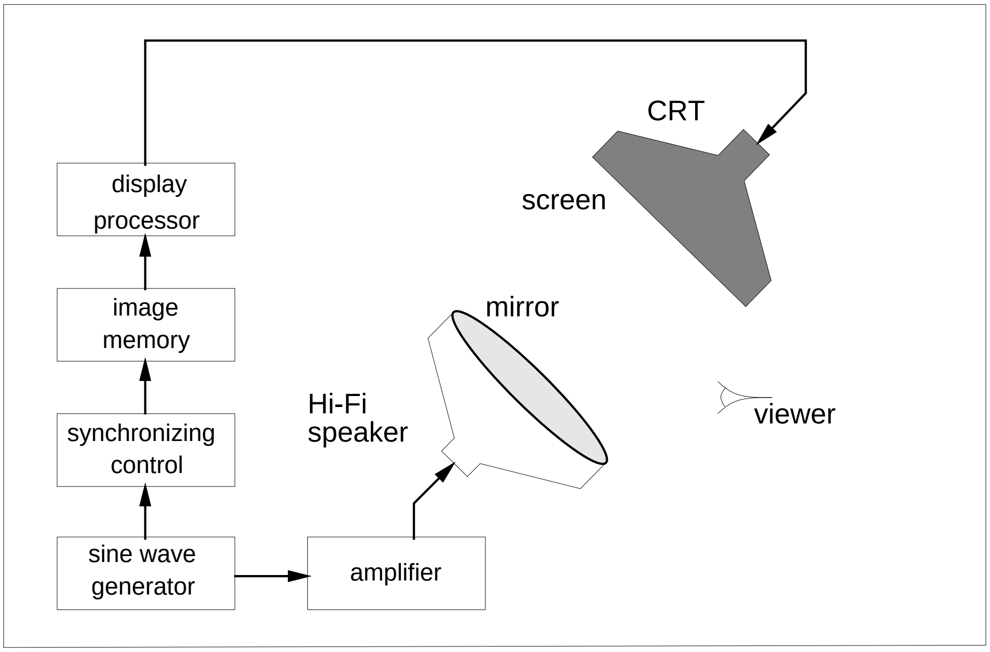

Figure 4.3: Block Diagram of Varifocal Mirror Display System

(c) **Auto-Stereoscopic Displays**
New liquid crystal displays developed by Dimension Technologies, Inc. [39, 158] use tiny lenses to let the viewer's eyes see different pixels at the same location. No glasses are required, however, the user's head has to be within a certain area in front of the display. A similar technique is currently being developed for CRT's in combination with LCD shutters [175].

4.3.1.6 Eye Tracking

A technology called *Eye Tracking* makes the human eye a computer input device by determining the direction in that the user looks. With a response time of 20 to 30 milliseconds the eye could be a very fast input device. A number of attempts have been made to utilize this (a short overview can be found in [174]):

Of course, one of the first organizations to show a serious interest in eye-tracking technology was the U.S. Air Force, and used it in flight simulators for fighter jets. Basically, a special camera "locks" on the eye, emits a beam of light onto its cornea and records the direction in which light is reflected. The resolution is high enough to determine on which word on a computer screen the user is focusing on.

At Texas A&M University, the user has to wear a set of eyeglasses equipped with infrared-emitting diodes and phototransistors. The position of the eye determines the amount and intensity of the light emitted to the phototransistors.

Sentient Systems Technology of Pittsburgh, PA, developed the "Eye-Typer 300" intended to be used by handicapped persons. It lets you enter information by focusing your eyes for a specified period of time on a certain point. The system sends the input to the computer over a standard RS232C interface, and sends feedback to the user using an embedded speech synthesizer.

Another system called OASIS (Ocular Attention Sensing Interface System) uses a combination of eye and voice input to present information without the users having to take their hands off, e.g., the controls of an airplane or car.

As a conclusion, the technology is new, immature and expensive. Also, the users have to wear glasses or helmets or have to stand in certain positions without moving their heads. On the other hand, eye tracking is an extremely fast yet natural user interface. In the Viewseum environment, the user could almost subconsciously select between items (e.g. pictures) presented on the screen and thus influence the course of events.

4.3.1.7 Gesture Recognition

Another experimental user interface technique is that of gesture recognition, especially hand gesture recognition.

Takahashi and Kishino [172] performed real-time hand gesture recognition experiments with a DataGlove (see figure 4.1 on page 132), in order to recognize the 46 gestures of the Japanese kana manual alphabet (similar to the gestures used by the deaf). The experiments showed that about 30 gestures could be distinguished with their coding method involving joint angle and hand orientation measurements.

Simple gestures like raising the left or the right hand should be distinguishable more easily, e.g. by infrared motion detectors or light beams.

4.3.1.8 Voice Recognition

While voice output seems to be a solved problem, voice recognition still isn't. However, there are advances: Kurzweil Applied Intelligence in Waltham, MA, has produced a 5000- to 10000-word voice recognition system that consists of special hardware and software for use with a 80386-type personal computer [174]. Unfortunately, the system has to be trained to recognize individual users' voices.

In the Viewseum, a large number of different users will communicate with the exhibits. Therefore, speech recognition must be speaker-independent. On the other hand, the set of words to be recognized is relatively small. Siemens has developed a prototype of a speaker-independent word recognizer based on phoneme recognition. The system chooses between a number of generated hypotheses and recognizes about 98 % [159]. Unfortunately, a lot of hardware (2-3 boards) is needed to recognize speech in real time.

4.3.2 n-dimensional Movies

The *Raster Image Document Manager* (section 2.3.5.3, page 93) of Hyper-G allows the display of images that are larger than the workstation screen. In that case the user can shrink the image (with loss of information) or scroll horizontally and/or vertically. This feature allows to incorporate some panorama (360°) images, creating the illusion of rotating a camera on the spot.

To some extent the opposite problem is to rotate a (virtual) camera *around* an object, e.g. a statue, to get views from all sides. Here the solution is to take images from a number of angles and compose a *Digital Movie* representing a flight around the object. As the *Digital Movie Document Manager* allows to play digital movies back and forth if a "backward channel" is supplied (section 2.3.5.5, page 95), the user gets the illusion of being able to move around the object and look at it from all sides.

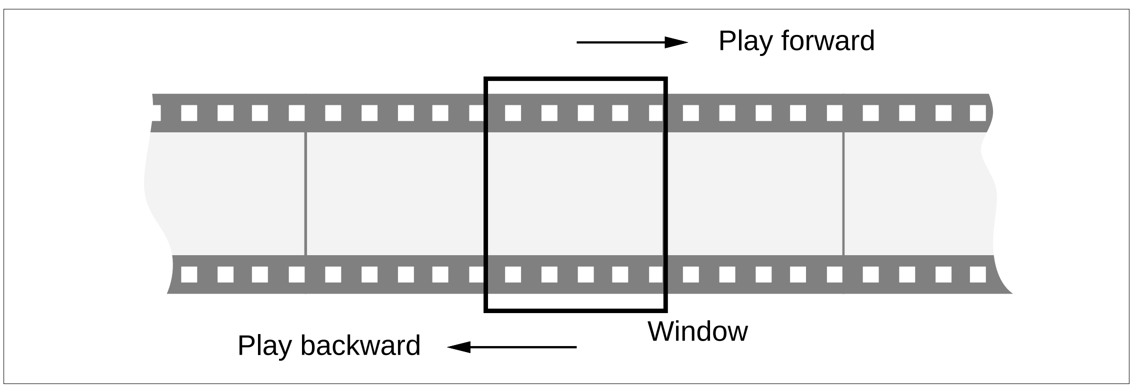

Figure 4.4: The 1-Dimensional Digital Movie

Figure 4.4 illustrates the concept of such a "1-dimensional" movie, where the user can move a "window" over the movie in one dimension. When moving the window forward, the "forward channel" (i.e. the differential images for the step from frame F_i to F_{i-1}) is used, when moving the window backward, the "backward channel" (i.e. the differential images for the step from frame F_i to F_{i-1}) is used. Alternatively, the movie may have been recorded using full images for each frame.

Things get more complicated if the user wants to see the object from above. In other words, the camera moves on a sphere surface during recording, creating a 2-dimensional digital movie. Now, every frame has not only a predecessor and successor, but is stored in a matrix with frames to the left, right, up, and down (figure 4.5). When replaying the movie, the user may step in either direction.

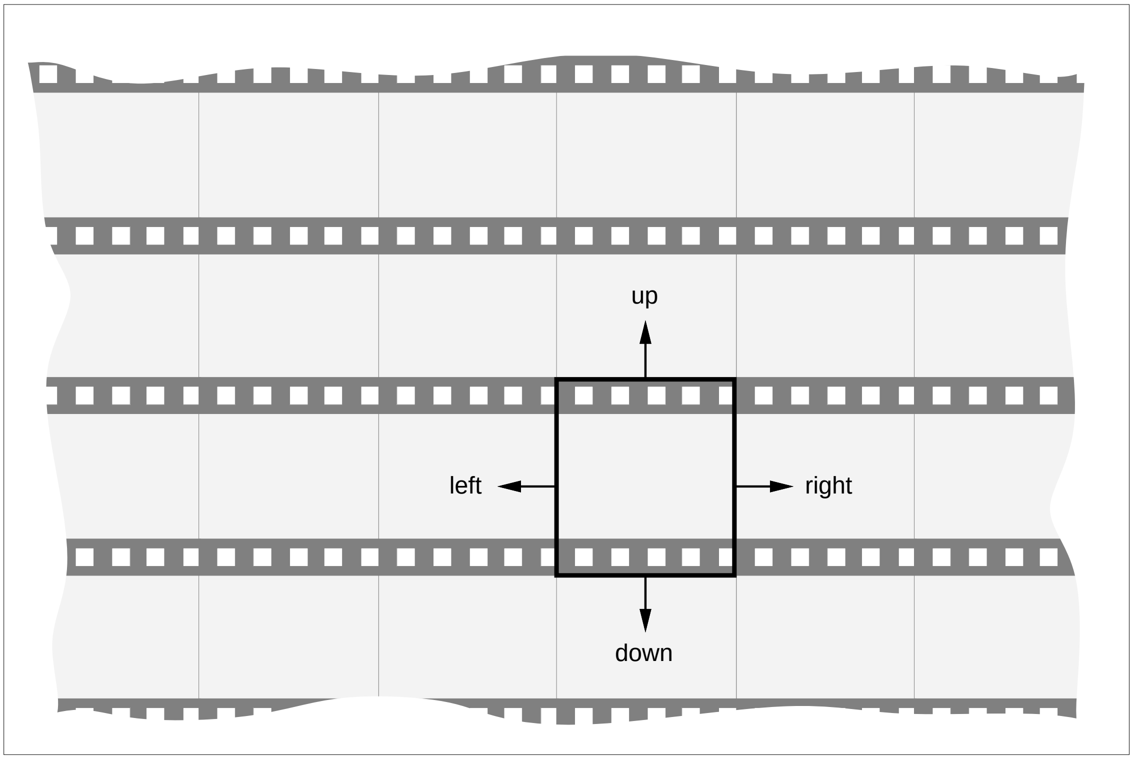

Figure 4.5: The 2-Dimensional Digital Movie

More dimensions may be added by allowing the object itself to move (e.g. a simulation of a running car engine), not restricting the camera position to a sphere, or special camera effects [109]. This leads us to the generic n-dimensional movie.

One big problem with n-dimensional movies is the exploding storage space required for each additional dimension. The problem can be reduced by the approach shown in figure 4.6 for a 2-dimensional movie: Not all frames $F_{i,j}$ are supplied, only a grid, e.g. 5 frames wide, is available. This essentially means that the user's navigation is a bit limited, however, there are still enough images to guarantee a smooth motion in any direction.

Applications of n-dimensional movies include visualizations of 3D objects, moving objects, panorama views of rooms and landscapes, and the like. Note that n-dimensional movies cannot be implemented using laser discs, because of their high access time to frames not in sequential order.

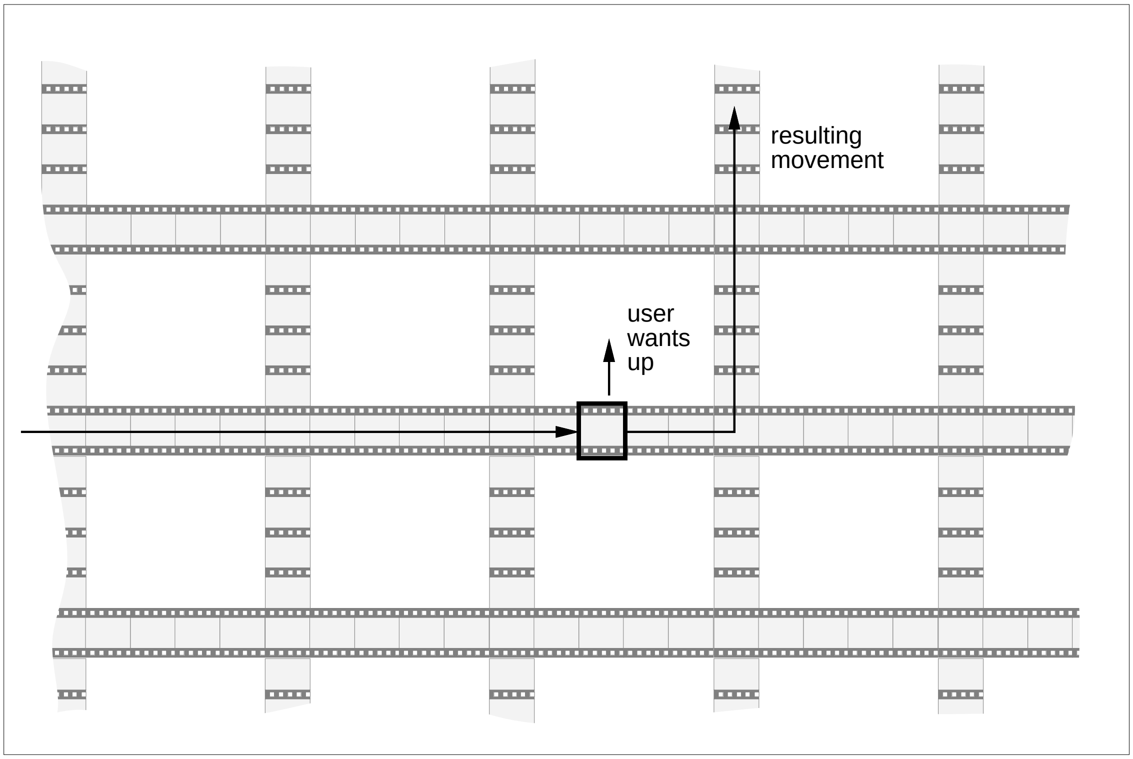

Figure 4.6: A 2-Dimensional Digital Movie Grid

4.3.3 The Interactive Movie

The *Interactive Movie* is a concept even more interactive than n-dimensional movies described in above section. Unlike a conventional (linear) movie, the contents of an interactive movie are not pre-determined, but may be different at any time the user watches the interactive movie.

The course of events, the appearance of objects and other parameters of an interactive movie may be influenced by:

- User interaction
- Random numbers
- Data from sensors
- Interaction with other interactive movies or programs

Depending on the story, an interactive movie may actually be one of 100,000 possible movies. This leads to fundamental properties of interactive movies:

- An interactive movie cannot be watched off a film or video tape. It has to be executed on a computer.

- Theoretically, interactive movies can be implemented as n-dimensional movies (see above section). However, an interactive movie with m frames per sequence and n decisions between k sequences each would require $m \times k^n$ frames, which is too much for a reasonable interactive movie.
- Interactive movies are best implemented using the *Hyper-Animation* concept (see chapter 3) by defining appropriate Hyper-Animation links between sequences of the interactive movie.

In the Viewseum environment, an interactive movie will be seen by the user as a computer monitor with high-resolution graphics. It will react to the user's presence (detected by one of the sensors described in section 4.3.1.4 on page 130). Communication with the user will probably take place using a touch screen (section 4.3.1.2, page 129), speech synthesis and perhaps speech recognition (section 4.3.1.8, page 136). With the use of ID cards, the interactive movie might even recognize its observer, choose the appropriate language and interaction level, etc.

The interactive movie is not totally interactive, however. Every possible sequence has to be prerecorded in advance. Only the combination and succession of sequences is determined at run time. Interactive movies are not as flexible as virtual realities (discussed in the next section); on the other hand, no expensive hardware is required ("poor man's virtual reality"). Still, interactivity is high when compared to similar systems found in museums nowadays, and image quality is much higher as in today's virtual reality systems. Also, users do not have to wear special equipment, which would be rather inconvenient in a museum environment.

4.3.4 Virtual Objects – Virtual Reality

"Virtual Reality", "Artificial Reality", "Cyberspace" are buzzwords of these days. Papers on it may not only be found in almost any computer magazine (e.g., [14, 40, 41]), but it also draws the attention of the general (American) public, as evidenced by articles in *TIME* [43], *Scientific American* [55] and even on the front page of the *New York Times* [149].

Virtual reality uses some of the 3D user interfaces described in section 4.3.1.5 plus powerful and expensive 3D workstations, and enables persons to move in a virtual world. There are multi-user systems; however, because of the need for a 3D workstation for every eye, the two-person variant ("RB2", for "reality built for two" [40]) cost \$ 430.000 (in 1989). Because the image quality is still to be considered poor, prices will remain high as pretensions increase. But there are efforts to bring virtual reality down to inexpensive hardware (PCs) [146], delivering even poorer quality (e.g. wire-frame rendering).

Serious applications include remote control of robots planned by NASA [149] (the user's motion data is gathered by the sensors and transmitted to the robot), and mission planning.

In the Viewseum, visitors could "experience" and "interact with" 3D objects that are virtually exhibited in the Viewseum, e.g. statues, living dinosaurs, molecules, the solar system, etc. To achieve this, the document type "3D object" has to be incorporated into Hyper-G, i.e. a document manager capable of displaying such an object on one of the 3D output devices described in section 4.3.1.5.2 on page 131 and interacting with it by means of 3D input devices (section 4.3.1.5.1, page 130) has to be supplied.

In addition, a user interface metaphor suitable for virtual reality has to be developed. Once again, XEROX PARC has unveiled a new user interface, one that incorporates color and real-time, three-dimensional, interactive animation. It is called *Information Visualizer* [20, 27], is implemented on Silicon Graphics platforms, and consists of an environment of rooms, perspective walls [102], data sculptures, and floating trees that the user can view from any side and wander around in. It is believed that this interface eases the managing of large quantities of information.

Bibliography

[1] Adobe Systems I. : *PostScript Language Reference Manual.* Addison-Wesley, Reading, Mass., 1985.

[2] Agha G. : "An Overview of Actor Languages". *ACM SIGPLAN Notices*, 21(10):58–67, October 1986.

[3] Andersen M. H., Nielsen J., and Rasmussen H. : "A Similarity-based Hypertext Browser for Reading the UNIX Network News". *Hypermedia*, 1(3):255–265, 1989.

[4] Andrews K. : *Distributed EDEN – Object-Oriented Design of the EDEN Kernel as a Distributed System.* Master's thesis, Technical University Graz, Austria, June 1991.

[5] Angstmann R. : *Implementation einer elektronischen Enzyklopädie.* PhD thesis, University of Karlruhe, Germany, 1978. In German.

[6] ANSI.: *Computer Graphics – Programmer's Hierarchical Interactive Graphics System (PHIGS); Functional Description, Archive File Format, Clear-Text Encoding of Archive File; ANS X3.144.* ANSI, September 1988.

[7] Apple Computer, Inc.: *Audio Interchange File Format: "AIFF". A Standard for Sampled Sound Files. Version 1.2.* Apple Computer, Inc., June 1988.

[8] Baran N. and Linderholm O. : "Fast New Systems from NeXT". *Byte*, 15(12):165–168, November 1990.

[9] Begeman M. L. and Conklin J. : "The Right Tool for the Job". *Byte*, 13(10):255–266, October 1988.

[10] Benest I. D.: "A Hypertext System with Controlled Hype". In *Proc. of the Hypertext II Conference, York*, 1989.

[11] Blaschitz P., Fellner W. D., Hofbauer A., Jones W., Kandler G., Kappe F., Kessler M., Klinger E., Rapatz K., and Yaganaki N. : *VTX EDEN – User's Manual.* INFONOVA Ges.m.b.H., Graz, Austria, 1990.

[12] Bowers R. A. : "The Oxford English Dictionary on Compact Disc". *Electronic and Optical Publishing Review*, 8(2):88–91, June 1988.

[13] Brown H. : "Standards for Structured Documents". *The Computer Journal*, 32(6):505–514, 1989.

[14] Burgess P. : "Head of VPL Research Virtually Makes His Own World". *MacWEEK*, 38–39, August 2, 1988.

[15] Bush V. : "As We May Think". In Lambert S. and Ropiequet S. (editors), *CD ROM – The New Papyrus*, pages 3–20, Microsoft Press, 1986.

[16] Bush V. : "As We May Think". *The Atlantic Monthly*, 176(1):101–108, July 1945.

[17] Byte. : "The 1990 Byte Awards". *Byte*, 16(1):147–165, January 1991.

[18] Campbell G., DeFanti T. A., Frederiksen J., Joyce S. A., Leske L. A., Lindberg J. A., and Sandin D. J. : "Two Bit/Pixel Full Color Encoding". *ACM Computer Graphics (Proc. SIGGRAPH '86)*, 20(4):215–223, August 1986.

[19] Card S. K., Morgan T. P., and Newell A. : *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, N.J., London, 1983.

[20] Card S. K., Robertson G. G., and Mackinlay J. D. : "The Information Visualizer: An Information Workspace". In *Proc. CHI '91: Human Factors in Computing Systems*, pages 181–188, ACM Press, New York, May 1991.

[21] Carlson P. A. : "The Rhetoric of Hypertext". *Hypermedia*, 2(2):109–132, 1990.

[22] Carlson P. A. : "Square Books and Round Books: Cognitive Implications of Hypertext". *Academic Computing*, 16–19; 26–31, April 1990.

[23] Carson G. S. : "Graphics, Networking, and Distributed Computing". In *Proc. Workshop on Graphics and Communications (ESPRIT Project 2463 - ARGOSI), Breuberg, Germany*, Springer, October 1990.

[24] Cavigioli C. : "Image Compression: Spelling Out the Options". *Advanced Imaging*, 64–85, October 1990.

[25] Cavigioli C. : "JPEG Compression: Spelling Out the Options". *Advanced Imaging*, 44–48, March 1991.

[26] Clark D. : "C-Cube's JPEG Image Compression: Where's It Headed?". *Advanced Imaging*, 50–71, September 1990.

[27] Clarkson M. A. : "An Easier Interface". *Byte*, 16(2):227–282, February 1991.

[28] Commodore Business Machines.: *Amiga ROM Kernel Reference Manual: EXEC. Technical Reference Series*, Addison-Wesley, Reading, Massachusetts, 1985.

[29] CompuServe.: *Graphics Interchange Format (GIF) – A standard defining a mechanism for the storage and transmission of raster-based graphics information.* CompuServe Inc., June 1987.

[30] Conklin J. : "Hypertext: An Introduction and Survey". *IEEE Computer*, 20(9):17–41, September 1987.

[31] Coulouris G. F. and Dollimore J. : *Distributed Systems: Concepts and Design. International Computer Science Series*, Addison-Wesley, Reading, Mass., 1988.

[32] Cox B. J. : *Object Oriented Programming – An Evolutionary Approach.* Addison-Wesley, Reading, Mass., 1986.

[33] Crowston K. and Malone T. W. : "Intelligent Software Agents – Using AI techniques in groupware has the potential to dramatically alter the way we organize our work". *Byte*, 13(13):267–272, December 1988.

[34] Davenport T. and Vellon M. : *Tag Image File Format (TIFF) – Rev. 4.0.* Aldus Corp. & Microsoft Corp., April 1987.

[35] Davies G., Maurer H., and Preece J.: *Presentation Metaphors for a very large Hypermedia System.* IIG Report 282, IIG, Graz University of Technology, Austria, April 1990. To appear in: Journal of Micro Computer Applications 14(2), 1991.

[36] Digital Equipment Corporation. : *Compound Document Architecture Manual, Ultrix Version 3.0, Order Number AA-LY28A-TE.* DEC, November 1988.

[37] Digital Research. : *GEM Programmer's Guide.* Digital Research, 1985.

[38] Digital Research. : *GEM Programmer's Guide.* Digital Research, 1985.

[39] Dimension Technologies, Inc. : "New Display Produces Stereoscopic 3-D Images Without Glasses". *Jandel Scientific Newsletter*, 4(2):2, 1990.

[40] Ditlea S. : "Inside Artificial Reality". *PC/Computing*, 91–102, November 1989.

[41] Eglowstein H. : "Reach Out and Touch Your Data". *Byte*, 15(7):283–290, July 1990.

[42] Ellis C. A., Gibbs S. J., and Rein G. L. : "Groupware: Some Issues and Experiences". *Communications of the ACM*, 34(1):39–58, January 1991.

[43] Elmer-Dewitt P. : "Through the 3-D Looking Glass". *TIME*, 65–66, May 1, 1989.

[44] Engelbart D. C. : "A Conceptual Framework for the Augmentation of Man's Intellect". In Howerton P. D. and Weeks D. C. (editors), *Vistas in Information Handling, Vol. 1*, pages 1–29, Spartan Books, Washington D.C., 1963.

[45] Engelbart D. C. and Lehtman H. : "Working Together – The human system and the tool system are equally important in computer-supported co-operative work". *Byte*, 13(13):245–252, December 1988.

[46] Fedida S. : "An Interactive Information Service for the General Public". In *Proc. European Computing Conference on Communication Networks*, pages 261–282, 1975.

[47] Fellner W. D., Andrews K., Kappe F., and Leitner H. : *Distributed EDEN – A Distributed Graphics Editing Environment.* Technical Report 9104, Memorial University of Newfoundland, St. John's, NF, Canada, May 1991.

[48] Fellner W. D. and Kappe F. : *EDEN – A General-Purpose Graphics Editor Environment.* IIG Report 266, IIG, Graz University of Technology, Austria, February 1988.

[49] Fellner W. D. and Kappe F. : "EDEN – An Editor Environment for Object-Oriented Graphics Editing". In Vandoni C. E. and Duce D. A. (editors), *Proc.* Eurographics '90*, Montreux, Switzerland*, pages 425–437, Eurographics Association, Elsevier Science Publishers B.V., The Netherlands, September 1990.

[50] Fellner W. D. and Kappe F. : "PIC – A Metafile Format for Distributed Graphics Applications". In *Proc. Workshop on Graphics and Communications (ESPRIT Project 2463 - ARGOSI), Breuberg, Germany*, Springer, October 1990.

[51] Fellner W. D., Kappe F., and Schaeffler J. : *CGI' Interface Version 1.0 (based on CGI* 2^{nd} *DP).* IIG Report 292, IIG, Graz University of Technology, Austria, November 1990.

[52] Fiderio J. : "A Grand Vision – Hypertext mimics the brain's ability to access information quickly and intuitively by reference". *Byte*, 13(10):237–244, October 1988.

[53] Fisher S. S. and Tazelaar J. M. : "Living in a Virtual World". *Byte*, 15(7):215–221, July 1990.

[54] Floyd R. W. and Steinberg L.: "An Adaptive Algorithm for Spatial Gray Scale". *SID 75 Int. Symp. Dig. Tech. Papers*, 36–37, 1975.

[55] Foley J. D. : "Interfaces for Advanced Computing". *Scientific American*, 127–135, October 1987.

[56] Frei H. P. and Stieger D. : "The Retrieval View, a Component of the Document Architecture". In Maurer H. (editor), *Proc. of Hypertext/Hypermedia '91, Graz, Austria*, pages 99–108, Springer, IFB 276, May 1991.

[57] Frisse M. : "From Text to Hypertext". *Byte*, 13(10):247–253, October 1988.

[58] Gloor P. A. : *Hypermedia-Anwendungsentwicklung – Eine Einführung mit Hypercard Beispielen. Leitfäden der angewandten Informatik*, Teubner, Stuttgart, 1990. In German.

[59] Grudin J. : "Perils and Pitfalls – To succeed with Groupware, you must first clear these hurdles". *Byte*, 13(13):261–264, December 1988.

[60] Halasz F. G. : "Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems". *Communications of the ACM*, 31(7):836–852, July 1988.

[61] Hammond N. and Allison L. : "The Travel Metaphor as Design Principle and Training Aid for Navigating Around Complex Systems". In Diaper D. and Winder R. (editors), *People and Computers III*, Cambridge University Press, 1987.

[62] Hardwick A. : "Error Correction Codes – Key to Perfect Data". In Lambert S. and Ropiequet S. (editors), *CD ROM – The New Papyrus*, pages 73–83, Microsoft Press, 1986.

[63] Harney K., Keith M., Lavelle G., Ryan L. D., and Stark D. J. : "The i750 Video Processor: A Total Multimedia Solution". *Communications of the ACM*, 34(4):64–78, April 1991.

[64] Heckbert P. : "Color Image Quantization for Frame Buffer Display". *Proc. SIGGRAPH '82, ACM Computer Graphics*, 16(3):297–307, July 1982.

[65] Helander M. (editor). : *Handbook of Human-Computer Interaction.* Noth-Holland Publ. Co., Amsterdam, 1990.

[66] Hewitt C. : "Viewing Control Structures as Patterns of Passing Messages". *Artificial Intelligence*, 8(3):323–364, June 1977.

[67] Hiltz S. R. and Turoff M. : "The Evolution of User Behavior in a Computerized Conferencing System". *Communications of the ACM*, 24(11):739–751, November 1981.

[68] Hiltz S. R. and Turoff M. : "Structuring Computer-Mediated Communication Systems to Avoid Information Overload". *Communications of the ACM*, 28(7):680–689, July 1985.

[69] Holweg G. and Huber F. : "Creating Highly Interactive Lesson Material or Learning by Doing". In Dalton D. W. (editor), *Proc. 32nd Annual. Conference, Association for the Development of Computer-Based Instructional Material, San Diego, Ca.*, pages 184–191, November 1990.

[70] Huber F., Makedon F., and Maurer H. : "Hyper-COSTOC: A Comprehensive Computer-Based Teaching Support System". *Journal of Micro Computer Applications*, 12:293–317, 1989.

[71] Hütter M. : *Design und Implementierung eines Hypermedia Basis Systems.* Master's thesis, Technical University Graz, Austria, June 1991. In German.

[72] Irler W. J. and Barbieri G. : "Non-Intrusive Hypertext Anchors and Individual Colour Markings". In Rizk A., Streitz N., and André J. (editors), *Hypertext: Concepts, Systems and Applications; Proc. ECHT '90*, pages 261–273, Cambridge University Press, 1990.

[73] ISO. : *Information Processing Systems – Computer Graphics – Interfacing Techniques for Dialogs with Graphical Devices (CGI), Parts 1-6, ISO DIS 9636.* ISO, December 1989.

[74] ISO. : *Information Processing Systems – Computer Graphics – Metafile for the Storage and Transfer of Picture Description Information (CGM), ISO IS 8632.* ISO, August 1987.

[75] ISO. : *Information Processing Systems – Computer Graphics – Programmer's Hierarchical Interactive Graphics System (PHIGS), Parts 1-3, ISO DIS 9592.* ISO, October 1987.

[76] ISO. : *Information Processing Systems – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1), ISO DIS 8824.* ISO, July 1986.

[77] ISO. : *Information Processing Systems – Open Systems Interconnection – Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1), ISO DIS 8825.* ISO, July 1986.

[78] ISO. : *Information Processing Systems – Text and Office Systems – Office Document Architecture (ODA) and Interchange Format, Parts 1,2,4-8, ISO IS 8613.* ISO, 1989.

[79] ISO. : *Information Processing Systems – Text and Office Systems – Standard Generalized Markup Language (SGML), ISO IS 8879.* ISO, 1986.

[80] ISO. : *ISO/IEC JTC1/SC2/WG12 – Multimedia and Hypermedia information coding Expert Group (MHEG) – Coded Representation of Multimedia and Hypermedia Information – MHEG Working Document "S", Version 3.* ISO, November 1990.

[81] Johnson J., Roberts T. L., Verplank W., Smith D. C., Irby C., Beard M., and Mackey K. : "The Xerox Star: A Retrospective". *IEEE Computer*, 22(9):11–26, September 1989.

[82] Joyce E. : "A Three Dimensional Space Tablet". *PC Magazine*, 4(12), June 26 1984.

[83] Kant K. and Zucker S. W. : "Planning Collision-Free Trajectories in Time-Varying Environments: A Two-level Hierarchy". *The Visual Computer*, 3(5):304–313, March 1988.

[84] Kappe F. : *Design und Implementierung eines EDEN-basierenden Bildschirmtext-Editors.* Master's thesis, Technical University Graz, Austria, February 1988. In German.

[85] Kappe F. : *Picture Interchange Coding (PIC); Functional Specification and Encoding of Profile '2D'.* IIG Report 266, IIG, Graz University of Technology, Austria, February 1989.

[86] Kappe F. : "Spezielle Eigenschaften großer Hypermedia-Systeme". In Maurer H. (editor), *Proc. of Hypertext/Hypermedia '91, Graz, Austria*, pages 164–173, Springer, IFB 276, May 1991. In German.

[87] Kappe F. : "Unorthodoxe Anwendungen von Hypermedia-Systemen". In Wallmannsberger J. (editor), *Hypertext – State of the Art*, R. Oldenbourg, Vienna, Munich, 1991. In German. To appear.

[88] Kappe F. and Maurer H. : "Animation in Hyper-G – An Outline". In Haase V. and Zinterhof P. (editors), *Proc. Future Trends in Information Technology '90, Salzburg, Austria*, pages 235–248, Austrian Computer Society, R. Oldenbourg, Vienna, Munich, September 1990.

[89] Kappe F., Nagler G., Sammer P., and Sigl H. C. : *Picture Interchange Coding (PIC) Addendum 3 - Profile 'RAST'.* IIG Report, IIG, Graz University of Technology, Austria. In preparation.

[90] Knuth D. E. : *Computers and Typesetting. Vol. A: The TEXbook.* Addison-Wesley, Reading, Massachusetts, 1986.

[91] Knuth D. E. : *Computers and Typesetting. Vol. B: TEX: The Program.* Addison-Wesley, Reading, Massachusetts, 1984.

[92] Kuhlen R. : *Hypertext – Ein nicht-lineares Medium zwischen Buch und Wissensbank.* Springer, 1991. In German.

[93] Laub L. : "Information Delivery Systems". In Ropiequet S., Einberger J., and Zoellick B. (editors), *CD ROM – Optical Publishing*, pages 11–30, Microsoft Press, 1987.

[94] Laub L. : "What Is CD ROM?". In Lambert S. and Ropiequet S. (editors), *CD ROM – The New Papyrus*, pages 47–71, Microsoft Press, 1986.

[95] Laurel B. (editor). : *The Art of Human-Computer Interface Design.* Addison-Wesley, Reading, Massachusetts, 1990.

[96] le Gall D. : "MPEG: A Video Compression Standard for Multimedia Applications". *Communications of the ACM*, 34(4):46–63, April 1991.

[97] Linton M. A., Calder P. R., and Vlissides J. M. : "The Design and Implementation of InterViews". In *Proc. USENIX C++ Workshop, Santa Fe, New Mexiko*, November 1987.

[98] Linton M. A., Vlissides J. M., and Calder P. R. : "Composing User Interfaces with InterViews". *IEEE Computer*, 22(2):8–22, February 1989.

[99] Lion K. : "DAT's a Solution". *Byte*, 15(12):323–328, November 1990.

[100] Loviera G. and Kinstler D. : "Multimedia: DVI Arrives". *Byte*, 15(11, IBM Special Edition):105–108, 1990.

[101] Lozano-Peréz T. and Wesley M. A. : "An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles". *Communications of the ACM*, 22(10):560–570, October 1979.

[102] Mackinlay J. D., Robertson G. G., and Card S. K. : "The Perspective Wall: Detail and Context Smoothly Integrated". In *Proc. CHI '91: Human Factors in Computing Systems*, pages 173–179, ACM Press, New York, May 1991.

[103] Magnenat-Thalmann N. : "Multimedia, Virtual Reality and Computer Animation". In Maurer H. (editor), *Proc. of Hypertext/Hypermedia '91, Graz, Austria*, pages 1–17, Springer, IFB 276, May 1991.

[104] Magnenat-Thalmann N. and Thalmann D. : "Abstract Muscle Action Procedures for Human Face Animation". *The Visual Computer*, 3(5):290–297, March 1988.

[105] Magnenat-Thalmann N. and Thalmann D.: *Computer Animation: Theory and Practice.* Springer, Tokyo, Berlin, Heidelberg, New York, 1983.

[106] Magnenat-Thalmann N. and Thalmann D.: "The Direction of Synthetic Actors in the Film *Rendez-vous à Montréal*". *IEEE Computer Graphics & Applications*, 7(12):9–19, December 1987.

[107] Magnenat-Thalmann N. and Thalmann D. : "An Indexed Bibliography on Image Synthesis". *IEEE Computer Graphics & Applications*, 7(8):28–38, August 1987.

[108] Magnenat-Thalmann N. and Thalmann D. : "MIRANIM: An Extensible Director-Oriented System for the Animation of Realistic Images". *IEEE Computer Graphics & Applications*, 5(3):61–73, March 1985.

[109] Magnenat-Thalmann N. and Thalmann D. : "Special Cinematographic Effects with Virtual Movie Cameras". *IEEE Computer Graphics & Applications*, 6(4):43–50, April 1986.

[110] Makedon F. and Maurer H.: "COSTOC – Computer Supported Teaching of Computer Science". In *Proc. IFIP Conference on Teleteaching, Budapest*, pages 107–119, North Holland Publ. Co., Amsterdam, 1988.

[111] Manes S. : "Archives in Miniature". *PC Magazine*, 8(2):185–200, January 1989.

[112] Marchionini G. and Shneiderman B. : "Finding Facts vs. Browsing Knowledge". *IEEE Computer*, 21(1):70–80, January 1988.

[113] Maulsby D. L., Witten I. H., and Kittlitz K. A.: "Metamouse: Specifying Graphical Procedures by Example". *ACM Computer Graphics (Proc. SIGGRAPH '89)*, 23(3):127–136, July 1989.

[114] Maurer H. : *Bildschirmtext muß ein Erfolg werden.* IIG Report B42, IIG, Graz University of Technology, Austria, February 1984. In German.

[115] Maurer H. : *Bildschirmtextähnliche Systeme.* IIG Report B11, IIG, Graz University of Technology, Austria, 1981. In German.

[116] Maurer H.: "Hyper-G – Ein echtes Hypermedia System". In Wallmannsberger J. (editor), *Hypertext – State of The Art*, R. Oldenbourg, Vienna, Munich, 1991. In German.

[117] Maurer H. : *The Viewseum – An Introduction.* IIG Report 279, IIG, Graz University of Technology, Austria, April 1990.

[118] Maurer H., Roszenich N., and Sebestyen I. : "Videotex without Big Brother". *Electronic Publishing Review*, 4:201–214, 1984.

[119] Maurer H., Schinnerl W., and Tomek I. : "Kommunikation in einem Hypermedia-System". In Gloor P. A. and Streitz N. A. (editors), *Proc. of Hypertext/Hypermedia '90; IFB 249*, pages 124–133, Springer, Berlin, 1990. In German.

[120] Maurer H. and Soral G. : *HOTACT: HOmeTrainer And Computer Technology.* IIG Report 283, IIG, Graz University of Technology, Austria, August 1990.

[121] Maurer H. and Tomek I. : "Broadening the Scope of Hypermedia Principles". *Hypermedia*, 2(3):201–221, 1990.

[122] MAURER H. and TOMEK I. : "From Hypertexts to Hyperenvironments". *e & i*, 107(12):614–616, December 1990.

[123] MAURER H. and TOMEK I. : *Hypermedia Bibliography.* IIG Report 286, IIG, Graz University of Technology, Austria, November 1990. An updated version appears in: Journal of Micro Computer Applications 14(2), 1991.

[124] MAURER H. and TOMEK I. : "Hypermedia in Teleteaching". In *Proc. IFIF Congress WCCE '90*, pages 1009–1015, North Holland Publ. Co., Amsterdam, 1990.

[125] MAURER H. and TOMEK I. : "A Model for a Next-Generation Hypermedia System". In *Proc. ACM/CSC '91, Texas*, 1991.

[126] MAURER H. and TOMEK I. : "Some Aspects of Hypermedia Systems and their Treatment in Hyper-G". *Wirtschaftsinformatik*, 32(2):187–196, April 1990.

[127] MAURER H. and WILLIAMS M. : *Hypermedia Systems as Infrastructure for Museums.* IIG Report 279, IIG, Graz University of Technology, Austria, 1991. To appear in: Journal of Micro Computer Applications 14(2), 1991.

[128] MCAVINNEY P. : "Telltale Gestures: 3-D applications need 3-D input". *Byte*, 15(7):237–240, July 1990.

[129] MEYROWITZ N. K. : "InterMedia: The Architecture and Construction of an Object Oriented Hypertext/Hypermedia System and Applications Framework". In *Proc. OOPSLA 86, Portland, Oregon*, pages 186–201, September/October 1986.

[130] MOLINE J., BENIGNI D., and BARONAS J. (editors). : *Proc. Hypertext Standardization Workshop*, NIST Special Publication 500-178, January 1990.

[131] MÜHLHÄUSER M. : "Hyperinformation in Instructional Tools Environments". In NORRIE D. H. and SIX H. W. (editors), *Computer Assisted Learning: Proc. 3rd International Conference ICCAL'90*, pages 245–264, Springer, LNCS 438, 1990.

[132] MÜLNER H. : *Die Behandlung von Textdatenbeständen in einem Hypermediasystem.* PhD thesis, Technical University Graz, Austria, 1991. In German.

[133] MÜLNER H. : "A System of Inter-Active Encyclopaedias". In *Proc. 4th Austrian-Hungarian Informatics Conference, Budapest*, pages 181–190, John von Neumann Society for Computing Sciences, Hungary, 1989.

[134] NAGLER G. M. : *Integration digitalisierter Bilder in Editiersysteme.* Master's thesis, Technical University Graz, Austria, September 1990. In German.

[135] NELSON T. H. : "A File Structure for the Complex, the Changing, and the Indeterminate". In *Proc. 20th ACM National Conference*, pages 84–100, 1965.

[136] NELSON T. H. : *Literary Machines, Ed. 87.1. (1987).* The Distributors, South Bend, IN, 1981.

[137] NELSON T. H. : "Managing Immense Storage: Project Xanadu provides a model for the possible future of mass storage". *Byte*, 13(1):225–238, January 1988.

[138] NIELSEN J. : "The Art of Navigating through Hypertext". *Communications of the ACM*, 33(3):296–310, March 1990.

[139] NIELSEN J.: *Hypertext & Hypermedia.* Academic Press, San Diego, CA, 1990.

[140] NIEVERGELT J. : "Errors in Dialog Design and how to Avoid them". In NIEVERGELT, CORAY, NICOUD, and SHAW (editors), *Document Preparation Systems*, North Holland, 1982.

[141] NYE A. (editor). : *X Protocol Reference Manual.* Volume 0 of *The X Window System Series*, O'Reilly and Associates, Inc., 2^{nd} edition, 1990.

[142] NYE A. : *Xlib Programming Manual.* Volume 1 of *The X Window System Series*, O'Reilly and Associates, Inc., 1989.

[143] NYE A. (editor). : *Xlib Reference Manual.* Volume 2 of *The X Window System Series*, O'Reilly and Associates, Inc., 1989.

[144] OPPER S. : "A Groupware Toolbox: These products prove that personal computer groupware is real". *Byte*, 13(13):275–282, December 1988.

[145] OWCZARCZYK J. and OWCZARCZYK B. : "Evaluation of true 3D Display Systems for Visualizing Medical Volume Data". *The Visual Computer*, 6:219–226, July 1990.

[146] PAUSCH R. : "Virtual Reality on Five Dollars a Day". In *Proc. CHI '91: Human Factors in Computing Systems*, pages 265–269, ACM Press, New York, May 1991.

[147] PINS M. : "Extensions of the Color-Cell-Compression-Algorithm". In MAGNENAT-THALMANN N. and THALMANN D. (editors), *proc. Computer Animation '91, Geneva, Switzerland*, pages 241–251, Springer, 1991.

[148] POLHEMUS. : *3SPACE Isotrak – User's Manual.* Polhemus – A Kaiser Aerospace & Electronics Company, Colchester, Vermont, May 22, 1987.

[149] POLLACK A. : "What Is Artificial Reality? Wear a Computer and See". *The New York Times*, 138(47836):1, April 10, 1989.

[150] POSKANZER J. : *Extended Portable Bitmap Toolkit. Beta Test Distribution of Feb. 5.* 1991.

[151] Pueyo X. and Tost D. : "A Survey of Computer Animation". *Computer Graphics Forum*, 7(4):281–300, December 1988.

[152] Quarterman J. S. : *The Matrix – Computer Networks and Conferencing Systems Worldwide*. Digital Press, Bedford, MA, 1990.

[153] Rapatz K.: *Maßschneiderung von EDEN auf GEM-Kompatibilität*. Master's thesis, Technical University Graz, Austria, 1989. In German.

[154] Raymond D. R. and Tompa F. W. : "Hypertext and The Oxford English Dictionary". *Communications of the ACM*, 31(7):871–879, July 1988.

[155] Reeves W. T. : "Particle Systems – A Technique for Modeling a Class of Fuzzy Objects". *ACM Computer Graphics (Proc. SIGGRAPH '83)*, 17(3):359–376, July 1983.

[156] Reynolds C. W. : "Computer Animation with Scripts and Actors". *ACM Computer Graphics*, 16(3):289–296, July 1982.

[157] Robinson M. : "Through a Lens Smartly – MIT's Information Lens project has laid the foundation for intelligent assistants for your E-mail system". *Byte*, 16(5):177–187, May 1991.

[158] Robinson P. : "Stereo 3D". *Computer Graphics World*, 68–74, June 1990.

[159] Rumpf C., Harke U., Leiner U., and Niemöller M. : "Natürliche Sprache und Computer-Animation – Eine multi-mediale Dialogoberfläche". In Maurer H. (editor), *Proc. of Hypertext/Hypermedia '91, Graz, Austria*, pages 238–248, Springer, IFB 276, May 1991. In German.

[160] Sammer P. : *Erweiterungen des klassischen computerunterstützten Unterrichts*. PhD thesis, Technical University Graz, Austria, May 1989. In German.

[161] Scheifler R. W. and Gettys J. : "The X Window System". *ACM Transactions on Graphics*, 5(2):79–109, April 1986.

[162] Schütt H. A. and Streitz N. A. : "A Hypermedia Engine Based on a Relational Database Management System". In Rizk A., Streitz N., and André J. (editors), *Hypertext: Concepts, Systems and Applications; Proc. ECHT '90*, pages 95–108, Cambridge University Press, 1990.

[163] Shneiderman B. : "User Interface Design for the Hyperties Electronic Encyclopedia". In *Proc. of Hypertext '87, TR88-013*, pages 199–205, University of North Carolina, Dept. of Computer Science, March 1988.

[164] Smith B. : "Personal Iris: The Dream Maker". *Byte*, 15(7):174–178, 1990.

[165] Stallman R.: *GNU Emacs Manual. 5th Edition, Emacs Version 18 for Unix Users*. October 1986.

[166] Stein R. M.: "Browsing through Terabytes – Wide-area information servers open a new frontier in personal and corporate information services". *Byte*, 16(5):157–164, May 1991.

[167] StereoGraphics Corp. : "3-D Viewing in Two Styles". *Byte*, 16(1):50, 1991.

[168] Streitz N. A., Hannemann J., and Thüring M. : "From Ideas and Arguments to Hyperdocuments: Travelling through Activity Spaces". In *Proc. Hypertext '89*, pages 343–364, ACM Press, New York, November 1989.

[169] Stroustrup B.: *The C++ Programming Language. Addison-Wesley Series in Computer Science*, Addison-Wesley, Reading, Massachusetts, 1987.

[170] Stubenrauch R.: "Touring a Hyper-CAI System". In Maurer H. (editor), *Computer Assisted Learning. Proc. of* 2^{nd} *Int. Conf., ICCAL '89, Austin, TX, USA*, pages 541–551, Springer, LNCS 360, May 1989.

[171] Stubenrauch R. and Tomek I. : "Navigation and Browsing Through Lessons in a Hyper-CAI System". In *Proc.* 7^{th} *International Conference on Technology and Education, Brussels, Belgium*, pages 125–127, CEP Consultants Ltd., March 1990.

[172] Takahashi T. and Kishino F. : "Hand Gesture Coding Based on Experiments using a Hand Gesture Interface Device". *ACM SIGCHI Bulletin*, 23(2):67–74, April 1991.

[173] Tanimoto S. L. and Runyan M. S. : "PLAY – An Iconic Programming System for Children". In Chang S., Ichikawa T., and Ligomenides P. A. (editors), *Visual Languages*, pages 191–205, Plenum Publishing Corporation, 1986.

[174] Tello E. R.: "Between Man and Machine". *Byte*, 13(9):288–293, September 1988.

[175] Theoharis T. A., Travis A. R. L., and Wiseman N. E.: "3D Display: Synthetic Image Generation and Visual Effect Simulation". *Computer Graphics Forum*, 9(4):337–348, December 1990.

[176] Tomek I., Khan S., Muldner T., Nassar M., Novak G., and Proszynski P.: "Hypermedia – Introduction and Survey". To appear in: Journal of Micro Computer Applications 14(2), 1991.

[177] Tomek I. and Maurer H. : "The Analyst as a Staring Point for a Hypermedia System". To appear in: Journal of Micro Computer Applications 14(2), 1991.

[178] Tost D. and Pueyo X.: "Human Body Animation: A Survey". *The Visual Computer*, 3(5):254–264, March 1988.

[179] WALLACE G. K. : "The JPEG Still Picture Compression Standard". *Communications of the ACM*, 34(4):30–44, April 1991.

[180] WELCH T. A. : "A Technique for High Performance Data Compression". *IEEE Computer*, 17(6):8–19, June 1984.

[181] WIENER R. S. and PINSON L. J. : *The C++ Workbook.* Addison-Wesley, Reading, Massachusetts, 1990.

[182] WIENER R. S. and PINSON L. J. : *An Introduction to Object-Oriented Programming and C++.* Addison-Wesley, Reading, Massachusetts, 1988.

[183] WILHELMS J.: "Toward Automatic Motion Control". *IEEE Computer Graphics & Applications*, 7(4):11–22, April 1987.

[184] WILLIAMS D. : "OnmiView – A Real-Time, Auto-Stereoscopic, Multiplanar 3-D Display System". *Byte*, 16(12):19, November 1990.

[185] WILLIAMS G. : "HyperCard". *Byte*, 12(14):109–117, December 1987.

[186] WINOGRAD T.: "Where the Action Is – Groupware brings clarity and simplicity to the coordination of human action". *Byte*, 13(13):256A–258, December 1988.

[187] WITTEN I. H. : *Principles of Computer Speech.* Academic Press, London, 1982.

[188] WITTEN I. H. and MAULSBY D. : "I, Metamouse". In *IIG Report 260*, pages 198–210, IIG, Graz University of Technology, Austria, June 1988.

[189] YANKELOVICH N., HAAN B. J., MEYROWITZ N. K., and DRUCKER S. M.: "Intermedia: The Concept and the Construction of a Seamless Information Environment". *IEEE Computer*, 21(1):81–96, January 1988.

[190] YANKELOVICH N., MEYROWITZ N. K., and VAN DAM A. : "Reading and Writing the Electronic Book". *IEEE Computer*, 18(10):15–30, October 1985.

[191] ZELTZER D. : "Motor Control Techniques for Figure Animation". *IEEE Computer Graphics & Applications*, 2(11):53–60, November 1982.