Cruising and Creating Hypertext with WHype

Michael Ledwidge
leddo@minnie.cs.su.oz.au

A thesis submitted in partial fulfilment of
the requirements for the degree of
Bachelor of Arts (Honours).

Basser Department of Computer Science,
University of Sydney

October 1993

Acknowledgments

Got here!!!
This tree owes its current shape to the following people and numerous others most of whom are, no doubt,sick of hearing about hyper-thingys.

Thanx muchly to

'WHyperactive' Bock,
'Hacker' Flower Clancey, 'Sing-a-long' Ellis,
'Vocabularian' Kaplan, the Jase Man, Smith,
Lord Langdale,
Betty, Katya,
??? Johnson & Fred de Club Basement,
'Manual' Maguire, 'Bush Tucker' Gardner, Felonious Forbes, 'Oh yeah' Jenkins,
Tyrone the Timekeeper, Bernie 'Bellows', Spiteri Van Gogh, Jaguar Johnny, Skunkhour,
Dr. Suess, Rob McCool, 'Doc' Matty,
and not forgetting
Associate Professor Bob Kummerfeld.

Peace, love, and mung beans

.M.

Abstract

Global information resources are not new. The Internet has been providing services for some time for those who had the time to learn how to use them. The World Wide Web project has been embraced as a revolutionary breakthrough in information technology because, for the first time, the resources of the Internet are available at the click of a mouse button. At present the potential of the Web is being hampered by the lack of 'authoring' facilities which would allow non-programmers to contribute their own material to an already-sizeable mass of global information. WHype is a lightweight distributed hypertext editor which permits local hypertext structures to be quickly created, modified, and linked to existing World Wide Web material. This paper uses the development of WHype as a basis for an analysis of the World Wide Web project and the implications that global information dissemination of this kind will have on society. This paper also looks at the evolving Hypertext Markup Language as a means by which this new technology can be exploited.

Contents

List of Figures

Part I

The World-Wide Web

Chapter 1

Introduction

1.1 Believe the hype

> 'The nirvana of personal computing is when end users can change their tools and build new ones without having to become professional-level programmers' Alan Kay [6]

It has already begun. Slowly but surely the advances being made in the field of computer science are beginning to leave their mark on society. This may sound like an naive observation. After all, computers have been unseparable from business, commerce, indeed most fields of human endevour, since the mid-80's. However, only very recently has this expanding sphere of influence begun to be viewed as relevant to everyone. Computer networks have remained the province of academics, researchers and the military for too long. In today's world the realisation is growing, that this resource, information itself, will be eventually widely available; not just to those involved with its development. The success of global information systems like gopher and WAIS has clearly demonstrated that there is a growing demand for on-line information services. The fact that, at present, the majority of computer users do not have access to such resources should not be taken as a sign that information technology will continue to remain the privilege of academics and researchers. It is only a matter of time before the information explosion taking place on the Internet is recognised and access is demanded by the general public.

At the same time, access to computer networks has not been subject to physical constraints alone. Even for those with access to global resources, the sheer complexity and varied nature of information services has proven to be an effective barrier to communication and information dissemination.

The *World-Wide Web* (WWW) project has brought a new dimension to information technology. A user can now explore most services that the Internet offers without having to deal with the intricacies of each separate system. The key to this flexibility is the use of hypertext as a navigational and contextual tool. Any two nodes of information can be connected through the Web

(regardless of operating system, storage mechanism, or format), provided the two locations are known and made public. This mechanism provides the end-user with a virtual information landscape that effectively conceals the protocol layers that coordinate the transfer of data. This 'web' of information bears little or no relation to the physical ordering of Internet data but presents its material in a package that more accurately reflects the cognitive needs of actual human users.

Some of the main difficulties involved with computer-related cognition can be related to a user's grasp of the interface. Only once the user is completely at ease with a particular type of interface can any system be used to its full potential. In the last decade, society has been sniffing around the edges of computer literacy with the increasing proximity. The diminishing cost of the personal computer and the growing acceptance of the graphical user interface (as popularised by Apple Macintosh and Microsoft Windows) has brought an increasing number of people into contact with computers. Software and control devices (ie. a mouse and keyboard) are familiar items to a growing segment of the population.

This growing familiarity with a particular interface design has had the added side effect of exposing many computer users to the concept of hypertext. Commercial software manufacturers have been quick to grasp the appeal of hypertext as an on-line help format for their applications.

Nevertheless, without the ability to quickly modify and extend the Web , it is easy to think of it merely as a more elaborate reference tool. This is certainly the most useful aspect of the Web at present but, as envisioned by its creators, it can also be much more. The ability to link personal hypertext structures into a global conglomeration is one that has the potential to irreversibly change the way information is stored and disseminated.

Whether or not this technology is exploited to its fullest potential is wholly dependent on social, political and economic factors that must be examined elsewhere. Information resources have never been accepted for their potential alone. Admittedly, the Web is still in its infancy but nevertheless its long term success will depend upon a broad spectrum of usage, reflecting not only the interests of the research community which brought it into being, but the community in general.

With the majority of Web material presently being provided by information technology developers and programmers, it is clear that this so-called global information structure has not yet reached any state of global significance. The problem lies along two fronts. Firstly the Internet is not accessible by the general public. The second, and more immediately addressable, problem is the lack of authoring tools aimed specifically at this new medium. As a result, most information has been converted by hand (or dedicated software) from existing formats into hypertext.

What is needed is, not so much a complex authoring tool that would cater to developers, but a tool that will allow novice users to interact with this

hyperspace environment.

Chapter 2

Non-linear communication

2.1 Definition

Any text which contains links to other texts is said to be *hypertext*. When the material being linked is not purely text-based, it is called *hypermedia*. Over the last few years, improved graphical displays and audio capabilities have encouraged the spread of multimedia resources and applications. Nevertheless, text-based communication remains the focus of this discussion. This is not only because traditional reading and writing methods form the basis of modern society but also because the all the ideas expressed here can be extended to included hypermedia.

2.2 Background

Hypertext is not a new concept, the first recorded mention of the notion of non-linearity in writing is attributed to Socrates. [10] However it was not until 1945 that Vannevar Bush [1] first proposed a theoretical machine, MEMEX, which would not only store vast quantities of information, could allow users to create 'information trails' through the data stored separately.

In 1965, Ted Nelson coined the phrase 'hypertext' in conjunction with his conceptual design for a global information system, Xanadu. The project aimed to combine 'all the world literature in one publicly accessible global online system'. [1] Both these projects have never moved beyond a design phase [2] although they have provided the impetus for a broad range of less ambitious hypertext systems of which Apple's Hypercard is probably the best known.

Over the last couple of years, hypertext has emerged as a valuable component of the graphic user interface. The immense popularity of personal computer

[1] Vannevar Bush was the Science Advisor to U.S. President Roosevelt during World War II.

[2] The Xanadu project was discontinued by Autodesk in 1992

operating systems like Apple Macintosh, Microsoft MS DOS/Windows, and to a lesser extent IBM's OS/2, has exposed a large number of people to hypertext. Commercial software companies have been quick to convert their help systems to this format. Such use has ensured that, while the phrase 'hypertext' is yet to enter the common vernacular, non-linear on-line text has become more and more familiar. What is this attraction to hypertext as a display medium?

2.3 Linear vs. non-linear

There have been enough discussions on the relative virtues of hypertext and conventional text to make it clear that, while non-linear representation of information is ideal in some domains (ie. on-line help screen references), it is arguably not well suited to other tasks such as unstructured information retrieval. There is no doubt, however, that the use of hypertext is an asset for computer interfaces. The intimacy that many people relish with paper-based text (ie. being able to fold a paperback novel up and stick it in a pocket) is simply not available with computer system interfaces. [3] Nevertheless hyperspace is a flexible environment in which information can be re-structured to suit the individual if need be.

The power of non-linear text is akin to a double-edged sword. Its greatest advantage, namely the ease with which one can move through the text is also the source of its biggest problem - navigation. The freedom within any large hypertext system is balanced by the difficulty in maintaining any sense of location within such a structure.

This is not so much of a problem when the user is *browsing*, moving without any specific goal through hyperspace, but it is obviously a major stumbling point when searching for certain information. The perceived structure of hypertextual material in the Web does not need to bear any resemblance to the physical layout of data. Nevertheless the presentation of original text is a factor of vital importance to its author. The temptation to exploit the freedom and flexibility provided by hyperlinks must always be tempered with an awareness of the target audience. With paper-based text, linearity is implicitly imposed on the text. Hypertext authoring requires greater responsibility on the behalf of authors themselves to ensure that their material is well structured.

Despite appearances, hypertext is not automatically a useful cognitive device. The majority of hypertext in existence today has been produced as part of commercial software or by information technology developers. It is misleading to judge the innate qualities of hypertext from the high quality of what is already available. [4] The World-Wide Web has been, so far, mainly used to represent well-structured material such as technical documentation and refer-

[3] Pen-based computers, such as the recently released Apple Newton, are only now addressing this issue.

ence material. Once authoring tools are widely available, it will be interesting to see how well less-structured information can be presented in a hypertext format. Producing a highly interconnected series of documents is not necessarily a guarantee that the material will be more easily understood. What is assured only is that browsing through the material, superficial analysis, will be made easier.

2.4 Education via hyperspace

Hypertext is a powerful educational tool. This statement is supported by the presence of hypertext help systems in most commercial applications. Hypertext offers a fresh individualistic view of text that allows the user to effectively filter the information being presented by selecting certain links ahead of others. For children, this medium offers more flexibility than conventional text, inspiring indivuals to follow personal modes of interaction with any given material.

Hypertext is not passive. Unlike television, this medium requires active participation from its audience. As more elaborate means are developed to utilise hypermedia, this participation could be increased dramatically from the present requirements of clicking mouse buttons. This aspect cannot be over-estimated in a society that appears to be actively seeking to simplify living in direct contrast to the state of its technology.

Infinitely flexible and highly connectable, the World-Wide Web is an ideal environment for expressing ideas. It is, however, not perhaps as relevant to the sciences as to other fields. The vision of this project is not limited to the technology that supports and maintains it. It is about the ability to create and explore the creations of others in a virtual space.

The World-Wide Web could be used as a valuable teaching aid. Rather than using conventional textbooks, teachers could lead their classes along cognitive trails through the Web, distinct to their individual teaching styles, which freely traverse the traditional boundaries of curriculum.

However, the WWW hyperspace will never be more than a gigantic library resource if the the public are not given both access and the means to contribute material.

2.5 Modelling thought

Could hypertext be, in some way, a valid attempt to map cognition in the same way that chaos theory is now attempting to model the infinite complexity of the natural world? Could the obvious appeal of present hypertext systems be somehow traced to a similarity between their non-linear structure and the undefined, but possibly, non-linear thought processes occurring

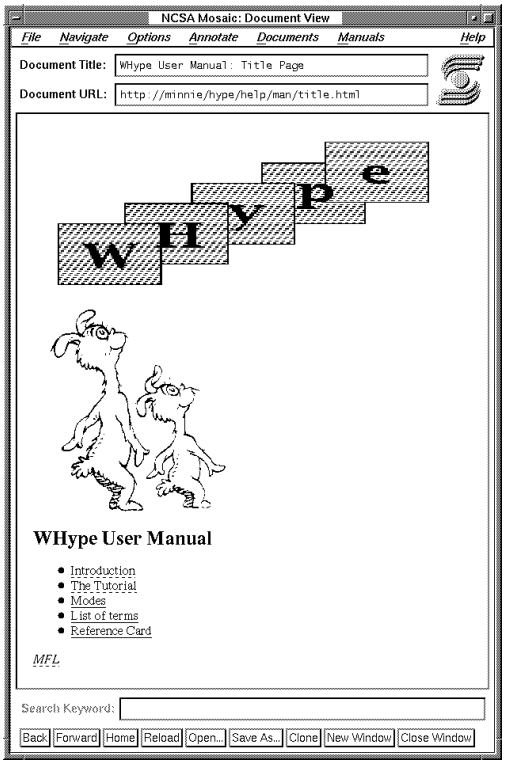


Figure 2.1: An example of educational hypermedia.

in a human brain? What cannot be denied is that there is more scope to human-computer interaction then can be expressed through linear methods.

The constraints of paper-based linear writing methods have always been undermined and stretched. The process of scanning a written page of text is such a 'natural' thing for the majority of literate people that the innate limitations are not often appreciated. Whether it is a UNIX technical manual or a Shakepearean play, non-linear devices are used to establish context. The former example uses underlining to indicate that a separate manual entry is devoted to that topic. The latter uses footnotes to give background information relating to historical and production notes. Both of these texts are presented,and can be interpreted, in linear fashion yet the contextual links embedded within them allow the material to be interpreted in a wider scope.

In general, people do not interact with the written word in a purely linear fashion. One has only to consider how reference material, like an encyclopedia, is accessed to be convinced of exactly how non-linear the process of reading can be. Even with material intended for linear access, such as a novel, this type cannot be imposed on the reader, merely suggested. There are an infinite number of ways to read it. Just as few people will read a book from start to finish in one sitting, few people never re-read certain material, either deliberately or simply to find where they left off. Writers have long devised means of working around the limitations of linear text. There is no reason to

suggest that the skills involved in producing literature will not eventually be used to exploit this new medium.

Similarly, there is no reason to suggest that linear text will be ever be superseded. Despite claims to the contrary [4] , traditional writing methods are too firmly embedded in human tradition to be easily abandoned. What is certain, however, is that hypertext will play an increasingly important role in human culture as paper-based resources are depleted and the computer becomes even less of a novelty.

2.6 A question of navigation

Moving within a hypertext system of any size, whether this is fifty pages of on-line help for a word processor, or the World-Wide Web itself, is a daunting prospect when trying to maintain a sense of location within the virtual information structure. The sheer volume contained within the Web gives a clear indication that hypermedia research should take this problem into consideration.

One approach to navigation is to take a local rather than global perspective. It is inconceivable that a single person could ever construct a mental map of the whole Web even at this early stage of development. Similarly, one should not assume that a user of a hypertext system will ever, even with assistance [5] , be able to visualise their progress through hyperspace, let alone the physical structure of the data. Instead users will need to be able to customise their information environment to reflect personal context and perspective. Hyperlinks are the means by which information can be re-structured to suit individual needs without changing any physical aspect of the data. What is needed, however, are the facilities to let users to do this. By building their own contextually-based maps of their environment, users will be able to make practical use of hyperspace.

At present, the World-Wide Web is still a novelty. The majority of users are involved in both building information sources and browsing others. Once this information infrastructure is familiar to enough people, it will, not doubt, encourage more sophisticated interaction.

In the process of setting up a local hypertext server, I spent a great deal of time traversing the Web looking for reference material and useful information sources. A personal hypertext system, consisting of little more than links to outside sources, was invaluable in establishing a good cognitive base for navigating the Web. I built up a framework of diverse links, essentially pages of lists, providing access to many, if not most, [6] of the public Web resources.

[4] In 1982, D.H. Jonassen took the bold step of proclaiming, 'in ten years or so the book as we know it will be as obsolete as is movable type today'

[5] XMosaic's Hotlist feature allows the user to record landmarks in their travels through the Web.

[6] Charting the entire Web is already virtually impossible since there is no formal regis-

In this light, WHYPE and other authoring prototypes will serve not only the user wishing to produce original hypermedia documents, but also the user primarily concerned with browsing the hyperspace for existing material. Only through the process of manipulating, and perhaps re-ordering, hyperlinks can the individual ever hope to orient him or herself completely. The process of creating something has always been the key to understanding it. A simple authoring tool, like WHype, provides the user with the power to create their own virtual landscape from new or existing material.

tration of sites.

Chapter 3

The World-Wide Web

3.1 Early threads

In March 1989, Tim Berners-Lee proposed a design for a new global information system that would serve the world-wide High Energy Physics community. This would allow physicists to exchange and share resources, primarily in hypertext form, while allowing the data to be stored in single locations. The system was not intended to exist in isolation. The aim of incorporating, and sharing information with, existing information systems was fundamental to the design. Based at CERN (the European Centre for Nuclear Energy Research), work began in earnest to develop the necessary software components. By November 1990, an initial prototype of the project had been implemented on the NeXT platform. In the initial documentation [12] , the designers emphasised the academic philosophy of 'information for all'. They demonstrated that use of the World-Wide Web did not involve a re-structuring of existing data management schemes but merely allowed greater access to the information.

3.2 Architecture

The World-Wide Web uses a client-server model (see Figure 3.1). The two components communicate via a simple protocol. The hypertext takes the form of plain text with additional markup to be interpreted by the client.

Both this protocol (HTTP) and the markup language (HTML) developed for the Web, have undergone revision throughout the course of this year. This paper, and WHype itself, utilise the original specifications that were in place until very recently. The revised specifications (HTTP2 and HTML+) are backwards compatible and consist mainly of new extensions. They will not be discussed here in any detail. [1]

[1] The HTTP2 and HTML+ specifications are available as working drafts.

3.2.1 Hypertext Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is the backbone of the Web infrastructure. Its lightweight nature allows a client to quickly retrieve information from any other Web site by communicating with that site's server. This communication involves the following phases. [2]

1. **Connection** The client initiates the protocol by making a TCP-IP connection on port 80 [2] to a designated host. The server on that host accepts the connection.

2. **Request** A request for a particular node is made by the client (in the form of 'ACTION object-description'). The original HTTP supports only one action 'GET'. This object description (a Uniform Resource Locator) refers to a specific data object and will be discussed in detail at a later stage.

3. **Response** The server sends back a byte stream of ASCII characters (or binary) depending on the content of the request. This will be either the requested material or an error message indicating why the request could not be fulfilled. Lines in textual material are delimited by an optional carriage return followed by a mandatory line feed character.

4. **Disconnection** The server disconnects when it has completed sending its response.

This basic protocol, despite the current revision of the HTTP specification, still handles the majority of Web traffic.

3.2.2 Clients

Clients are the end-user applications which permit access to the Web. These are currently referred to as 'browsers' since they essentially provide an environment for looking at existing material. These facilitate movement around hyperspace by traversing links and presenting each node of information to the viewer. The style of presentation is what presently differentiates the available clients. The more sophisticated clients can display hypermedia documents by calling external viewers and audio players when necessary.

The ever-growing number of Web clients has been responsible for the explosion of WWW usage that has taken place this year.

3.2.3 Servers

Along with information material, a WWW site requires software dedicated to passing this information onto other sites. This is the server (or daemon)

[2] The default World-Wide Web port.

,allowing files and directories in a local file system to be 'served' to clients worldwide. Since the server communicates with its clients via HTTP, it is often referred to as the *HTTP daemon* (HTTPD).

The server handles a request at a time. It is only active while processing a client request, shutting down once a message has been transmitted. Therefore it can remain inactive for long periods of time. Rather than waiting for client to make a connection, the server is typically only started when notified that a request is forthcoming. This notification can be provided by the internet daemon which monitors a host's ports for service requests such as ftp and telnet. When the internet daemon (inetd) notes a http request, it initiates the HTTPD to make the connection.

Whether or not a client request can be satisfied or not depends on a particular server's configuration file. Access to all public material on a site does not have to be automatically supplied. A simple rule file (examples of which are given in the next chapter) can restrict outside access to specific directories.

3.2.4 Uniform Resource Locators

The designers of the World-Wide Web envisaged the need for a generic syntax which 'can be used to refer to objects available through existing protocols and may be extended with technology.' [3] It is probably easiest to consider the *Uniform Resource Locator* (URL) as an extension of the full pathname syntax which includes format, Internet host address, and port number details (see Figure 3.2).

The format component of this syntax allows a WWW client to determine which protocol should be used to retrieve an object. The software which handles the interaction between the WWW and any information protocol other than HTTP is commonly referred to as a *gateway*. At present the following protocols are supported through the World-Wide Web.

- **HTTP** Hypertext Transfer Protocol as outlined above.
- **FTP** File Transfer Protocol. Any anonymous FTP site can be accessed through the Web via an FTP gateway.
- **News** Any Usenet newsgroup can be accessed through the local NNTP [3] server via a gateway.
- **Telnet** Telnet connections to most sites can be made using a specific gateway. Once a connection is established, a client typically opens a new login shell to handle the remote session.
- **Gopher** Gopher is a hierarchical global information system pre-dating the WWW. Gopher servers can be directly accessed through a specialised gateway.

[3] Network News Transfer Protocol

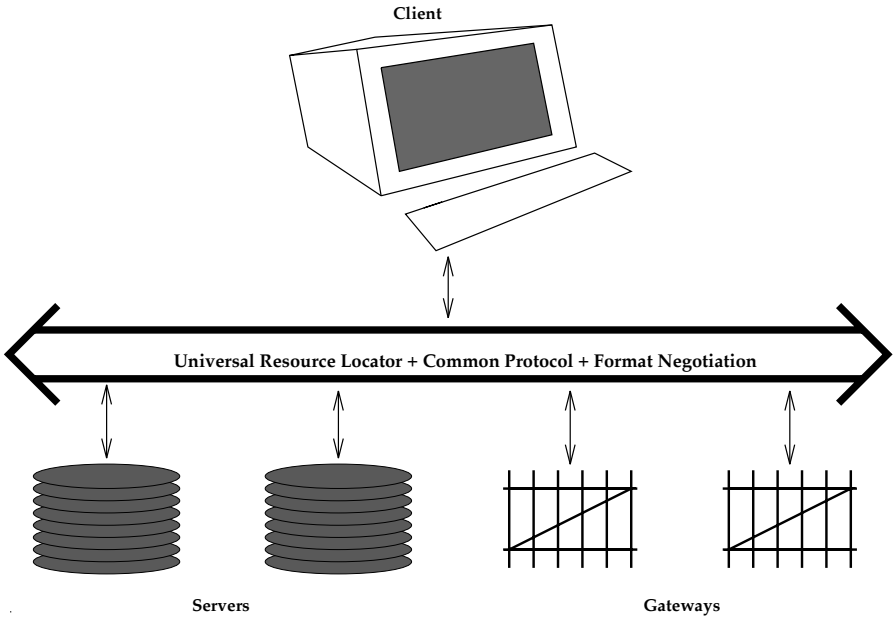


Figure 3.1: The World Wide Web Client-Server Architecture.

- **WAIS** Wide Area Information Server. This full-text information retrieval system uses an extension of the Z39-50 search and retrieve protocol. Queries can be made in a Web client which are then converted to WAIS format and sent through a gateway.

3.2.5 Hypertext Markup Language

The WWW was designed specifically to disseminate hypertext information. WWW hypertext is simply plain text which contains markup (formatting and hyperlink information surrounded by angle brackets). This markup, *Hypertext Markup Language* (HTML), is used by Web clients to determine the presentation of a node and to describe any links to other nodes. In most cases, markup consists of an opening and closing tag surrounding the section of text which is to be affected.

For example, to make the phrase 'Basser Department of Computer Science' an anchor (start of a hyperlink) to the Basser WWW Home Page, you would insert the following tags before and after it as shown here.

```
<A HREF=http://minnie.cs.su.oz.au:80/basser/basser.html>
Basser Department of Computer Science
</A>
```

Similarly, to markup the word **WHype** in bold, the following would be used.

```
<B>WHype</B>
```

In general, the closing HTML tag is the same as the opening tag except that the label is preceeded with a slash character.

In this paper, the term 'HTML document' will be used to indicate a single node of hypertext or hypermedia material. With HTML, creating 'hypertext views of existing bodies of information' [9] is made easy. The HTML+ specification, furthermore, provides for more sophisticated hypermedia facilities and supports a greater number of text formatting styles. Nevertheless, HTML is sufficient to produce good quality documents. A full list of HTML tags is provided in an appendix.

Interaction with hypertext material does not, and should not, require knowledge of HTML. Users of most clients can access the 'source' HTML but this information is normally used by the display software to determine the presentation and not displayed. It is advised that HTML clients ignore any markup they do not understand. [2]

There exist already a large number of conversion utilities which automatically generate HTML documents from existing material. These programs take text in existing formats such as laTex, rtf, [4] and troff and attempt to manufacture an equivalent HTML document. This has allowed information providers to quickly produce vast quantities of hypertext information. The World-Wide Web developers, who are familiar with HTML, have so far tended to utilise existing text editors in conjunction with pre-defined macros that produce HTML tags. At present these two methods are responsible for the majority of material available on the Web. Work on applications which focus on original HTML document creation has only just begun.

3.3 Crossing hyperspace: a test flight

The simplest way to consolidate the above description of how the World-Wide Web works is to see how a single hyperlink is traversed.

Let us say that from our Web client we wish to access the World-Wide Web 'What's New document' stored at NCSA. [5] All we need is the URL, in this case:

```
http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/whats-new.html
```

The initial acronym indicates that the subject of this particular URL is to be retrieved using HTTP. Our client then initiates the protocol by making a connection with the site referred to in the URL.

```
www.ncsa.uiuc.edu
```

[4] Rich Text Format.

[5] One of the useful properties of the Web is that along with global access comes the ability to centralise certain resources. This particular document , kept at the National Center of Supercomputing Applications (NCSA), is a list, in reverse chronological order, of all the interesting services as they have become available in the Web.

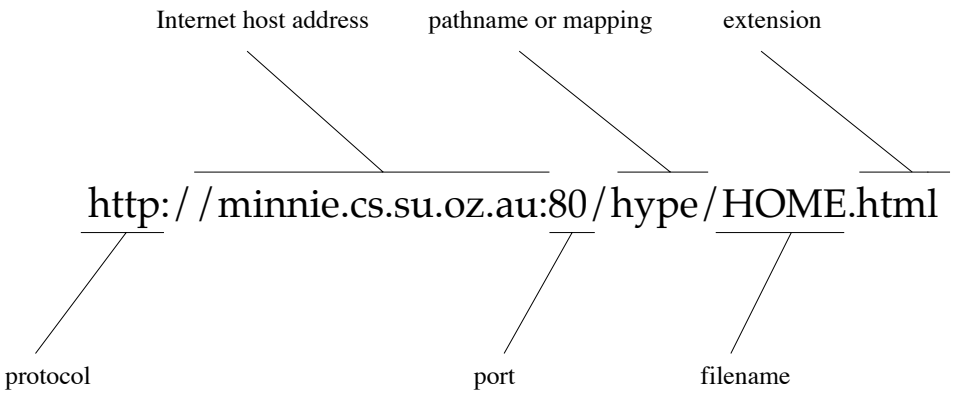


Figure 3.2: Components of a Uniform Resource Locator.

Once this connection is made, the client continues to use information contained from the URL. The client sends a request for the particular node it is after. Our specific request takes the form of

```
GET /SDG/Software/Mosaic/Docs/whats-new.html
```

The HTTP daemon, on receiving this request, must first ensure that the document exists. If it does, and it has been given permission to serve it, then it simply transmits the HTML document back to the client. If for whatever reason the request was not available to be fufilled, the server sends back an appropriate error message also in the form of an HTML document.

The WWW client handles the delivered information according to its file extension. In this case, since the extension indicates a HTML document, the client must parse the document, interpret the markup instructions and display the result to the best of its abilities. Since error messages are just simple HTML documents, the client would have followed the same procedure had the desired file been unavailable.

3.4 What is out there?

The value of the World-Wide Web is that information available is not restricted to hypertext and hypermedia. Since a URL can effectively point to any information source, the Web provides access to other existing information services. This makes the Web relevant to information providers who have committed themselves to earlier services such as gopher. [6] At present, the body of users interacting with the Web reflects the composition of the Internet. The majority of information available has been provided by academic, research, and government sites while commercial Internet sites have, so far,

[6]The University of Sydney gopher service is available through the URL gopher://gopher.su.edu.au:70/11/su.

been content to access these without contributing directly. The nature of the WWW project will change once this rapidly expanding sector of the Internet begin to exploit Web usage.

3.5 The expanding Web

The potential of this information resource has been quickly realised by other Internet-connected organisations. At the start of this year, there were around fifty known WWW servers. By August, an estimated 100 WWW servers existed. These figures only account for the hosts running HTTP servers. This quote from one of the more publicised WWW sites more eloquently describes the phenomenon.

> Honolulu Community College officially announced their opening of their hypermedia server - the first Web server in Hawaii - at the end of May 1993. By September of that year (after 105 days of service), they had received over 23,000 requests for documents and over 112,000 requests for assets from nearly 5,000 separate hosts on the network. From September 1 to 7 they received traffic from over 600 separate hosts, an all-time high. It is expected that traffic will increase further as the school year begins and student involvement in the Web increases. [8]

Information dissemination has become a buzzword. Universities, institutions and government services are racing to produce ever-more elaborate information services publicising their endevours. It remains to be seen how long this friendly spirit of free access remains. An 'Access Authorisation' implementation scheme for the WWW has just recently been announced. [7] Such a feature will no doubt eventually be included in all Web servers. When this occurs, the nature of the Web is bound to change dramatically.

At present, the users of the Web are involved in what could be thought of as a gigantic social experiment. Traditional copyright laws are clearly incompatible with a global system like the Web where information like song lyrics is widely disseminated. As it stands, these issues are not so much being ignored as avoided since Internet usage is not yet considered wide-spread enough to warrant the revision of civil law.

The natural evolution of the Internet newsgroups suggests that the material available on the Web will continue to diversify. However, the architecture of the Web means that eventually the novelty of devoting local system resources to a world-wide body of users will probably wear off. Similarly, once the World-Wide Web sheds its experimental tag, services which impose most heavily on computer resources (ie. hypertext clients available by telnet) may be withdrawn or at least restricted.

[7] October 1993.

Of course, all these predictions will come to nothing if the World-Wide Web does not continue to expand. This, however, seems unlikely at this moment. The World-Wide Web appears to be growing at a rate proportional to the Internet itself. The initial CERN Web server has been recording a doubling in its traffic every four months. Such momentum will be hard to slow down, and any efforts to impose widespread restrictions would no doubt provide even greater publicity for a service which does not appear to have been noticed yet by the Australian media.

3.5.1 Maintaining the Web

Hyperlinks can save considerable space on local sites by providing access to existing information resources stored elsewhere. At the same time, they raise the issue of network reliability. Even with the information presently available, the task of maintaining even a single site's worth of hyperlinks has quickly escalated. As of yet, facilities which would automatically update and maintain links for a site have not been made available, if indeed they have been created. Similarly, if a server supplying unique and valuable information is cut off from the Internet, [8] then that information will be completely unavailable. With increased use, and reliance, on Web-based material, the present casual approach to site maintenance will have to be tightened.

3.5.2 Will it last?

As of yet, no one is using the Web for critical material that is not available through other means. This state cannot last too long if the WWW is to continue to grow. The Viatel information service, offered and subsequently withdrawn by Telecom in the mid-80's, can be viewed as a warning that on-line information resources will never be widely accepted for their potential alone. If the World-Wide Web does not become utilised as a reliable source of valuable, not just trivial, information, it is doubtful whether it will last in its present form. Only time will tell if the WWW project is ahead of its time. If, however, the WWW infrastructure is ever commercially supported in the manner of the Minitel information system in France then this current information explosion may prove to be merely the catalyst for information technology development.

In a sense, the altruistic spirit with which the Web was constructed may yet prove to be its downfall. The dominant motivation behind most WWW sites at present is information dissemination. This not only means that information, some potentially lucrative, is being given freely to anyone with access to a Web client, but also that use of this information is essentially ungovernable. The dilemma of the WWW project is the same one faced by the Internet as a whole. Either this technology is going to provoke wide-scale change as a

[8] This is already become a regular aspect of Web use.

direct result of previously unavailable, and largely unimagined, global communication facilities, or these resources will somehow be restricted through political and financial means. While some might see this as the dawn of a new age of communication for mankind, it must be remembered that withholding information has always been the proverbial stick wielded by those in authority.

From the point of view of the pragmatist, it is hard to see how such resources will be maintained without financial incentives. Those anticipating wide scale access to the Web might well look to cable television as an indication of how the information systems may be available in the near future. Social inequality will no doubt continue to limit the relevance of global communication facilities to many people.

Nevertheless there are some positive signs. The recent U.S. National Information Infrastructure Agenda proposal pledges to

> ... transform the lives of the American people – ameliorating the constraints of geography, disability, and economic status – giving all Americans a fair opportunity to go as far as their talents and ambitions will take them. [5]

If nothing else, such rhetoric has helped to bring global information systems into the world media spotlight.

name	*tags*		*WHype*
Head	$<HEAD>$	$</HEAD>$	$\surd$
Title	$<TITLE>$	$</TITLE>$	$\surd$
Next ID	$<NEXTID\#\#>$		
Body	$<BODY>$	$</BODY>$	$\surd$
Plain Text	$<PLAINTEXT>$		
Anchor	$<A...>$	$</A>$	$\surd$
New Paragraph	$<P>$		$\surd$
Header level 1	$<H1>$	$</H1>$	$\surd$
Header level 2	$<H2>$	$</H2>$	$\surd$
Header level 3	$<H3>$	$</H3>$	
Header level 4	$<H4>$	$</H4>$	
Header level 5	$<H5>$	$</H5>$	
Header level 6	$<H6>$	$</H6>$	
Unordered list	$<UL>$	$</UL>$	$\surd$
Ordered list	$<OL>$	$</OL>$	
Directory	$<DIR>$	$</DIR>$	
Menu	$<MENU>$	$</MENU>$	
List item	$<LI>$	$</LI>$	$\surd$
Term	$<DT>$	$</DT>$	
Description	$<DD>$	$</DD>$	
Bold font	$<B>$	$</B>$	$\surd$
Italic font	$<I>$	$</I>$	$\surd$
Address font	$<ADDRESS>$	$</ADDRESS>$	
Typewriter font	$<TT>$	$</TT>$	
Pre-defined	$<PRE>$	$</PRE>$	$\surd$
In-line image	$<IMG...>$	$</IMG>$	$\surd$

Table 3.1: Hypertext Markup Language tags supported in WHype.

Chapter 4

Spinning a local Web

4.1 The project

The project, which I undertook, was to establish a lightweight, distributed hypertext system that was small, but powerful, with good authoring facilities. It would also, ideally, be portable.

4.2 Motivation: spreading the hype

As I began preliminary research into what had been already done in the field, it became apparent that any information system by itself serves little purpose if not utilized extensively. In other words, in order to set up a system that would be of some pragmatic value, it would be essential that people have access to its components as soon as they became available.

This was to prove a most valuable aspect of the project as the year progressed. Not only was I able to encourage people to browse the World-Wide Web, but opening up my work to onlookers generated a lot of useful feedback which was able to be incorporated into both the local hypertext and WHype itself.

In order to generate interest in the project, I intended to provide a comphrehensive series of links to existing Web material to demonstrate the scope of available resources. The extent to which I was successful in this endevour will be discussed further on.

4.3 The proposed system

The proposed hypertext system was to consist of these components, set up in the following order.

1. Clients for browsing the Web.

2. A WWW server which would provide any clients with locally-produced hypertext.

3. Local information resources as well as links to useful existing material. A 'local Web'.

4. A new Web client which would provide authoring facilities.

For the most part, I was able to stick to this plan throughout the project although work on the various components overlapped. Each of these developmental stages deserves separate consideration. The new client, WHype, itself will be the focus of the following chapters.

The process of compiling each of the WWW software products in turn was very frustrating. The mix of SYSV and BSD.4.1 UNIX particular to our MIPS R3000 (upgraded halfway through the year to an R4000) contributed to the difficulties I had installing most of the software, some of which (eg. the server) was vital to the system. The amount of time and effort spent trying to get products to compile on the MIPS was not beneficial to the task of designing and implementing my own program.

4.4 The clients: information explorers

The World-Wide Web is still in its infancy. Whether or not this information system continues to expand at the current rate of growth will remain to be seen. As the birthplace of the World-Wide Web, visiting CERN was the logical first step to discovering the Web.

```
telnet info.cern.ch
```

This command allows remote users to browse the WWW using the LineMode Browser client. This service, still available, allows people to not only learn about the Web without going through the effort of compiling Web software but also is a good way of researching which particular software would best suit their information needs. From this preliminary research, I was able to download a range of existing WWW software.

4.4.1 LineMode Browser

This was the prototype UNIX WWW client. First released in March 1991, it presents an ASCII-only view of the Web. Designed to be readily portable, the display is extremely rudimentary but nevertheless now provides access to all text material available through the Web. It uses numbers enclosed in square brackets to denote links to other nodes (eg. [42]). Users follow a link by typing in the corresponding number. This kind of display has the obvious limitation that the anchors are not easily distinguished from amongst the

text. Another inadequacy of the display is its upwards scrolling mechanism which makes nodes larger than the screen size awkward to read.

Moving around the Web is supported through a small number of navigational commands.

Back returns to the last node visited.

Home returns to the first node visited in the session.

Forward allows the history list to be re-traced.

Go jumps to a specified URL.

As with all the other clients, the LineMode browser contains a history mechanism which keeps track of which nodes have been visited. This feature is a vital component of any browser since it gives the user some sense of position within the mass of hyperlinks.

Despite its primitive interface , the Line Mode Browser remains a valuable addition to any hypertext system as it allows the Web to be accessed through any dumb terminal (ie. when using a modem). The simplicity of the program, which uses the public domain World-Wide Web libraries, meant that, out of all the software that needed to be compiled for the project, this was the only program that was easily installed.

4.4.2 Lynx

This browser, like its predecessor, is designed to provide vt100 terminal users with access to the Web. Unlike the LineMode browser, Lynx utilises a full screen display with cursor controls and anchor highlighting. This program is an example of software which, by virtue of its limitations (in this case an ASCII-only display), may never be desired by many users who have access to more elaborate hardware but nevertheless serves a large market. Its existence means that an information system of the sophistication of the Web does not necessarily have to require equally sophisticated resources to access it comfortably and efficiently. As access to Internet resources continues to grow, it is likely that text-only clients like Lynx will be the first glimpse of global information systems for many users.

4.4.3 ViolaWWW

The Viola object-orientated toolkit, under development at Berkeley, was first used to implement a Web browser. This product, released May 1992, provided the first instance of a windows interface for accessing the Web. Viola distinguishes anchors by underlining them. This is a vast improvement over the LineMode browser approach. Selecting anchors by a single mouse click appears to be a much more convenient way of moving around the Web. The

display is further improved by the browser's multi-font representation of the seven HTML headers and other formatting styles.

The well-structured interface contains the previously mentioned navigation controls as well as optional 'active help' wherein Viola tracks mouse movement and gives information about its whereabouts in a message box. However, the Viola icons which are quite small relative to the size of the window, are not very informative. Help documentation is available via a small globe icon although the fact that it is an icon at all is not immediately obvious.

There were some problems with Viola. Occaisional hangups would occur when the program could not find the fonts it requested. Also, the anchor selection mechanism was awkward in that an anchor extending over a line would only be selectable by clicking on the first of the lines. The speed of the display, especially when loading in large documents, was similarly just slow enough to be frustrating.

Nevertheless Viola provides a good representation of the Web in a package that is still being improved. The challenges of incorporating both multimedia support and editing facilities into a hypertext browser had still, however, not been met.

4.4.4 NCSA's Mosiac for X

With the alpha release of this product in January this year , the World-Wide Web took on new significance. For the first time, a browser supported not only multi-font text but also in-line graphic images and audio support.

There are several other features of XMosiac worth noting for their efforts in making the Web a more comfortable environment. Personal annotations are able to be added to a document. This allows users to append their own text to any existing node. Quick notes can be appended to any Web node not just ones which are write-permissible by the user.

The browser also maintains a user 'global history' allowing it to indicate which anchors in any node have already been accessed (by underlining the anchor with a dashed line). This is particularly useful when accessing a node for the first time. Any anchors leading to nodes already visited (even in earlier sessions) are immediately obvious. This feature is one that, no doubt, will be supported in future hypertext applications.

Perhaps the most useful navigational aid provided by XMosaic is its 'Hotlist' feature. This allows a user to maintain a list of interesting nodes, separate from the history list itself. This list, unlike the history record, is directly modifiable from within the browser.

XMosaic improves on the appearance of previous browsers, incorporating the now-standard navigational functions into a slick, Motif [7] interface complete with pop-up windows and pull-down menus. It supports the display of bitmaps, as well as the playback of audio material and animation clips

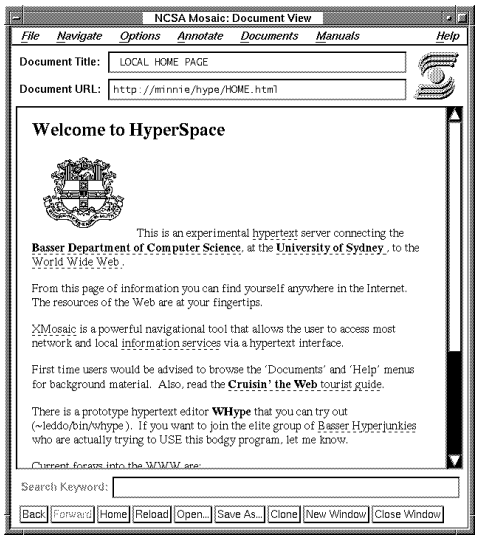


Figure 4.1: The NCSA Mosaic for X browser.

by calling external viewing and audio software. This interface gives uniform access to most of the existing media formats available on the Internet.

As such, XMosaic is the browser of choice for most users. The software is available to run on both UNIX and VMS platforms. There are Apple Macintosh and Windows releases as well as a planned release for the Windows NT operating system.

The design emphasis behind XMosaic has been to produce a sophisticated means of browsing World-Wide Web material. As such, the product does not provide any authoring facilities. Nevertheless an examination of the complexity and size of the XMosaic code resulted in a decision not to attempt to provide more than elementary browsing facilities within my own client. The sheer versatility of this browser was something that could not be replicated by a single developer. With this in mind, it was decided that any authoring facilities would be provided elsewhere, using XMosaic as a highly elaborate and flexible viewer.

4.4.5 TkWWW

As the first, and as yet only, released WWW editing tool for UNIX, [1] it was vital that I examine this program. However, tkWWW, built from the tcl library [2] had already gained a reputation as a difficult program to compile on different systems. I was fortunate enough to be given a temporary account on a NCSA host in order to review this product.

[1] The original NeXt browser also supports HTML editing although this was not available for testing.

[2] tool command language

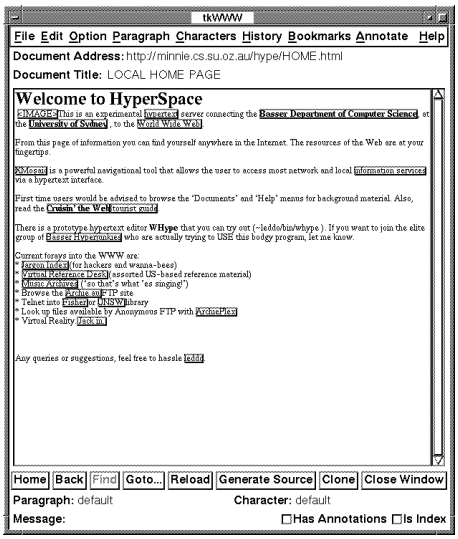


Figure 4.2: The tkWWW hypertext browser/editor.

TkWWW is described as a 'WYSIWYG hypertext browser/editor'. [14] Its browsing features are similar to what is available in graphical browsers like XMosiac and Viola. As with XMosiac, tkWWW uses pull-down menus and radio buttons. I found this style of interface easier to follow than the somewhat ambiguous icons of Viola. TkWWW offers the ability to enter modify text in the displayed node. From the menus, the entire range of HTML tags is available (by first highlighting some text and then a particular HTML tag). The multi-font display allows a user to construct new hypertext and create links. However, despite being touted as the first UNIX HTML browser/editor, it is doubtful whether the editor is being used extensively by developers.

Anchors are denoted in tkWWW by surrounding boxes. Although clearly a matter of personal taste, I found the box approach clumsy and the surrounding text, especially when there were many anchors close together, difficult to read. [3]

The most serious flaw in tkWWW was its inability to cache documents, or otherwise retain modified node information, when browsing. When constructing a hypertext document it is reasonable to assume that the user will wish to test the links within a node. If a link was traversed in tkWWW from a node in which there are modifications, these modifications are lost without any kind of warning. The program requires users to continually save their modifications before leaving the changed node. In a sense, this re-enforces the impression that tkWWW is only intended to provide the facilities to make occasional modifications to existing nodes. The program was far more useful as a browser than as a fully-fledged authoring tool.

[3] This style of presentation is also used in the recently-released Cello WWW browser for Windows.

Nevertheless tkWWW does support the full range of HTML. More direct comparisons between WHype and tkWWW will be made in the next chapter.

4.5 The servers: information providers

In order to make any local hypertext available to remote WWW clients, it was necessary to install an HTTP server.

The first server I set up was the one provided by CERN. As with the Web clients, the server code relies on the public domain World-Wide Web standard libraries. These libraries, forming the bulk of the code, provide the TCP/IP routines as well as an HTML parser and other generic hypertext utilities. Essentially a daemon which listens to the assigned HTTP port 80, the program examines a simple rule file to determine whether or not the material requested by a certain client is indeed available. A file consisting of these few lines was all that was needed to ensure that only two directories of hypertext were accessible.

```
map    /              /hype/HOME.html
map    /hype/*        /usr/hons/leddo/hype/*
map    /basser/*      /usr/hons/oliver/CI/*
pass   /usr/hons/leddo/hype/*
pass   /usr/hons/oliver/CI/*
fail *
```

These rules allow names to be mapped onto certain components of the file system, a useful feature which means that entire tree structures of material can be easily moved without having to modify all the anchors in each and every document. Only the relevant mapping in the rule file would need to be modified.

Towards the end of the year the limitations of this release became more obvious. It could not recognising certain graphics formats (which were being displayed as text files). I then installed the 'rival' server developed at NCSA. This was a much more compact program . It uses an alias, rather than a mapping, approach to identifying permissible directories. This was the configuration file used this year.

```
/hype:/usr/hons/leddo/hype
/Daemon:/usr/hons/leddo/src/WWW/Daemon
/WWW:/usr/hons/leddo/hype/WWW
/basser:/usr/hons/oliver/CI
/katya:/usr/hons/katya/www
/jamesr:/usr/hons/jamesr/cruise/www
```

Both these servers supported the single action, 'GET', used in the original HTTP. By June, however, some servers were also providing searchable index

facilities. These customised servers used small shell scripts to carry out client search queries on local databases. The latest NCSA server, just released, comes complete with a directory of utility programs that can be executed by clients. In this way, local services like 'finger' can be accessed through the Web. The output of each program is transmitted by the server as if it was the request itself.

4.6 Information services

4.6.1 Linking in the world

In an effort to encourage people to investigate what the World-Wide Web had to offer for them, I spent some time exploring and gathering together a collection of URLs that I felt might be useful reference material. I assembled this information into a home page (see Figure 4.1) that would be a useful local starting point for local users. I included links to a 'jargon' dictionary, the Archie.au FTP site as well as telnet connections to both Fisher Library and UNSW Library.

Despite the availability of all these services elsewhere, there is no doubt that this package of services encouraged many people to take advantage of them for the first time. For myself, this hypermedia reference base became the environment in which most of the research for this project took place.

4.6.2 Basser material

As mentioned earlier, the value of any information system is measured by the extent to which it is utilised. It was therefore important that my hypertext system incorporate more than just links to outside information. I was however reluctant to devote much effort to providing this material myself. Not only would it be counter-productive to appear to be monopolising any information service, but I felt that it would be more interesting to see if anyone would be motivated to contribute through publicising the available tools.

Three of the Honours students took up the challenge of hypertext authoring this year. Two of these three authors used WHype (to different degrees) to create their material. A discussion of this actual use of the editor will be included in chapter six.

The contributions to the local Web were

- Basser Department of Computer Science information, including the entire departmental handbook, as well as maps, research groups information and staff information. [4]

[4] By Oliver Bock. URL http://minnie/basser/basser.html

- A guide to Internet-based meteorological information. [5]
- Focus: Earth, a graphical tour of the Internet geographical resources. [6]

4.6.3 Other material

Along with the previously mentioned home page (now the Basser local home page) I also produced two hypertext documents providing information on authoring.

- The WHype User Manual [7]
- The Cruisin' the Web tourist guide [8]

The latter document discusses what I perceived to be three possible modes of interaction with the World-Wide Web; namely browsing, customising (restructuring existing material), and extending (authoring).

Nevertheless all these information resources and the Web infrstructure itself were only a fraction of the project itself. The main focus of the year was to provide a useful editing application. The design, implementation and subsequent usage of this tool, WHype, forms the content of the next two chapters.

[5] By Kateryna Ostrowsky. URL http://minnie/katya/meteo.html
[6] By James Robertson. URL http://minnie/jamesr/MainPage.html
[7] URL http://minnie.cs.su.oz.au/hype/help/man/title.html
[8] URL http://minnie/hype/help/nav/intro.html

Part II

The WHype HTML editor

Chapter 5

Design and alternatives

5.1 Design aims

WHype was designed as a tool that would to allow users to quickly and easily create their own hypertext structures. Knowledge of HTML is, however, a considerable hurdle for a novice user to overcome so the design concentrated on providing an interface which allows users to concentrate on developing their material, leaving WHype to generate any markup.

Based on an analysis of the resources available on the Web at the start of the year, it was felt strongly that only a small subset of HTML would be needed to give users the ability to create hypertext of a standard equivalent to what was available. In this regard, I was subconsciously eliminating those users who would wish to use all of HTML. The description of this project suggested that a simple, therefore easier learnt, hypertext application would be more immediately useful than one that provided elaborate facilities.

This could be viewed as an excuse to account for the relatively simplistic functionality of WHype. However, as a prototype, it can be viewed as an effective tool for experimenting with the concept of hypertext authoring.

Since HTML material could be simply manufactured using existing text editors (using macros or by inserting HTML by hand), it was imperative that WHype offer some simplification of the authoring process to encourage its use. This implicit requirement of the design suggested that the handling of hyperlinks would be a primary consideration of any design.

5.1.1 Choosing an interface

WHype was intended to be portable. In particular, it was seen as desirable that WHype be portable to the Plan9 operating system, presently under development at Bell Labs. To enable a smooth transition from UNIX, the interface was implemented using the plan9 text and graphics libraries (libXg).

1

This decision was perhaps the most important factor of the design phase since it involved considerable effort in identifying the trade-offs and limitations of building an interface with either the Motif toolkit or the plan9 libraries. I intitially had several misgivings about using the plan9 libraries (most of which were proven to be justified) that prevented the design phase from being quickly completed. In this chapter, I outline the context of the design decisions.

5.2 Motif: design or convenience?

The Motif toolkit offers several advantages to developers. 'Widget' object classes allow the quick assembly of standard window interface components into a user-specific framework. Not only does this provide for conformity between applications (aiding user cognition) but it frees the developer from the often tedious task of constructing such 'standard' features from scratch.

Using Motif would have allowed for the implementation of a WYSIWYG [2] authoring tool in the sense that it could have used a presentation consistent to that used in the Motif-based XMosaic browser. This was therefore a very tempting design approach. Nevertheless, the aim of producing an portable client warranted a closer examination of Motif design. [3]

The Motif design philosophy advocates three basic ideas of interface design

1. 'Programs that deviate from the expected design will almost assuredly confuse the user even if the changes were intended for the user's benefit.'

2. 'Users rely on rote memory....There is a limit, however, to how much users want to remember.'

3. 'Users, especially novices, will probably not want to customize or alter their applications in any way.' [7]

While not condemning any of these ideas, it is clear that this fixed approach to design should be treated with caution by those involved with active research. It cannot be denied that familiarity with a style of interface reaps great rewards with users; it allows users, new to a particular application, to find their way around purely through their pre-conceived notions of where objects 'should' be. The commercial emphasis on, what I will refer to as the 'standard windows interface design' [4] , has been beneficial in providing

[1] The UNIX ported versions are available via FTP (ftp research.att.com) from Bell Labs

[2] What You See Is What You Get

[3] Had WHype relied on the Motif toolkit, it would not have been easily portable.

[4] The term GUI, or graphical user interface, is used often as a synonym for this.

novice, and insecure, users with an immediately recognisable, even if not familiar, environment. Nevertheless the complete acceptance of current designs is not a healthy state of mind for those wishing to improve on an interface that owes as much of its appeal to its exposure and stranglehold on popular perception (of what an interface 'should' look like) than to its innate design qualities and potential.

5.2.1 The Windows word factory

The Motif style guide advises developers to 'make direct connections to real-world objects'. [7] For years software companies have utilised the desktop, notepad, clipboard, and filing cabinet metaphors to encourage usage of their programs. But what real connection does a labelled button have to the task of writing a letter? Before the advent of word-processing, none at all. Yet today, people have come to associate 'easy' software with an interface that presents an increasingly complex array of buttons and other virtual devices that help the user grind out their product. Those who have associated the writing process with a less mechanical perspective have not suffered; the $700 word processor is not yet an obligatory household item. However it remains to be seen whether this approach, to what was once a matter of sharpening a pencil and finding a piece of paper, will, as often touted, be the future of the written word.

5.2.2 'Horses for courses'

There appears to be a basic human drive to create perfection. The perfect car, the best computer, the ideal word processor. At the same time, it is no revolutionary observation that people are different with different needs. Any dissatisfaction expressed by academics concerning commercial software is clearly the result of one group of people viewing products not designed for them. Nevertheless, it would appear that commercial software designs which continue add features to their products, with every new version, are just as irrelevant to many users who may not desire the most intricate features. It is no disgrace to crave for a simplistic approach. Indeed, the UNIX philosophy of using a collection of small shell scripts as tools is indicative of the flexibility and power of this approach. Complexity in an application is fine as long as it does not needlessly distract from the fundamentals. Clearly all the complexity of a contemporary word processor is not required by every user.

A standardised approach to interface design encourages designers to add features without considering whether or not the means of accessing these features is relevent to the task at hand. The various components of the interface are never questioned for their intrinsic suitability to the specific task.

As an example, consider the tkWWW client - billed as a hypertext editor. The program provides the user with enough pull-down menus to support the entire HTML. Nevertheless this standard toolbox approach makes no effort to

investigate the possibility that a previously unavailable medium, distributed hypermedia, may be better supported through an interface developed specifically to exploit it.

Software has remained exclusively in the scientific domain for too long. The substantial user population that utilise text editors and word processors are, for the most part, unaware that they are using products that are, in themselves, creatively engineered. Most people view software as tools to enhance their own productivity. This produces a conflict of interest whenever software engineers are clearly not building applications for people like themselves. A computer program can not be appreciated for its code design by an end-user. It is therefore hard to reconcile the ambitions of both the tool user and the software designer who created it. More features and technological innovation is obviously important for the designer but less so for the user who may have been happy with a simpler product. It is also clear that, for many computer applications, this conflict of interests is not so apparent when the tools are designed for people for whom this programming mentality is understood. However this paper is only concerned with information applications. Information technology research should not be allowed to reflect only the perspective of programmers since it affects the whole of society. It is a fact sometimes not well appreciated by the scientific community that communication is often best achieved through simplistic means. Providing more elaborate communicative devices may enable good information to be improved dramatically but the basic mechanism must remain accessible.

The reason that the hammer has remained in such a simple form is that it fulfills its design requirements perfectly. Any frills and extensions would obviously be detrimental to its functionality. If a hammer was implemented in software than it is conceivable that someone would attempt to apply the Motif style guide.

> 'Underuse of visual cues may make an application look bland and uninteresting' [7]

Does a word processor need to be interesting? It is, after all, a tool for expressing hopefully interesting material. The humble sheet of paper looks pretty bland to begin with. Similarly the sam text editor, providing few visual cues to the user, begins with a blank window. [5]

This is not to say that there is not a place for highly sophisticated (and complicated) interfaces. After all, just as one person may use coloured designer writing paper, a computer user may wish to take advantage of advanced editing features. Nevertheless, a visible trend in software design, to equate cluttered 'feature-packed' GUIs with sophistication, is disturbing. Complexity in applications should only be visible to those who wish to use it. It is obvious that users are becoming increasingly disorientated in the midst of

[5] The fact that, even after running sam with file arguments, the novice user is presented with a blank window, is nevertheless an obvious cognitive problem which must be addressed through education.

such 'useful' programs. With conventional text editors, let alone hypertext authoring tools, basic features should be kept independant from the toolkit of advanced features. A plumber does not try to juggle twenty different tools at once when grappling with a broken faucet. Similarly until use of hypertext is more familiar, there would appear to be little cognitive advantage of flooding a user's conceptual notion of hypertext by providing a multitude of extra features.

5.3 Interface design: whose cognition and why

Of course, none of the above discussion would be necessary if it were possible to design interfaces that enabled sophisticated use (satisfying the demands of high level users) while actively encouraging new users to attain this sophistication. However this design philosophy is either not widely accepted or simply not recognized.

5.3.1 Commercialism: giving the public what it wants

Computers are always being pictured as labour-saving and time-saving devices; a perception not entirely unnoticed by the advertising industry. The GUI has been proclaimed as the panacea for all users uncomfortable with computers. Such enthusiasm is not shared by all the academic community. There is a growing appreciation of the limits of any approach to interface design that attempts to 'simplify' complexity in applications through graphical means. It is becoming more and more obvious that a screen-full of thirty buttons, or coloured icons, is not really any more helpful than having thirty text-based commands if help facilities are readily available. Communication based on graphical information has not yet evolved to the state where words are unnecessary. In this light the extent to which the commercial GUI has been expanded can be viewed as more of a gimmick than as a valuable aid to the user. Widespread usage of a commercial text editor of the calibre of Microsoft Word for Windows, which with each release adds more and more features to its GUI (see Figure 5.1) , does not seem to have improved the ability of the average user to write.

Such an outlook does, admittedly, seem to suggest that some software developers have not considered the needs of their users. Unfortunately the problem is not quite that simple. Rather than intentionally slandering the majority of commercial program developers who make use of the standard windows design, this analysis is intended to suggest that the direction taken by contemporary software is somewhat misguided in not fully appreciating basic human traits like adaptability. The arguments for a standard , and instantly recognisable, interface are valid. However it may be that the negative aspects of non-conforming interfaces (eg. the inability of users to cope

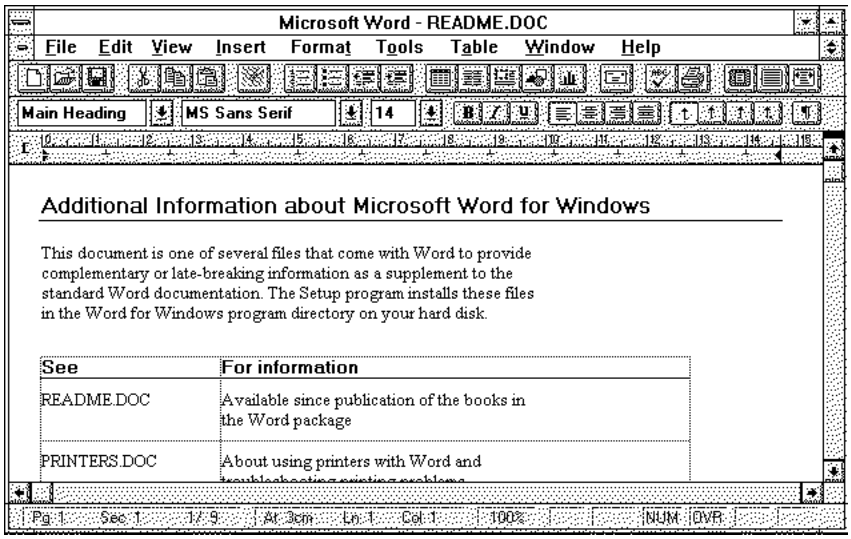


Figure 5.1: The Microsoft Word for Windows graphical user interface.

with unfamiliarity) have been overestimated. It is obvious that the decision for many programmers to adopt this form of interface is guided by factors that have little to do with a deep-felt conviction of its worthiness. There is little motivation for fighting public perception of what a computer interface can be, not in the future, but right now. After all, what is the use of a innovative new interface if no one will use it? The safe and financially-secure path to development is often to continue expanding on what has been already accepted.

It is, nevertheless, unfair to suggest that commercial software design is in any way irresponsible. After all, the commercial sector can be seen as a mirror into popular perception. The surge in computer-usage in recent years is not something that could have been anticipated, let alone planned for. This discourse is attempting to emphasise the fact that interface design is still too young a field of endevour to be standardised at the current level of sophistication.

5.3.2 Academia: learning for mankind?

It is not only the commercial sector which has encouraged standardisation. The inability of the academic community to publicise any potentially superior interface design could be interpreted as passive condoning of the spread of 'Microsoftism'. Within any academic domain there has always been a degree of superiority towards those outside the field. This is especially true with computer science for the simple reason that we are dealing with complexity which is never revealed to most users. Interface developers have continued to produce designs that challenge the public perception of what an interface should be. Unfortunately for them, the public has been given no choice, and

appears to have no understanding that a computer program need bear little resemblance to an Apple or Microsoft product. The commercial markets for these designs have been established and sheltered from any potential competition.

At the other end of the spectrum, the resources of contemporary academic computing environment have been jealously guarded by those who have access to them. The quite drastic contrast between an Internet-connected UNIX account and the rather meagre resources of the home computer user running MS-DOS/Windows on a 386SX is not something that is widely publicised. Why? It could be the unspoken opinion that the majority of the population would not be able to use these resources even if they were made available. Equally, it might simply be that the potential for universal access to the Internet is, for many, too problematic, and perhaps too frightening, a prospect to deal with. Nevertheless the social pressure to grant wide-scale access to the Internet is mounting slowly as the resources currently available to an estimated 12 million users world-wide are being publicised. Information and communication technology has negated any argument that networked computer systems are not relevant to the general public. It is doubtful whether the home computer user will be content to be patronised, and financially drained by commercial software companies, once the full extent of present academic resources are made clear.

5.3.3 The value of consistency

Without wishing to conjure up Orwellian visions of mind control , it is clear that people can be conditioned into using just about any form of interface through the proper circumstances. The widespread use of use of an application like the vi text editor bears testimony to not only the flexibility and adaptive ability of humans as computer users but also to the fact that people can, and will continue to, use applications through bad, or otherwise inferior, interfaces if necessary. There seems to be little criticism from expert vi users regarding this product. Is this to be taken as a sign that it has a good interface? Not really, most admit that learning to use vi is tedious and difficult. However, once a large subset of its one-letter commands are committed to memory, complex editing procedures can be achieved very quickly. In this way, it is clear that for many people the inconvenience of learning to use vi is worth the effort.

All this is not mean to insinuate that the standard windowing interface, as supported by Motif, is a bad design but rather that a condescending ('bells and whistles') approach to interface design should not necessarily be viewed as ideal, especially in the area of research. Equally, to withhold alternative designs from the public eye on the basis of whether or not people would be able to use them 'comfortably' is short-sighted.

The consistency provided by Motif interfaces should not be considered sufficient reason to automatically choose that style of interface. Admirable as the

efforts to aid user cognition might be, it is doubtful that these gains would override the implicit advantages of using a system that deals with the same task in a simpler and conceptually more relevant fashion. For a program as simple in concept as WHype, there seems to be little advantage to be gained from adhering completely to this style of interface purely for its familiarity.

5.3.4 Summary

As a conclusion to this brief examination of the standard windows interface, it must be made clear that I am not suggesting that this design will need to be phased out immediately. Nevertheless, as computer usage continues to climb, diverse, and deepen, the reluctance of most commercial software companies to offer alternative designs can be viewed as a lack of vision.

It may be naive to suggest that the average member of the public will ever make use of computer applications that require programming skills. Equally it is obvious that more and more people are interacting with computers in increasingly elaborate fashion. To satisfy both those who want to simply use and those who wish to exploit available technology through a computer interface, there must be specifically designed tools. A global population is more than enough of a market to warrant a variety of interface styles.

Rather than assume that a similar GUI approach is ideally suited to every application, designers ought to consider specific application design requirements with with greater detail. We live in a society that endevours to make things 'user-friendly'. As attention spans lessen, literacy plummets, and fast food chains multiply, people are starting to assume the computer, like everything else in our world, will be somehow rendered more palatable. It may yet be. Nevertheless for those who wish to exploit the ability of computers to increase their productivity, the GUI is not going to remain relevant.

Alternative interfaces designs, like the one used by the sam text editor, suggest that a minimalistic approach to interface design might be more beneficial to programmer and non-programmer alike. More 'friendly' interfaces may, however, continue to support the ever growing number of products that are only intended for casual use. In the same way as the World-Wide Web XMosaic browser uses an appealing GUI to allow immediate access to global information, there will always be a need for products that directly address the needs of casual users.

At the same time, it must be noted that the cognitive barriers which currently prevent people from using programs that use minimalistic visual feedback are easily addressable through education. [6]

Developing a hypertext system for a body of users adequately familiar with both Motif and plan9 interface designs [7] meant that any WHype interface

[6] Education or conditioning? At primary school level, at least, the difference is often hard to tell.

[7] The Basser Department of Computer Science.

design would not have to focus on the cognitive needs of novice computer users. In any case, it would be appropriate for WHype to use hypermedia documentation for this purpose.

5.4 The Plan9 libraries: benefits and limitations

The plan 9 libraries are in a developmental stage. They support a collection of graphics and text utilities (added this year) that facilitate the construction of a interface along the lines of the sam text editor. There is no support, however, for Motif-style interface components within the libraries. [8]

The sam editor interface is built around a set of library functions supporting *frames* of editable text. As of yet, this library only supports a single font per frame. The text library, *libtext*, provides dedicated editing functions that operate on a higher level *text* structure incorporating the frame object. While certainly simplifying the process of creating editable text windows, there was no direct means of supporting the kind of multi-font display that hypertext clients are now expected to provide.

Sam is characterised by a minimalistic display which includes no pull-down menus, and no pop-up windows in clear contrast to most interface designs commercial or otherwise. The basic editing and file operations are chosen from menus which are triggered from buttons two and three of a three-button mouse. The design was first used in the mux windowing system. It reflects the conviction of its author, Rob Pike, that user comfort is achieved by making the interface transparent and simple. However, many have questioned this definition of simplicity when learning to use sam since it offers negligible feedback to the user.

The motivation behind the original design gives an indication of how alternative approaches to interface design can diverge completely without becoming irrelevant. Pike's complaint with modern window systems stems from his conviction that 'simple systems are more expressive'. This statement requires some more qualification. [11]

There appears to be little opposition to the claim that 'simpler' yet equally flexible interfaces are to be admired for their aesthetic qualities. Nevertheless what is less clear is to what definition of simplicity an interface design should be held to. The sam-style interface is simplistic in the sense that there is little, if any, feedback given to the user and also that the visual landscape presented to the user is fairly unencumbered. While this approach obviously could suit a programmer, it is far more controversial to suggest (as Pike does [11]) that this style is suitable for a novice user. In fact, it is not unreasonable to suggest that this 'simple' approach to design is an incredibly complex hurdle for some people to overcome.

[8] WHype interface components, like the pop-up window, had to be built from scratch.

Admittedly, once past this cognitive barrier, the interface can be just as transparent and unintrusive as Pike claims. However, it is doubtful whether this style of interface will ever be available to the majority of computer users as long as the currently favoured interface designs remain in such widespread use. This is not so much a reflection on the worth of the interface design itself, but more a reflection on the lack of communication between the 'real world' and the research sphere. For those who are exposed to sam, as are undergraduates in this university, there is also the lack of adequate documentation that might have enabled less technically-minded users to use the product more effectively.

As mentioned earlier, it was desirable to produce an application that would be as easy as possible to learn to use. The text library allows for familiar [9] sam-style cut and pasting modes to be replicated.

While accepting that this was obviously not going to have any benefits for people not already familiar with sam, it did mean that the user group who would be most likely to use WHype (ie. local users) would have little difficulty in using a similar interface.

5.4.1 WYSIWYG in an HTML editor

The WYSIWYG approach has become universally accepted as the best way of displaying any constructionist task such as editing. It certainly enables the novice user to concentrate on actual editing in a fashion that was simply not available with primitive application like the infamous ed line editor. [10] It has been suggested,by many in the World-Wide Web development community, [11] that this approach should be used in hypertext authoring applications as well. I was obviously extremely reluctant to use a interface design that could not support this kind of display that was being suggested.

Nevertheless, despite the conviction felt amongst many Web developers that WYSIWYG editing tools are needed, there is a fatal flaw in associating this approach with HTML editing. This needs to examined more closely.

What you see in a WYSIWYG hypertext editor is not necessarily what you would get displayed through another hypertext client. It would be impossible to construct a WYSIWYG HTML editor for the Word-Wide Web.

As mentioned in chapter four, HTML is used to provide Web clients with information relating to a document's presentation. This information may just consist of hyperlinks or it may be elaborate formatting and graphical material. The current evolution of HTML into HTML+ indicates quite clearly that this markup language will continue to provide more and more sophisticated features to authors. Obviously it would be desirable if authoring tools could

[9] The sam editor, while not widely used, is used throughout this particular department.

[10] This early UNIX tool allowed editing on a text file, one line at a time.

[11] Opinions expressed on the www-talk mailing list, comp.infosystems.www newsgroup and direct correspondence with developers form the basis of the 'WWW development community' references.

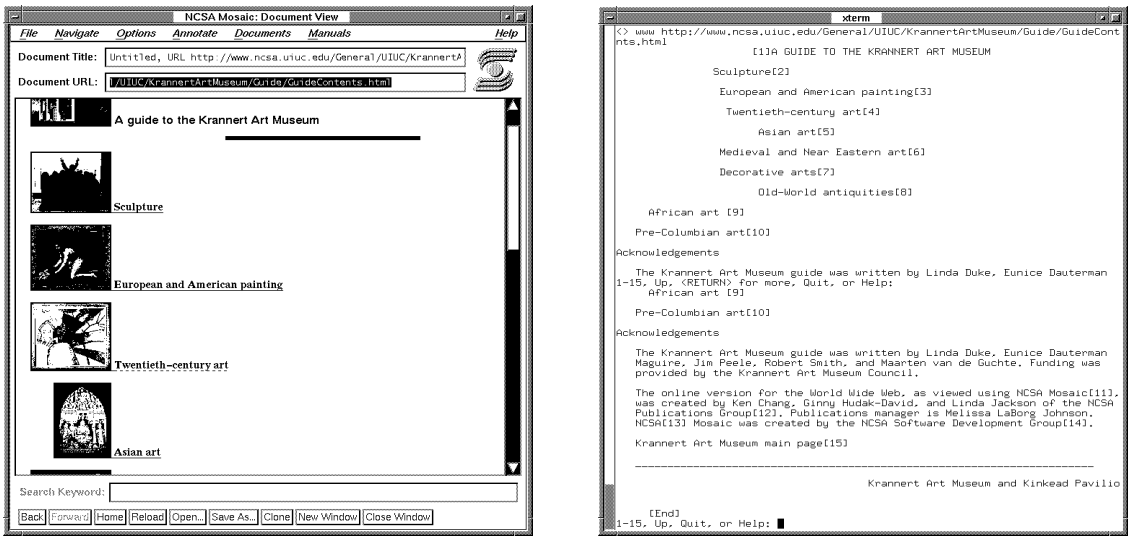


Figure 5.2: Views of a HTML document from XMosaic and the Line Mode browser.

present the documents as they would appear in other clients.

Unfortunately the sheer divergence amongst computer systems which have access to the Web has ensured that, while some sites may be able to view Web hypermedia in all forms, many others can only display more primitive presentations of the information being accessed. A Web client only interprets the HTML that can be properly utilised by the host environment. In other words, a home PC user, accessing the Web via a modem connection to a UNIX site, can still read hypermedia material using clients like the LineMode and Lynx. Any non-textual components of this material are simply not included in the screen display.

As hypermedia Web material continues to accumulate, the limitations of a text-only display are becoming more and more evident. Nevertheless, any author creating material for the Web, at this moment, must be careful that the content of his or her documents are not relying too heavily on the graphical component. Of course this statement implies that authors should limit their work to material that is interpretable by the vt100 client. Without wishing to deviate into any sermon on how the Web should be used, it must be noted that the original motivation behind the Web, namely global sharing of information, is not helped by effectively concentrating on using information resources that, while undeniably desirable, are arguably exploited as much for their novelty and aesthetic qualities than through any desire to communicate. The number of sites currently accessing would be severely depleted if only the sites with graphical clients were to be counted. It is doubtful whether systems like the Web will ever be relevant to the public if access requires sophisticated resources presently only available to a minority of users.

This issue has not been ignored by the development community which has

urged authors of hypermedia material to ensure that non-textual components of documents complements rather than replaces the hypertext. Whether this style guide will continue to be respected remains to be seen but it is clear that the demand for graphical hardware and network access that supports hypermedia will continue to rise.

With this in mind, the recommended WYSIWYG approach to HTML editing can be viewed in context. It is not so much that developers feel that this approach ideally reflects the nature of HTML but rather it is felt that widespread familiarity with modern word processors is something that should be taken advantage of. It is doubtful, however, whether this approach can ever truly reflect the nature of distributed markup-based hypermedia.

There is considerable contention on whether or not this approach to HTML authoring is even required. Many argue that the sophistication of existing editors renders any attempt to build HTML-specific editors pointless. They point to the already popular support for HTML converters and filters. These will no doubt continue to be used by many people who are less concerned with creating hypermedia than presenting existing material in the format. At present, there is a lack of emphasis on authoring tools. However, the need for such applications to provide direct support for the Web environment cannot be doubted. [12]

There is a clear reluctance amongst developers to work on applications that might require users to acquire new skills and methods specific to this new medium. As with the philosophy behind the standard window interface, there may be considerable conviction that exposing users directly to the flexibility and scope of a non-linear canvas will be in vain. Whether or not users will be able to come to terms with the unfamiliar process of creating hyper-structures, or simply refuse to try, remains to be seen. Nevertheless a simplistic application like WHype provides the essential tools with which a user can begin to experiment with non-linear expression and communication.

5.5 Tradeoffs

From the above discussion it should be clear that I felt that neither the Motif or plan9 approach was ideally suited for the task at hand. Using the Motif toolkit would speed up the task of assembling the interface but would render the job of porting the editor (at some later stage) extremely difficult. At the same time, an interface that fitted into the plan9 idiom would be both elegant from a design perspective yet fairly intimidating to a novice user.

The compromise I came to was to use the Plan9 libraries,replicating features from the 'standard' windows design that I felt would compensate for the

[12] Although the almost universal use of HTML converters this year by Web developers did cause some intial apprehension! It is clear that the current aim of information developers is to convert, and provide means of converting, existing material to hypermedia form rather than to develop new authoring methods.

tendency of Plan9 software to provide 'unintrusive' feedback to the user. This decision produced a hybrid interface that will not satisfy either the Motif or Plan9 purists. Instead, by combining what I perceived to be good features of the two divergent designs, I hoped to create a prototype editor that would be immediately useful to the target user group.

Chapter 6

Building WHype

6.1 sam: a role model

I had planned to closely follow the design of the sam text editor. Besides the obvious appeal of conforming to an elegant and pre-conceived design, this was desirable in light of the obvious time restraints. Nevertheless I avoided using the command window approach that sam uses (allowing a large set of detailed operations to be performed on the text). in favour of a Motif-style pop-up window (see Figure 6.1).

The sam command window allows the editor to support highly complex editing procedures while retaining short, and therefore more 'desirable', [11] button menus listing the most frequently used options.

One problem with the command window is that, to a novice user, it is not immediately obvious. After all, it is possible to perform all the basic editing functions from the button menus alone. In fact, the command window could almost be viewed as a black hole deliberately avoided by many novice users. First-year computer science students have repeatedly demonstrated the validity of this observation. Rather than typing 'q' on the command line (quit being a notable exception from the button menus), many users exit the editor by killing the entire window. After observing an almost universal disregard for this component of the sam interface amongst a body of potential WHype users [1] , it was decided that a more explicit means of interacting with the user would need to be found.

The pop-up window approach may not have suited an application which required complex user interaction but the user-dialogues in WHype were intended to be simplistic and brief.

One problem associated with the command window approach to user dialogue is that the syntax of each command needs to be specified and learnt by users. A pop-up window approach, while making any complex command support non-trivial to design, does allow the simple operations like prompting for a

[1] First-year students in this department.

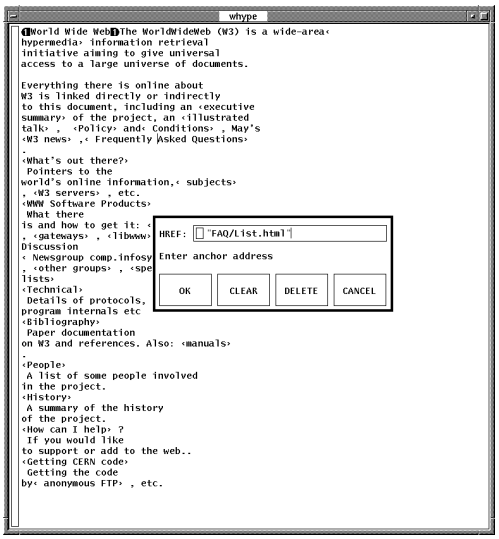


Figure 6.1: WHype displaying the anchor pop-up window.

filename or the destination of a hyperlink to be easily carried out by a novice or occasional user.

6.1.1 Building the interface

The process of implementing the interface and text handling functions of WHype was complicated through the instability of the local copies of the libraries themselves. [2] Rather than using a stable old version of the libXg and libtext, I attempted to keep up with any modifications that invariably were made to the libraries without notification. The lack of adequate documentation (approximately two manual pages of information for each library's worth of functions) also proved to be a major factor affecting the intial speed of implementation. [3] This approach was worth the effort since it ensured that WHype has always compatible with the latest release of the libraries.

6.2 What you see is not what you get

Hampered by the lack of multiple fonts, WHype indicates the presence of markup by using simple tags as brackets. As described earlier, this approach was more-or-less imposed on the editor by the limitations of the libraries. Nevertheless this did not affect WHype's performance, only its aesthetic appearance. To avoid conflicting with user input, the characters used for the

[2]The Basser Department of Computer Science is one of two beta sites for Plan9.

[3]Inadequate documentation, related to academic computer products, appears to be widely tolerated. It would be hoped that new information systems like the Web could spark a reform in this area.

tags were specially created as *unicode* characters. Unicode is best described as an attempt to map all the known character sets onto a single font. The WHype special characters were inserted into an unused section of the character set (0xe000 - 0xe020). This method has one clear advantage over the WYSIWYG approach - style information (ie. header style numbers, italics) is given directly. WHype denotes the presence of an anchor with unicode characters that appear as tiny angle brackets. This presentation was decided on with the view that anchors should not be too strongly emphasised in the text.

This solution to the fonts problem proved to be satisfactory in testing WHype this year. However, if WHype were to be ever released or worked on, this would obviously have to be replaced with a more permanent solution.

6.2.1 Comparisons with tkWWW

Graphical browsers providing a WYSIWYG representation of HTML documents do not need to provide explicit information concerning markup. However, such information should be available to anyone actually constructing a document. TkWWW has a cascading menu devoted to six headers but only uses three fonts to display them with. Clearly the WHype tag information is more useful to the serious author even if it is not so visually appealing. TkWWW users must view the document HTML source for clarification as to the exact markup used. Clearly this undermines the usefulness of having a WYSIWYG approach anyway.

WHype is not a big program. It does not attempt to support the full range of HTML tags as tkWWW does. This is, in part, due to project time constraints. What WHype does offer is a simple editing environment in which a user can become familiar with the concept of authoring. The main complaint that I had with the tkWWW product is that it does concentrate on treating the creation of hypertext as if it was merely another word processor. Authoring implies more than just converting footnotes to hyperlinked material. This is not to say that WHype is, in any way, a more sophisticated authoring tool; it is not. Nevertheless by providing a focus on the fundamental components used in hypertext, it was hoped that the user would be encouraged to experiment more with non-linear structures. The measure of a client such as WHype would ideally be measured by what material was created using it.

For a naive user, anyone not associated with World-Wide Web development, the variety of options in tkWWW does not encourage the development of hypertext structures. Rather, the facility to generate hyperlinks is just one of a large number of pull-down list items. Although this approach is perhaps more useful for someone familiar with HTML-based hypertext, such people were not part of WHype's target user group.

The overwhelming use of the NCSA browser Mosaic for X suggests that people will tend to be attracted to the most elaborate tools available. It is hoped

that WWW developers do not merely concentrate their efforts on producing authoring tools that simply replicate linear text editing methods.

6.3 The WHype User Manual

The user manual (see Figure 2.1) is in hypertext form and, for the most part, was written using WHype itself. It is available through the URL

http://minnie.cs.su.oz.au/hype/help/title.html

One clear advantage of WWW-based documentation is that it can be easily kept up-to-date without the need to re-circulate it. At the same time, it is difficult to convert material written in a completely non-linear fashion to paper. The text is therefore not included as part of this paper. The motivation for this is to show how WHype can be used to author highly inter-connected hypertext that provides multiple forms of access to the structure. The colloquial nature of this documentation is likewise intended to illustrate how hypertext offers a freedom of expression through an inter-connected structure that is not possible in conventional text.

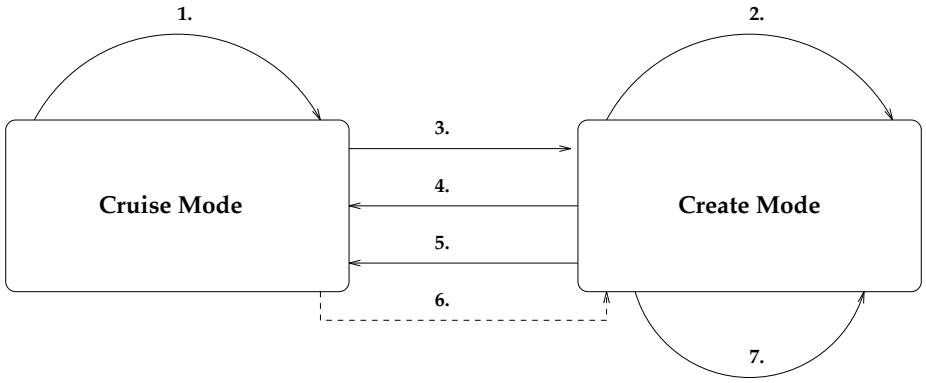
6.3.1 The Tutorial

As mentioned earlier, one of the aims of this project was to illustrate that the World-Wide Web should not be treated as a passive medium. WHype was envisaged as a means of encouraging users to customise and extend their view of the Web.

The tutorial leads the novice user through the process of creating hyperlinks and node. The emphasis is on creating personalised hypertext structures rather than merely producing a linear sequence of presentable documents.

Of course, this approach assumes that users will respond in kind by trying to build hypertext of their own. The tutorial ends after showing how to incorporate the WHype manual itself into a personal hypertext framework. An included tutorial leads the user through the process of incorporating the manual into a personal hypertext structure.

This document is intended to be self-contained. Nevertheless it is important to examine WHype's main features and explain the motivation for their inclusion.



1. Navigational controls: 'back', 'home', 'forward', or click on anchor.
2. a) File handling: 'create' new node, 'open', 'close', or 'write'.
b) Select 'cruise' after having selected an invalid anchor to create the referenced node.
3. Select 'create'.
4. Select 'cruise' after having selected an anchor (traverse the link).
5. Select 'cruise' without having selected an anchor (return to previous location).
6. Select 'add' to append current node to the create file list without exiting mode.
7. Editing tools: 'cut', 'paste', 'snarf', 'exch', 'anchor', or 'style'.

Figure 6.2: Interaction between 'cruise' and 'create' modes in WHype.

6.4 Features

6.4.1 The Modes

The focus of this project was to provide authoring facilities. Nevertheless once the basic editing functionality was achieved, it became clear that WHype should also support browsing capabilities similar to what is available in other Web clients. To this end, it was decided to provide two different modes of operation. *'create'* mode, the editor, was to provide the user with the means of extending and modifying the Web. A separate 'cruise' mode would support conventional browsing. It was envisaged that the modes could be used independantly but would be more useful when used together (see Figure 6.2).

The following quote from the manual introduction illustrates the the concept behind the mode design.

> On the WHype screen at any time there is a 'gunsight', 'pencil', or 'expensive watch' cursor. These correspond to the three modes of interaction you can have with the World Wide Web in late 1993.
>
> - **CRUISING.** You can travel through the world's information resources, shooting from one node to the next via hyperlinks.

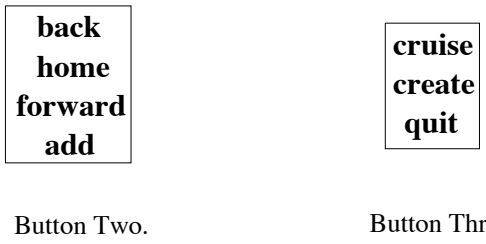


Figure 6.3: Mouse button menus in 'cruise' mode.

- **CREATING.** You can, for the first time, carry with you a virtual clipboard of things to edit and manipulate. Things like bitmaps and, of course, hypertext nodes.

- And since network technology (while too complex to be fully understood by this writer) cannot yet keep up, the third mode of Web usage is **WAITING!**

6.4.2 Mouse Button Menus

The second and third mouse buttons trigger separate menus in WHype as they do in the sam text editor. Unlike sam, different menus are associated with each of the two modes. In general, the tools and functions are listed on the button two menu while modes (and file handling operations in 'create' mode) are on the button three menu.

It was intended that the third button menu be reserved for selecting different modes. However the cognitive advantage of providing familiarity to sam users (by replicating the sam menu design) took priority.

6.4.3 Anchors

The focus of any application which claims to support authoring is its handling of hyperlinks. This is, after all, what separates a conventional document from a hypertext one. The means for handling anchors should therefore be easily accessible. As with all WHype features requiring user dialogue, anchor information is requested by the pop-up window.

There are seven forms of anchor information that can be expressed in markup. [9] WHype only fully supports one type (although the NAME field is recognised and maintained internally).

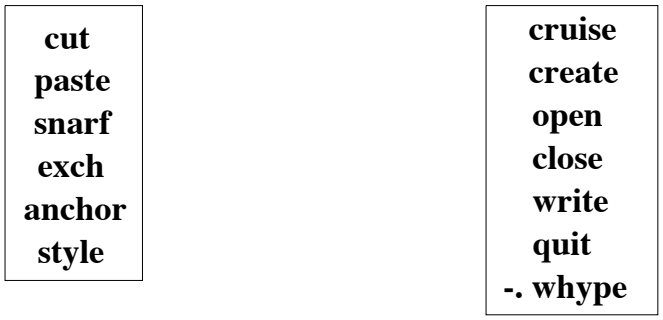


Figure 6.4: Mouse button menus in 'create' mode.

References

In accordance with the WHype Way [4] the process of creating an anchor is kept simple (see Figure 6.5). The supported anchor type is simply a reference to the destination of a hyperlink (HREF). This still is, by far, the most frequently encountered form of anchor when traversing the Web. Initially only supporting one type of anchor was seen as advantageous, from a cognitive perspective, in not confusing the novice user.

Labels

On the other hand, once this elementary stage of use has been mastered, users may well wish to use the NAME tag which allows the start of a hyperlink to be labelled. In most applications this information is not really vital, after all the destination of a hyperlink is most often the document as a whole (a client displays the start of the document). Nevertheless, giving an anchor a NAME allows that positions within the document to be used as a destination points of a link. This feature is of particular use when accessing large documents. The ability to traverse internal node links is also valuable. WHype recognises the value of such flexibility by parsing NAME information and re-generating it on when it writes out a file. Nevertheless within WHype itself, it is not yet possible to alter or add NAME information.

Needless to say, the line had to be drawn at some stage between what was feasible and what was crucial under considerable time constraints. No doubt, supporting NAME anchor fields will be necessary with any extensions to WHype.

[4] 'The simple things in life are often the best.' Kellogs

Manipulating anchors

An anchor, displayed in a WHype document, can be manipulated in three main ways. These are

1. Modifying an existing anchor name
2. Modifying an anchor's reference address (HREF)
3. Traversing the anchor to its reference document

Each of these actions is equally important in the context of a hypertext editor. Therefore, it was felt that to make one of these actions the default, so to speak, to use the double click action would be imposing. Interface design should strike a balance between providing the necessary functionality and inflicting the author's personal style on the user. WHype does not overload the double-click function with any action, it is used only to select text in the standard fashion. The actions of traversing or modifying an anchor reference are given equal weight in WHype, in terms of the amount of user input required to execute them. This reflects the anticipated mode of using the editor efficiently; quickly traversing a hypertext structure, modifying and creating anchors at will.

6.4.4 Lists

I tried to implement features that could be expressed in as simple a means as possible. Tab characters hold no significance within an HTML document, this character can therefore be used to indicate some special significence to WHype. I chose to use this to denote a bulleted list item because it appeared that this was one of the more common formatting devices used in existing Web material. It is very hard to keep subjective vewpoint separate from any product. The decision to include bulleted lists as a fundamental feature was also guided primarily by my personal belief that the list is a valuable cognitive device. I found lists of related hyperlinks to be a useful navigation tool that was used much in the same manner as a table of contents is. This feature may not be appreciated by everyone but it seems to have been exploited by WHype users so far.

6.4.5 Style

WHype, at present,supports only a subset of HTML. Unlike tkWWW, a single mechanism is used to enter in style information. Users type in a single character (corresponding to the first letter or header style number) in the pop-up window dialogue box. This mechanism obviously requires documentation so the available tags are emphasised in the User Manual. At present WHype supports

1 Header style one.

2 Header style two.

b Bold font.

i Italic font.

As with the sam command window, this pop-up window could be used to access a large variety of commands should WHype be extended to require them.

6.5 Using World-Wide Web libraries

I had intended to use the public domain libraries provided by CERN to WWW developers. This would have allowed me to concentrate on new authoring facilities rather than replicating existing functions. However, in practise, this proved to be very difficult as the WWW code was highly structured [5] and difficult to interpret. This was, in part, due to my inexperience in tracing code on this scale which was able to compile on a number of platforms.

I was not able to incorporate my data structures into those used by the WWW library without rewriting my entire program. This meant that I had to design my own routines, guided by the above mentioned code, to do the following tasks.

- Parsing HTML documents.
- Communicating with Web servers using HTTP.
- Recognising URL formats.
- Generating HTML from internal data structures.

An independant approach to coding meant, however, that WHype should be easily portable to the Plan9 operating system. Nevertheless to maintain compatibility with the HTTP2 and HTML+ specifications, it will be necessary to continually revise the program or port the public libraries themselves to Plan9.

It would obviously have been a lot more convenient had I been able to use these libraries. Nevertheless the Web libraries reflect the growing size of the infrastructure being assembled to allow complex hypermedia construction. WHype does not really fit the role of a fully-featured hypermedia workstation. It was never conceivable that I would be able to replicate the efforts of a fully featured Web client like Mosaic for X. Furthermore, not being able to

[5] WWW public domain code is written in C and structured in a macro-intensive fashion to facilitate its eventual translation to C++.

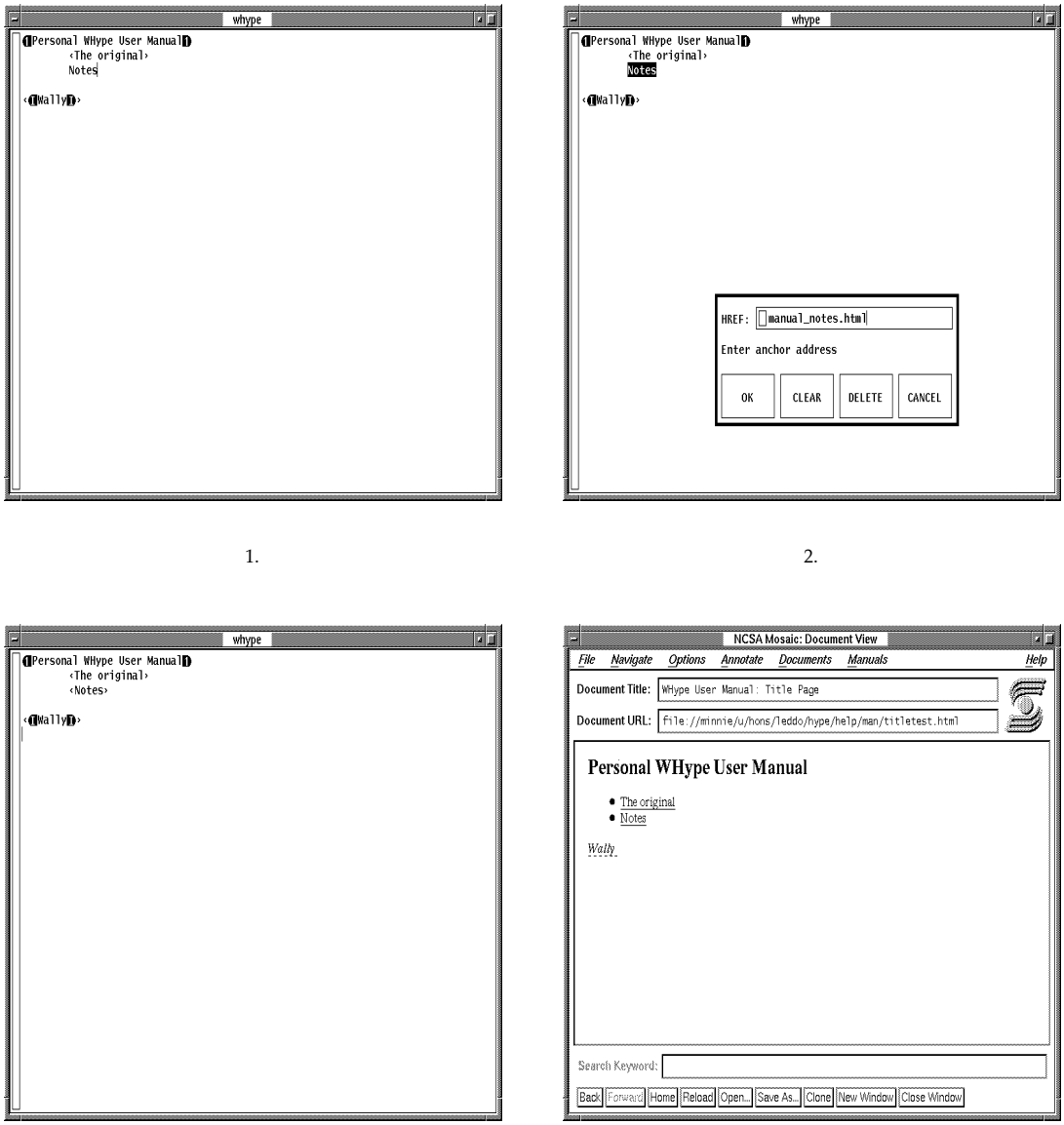
rely on WWW code gave me some perspective on how useful a lot of the implemented features actually were. Rather than being able to call on all that had previously been done, I was forced to look closely at the available functionality and evaluate which features were truly essential to an authoring tool. After some consideration, it would appear the hyperlink itself is the only feature that really falls into this category. Future work on hypertext authoring tools should concentrate on investigating this further.

6.6 Intended use vs. actual use

The WHype interface design aimed for simplicity. This was to give immediate benefit to those familiar with the sam-style interface, but it was nevertheless quickly frustrating to those users who want to produce sophisticated HTML material. It was perhaps, naively, assumed that WHype's limitations would be overlooked due to its ease of use. Feedback from the first WHyperactive [6] was a good indication of how the editor would be accepted by a capable user group. The immediate reaction was favourable; WHype provides sam-style editing as well as an intuitive means of creating hyperlinks. However the facilities offered by the editor are limited in their flexibility and scope. From the reactions of the two users who produced hypertext with WHype,it would appear that while the editor interface might be satisfactory in principle, formatting and style support was found to be lacking.

Nevertheless this feedback enabled me to use the remainder of the project's duration to implement some of the suggested features. This process could have continued indefinitely until the entire HTML specification was supported.

[6] WHype user?!



1. 2.

3. 4.

Figure 6.5: The steps involved in creating an anchor in WHype. 1) Enter text. 2) Highlight it and select 'anchor'. 3) Enter a URL to link to. 4) XMosaic presentation.

Chapter 7

Conclusion

7.1 WHype under the microscope

In this section, I outline the major design flaws in the WHype editor and suggest the direction that any future work in this area should take to avoid the same problems that I had this year.

7.1.1 Parsing HTML

WHype was always seen as an editor supporting *basic* HTML features. As suggested by WWW literature [2] tags not recognised by my parser are disregarded. I chose to interpret this as meaning "don't display the unknown tags". This was consistent with the XMosaic presentation. This also meant that comments could be included in a node by surrounding text with angle brackets. Nevertheless the continual expansion of HTML throughout this year proved to be a problem. There were many desirable features, such as in-line image handling, that were used in local hypertext yet not directly supported in WHype. Any node that was loaded into WHype, and subsequently saved, would therefore lose any unsupported tag information. On hindsight, WHype would have been more useful if it displayed any unknown tags as part of the text.

7.1.2 Parsing URLs

The URL specification [3] supports a wide range of protocols. In the process of building WHype, I focused on recognising HTTP document requests and any relative path addressing. Once these were working, I then began to extend the URL parser to recognise a wider variety of URLs. On hindsight, I should have built the full URL parser at the start of the implementation phase even if some forms were still not recognised.

7.1.3 Manipulating anchors

The TkWWW browser/editor removes all anchor information when cutting, and subsequently pasting, text. Like many aspects of the program, this appears to be more of a 'quick fix' than any concerted effort to support hypertext editing. However, due to time restraints, WHype does not offer any better approach. The cut and paste facilities in WHype cannot deal with text containing anchors. Any further work on WHype should involve parsing any 'snarfed' text for anchors (that may need to be copied or removed from the text). This would allow a user to transfer whole sections of hypertext.

7.1.4 Bad Markup

The WWW development team have emphasised that any HTML-generating applications should take care to produce HTML with care to relieve the pressure off client developers to detect 'bad markup'. The code generating HTML is guilty of inserting too many paragraph markers into the text ($\langle P \rangle$). While these extra tags are simply not used by most client browsers, this is an example of bad markup specifically noted by the HTML RFC. [9] With more time, this eccentricity of WHype could be fixed.

7.1.5 Presentation

The lack of emphasis placed on developing WHype's visual appearance is clear. The special character tags, used to indicate the presence of markup in the text, do not improve the presentation of HTML material. This style of presentation, while adequate for a prototype, is hardly aesthetic. A document which contains a lot of HTML looks quite cluttered in WHype. Nevertheless once libtext supports multiple fonts, it will be a simple matter to convert WHype to display the fonts themselves rather than tags.

An alternative approach to the multiple font problem posed by libtext would have been to 'fake' additional fonts by inserting extra style character sets (like bold and italics) into unused sections of the unicode character set. This would have been an obvious short term solution to the limitations of the text library. This option was not considered feasible at the time, although it would appear to provide better presentation than what is now available.

7.2 Relevance

7.2.1 Who is it designed for?

The sam-style interface was designed by programmers for programmers. The challenge, as I saw it, was to take the tools used to build applications for

programmers and use them to create a product that did not reflect its technical heritage. The WHype interface, itself, makes only one concession to the novice user, the pop-up window. Its display is almost as minimalistic as that which is offered by the sam editor. Nevertheless the WHype user manual attempts to directly address issues of cognition through a tutorial-style guide which blatantly tries to disguise the technical infrastructure of the WWW behind colloquial language and colourful illustrations; thus hopefully not restricting the use of WHype to a fraction of the general population (programmers). By not providing explicit help facilities in the interface, it was hoped that WHype would avoid the often-made criticism of 'friendly' interfaces that the features, which make them easy to learn, are distracting once learnt.

This desire was not guided by commercial instincts rather than a deep-rooted conviction that this is the direction that software design must take if it is to be of relevance to the public at large. Needless to say, WHype falls short of satisfying either the novice or the experienced computer user. The state of the problem can be summarised as follows.

The novice user (someone who has never used the UNIX operating system, let alone the sam text editor) is required to interact within a text editing environment that is completely alien to the style of text editing presented by commercial software. Without wishing to recommence an interface and design discussion, it is clear that unfamiliarity with an interface can be as effective a cognitive barrier as never having used a computer at all.

At the same time, WHype does not satisfy the needs of experienced users. As mentioned earlier, the first WHype user was quickly consumed by the desire to create material that was outside the scope of this program.

Does this mean that a year's work has been in vain? In the context of Computer Science Honours it is perhaps an unreasonable ambition to build a product that will be embraced by your peers. After all, it is within the scope of the course to attempt to challenge the notions of perfection; if a program was not ideal for a task, you would write one that was better. For this reason, it was never felt that WHype need cater for the needs of Computer Scientists. Whether this approach was lazy, naive, or otherwise mis-guided is for others to judge. Nevertheless, WHype set out to introduce people, not programmers, to the concept, and freedom, of authoring hypertext.

7.2.2 Back to basics

In keeping with the philosophy of the World-Wide Web project, WHype is an editor for 'spreading the word'. Simplicity has a certain strength when it comes to computers. For many people, the sheer complexity, and potential, of computer systems is mind-boggling. This is not a state of mind that encourages familiarity and comfort WHype tries very hard to be non-threatening. Even the minimalistic look of the program is an attempt to draw parallels

with the challenge presented by a blank sheet of paper; anything is possible but its entirely up to you what you do with it.

The functionality of WHype in its present state is barely sufficient to allow the creation of simple hypermedia. This statement bears closer examination. This means that WHype can be used to create a previously unavailable form of human expression, one that can be linked easily into a global information network for users on other sites to access.

Clearly, the presence, or absence, of desirable features in WHype is inconsequential to the fact that this expressive power is now available.

At the same time, it is obvious that WHype is merely an simple application of this new approach to information dissemination that this author can take no credit for. Rather than examine the existing WHype editor in too much detail, it would be more productive to look at exactly what directions hypermedia applications will be taking. Of course this discourse will not be totally unaffected by my personal viewpoint.

7.2.3 A warning

As mentioned throughout this paper, both the HTTP and HTML specifications are constantly being revised. Discussions on the *comp.infosystems.www* newsgroup and the *www-talk* mailing-list indicate that this expansion of the World-Wide Web infrastructure will probably continue for some time. HTML+ gives information providers a variety of new features to incorporate into their hypermedia. This has, however, introduced new levels of complexity into what was previously a simplistic method of disseminating information. From the scientific perspective, this can only be a positive step; the next few years should see a dramatic rise in, not just the quantities, but the sophistication of hypermedia resources.

Nevertheless, this does not come without a price. The World-Wide Web is revolutionary in the sense that it allows people with little or no computer experience to use global information systems. The WHype client was aimed at encouraging this level of use by providing the tools to create basic HTML documents. It could be argued that the WWW development community may be moving this technology out of the realms of public comprehension with their efforts to make HTML more 'valuable.'

It is entirely possible that people, who may have otherwise experimented with the possibilities offered by the WWW, may be dissuaded from doing so. [1]

[1] The original version of the NCSA HTTP server, used in this project, required a configuration file of six lines. The latest release, still in the process of being configured, uses three configuration files (around two hundred lines).

7.3 Extensions

7.3.1 Additional modes

In developing a modal approach, I attempted to foresee how an authoring tool like WHype might conceivably evolve. As new modes of interaction with the Web are enabled (and subsequently supported in WHype), they could be added to the third button menu. Each mode would then have its corresponding tool menu (on button two). Most of the suggestions below are already feasible if not immediately pragmatic. Possible future modes would include

'comm' Hypermedia presentations of mail and news groups. Already HTTP2 supports server interaction with local host services. [2]

'compose' Music composition and recording tools and other audio editing facilities.

'cinema' MPEG animations are already supported in WWW. Future applications could link to dedicated animation editing facilities.

'custom' Options allowing users to customise their clients .

Clearly there is much scope to for a client like WHype to use a wide range of creative tools much in the same way that XMosaic can call upon external viewers and audio players.

7.3.2 Supporting HTML+

The new HTML+ specification has introduced a new level of sophistication to the World-Wide Web. The capabilities of modern workstations have inspired the recent [3] inclusion of varied features such as *ISMAP* (the facility to utilise hotspots (regions) on any bitmap as hyperlinks), and *FORM* (dialogue boxes, radio buttons and other input devices can now be included within a HTML document). The lack of certain facilities within this department (specifically audio support and colour terminals) should not be viewed as reason to ignore the possibilities that these hypermedia components offer. A hypermedia authoring tool that incorporated sound editing, bitmap drawing utilities, and well as hypertext support of the kind attempted by WHype cannot be far off.

[2] The World-Wide Web infrastructure does not yet support direct client-to-client communication. Otherwise, facilities like 'talk' would be desirable.

[3] October 1993.

7.3.3 Computer Aided Authoring

This is not as redundant a concept as it may seem. There are lessons to be learnt from commercially available CAD/CAM [4] systems designed for non-programmers. As use of hyperspace becomes more and more sophisticated, the need for specialised tools, supporting the creation of complex hypertext structures, should become more apparent.

WHype does not satisfactorily address the problems involved in structuring hypertext. This is partly due to the program's mimicry of a conventional text editor. Obviously the authoring facilities of the future need bear little resemblance to contemporary editors just as a hypermedia presentation bears little similarity to a newspaper. Utilities that specifically address the complexity in structuring media presentations will also need to be developed. This has already been attempted with hypertext [13] although it would appear that the boundary between utility and contrivance is still difficult to judge.

An authoring tool that automatically generates visual information about the hypertext structure (eg. a diagram showing the interconnectivity of nodes) would be both useful and quite feasible to implement. Certainly as non-linear material becomes more and more familiar, the need for further navigation and visualisation tools will become clearer.

[4] Computer-Aided Design/Computer-Aided Modelling.

7.4 The last word

This paper has not been written from a purely science background. The research that has gone into this project has not just concentrated on analysing the technical side of the World-Wide Web. It is perhaps even appropriate that this analysis of the WWW attempts to provide a more cross-disciplinary perspective to a computer science paper than normal. After all, WHype directly supports contextual links through hypertext. I have attempted to examine hypertext in a wider context.

A lot of what has been discussed here will be of little relevance to some scientists. This should not come as any surprise. Just as many people still reject the notion that a computer has any part to play in their lives, others will have difficulty in seeing beyond the technological achievements of distributed networks. Nevertheless, cultural, as well as information, infrastructure is already in place. It would appear that attempts to restrict the use of the Internet to the scientific disciplines are doomed to failure.

Without launching once again into hyperbole, it should be clear that as more and more people become familiar with current standards in interface and personal computing, the level of sophisticated use will continue to spiral upwards. Perhaps it is still too early to try and predict what relationship non-programmers will have with computers in a society that cannot function without them. Perhaps the current trend of dependence will continue. Already computer malfunction within the public domain is often treated as a catastrophe - witness the chaos within any large supermarket when the check-out computer goes down. Isaac Asimov's fictitious empire, which collapses under the cumulative weight of ignorance regarding its infrastructure, is no longer merely science fiction.

Modern computer science research cannot continue in isolation. It would appear that interface designers control the destiny of society. Without the ability to relate to, and interact with technology, many people will become increasingly alienated from mechanisms more and more prevalent in their lives. The implications of distributed hypertext systems are therefore relevant to the general population. Even if the the public have not yet access, or even knowledge, of technological advances such as global information systems, this does not mean that academics should assume that their current isolation from outside influences will necessarily last.

The World-Wide Web project is actively promoting a virtual environment that directly supports the eventual widespread use of computers as tools. Prototype exploratory tools like WHype will be needed to investigate this hyperspace more intimately. The next decade of distributed information systems development promises to be interesting.

Michael Ledwidge
http://minnie.cs.su.oz.au:80/hype/MFL.html

Appendix A

Terminology

There are many terms used in this paper which may not be familiar. Most of these are now directly associated with the WWW project.

Anchor A region within a document that is designated as the source or destination of a hyperlink. In the context of a browser, selecting a anchor (typically by clicking on the region) allows the link to be followed to its destination. The usual convention is to highlight the anchors in hypertext through some means. The destination of an anchor can either be a node itself or another anchor within that node. At present anchors are one-directional, ie. a link to a node does not necessarily imply that there exists a reverse link in that node.

Authoring The process of creating material for the Web. More than just editing, this may involve designing a local web structure, creating links, and preparing audio and graphical material.

Browser Client software that allows a user to access and interact with the World-Wide Web.

Hyperlink An established relationship between two nodes that allows quick traversal from one to the other. Hyperlinks can link material on different sites.

Hypermedia Implies hypertext which is linked to other non-textual material. The Web supports audio, graphical, and animation material as well as text.

Hyperspace The exponential growth of the Web has produced a similar explosion in the number of terms used to describe the global information environment it encompasses. I have adopted this term for my own usage, which, although unoriginal, is less immediately traceable than the often-used term **cyberspace**, first used by the novelist William Gibson to describe his vision of a virtual information landscape.

Hypertext Any text which contains links to other text. The material is not necessarily intended to be read in a linear fashion.

Hypertext Markup Language (HTML) By inserting markup into plain-text, display-based information as well as hyperlink information can be represented. Browsers recognize the markup according to their system capabilities; systems that do not support certain features (eg. sound) can ignore the corresponding markup.

Hypertext Transfer Protocol (HTTP) This is the backbone of the Web information structure. According to the original specification (HTTP0),an HTTP server receives a request for a node (in the form 'GET $< filename >$') and either returns the node itself as stream data or an error message. This protocol is presently being revised.

Hypertext Transfer Protocol Daemon (HTTPD) The standard Web server. This program validates client requests for local site information and sends back a copy of the particular resource or an error message if unavailable.

Internet The physical 'network of networks' connecting 1.7 million [1] sites world-wide. Not synonymous with the World-Wide Web.

LibXg The plan9 graphics library ported to UNIX. The functions contained within were used to build the WHype interface.

Libtext The plan9 text library ported to UNIX. As with libXg, these were used in WHype. As yet the library only supports the use of a single font at a time.

Node One of the many terms used to indicate a specific unit of information obtainable through the Web. This may be a hypermedia object. The term 'document' is often used in the context of hypertext for the sake of familiarity.

Plan9 (from Bell Labs) A distributed operating system under development from Bell Labs. The ported UNIX graphics libraries (libXg) are available from research.att.com.

Traversal The process of following a hyperlink from its source node to the destination.

Universal Resource Locator (URL) Extension of the filename concept to include a format code, Internet address and port number details. The variety of accepted URL formats ensures Web access to a large number of Internet resources.

[1] Statistics August 1993

Appendix B

A sample HTML document

This appendix contains the source HTML for the Basser Local Home Page. Figure 4.1 shows the corresponding display in XMosaic.

```
<Since HTML parsers ignore any markup they do not understand,
one can put comments in angle brackets>
<TITLE> LOCAL HOME PAGE </TITLE>
<H1>Welcome to HyperSpace</H1>
<IMG SRC="http://minnie/hype/crest.xbm">
This is an experimental
<A HREF=http://info.cern.ch/hypertext/WWW/WhatIs.html>
hypertext
</A> server connecting the
<A HREF=http://minnie.cs.su.oz.au:80/basser/basser.html>
<B>Basser Department of
Computer Science</B>
</A>, at the
<A HREF=gopher://gopher.su.edu.au:70/11/su><B>
University of Sydney</B>
</A>
, to the
<A HREF=http://info.cern.ch/hypertext/WWW/TheProject.html>
World Wide Web
</A>.
<P>
From this page of information
you can find yourself anywhere in the
Internet. The resources
of the Web are at your fingertips.
<P>
<A href=http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/help-about.html>
XMosaic
</A>
 is a powerful navigational tool that allows the user
```

```
to access most network and local
<A HREF="http://minnie.cs.su.oz.au:80/hype/sources.html">
information services
</A>
via a hypertext interface.
<P>
First time users would be advised to browse the ‘Documents’ and ‘Help’ menus
for
background material.
Also, read the
<A HREF=file://minnie/u/hons/leddo/hype/help/nav/intro.html>
<B>Cruisin’ the Web</B>
 tourist guide
</A>.
<P>
There is a prototype hypertext editor
<B>WHype</B>
that you can try out (~leddo/bin/whype ).
If you want to join the elite group of
<A HREF="http://minnie.cs.su.oz.au:80/hype/help/jasonp.html">
Basser Hyperjunkies
</A>
who are
actually trying to USE this bodgy program, let me know.
<P>
Current forays into the WWW are:
<UL>
<LI>
<A HREF=http://iicm.tu-graz.ac.at:80/Cjargon>
Jargon Index
</A>
(for hackers and wanna-bees)
<LI>
<A HREF=gopher://mothra.nts.uci.edu:7000/11/gopher.welcome/peg
/VIRTUAL%20REFERENCE%20DESK>
Virtual Reference Desk
</A>
(assorted US-based reference material)
<LI>
<A HREF=gopher://gopher.uwp.edu:70/1/pub/music>Music Archives</A>
 (‘so that’s what ’es singing!’)
<LI>
Browse the
<A NAME=Archie.au HREF="ftp://archie.au">
Archie.au
</A>
FTP site
```

```
<LI>
Telnet into
<A NAME=SU HREF="telnet://library2@lib2.su.OZ.AU">
Fisher
</A>
or
<A NAME=NSW HREF="telnet://libcat@libprime.libsys.unsw.OZ.AU:23/">
UNSW
</A>
library
<LI>
Look up files available by Anonymous FTP with
<A NAME=ARCHIE HREF="http://web.nexor.co.uk/archieplex">
ArchiePlex
</A>
<LI>
Virtual Reality.
<A NAME=JAY HREF="http://minnie.cs.su.oz.au:80/hype/virtual.html">
Jack in.
</A>
</UL>
<P>
Any queries or suggestions, feel free to hassle
<A NAME=3 HREF="http://minnie.cs.su.oz.au:80/hype/MFL.html">
leddo
</A>.
```

Bibliography

[1] Tim Berners-Lee. An overview of hypertext and ir systems. Available on the World Wide Web, URL: "http://info.cern.ch/hypertext/Products/Overview.html", May 1993.

[2] Tim Berners-Lee/CERN. "the world-wide web book". Hypertext documentation on the WWW, not available on paper, is cited here by URL references.

[3] Tim Berners-Lee/CERN. *Uniform Resource Locators*, March 1993. work in progress.

[4] P. J. Brown. Assessing the quality of hypertext documents. In *European Conference on Hypertext, INRIA, France*, November 1990.

[5] Ronald H. Brown. National information infrastructure agenda. Available on the World Wide Web, URL: "http://sunsite.unc.edu/nii/NII-Executive-Summary", May 1993.

[6] Clive Davidson. The man who made computers personal. *New Scientist*, 1933.

[7] Dan Heller. *Motif Programming Manual*, 1991.

[8] Kevin Hughes. Entering the world-wide web: A guide to cyberspace. Available on the World Wide Web, URL: "http://pulua.hcc.hawaii.edu/guide/www.guide.html", September 1993.

[9] Tim Berners-Lee/CERN Daniel Connolly/Atrium Tech. Inc. *Hypertext Markup Language*, June 1993. work in progress.

[10] Cliff McKnight. *Hypertext in Context Information filtering and information retrieval: two sides of the same coin.* Cambridge University Press, 1991.

[11] Rob Pike. Window systems should be transparent. *Computer Systems Summer 1988*, 1(3):279–296, November 1988.

[12] T.J. Berners-Lee / R. Cailliau / J-F Groff / B. Pollermann. World-wide web: The information universe. *Electronic Networking: Research, Applications and Policy*, 2(1):52–58, Spring 1992.

[13] W. Schuler & J. Smith. Aaa: A hypertext-based authoring tool for argumentative texts. In *European Conference on Hypertext, INRIA, France*, November 1990.

[14] Joe Wang. tkwww: Readme. Available on the World Wide Web, URL: "http://info.cern.ch/hypertext/WWW/TkWWW/README.txt", 1993.