ROCHESTER INSTITUTE OF TECHNOLOGY

A thesis submitted to the Faculty of
The College of Imaging Arts and Sciences
in Candidacy for the Degree of
MASTER OF FINE ARTS

Intelligent User Interfaces and the Internet

by

William F. Colgrove

May 1995

Certificate of Approval

M.F.A. Degree Thesis

9.6.95

Jim VerHague, (Thesis Committee Chairman)

9-6-95

Bob Keough, Thesis Committee Member

9-8-95

John Ciampa, Thesis Committee Member

9-20-95

Mary Ann Begland, Chairperson of the Graphic Design Department

I ______________________, hereby grant permission to the Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Intelligent User Interfaces and
the Internet

William F. Colgrove

ABSTRACT

The purpose of my thesis was to have the opportunity to explore the Internet and to attempt to formulate and put into use ideas which would enable the user to sort through the vast amount of information which the Internet has put at our fingertips. I examined the culture of the Internet as well its various uses and difficulties encountered by having so many different types of machines connected to one network. I used the Macintosh computer for research on the Internet as well as a method for presenting and distributing my ideas. I studied the public's perception of the Internet through its exposure in mass media and made a conscious effort to dispel the hype and fictions surrounding the Internet.

Table of Contents

Introduction:

I must begin by saying that the purpose of my Thesis project evolved and changed. I began with an interest in my chosen subject, this new form of communication and all that it had to offer. I was frustrated and a bit disappointed by two things: a lack of standardization of the content, and no immediate feedback. This caused me to attempt to develop a hypothetical interface for exploring the content of the Internet as well as creating an interactive multimedia project which informed the user about the Internet by using the interface that I developed. Over the course of doing research I am glad to say that I was able to get a glimpse of the future of the Internet and see it come true. The advance of computer technologies has had a great impact on our communication environment in the past few years. This rapid growth in connectivity and communication has not been accompanied by organization of content.

Reasons:

The Internet, despite the fact that it has been around and in widespread use for almost 20 years, has only just begun to have its full impact on day-to-day life. As an educational tool, the possibilities are limitless. As Ted Nelson outlined in Project Xanadu there would be a need to create a front end to this library of knowledge. I wanted to try my hand at doing that.

It is essential to understanding the nature of the Internet that it is chaotic. There is a lack of order to its structure which makes it what it is. This chaos calls for some form of order. Although there will probably never be one 'killer application' or definitive tool for Internet navigation, great strides have been made toward developing a system which is fast, effective, and can work on all machines. It is noteworthy that many machines which can connect to the Internet are machines which are capable of displaying only 1 bit of color. So it was clear that my thesis project would not be meant for all machines, because the platform which I chose to develop the program for was the Macintosh. I was most familiar with the available Macintosh tools for developing such a simulation.

Although it seems like the Internet and the Information Superhighway should be accessible to all people everywhere it is painfully clear that electronic modes of communication are either provided for people by an employer or a university which is usually free to the user or by means of a dial up service which is paid for by the user. So we have what amounts to a class of people which are most likely employed, educated, and/or wealthy enough to afford a home computer, a modem and the means to pay for a monthly connection fee. Nonetheless this does not stop the daily influx of new users onto the Internet. A whole new culture has emerged, a new means of publishing. The Internet is certainly a way of cutting out the middle man in many instances and that is exactly the reason why many companies are seizing this opportunity to stake their claim on the Information Superhighway's roadside, which is all too easy due to the lack of governmental regulation of the Net. At this point strict government regulation would be disastrous to the spread of global electronic communication. The Internet has made the world a much smaller place to live in. Letters can now take minutes to reach their destination. Digital files can be transferred with sound, graphics and movement. People from different places and different backgrounds can meet and discuss ideas, technology, the latest music or trends. Never before in the history of the world has this kind of widespread instant communication been possible, and the Internet is still in its infancy. All of these things are my reasons for choosing to use the Internet as the basis for my Thesis. The Internet was clearly where communication was headed and I wanted to know more about how it worked and where it was going.

Because the Internet is a means of communication it is important that the interface fulfill a role socially as a tool for communication. The interface for the Internet should, at a certain level, function like a telephone would. Many on-line services offer a means for people to meet and chat while logged in to the 'Net. With the abrupt rise of electronic communication there was also a shift in attention to syntax. Mass communication had moved a vast distance from the written word. Television, fax machines, photocopiers and especially the telephone had caused an evolution of language. When sending an electronic mail message or posting a message to one of the Internet's many discussion groups or news groups, writers of these notes soon became aware of the consequences of ill-chosen words. New language surrounding the culture of the Internet appeared; "newbie" for new user "IMHO" an acronym for In My Humble Opinion or "flame" for openly ridiculing or blasting another person or their point of view. All of these things in one way or another contributed to the idea that an interface for the Net must be unifying and variable, that all people can use it regardless of their skill level and still not feel lost or hindered in any way.

Research:

About three years ago I was introduced to the Internet. Before too long it ruled my life. I was fascinated by the variety of content and conversation which ensued day after day. I soon became aware of other electronic services which offered a more user friendly front-end to the Internet such as America Online, CompuServe and Prodigy. Most of these services were for the home and few at that time offered full-access to the Internet and its contents. While spending a summer working at an on-line newspaper I got wind over the news line of a graphic interface for the Internet: a point-and-click tool which was easy to use. It was called Mosaic and it was produced by students at NCSA in Illinois. When school resumed in the fall I made it a point to see if it was possible to get our computer lab wired up to the campus Ethernet, thereby hooking us directly to the Internet. As of this writing much time has passed and Mosaic and the Web have emerged as 'the next big thing'. It has given birth to offspring all operating on the same method of reading files which display text and graphics which can link to other files located just about anywhere in the world. The Web, which was once an experiment for education and research, largely similar to Ted Nelson's Xanadu, has become a widespread attempt to create the shopping malls of the future. The Web is the most visceral and immediate of all online media which is probably why it is the most successful.

It was important for my thesis program to work at some level as its hypothetical fully implemented version really would function on the Internet. If this program were fully developed and implemented for the 'Net it would act as an Intelligent Agent (IA) for the user, meaning that the program would actively edit which content reaches the user's computer based on the type of decisions which the user makes and what kind of information he or she is looking for. This could be facilitated by asking the user a series of transparent questions concerning his or her search or by building in some form of user log-in for each member of the family. Parents could exercise control over what kind of material that their children had access to, an option which has become available in many online services. The program would also be its own best judge of monitoring network traffic and speed, having an internal clock which would monitor time elapsed to make requests and transfer files from other servers. This file could be maintained on the computer's hard drive and serve as a list of what kind of information is where. For example, if the user keeps looking for information on a certain topic, the computer would know where the quickest and most reliable places were to get that information. For purposes of my thesis, however, I chose to provide the user with three varying levels of content in the form of articles which were acquired from the pages of Wired, Time and the Internet itself.

Macromedia Director was chosen because it was emerging as the standard for developing multimedia titles and it was something I felt that I needed to know in order to find a job in the industry. I first had to construct a list of topics which I wanted to cover. I spent a great amount of time on the Internet, reading the news groups and learning how to download files. Much time was also spent on the campus VAX system. The files which I found were text since most of the terminals on campus have monochromatic monitors. I then spent some time using Dartmouth University's Fetch program for the Macintosh.

Over the course of creating my thesis it became less of an application, as I has originally intended, and became more of an interactive program. I began with the concept of a great deal of user cross-referencing within the program and spent an equally great amount of time developing the hypertexting system for Macromedia's Director. Hypertext is a word for files which are linked, (these files may be in any format; sound, graphics, text) in terms of content as well as their location, so that one may be accessed from the other. This method of learning allows the mind to explore more freely and follow the paths of the topics which most interest the user. In addition to developing the hypertext system I also developed a method which allowed the computer program to remember where you had been, or which sections you had visited and present you with a list of these documents. When an item from the list was chosen the computer would return you to that section or open that document. Unfortunately these items did not make it into my final program but I have included the scripts here since doing the programming to create this was invaluable to my learning how to program in Director.

Hypertext is the basis of the World Wide Web. All information is linked to more information, allowing the user to cross-reference any number of works in a matter of seconds. The main problem is that someone, somewhere has to establish those links. Learning is enhanced by letting the user explore the aspects of a subject that he or she finds interesting. Since hypertext is employed not only by a great deal of educational multimedia titles currently on the market and, in fact, is employed on the Internet itself, it seemed to be a natural choice to adopt it for use in this program.

Part of the goal of this thesis was to be a hypothetical user interface for navigating the Internet. Since the Internet is comprised of hundreds of thousands of files with varying file formats flexibility was a great issue. When constructing the user interface, I wanted to give the user as many options as possible for the retrieval of files and the presentation of information. The whole interface was constructed in terms of being flexible. Placeholders were used where data, images, or sound could be played or previewed before they are downloaded to the user's machine. Part of the problem of many dial up services which offer Internet access is that they often provide no means of previewing a file to see whether or not you want to take the time to transfer it to your machine. Multiply this by the fact that most machines which comprise the end-users of the Internet are UNIX or PC based machines. In many cases file formats are not compatible or recognizable across different platforms, and are therefore rendered useless to the casual user. There are utilities which will convert files from one platform to another but these usually require the user to have some form of technical knowledge.

The project was constructed so that the user would be asked a question, or series of questions, in order for the computer to help determine the user's skill level. If fully implemented the computer would ask a series of transparent questions or be smart enough to identify the users on a home machine by having a record of the occupants of the house and the users on the machine.

Once the user's skill level is set the computer then knows what information to present to the user from a list of all available files. The list was of a series of ASCII text files which contained definitions on the uses and the features of the Internet. The ASCII text format was chosen because it is the file format which allows Director to "see" the file so that it may be retrieved and flowed into a text field which served as a placeholder for the information. This text is read into Director unformatted, without text styles, sizes and so forth.

In order to solve the problem of informing the viewer which words are hypertext since I could not bold, italicize, underline or color the words in the text file and I could not efficiently script a filter to do so in Lingo, I chose to place the words which were sensitive in all caps to alert the viewer to the fact that they were hyperlinked. The Apartment, which is a series of demonstration movies done in Director with Lingo, proved to be quite useful in determining what can and cannot be done with Director to achieve the goal of hypertexting.

The early scripts for the hypertext amounted to getting the computer to recognize the word or successive string of characters which the mouse clicked upon. Once this string was recognized it was placed in a global variable called an array. The array was then compared to a list, which was stored as text in a field of possible destinations which did not appear on the screen.

At this point in development there was a unique problem because the program did not really "go" anywhere in the spatial sense. This program could not be mapped in the traditional manner as is often done with interactive programs. There are no different areas for the user to enter, just a static interface. Since there is no variation of the position of the elements of the screen, the user quickly becomes familiar with the places where information is entered and where information is presented. I felt that forcing this familiarity was absolutely necessary in order to make the user comfortable with what can be an overwhelming and intimidating subject matter.

Because the program does not have screen variation in which different screens could be crafted to present different levels of information to the user, all the variation in information had to be handled through scripting the computer to retrieve the files which were appropriate to the stored skill level of the user. The user could then click on the ASCII text file and expect to receive some additional information by way of links on any word which was in all capital letters. No matter where the word appeared in the program it could be clicked upon and the user could expect to go to a screen OR be presented with more information about the word which was clicked.

These links would be a synthesis of hard and soft links. A hard link is the kind of link that is not variable, it is an absolute. This kind of link will always take you to the same place in the program no matter where the user is. A soft link is the kind of link that is variable and may take the user to one place when he is on one screen or somewhere different when he is on another.

If the comparison returned "true", meaning there was additional subject matter on the word which was clicked, the computer then disposed of the opened text file and sent the program to the screen which contained this additional information. If the comparison returned "false" the computer simply beeped, informing the user that word was not a link.

Having completed this much of the program my next task was to create a user-friendly environment which enabled the computer to "remember" which sections had been opened and allow the user to choose from a list of those sections in the order in which they were visited. This would give the user the chance to page back and forth between the places that he or she had visited.

This idea was triggered by NCSA'S Mosaic which allows the user to select from a pull menu the pages on the Web that he or she has visited and return there. In this respect Mosaic and the Web actually encourage exploration. If users know that they can return to a given spot, electronically dog-ear the page, they will be more apt to go off on a tangent which interests them.

In order to get Director to do these pull menus I had to set up a certain number of additional invisible fields; a field for temporary storage, one for permanent storage, and one to store the contents of the pull menu. The temporary field stored the word on which the mouse clicked. The main list was the list of all the topics which I had covered in my research and which existed as possible destinations for the hypertext. The menu field was used to create the pull menu, like the one in the Macintosh Operating System (OS) by utilizing the installMenu Lingo command. A global variable was created to keep track of how many times the hypertext comparison returned "true." Each time this comparison returned true, one was added to the variable and that word was added to the menu field in the line number equal to the global variable. The menu was then installed and, through the use of an array, a command was associated with each item on the pullMenu.

Thoughts on User Interface Design:

The interface to a program, game or utility is essential to how useful the program is as a tool. One does not need much instruction to learn how to operate a pencil and paper or even a keyboard to a word processor. These interfaces are familiar to us having been used since we were children in the case of a pencil and paper or later in life in the case of a keyboard. One becomes more and more proficient in using an interface the more that he or she spends time immersed within it. But by the same token if a program interface is too simplistic the user may very well become bored with it and stop using it. I felt that at no time in the act of creation or exploration should the user be totally conscious of the interface. Programs which tend to be interface-heavy tend to get annoying.

The ideal situation would then be to create an interface which is easy to use and understand at the lower levels while, as the user's proficiency increases, so does the complexity of the computer program. A prime example would be the interface of Adobe's Photoshop, a raster-image editing and creation program. Photoshop provides a multitude of ways to get results, and as the user's familiarity with the program grows so does their ability to find better and faster ways of doing things. This type of interface challenges the user to explore and find new ways of doing things.

The interfaces of some computer programs tend to borrow tools or items from other programs. Apple computer took a bold step forward in terms of interface design when it introduced the Macintosh. The Macintosh came with a set of built-in tools that programmers could call upon when designing the interfaces for their computer applications. These built-in items include icons, pointers, and pull menus. Learning the interface of one program on the Macintosh operating system became like learning to drive a car - the tools became familiar to the user so that he or she could switch programs as easily as one would switch from driving one make and model of a vehicle to another.

The interface is the filter through which we run our ideas to transfer them to a digital format where they become much easier to manipulate and transform than on paper. The interface to a computer program must allow the maximum throughput of ideas so as not to hinder the creation or exploration process. In light of all this a computer program interface must be intuitive, flexible, and encourage exploration.

The interface is the tool with which the user navigates or manipulates the program. There hasn't really been a great deal of advancement in this area since the great flood of ideas emerged from Xerox PARC in the late seventies/early eighties. At the highest level the UI should be a tool. Like any tool it should be simple to operate, or to paraphrase Marshall McLuhan, 'an extension of the hands'. In order to combat the chaos and the confusion of the Internet as well as the general level of frustration which comes with owning a home computer and getting connected to the Internet, the interface which I planned on developing had to have a certain functionality as well as a certain look.

The interface itself was to be very clean and organized, combined with the way that it was styled and animated as to closely resemble a Personal Digital Assistant (PDA). The same summer that I discovered the Internet, Apple Computer Inc. unveiled the Newton PDA at MacWorld Boston. The device had caught the interest of the attendees as well as the news media. The Newton was a hand-held device which kept track of and organized day to day routines and information, and also had a certain toy value. The futuristic styling and animation of my interface would give the user the feeling that he/she was manipulating an actual object like the Newton.

Functionality of an Interface for the Internet:

If there is to be one application which will help to guide a user successfully through the Internet it must be able to serve many functions, and perform many tasks. When creating my program I tried to take into account many of the problems that face the novice computer operator when looking for something on the 'Net. I found this to be particularly easy since I was learning the ups and downs of the Internet as well as computing.

The interface must be capable of providing two types of choices; one would be consistent tools which the user could always rely upon in a panic situation, the other would be variable and location dependent. For example, the information stored on one machine differs from that on another and that must be made evident as well as presentable. Therefore, I would provide the user with two areas to make choices which reflect these findings.

I also had to provide the user with a method of viewing graphics, so a window was created for that purpose. Images and movies could be presented there in a smaller format so the user would not have to waste time downloading a graphics file, the nature of which is unclear.

Most current Internet applications require that you use separate external applications for viewing movies, pictures or playing back audio files. In theory, the interface and program which I constructed would have these items built into it in a way that the program would be able to import and play any file from any platform to the machine you are currently working on without having to open another application to do so and converting the file type internally. This would not require the user to know how to make an X-Windows sound file play on a Macintosh. This would also take a great deal of the guesswork and frustration out of surfing the Internet.

The other item which was included in the interface is a command line to provide feedback not only about where the user is and where he/she will go if a button is clicked. Where, in this case, means which computer is at what location on the Internet or for the purposes of the prototype what section of the interactive program.

Process:

When selecting and creating content for the interactive program I often thought of my parents as a target audience. Logically I figured that if I could get them to understand what the Internet is and how it might effect them I would have accomplished a great task.

I decided to do animations on each of the topics which I had chosen to explain the Internet and its services. The animations were created using Macromedia Director's animation capabilities, Strata StudioPro's three-dimensional modeling, rendering, and animating capabilities, and Adobe Premiere's ability to edit and create QuickTime movies.

When I began doing these animations I had planned on using Premiere to cut between or overlay text that I had animated in Director and the models which I had constructed in StudioPro, but while using these programs I stumbled on a feature which enabled me to integrate both text and image more fully. StudioPro allows the user the option, when rendering a still or moving image, of rendering the background as an alpha channel.

An alpha channel is an 8 bit grayscale image which is used for functionality rather than display purposes. In a 24 bit color (or 3 byte) image, 8 bits each are used for the Red, Green and Blue layer which when combined create a picture containing millions of colors. Additional layers of data can be stored in 32 bit images, 24 bits for viewing and the remainder is an 8 bit image which can be used to store a mask or other data.

I did not really pay much attention to the function in StudioPro until I opened an animation which I had created in StudioPro and saved as a QuickTime Movie in Premiere. Premiere gave me the option of viewing the Alpha Channel of that movie, which was a perfect mask of the background of the scene I had created and animated. After a little exploration in Premiere, I found that I could place the StudioPro animation in the Special Effects track in Premiere's Construction window and, by using the

Transparency Settings option, declare the movie's Alpha Channel as a mask. This would allow another movie or animation to appear in the background behind the objects which I had created and rendered in StudioPro.

After this, I went ahead and used Photoshop to create text, so that it had a smooth anti-aliased edge, and animated the type scrolling in Macromedia Director. I then used Director's Export function to save the animations as a 320 by 240 pixel QuickTime Movie, the same size which I was creating my object animations in StudioPro. These type animations were created to show data being sent and received by the computers which I had modeled. I used lighting, various spotlights turning on and off, in order to help show that the computers were communicating.

It was important to have the layout of the interface be as clean and as organized as possible plus be flexible and free of any inconsistencies. I made many decisions on the location and the actions of the buttons which would be used to control the program. I felt that having the buttons to the right of the interface were similar to having tabs or bookmarks in an address book, thereby reinforcing the Personal Digital Assistant analogy. They also helped to present a constant menu from which the user was able to choose. Like the Macintosh Operating System in which certain items in pull-menus are dimmed when they are not available, certain tabs would not pull out when their respective options were not able to be used. In a similar fashion, input and output were arranged in certain areas of the interface to give the user a consistent method of getting information in and out (*Figure I*).

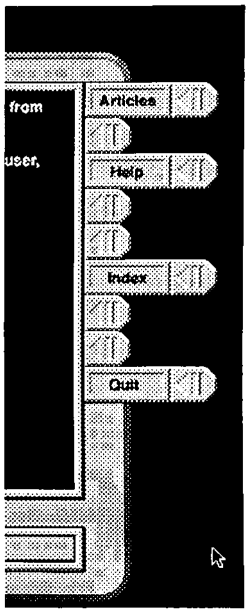

Figure I: Buttons

Animating the buttons proved to be quite a problem. In Director, once a graphic element (or sprite) is placed under the control of Lingo (called a puppetsprite), it cannot be animated traditionally in the Score Window as I was accustomed to doing. Instead of spending a great deal of time writing an elaborate routine to animate the puppetsprites through scripting with Lingo, I found another simpler and somewhat less elegant solution. The program would idle on a specific frame waiting for the user to make a choice from the buttons or tabs presented to him/her. All the available tabs are declared puppets which, when clicked would put a value into a global variable. The button would swap with a graphic highlight state and once the variable was filled it would turn off all the puppets and play a series of frames which animated the tabs closing and then an animation of them reopening depending on which button was pressed.

I decided to hand out a version of my thesis on 3.5" floppy disks at the Thesis Show. The program contained no QuickTime movies but contained all graphics, text and the interface. My philosophy behind it was that it would serve as an off-line primer for attendees of the thesis show to learn about the Internet as well as interactive media.

Conclusion:

It has been a year since finishing my thesis and much has happened to the Internet. The Internet has literally gone from rags to riches as major companies have populated it with electronic storefronts. The Web, and hypertext, has emerged as the standard method of presenting information and Mosaic has spawned a host of similar applications, like Netscape for navigating. In my research I tried to explore how information should be presented and stored on the Internet while acquainting myself with 'Net culture and protocol. There are still many subjects I discovered which I might have touched upon but could not due to time. One thing is for certain, that the Internet will continue to change rapidly over the coming years.

Bibliography

Burr, Ty. "Future Hype" Entertainment Weekly, 10 December 1993, 38-49.

Cotton, Bob. Understanding Hypermedia: from multimedia to virtual reality, London: Phaidon, 1993.

Elmer-Dewitt, Philip. "Who Should Keep the Keys?" Time , 14 March 1994, 90-91.

Elmer-Dewitt, Philip. "Take a Trip to the Future on the Electronic Superhighway" Time , 12 April 1993, 50-55.

Elmer-Dewitt, Philip. "Battle for the Soul of the Internet" Time , 25 July 1994, 50-56.

Krol, Ed. The Whole Internet: user's guide and catalog, Sebastopol: Microsoft Press, 1987.

Nelson, Theodor. ComputerLib / Dream Machines Rev. ed., Redmond: O'Reilly & Associates, Inc., 1993.

Nelson, Theodor. Literary Machines 1981.

Powell, Bill. "Eyes on the Future" Newsweek, 31 May 1995, 39-50.

Rucker, Rudy ed. Mondo 2000: a user's guide to the new edge, New York: Harper Collins, 1992.

Appendix A: Thesis Diary

12.4.93 - 1.4.94

These weeks were spent researching the kinds of topics that I will be presenting as an information database as part of my thesis. As I see it my thesis will be comprised of many parts: mainly a working multimedia environment which is 'aware' of the user's level of experience with a computer and displays information which the user will understand, a resource tool for new media and computerized communications, a stage for some projections on my part about the future of multimedia and digital networking. I assembled a library of sources which represent a broad range of audiences, from specialized textbooks to popular magazines all hyping the Information Superhighway.

1.17.94

I decided to begin by developing a understanding of Lingo since my knowledge of it would be the limiting factor in what I could do with the interactive piece I was planning on creating and distributing. By this time I had created a series of six Macromedia movies which represented 'steps' in the development of this piece, each time that I made a major revision to my work I would re-save with the same title followed by a higher numerical value. At this time the project consisted of two screens; the first set the user's level by clicking on a button which corresponds to that level (i.e. beginner, intermediate, advanced) a global variable is assigned and will be called upon throughout the program. The second screen contains buttons and a field which retrieves a text (TXT) file based upon the user's level. The contents of the field are printable and it is worth mentioning that the same field and 'Print Field' button gets used throughout the program. Different text files are retrieved depending on the location of the user. The Print Field command was downloaded from 'America Online' and is a Hypercard XCommand which someone had adapted for use in Director.

1.19.94

I had posed the question of offering some form of help to the user. To this effect I thought of creating a floating palette which would allow the user to select a topic and see the definition of that topic displayed in the palette or click a Go To button which would send the user to the section on that particular subject.

I also looked into using ResEdit to make the Text files invisible, and saving the users level (which would effectively personalize the stack but would become boring if the user could not reset their level to gain access to more information).

1.24.94

I purchased a copy of the ResEdit 'manual' which informed me how to create custom pointers for use and how to make the text files which were being called by Director invisible. I also made some discoveries about the Help palette; the windows script in The Apartment (a Lingo resource provided by Macromedia) could only display static text, radio buttons, click boxes, or bitmaps not scrolling text fields. I determined that possibly HyperCard would prove to be of some use, so I decided to look into it further. I also had an idea concerning saving the user's level: it could be dumped into a text field not present on the stage and saved, the only thing I wasn't certain about was whether Lingo could perform a DoMenu type command.

1.25.94

Created an arrow button to move between subjects which also cleared the contents of the field so that when a new topic was reached the material from the old topic wasn't still sitting in the field. At this stage I had to attach the Read Text File script directly to the button instead of using a handler to get the text files for that specific topic. I tried the XCMDGlue Apartment movie and found a movie which creates a Mac Window which has click lines and would return the selection. I wondered if this and another script called MakePalette from the HyperCard PowerTools stack could work together.

1.31.94

I made the text files invisible and the movie was still able to access them (they were sitting in the same folder). For comparison I placed a duplicate of the movie in another folder without the text files present and the movie could not 'see' out of the folder to get to the text files. At this point I thought that I had reached a conclusion about what I was going to do to hide the text files from the user. Unfortunately making them invisible would prevent the user from copying it to their hard drive without introducing a serious risk of disabling the functions of the program, and if the text was invisible I would be doing a great injustice to the user...who would want invisible files on their computer that they don't know about? I also created a standalone movie to make sure that the standalone application could still call the text files and it could.

I thought about making the text files resources using ResEdit but I found out that TXT files (text only) have NO resource fork and therefore could not really be 'installed' into the standalone. I also recognized the need to script an alert to the user so that when he or she opens the movie it checked to see if the folder containing the text files is present and alerts him or her to this fact possibly providing instructions or some other kind of help.

2.6.94

I decided that the best way to help the user was not through the use of the palette, which I could not create the way that I wanted. Instead, I created a help screen which offered the user two choices, an Index or a Definition. Both utilize the script that I found in the Apartment called XCMDGlue. The contents of the field are the contents of a Director text cast member. By clicking on a line and hitting the 'Select' button, an array is created and then put into a text field which is used to send the user to a corresponding topic or enable the user to open the definition on that topic. I also decided to add the 'Print Field' command to this but did not do so at this point. The computer lab was also connected to the campus network which allowed me access to the Internet and, more particularly, the World Wide Web via NCSA Mosaic. Mosaic is an application which reads Web pages that can contain hyperlinked text and images. At this point I made the assumption that Director was incapable of such things, although I knew that it was capable of determining what word was clicked when over a text field.

2.15.94

I fleshed out the remainder of the program to include screens for all the major topics that I plan on including in my thesis. This enabled me to make the Index button work so that when the user went to the Index and selected a topic, he or she was taken to a screen on that topic. At a meeting with Jim it was suggested by him that I create a pull menu which records where the user has been and would send him back there if he were to choose a topic from it. My first inclination was to create an option which allowed the user to 'dog ear' pages which would be added to the Pull Menu, but I wasn't sure how many people would actually use that feature. Better to do it for them without even asking, just to make things more convenient. I altered the Definition to automatically access the text file when the topic was selected from the dialog box. Previously a button had to be pressed after the topic was selected to get the text. Something interesting happened here without really knowing what I was doing. I began to use the Array which was created when the item was selected to open the corresponding text file. This gave me an idea to use an Array to open all the other text files instead of the series of if/then statements as I had been doing previously. Since a global variable was in use to declare the user's level I could use this to tell the program which text file to retrieve. I wondered if I could add this to the name of a text file and then use that to open it (ie. if the user's level = L = 1 and the name of the file was 'Internet' could I open a file called 'Internet1'). This would greatly simplify things without all the hassle of a series of if/then statements to determine which file to open. This gave me another idea as to where to place the text files. I moved all of the text files to a folder called 'Resources' and placed it in the same folder with my Director file. I then went into my scripts and changed my File I/O statements to include " the PathName & 'Resources:' ". It worked so that cleared up the problem that I had with making the text files invisible since in most cases there will be three versions of each article. Having them remain visible could also mean that the program could be 'updated' in terms of the information it contains simply by replacing the text files.

2.19.94

The idea of hypertext started with me seeing NCSA Mosaic and was pushed by Jim as something to try to include. I had thought earlier that doing hypertext with Director would be impossible and I was wrong. I used two Apartment files as starting points "Menu Manipulation" and "Mousies/Chunky Text". I started with "Mousies.." which identifies what words are under the mouse when it is clicked. I created a text cast member into which the clicked word would be dumped. The same word would also be put into an Array which would then be used to send the user to a screen with the same name as that Array (ie. go frame xArray). This worked except that when I clicked on a word which was not also the name of a frame, an error would occur. Therefore, the MouseWord would have to be compared to a list of frames which exist as possible destinations. If field "DIR", which was the same field being called by the Index and the Definition help screens, contained the MouseWord then it would proceed to send the user to that screen otherwise nothing would happen, a simple if/then procedure. I wrote a script which went 'if field "DIR" contains the MouseWord then go...', this would cause problems later because I was comparing the larger item to the small. Words like 'in' would still pass the test while longer words would not. This was fixed by switching the order in which the comparison was made, comparing the smaller word to the large list of words. At this point I also had to fix my wording in the script a second time. The computer program was recognizing the words in my list plus the comma as a whole unit. Dave Seah mentioned that he and Keith Watson had used a script which called the 'item' instead of the word. This eliminated the error that the punctuation caused. The next problem I had was that topics which were more than one word weren't getting added to the list or hyperlinked. So I had to change all instances of multi-word topics to include underlines (ie. World_Wide_Web) so the computer would see them as one chunk.

The next thing that I had to do was to create the menu. I called upon the "Menu Manipulation" movie to do this. I adapted the scripts in this movie so that when the MouseWord was clicked it was dumped into a temporary field and subjected to the same test as the hypertext. If the MouseWord passes the test it is added to a

'Pseudo' field and to a field which is the actual menu. At the start of the movie, "menu: file" is placed into line one of a field. Every time the mouse is clicked on a sensitive word 1 is added to a global variable which controls which line the word is put into. The word is then added to that line # of field 'Pseudo' and that line # + 1 of the field containing the menu. The menu is then installed and enabled. The enabling took some trial and error to figure out. Originally I had a button enable the menu but if the menu was chosen before the button was pressed nothing would happen. The EnableMenu handler was written and called in each instance that the InstallMenu command was used. After this was tested by Gedeon Maheux, it was found that the Index and Directory buttons did not add items to the pull menu, only hypertext did. This was pretty much an oversight on my part and was corrected by adapting the hypertext script which tests the MouseWord to the Array which is used to manage the activities on the help screen.

I found limitations because I was flowing in a text only file. Director didn't have any formatting features which can color text, nor could you set the color of text by using Lingo. The solution was to type all hyperlinked text in all caps to notify the viewer that it was clickable.

2.20.94

I added a 'Print Field' button to the Definition screen, and a button to the screen which would allow the user to go to the screen on that topic so he or she would not have to go back to the Index button to do so. It was a simple procedure but a nice feature for the user. The 'Print Field' button worked well for the definition screen although it seemed to be a bit of a waste of paper because the definitions themselves were not all that long. I decided to script something which I called a Dictionary which is a text cast member that does not appear on the stage. There were three buttons added to this screen concerning the Dictionary: Add To Dictionary: which added the current definition being displayed on the screen to the Dictionary list. A global variable is set to the # of lines in the Dictionary to which 2 is added so the user won't erase any information that he or she has put in the Dictionary (when a new topic is added). Print Dictionary: prints the field and Clear Dictionary: which empties the field.

Upon quitting, the computer checks to see if something has been added to the Dictionary and not been printed. Through the setting of a true/false statement after the print and add commands, if the user has added something to the Dictionary and not printed it, the computer does not quit the program. Instead the user is prompted to Print, Quit Anyway, or Save the Dictionary as a text document.

The final addition to the program was to make the definition text fields hypertext sensitive. This caused some confusion because I had named the cast member 'Info' so the program wasn't always going to the right spot when I clicked on a word. Once this error was corrected the hypertext worked.

2.28.94

After a brief meeting with two of my thesis advisors I was redirected. I was told not to throw away anything that I had done up to that point but I was encouraged to try one of two different things: an Internet of the future simulator; or a program which would explain the functions of the Internet. Not to be discouraged I decided to try both.

The focus of the thesis was to try to achieve a new aesthetic in terms of human-computer interface design. The typical command-line interface which was currently in widespread use on the Internet would not prove useful to parties who were interested in being connected but didn't want to have to learn a new language of archaic commands which in effect produced results but were not very dynamic or visceral for the viewer. The end-user needed to feel involved, informed, and most importantly in control of the Internet of the future. In short, the interface would have to become user-sensitive and weed out useless information while presenting that information which is useful to the person navigating the Information Superhighway. As I looked around there were two very different fields of thought concerning the horizon of interactive communication. One group held the belief that the Information Superhighway would be in everyone's living room in a year of so with little or no attention to HOW it would get there. But there was a dollar to be made off of it and get there it would. The other group raised the battle cry that the Highway was already HERE and it was called the Internet and something for everyone could be found there.

3.12.94

On this date I had begun to do the graphics for the interface and the animations which would inform the user about the features of the Internet. I created several three-dimensional models on the computer in Strata StudioPro. I also began sketching ideas for the structure of the interface that could be presented to the user and how. And I began searching the Internet and the World Wide Web for articles to include in the program.

4.9.94

For the past month I have been at work on the actual interface of the program. The concept was originally inspired by a combination of the MARSBook multimedia program interface and the shape and look of Apple's Newton Personal Digital Assistant. While I was creating the program, I was forced to change the interactivity of the program The initial concept was to establish a metaphor of a computer within a computer, in this case a personal digital assistant. I had examined many interfaces in interactive programming. I wanted to give the user the impression that he or she was effecting something above and beyond that of simply making buttons depress and sounds play. I began with the concept of an animated interface with control panels which would enable/disable themselves according to where the user was. The tabs on the right were created to give the user a familiar, menu-based, system similar to the Macintosh OS. Additional feedback is given in the window on the right to make the user aware of his/her position in the program. Most of the difficulty arose in the programming process with the animations. Two separate animations were created for each section (at least), one for going "in" and one for going "out". Whenever a tab is chosen a variable is set and the exit animation is played. At the end of that animation, a check is executed which informs the computer which button was selected and sends the program to the frames of the animation which "opens" that section.

```
on DoTell
 global MW,MC
 put "" into cast "two"
 put the mouseWord into MW
 put the mouseCast into MC
 if field "dir" contains word MW of field the name of cast MC then
  exit
 else
  set MW = ""
  beep
 end if
end DoTell

on Choice
 global MW, Q
 put the number of lines in field "Pseudo" into total
 repeat with x = 2 to total
  if line x of field "Pseudo" = MW then
   set Q = 1
   exit
  else
   set Q = 0
  end if
 end repeat
end Choice

on Teller
 global MW,MC,M,Q,PLine
 if MW = "" then
  exit
 else
  if Q = 1 then
   put the text of cast a13 into bArray
   go to bArray
   exit
  else
   put word MW of field the name of cast MC into line 1 of field "Two"
  put the number of lines in field "dir" into total
  repeat with x = 2 to total
   if word 1 of field "two" contains item 1 of line x of field "dir" then
    put the text of cast a13 into bArray
    go to bArray
    set M = M + 1
    set PLine = PLine + 1
    put word MW of field the name of cast MC into line PLine of field
"Pseudo"
    put word MW of field the name of cast MC into line M of field "One"
    installMenu a12
    EnableMenu
   end if
  end repeat
 end if
 end if
 updatestage
end Teller
```

Menu Item Script Handlers

Determines which menuItemNum was selected and executes the corresponding script. Called when a menu item from the user created menu is selected Each menuItem in the menu created is set to call this script with its name as the menuItemNum that is passed. These scripts were set on clicking the "Pseudo" menu button which also creates the menu.(See the script of cast A26) When a menuItem is selected the number of the menu item is returned Since the items can be in any order, we need to determine the name of the menuItem number.

```
on fileMenuChoice menuItemNum
put the name of menuItem menuItemNum of menu "File"
 put the name of menuItem menuItemNum of menu "File" into
itemSelected
 go to itemselected
end fileMenuChoice

on EnableMenu
 if field "Pseudo" = "" then
  alert "Please Select a menu item to add to the menu first"
  exit
 end if
 installMenu A12
 repeat with x = 1 to the number of lines in field "Pseudo"
  if line x of field "Pseudo" <> empty then
   set the name of menuItem x of menu "File" to line x of field "Pseudo"
   set the script of menuItem x of menu "File" to
"fileMenuChoice("&x&")"
```

```
  end if
 end repeat
end EnableMenu
```

The Movie Script is as Follows:

```
on initialize
 set the text of cast a33 = ""
 global file, LastFrame
 set the immediate of sprite 1 to true
 set the immediate of sprite 2 to true
 set the text of cast a21 to "" --Articles
 set the text of cast a43 = "" --Definition Text
 set the text of cast a45 = "" --Definition Topic
 set the text of cast b11 = "" --Dictionary
 set LastFrame = 7
end initialize

on StartMovie
 global M, PLine, DefLine, BC, T, AR, GT, IX, DD, DX, DICT
 set DICT = 0
 set DX = 0
 set DD = 0
 set IX = 0
 set GT = 0
 set AR = 0
 set T = 0
 set BC = 0
 set DefLine = -1
 set M = 1
 set PLine = 0
 set the text of cast a13 to "" -- Temporary
 set the text of cast a22 to "" --Pseudo
 set the text of cast a51 = "" -- Index
end StartMovie

on StopMovie
 global file
 if the text of cast a21 <> "" then
  file(mDispose)
 end if
 set the text of cast a13 to "" -- Temporary
 set the text of cast a21 to "" --Articles
 set the text of cast a22 to "" --Pseudo
 set the text of cast a43 = "" --Definition Text
 set the text of cast a45 = "" --Definition Topic
 set the text of cast a51 = "" -- Index
 set the text of cast b11 = "" --Dictionary
end StopMovie
```

```
on FinishRead
 global File
 file(mDispose)
 set the text of cast a21 to " "
 go to frame "menu"
end finsh read

on TextSet
 set the textFont of field a21 to "Geneva"
 set the textSize of field a21 to 12
end TextSet

on PrintField
 set aname = CallBackTracer(mNew)
 SetCallBack PrintDoc, aName
 set PageHeader = "III"
 set container = the text of cast a21
 set PrintDialogString = "Internet Information Index"
 set FontName = "helvetica"
 set FontSize = "12"
 set BooleanSetUp = "true"
 PrintDoc PageHeader, container
PrintDialogString,FontName,FontSize,BooleanSetUp
End PrintField
```

Reading Text Files

```
on GetIt
 puppetsound "switch.iff"
 global L, file, FileIO, f, d, c, t
 puppetsprite 22, true
 put the clickon into c
 put the mousecast into t
 put the name of cast the mousecast into tempArray
 set the castnum of sprite c to (t + 8)
 updatestage
 set the castnum of sprite c to t
 set f = 1
 set file = FileIO (mNew, "read",the PathName & "Resources:" &
TempArray & "." & string(L))
 if not objectp(file) then exit
 set s = file(mreadfile)
 if s <> "" then
  set the text of cast a21 to s
  TextSet
 else
  alert "That is the last line of the file." & return & "Click Done to exit."
 end if
 set the castnum of sprite 20 to (t + 32)
 set the castnum of sprite 22 to (t + 36)
 updatestage
end GetIt
```

Button Depressing-- Used for tabs

```
on switchbutton
 global sb, ob
 updatestage
 set ob = the mousecast
 set sb = (the mouseCast - 79)
 puppetsprite sb, true
 set the castnum of sprite sb = (ob + 16)
 puppetsound "switch.iff"
 updatestage
 puppetsprite sb, false
 set sb = 0
end switchbutton
```

Used for Level Setting

```
on switchbutton2
 global sb, ob
 updatestage
 set ob = the mousecast
 set sb = (the mouseCast - 71)
 puppetsprite sb, true
 set the castnum of sprite sb = (ob + 5)
 puppetsound "switch.iff"
 --go to frame the name of cast sb
 updatestage
 puppetsprite sb, false
 set sb = 0
end switchbutton2
```

Used for Quitting

```
on switchbutton3
 global sb, ob
 updatestage
 set ob = the mousecast
 set sb = (the mouseCast - 135)
 puppetsprite sb, true
 set the castnum of sprite sb = (ob + 16)
 puppetsound "switch.iff"
 --go to frame the name of cast sb
 updatestage
 puppetsprite sb, false
 set sb = 0
end switchbutton3
```

Scroll Articles UP

```
on up
 set the castnum of sprite 13 to 123
 puppetsound "switch.iff"
 put the castnum of sprite 15 into y
 if y = 129 then
  go to frame "up1"
  set the castnum of sprite 15 to 130
  set the castnum of sprite 16 to 131
 else
  if y = 130 then
   go to frame "up2"
   set the castnum of sprite 15 to 131
   set the castnum of sprite 16 to 132
  else
   if y = 131 then
    go to frame "up3"
    set the castnum of sprite 15 to 132
    set the castnum of sprite 16 to 129
   else
    if y = 132 then
     go to frame "up4"
     set the castnum of sprite 15 to 129
     set the castnum of sprite 16 to 130
    end if
   end if
  end if
 end if
 set the castnum of sprite 13 to 115
end up
```

Scroll Articles DOWN

```
on down
 set the castnum of sprite 14 to 124
 puppetsound "switch.iff"
 put the castnum of sprite 15 into y
 if y = 129 then
  go to frame "down1"
  set the castnum of sprite 15 to 132
  set the castnum of sprite 16 to 129
 else
  if y = 130 then
```

```
   go to frame "down4"
   set the castnum of sprite 15 to 129
   set the castnum of sprite 16 to 130
  else
   if y = 131 then
    go to frame "down3"
    set the castnum of sprite 15 to 130
    set the castnum of sprite 16 to 131
   else
    if y = 132 then
     go to frame "down2"
     set the castnum of sprite 15 to 131
     set the castnum of sprite 16 to 132
    end if
   end if
  end if
 end if
 set the castnum of sprite 14 to 116
end down
```

Previous Topic

```
on back
 set the castnum of sprite 13 to 121
 puppetsound "switch.iff"
 updatestage
 if the frame = 391 then
  go to frame "WORLD_WIDE_WEB"
 else
  go marker (-1)
 end if
 set the castnum of sprite 13 to 114
end back
```

Next Topic

```
on forward
 set the castnum of sprite 14 to 122
 puppetsound "switch.iff"
 updatestage
 if the frame = 461 then
  go to frame "Archie"
 else
  go marker (1)
 end if
 set the castnum of sprite 14 to 115
end forward
```

Where to Go?

```
on ButtonCheck
 Global BC
 if BC = 1 then go to frame "Articles"
 if BC = 2 then go to frame "Topics"
 if BC = 3 then go to frame "Help"
 if BC = 4 then go to frame "menu"
 if BC = 5 then go to frame "index"
 if BC = 6 then go to frame "define"
 if BC = 7 then go to frame "quit"
end ButtonCheck
```

Index

```
on Index
 global idArray, BC, GT
 put the name of cast the mousecast into idArray
 set GT = 1
 set BC = 2
 go to frame "IndexDone"
end Index
```

Dictionary

```
on Dictionary
 global DictArray, DX, file, FileIO, f
 puppetsprite 13, true
 put the name of cast the mousecast into DictArray
 set the castnum of sprite 13 to ( the mousecast + 16 )
 set the loch of sprite 13 to 161
 set the locv of sprite 13 to 120
 go to frame "Diction"
 set f = 1
 set file = FileIO (mNew, "read",the PathName & "Resources:" &
DictArray)
 if not objectp(file) then exit
 set s = file(mreadfile)
 if s <> "" then
  set the text of cast a21 to s
  TextSet
 else
  alert "That is the last line of the file." & return & "Click Done to exit."
 end if
end Dictionary
```

PrintDictionary

```
on PrintField2
 global QuitTest
 set aname = CallBackTracer(mNew)
 SetCallBack PrintDoc, aName
 set PageHeader = "Dictionary"
 set container = the text of cast b11
 set PrintDialogString = "You can put the name of your field here"
 set FontName = "times"
 set FontSize = "12"
 set BooleanSetUp = "true"
 PrintDoc PageHeader, container
PrintDialogString,FontName,FontSize,BooleanSetUp
 set QuitTest = 0
End PrintField2

factory CallBackTracer

method mNew

method mSendHCMessage x

method mSendCardMessage x

on ReadText2
 global L, file, FileIO, myArray,f ,M ,Pline
 set f = 1
 set file = FileIO (mNew, "read",the PathName & "Resources:" &
myArray)
 if not objectp(file) then exit
 set s = file(mreadfile)
 if s <> "" then
  set the text of cast a43 to s
  put the number of lines in field "dir" into total
  repeat with x = 2 to total
   if word 1 of field "Def" contains item 1 of line x of field "dir" then
    set M = M + 1
    set PLine = PLine + 1
    put myArray into line PLine of field "Pseudo"
    put myArray into line M of field "One"
    installMenu a12
    EnableMenu
   end if
  end repeat
 else
  alert "That is the last line of the file." & return & "Click Done to exit."
 end if
 updatestage
end ReadText2

Cancel Dictionary
```

```
on ClearDictionary
 global QuitTest
 set QuitTest = 0
 set the text of cast a43 to ""
 set the text of cast a45 to ""
 go to frame "help"
end ClearDictionary
```

Quitting

```
on QuitMovie
put the number of lines in field "Dictionary" into total
if total > 1 then
go to frame "quitcheck"
else
 play frame "Main" of movie "Thesis v1.01"
end if
end QuitMovie

on writeDictionary
 global file, FileIO
 set file = FileIO(mNew, "?write", "Dictionary" )
 if not objectp( file ) then
  set theProblem = string( file )
  exit
 end if
 set s to the text of cast b11
 file( mWriteString, string( s ) & return )
 file(mDispose)
end WriteDictionary
```