

**Design of the User Interface for a Document Browser**  
**Supporting Interactive Search**

by

Frank Y. Ho

Submitted to the Department of Electrical Engineering and Computer Science  
in Partial Fulfillment of the Requirements for the Degrees of  
Bachelor of Science in Computer Science and Engineering  
and Master of Engineering in Electrical Engineering and Computer Science  
at the Massachusetts Institute of Technology

May 1994

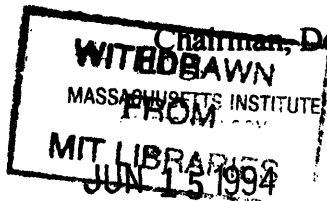
Copyright Frank Y. Ho 1994. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce  
and to distribute copies of this thesis document in whole or in part,  
and to grant others the right to do so.

Author [Signature]  
Department of Electrical Engineering and Computer Science  
May 16, 1994

Certified by [Signature] Professor Randall Davis  
Thesis Supervisor

Accepted by [Signature] F. R. Morgenthaler  
Chairman, Department Committee on Graduate Theses



**Design of the User Interface for a Document Browser  
Supporting Interactive Search**

by  
**Frank Y. Ho**

Submitted to the  
**Department of Electrical Engineering and Computer Science**

**May 1994**

In Partial Fulfillment of the Requirements for the Degree of  
**Bachelor of Science in Computer Science and Engineering  
and Master of Engineering in Electrical Engineering and Computer Science**

## **ABSTRACT**

A White House document browser is designed and implemented using a hypertext markup language, HTML, that allows the application to be viewed with a World Wide Web document browser. An extension to the standard HTML specification that supports multiple pages within a document node, as well as the logic control of text elements, is implemented by modifying NCSA Mosaic, a WWW browser. The new features provide a powerful means for extending the static and sequential view of a document, enabling the Mosaic browser to support highly interactive documents such as adaptive survey forms. This capability is especially useful in distributing survey forms targeted at a wide range of users in which the user's response to previous questions are used to determine the presentation of subsequent questions. This effort represents an attempt to achieve higher user interaction within the traditional document model.

**Thesis Supervisor: Professor Randall Davis**

**Title: Associate Director, MIT Artificial Intelligence Laboratory**

*To my parents*

# Table of Contents

<b>1.</b>	<b>Introduction</b>	<b>1</b>
1.1	Context	1
1.2	Nature of This Project	3
<b>2.</b>	<b>Search Model</b>	<b>5</b>
2.1	Design Issues	5
2.2	New Ideas on Interface Design	8
2.3	Indices For Document Search	14
<b>3.</b>	<b>Implementation Issues</b>	<b>16</b>
3.1	Choosing the Interface Model	16
3.2	Implementation Platform	18
3.3	HTML Features	19
3.4	The Prototype Browser	21
3.5	Limitations of the Prototype	23
<b>4.</b>	<b>HTML Extension</b>	<b>25</b>
4.1	Description of HTML functions	25
4.2	HTML+ Features	27
4.3	Additional Features For The HTML Extension	30
4.4	Graphics Extension	35
4.5	A Virtual-Reality Composition Language	37
<b>5.</b>	<b>Hypermedia Design</b>	<b>38</b>
5.1	Document and Program Model	38
5.2	Interactive Document	39
5.3	Functions	40
5.4	Interface Features	43
5.5	Authoring Tool	45
<b>6.</b>	<b>Conclusion</b>	<b>47</b>
	<b>Appendices</b>	<b>49</b>
	Appendix A. Search Interface	49
	Appendix B. Sample Usage Survey	51
	Appendix C. Quick Reference Guide to the Extension	61
	References	85

# 1 Introduction

## 1.1 Context

### ***The Information Superhighway***

As part of our government's effort to promote applications of technology to society, the administration has launched the Information Superhighway initiative. The project oversees the use of computers linked with the internet that spans across the nation and around the world. Increasing volume of information will flow across the network, reducing the current dependence on paper-based information flow. Electronic versions of newspapers, magazines and books will supplement traditional paper versions. On-line services for banking, shopping, education and entertainment will make their way into our everyday life. As more documents and services are available over the network, people will begin to see the convenience of being able to access such a diverse range of resources from their own terminal. The power of the internet, and the role it will play in our future lives, cannot be underestimated.

### ***The White House Electronic Mail Project***

A recurrent issue that echoes the Information Superhighway initiative is the administration's effort to *reinvent government* -- a set of goals that includes making it more efficient and more responsive to the public. This, along with the information superhighway plan, results in the White House Electronic Mail Project. The project attempts to improve government communication with the people by starting the first-ever electronic mail service at the White House. The president and the vice president each has an e-mail address, to which people can send comments and opinions about the

administration over the network. The incoming mail is currently handled by a staff, who reads each message and presents the collective opinion of the senders to the president or the vice president. A standard thank-you reply is automatically generated and mailed back to the sender.

### ***Subscribing to White House Publications***

The White House e-mail service is only the first of many computer-based services that the government intends to introduce to the public. Currently, a White House publication subscription service is available. People interested in receiving particular types of White House press releases can send an e-mail to the service and request to add their name to the subscription list. The subscription service is available only by e-mail, through which the user has to send in a number of survey forms to the server. The user must therefore go through several rounds of e-mail messages before receiving the first delivered document. Publications classified under the user's chosen subject area periodically arrives as e-mail messages.

### ***An Interactive User Interface***

In an attempt to further improve the service level provided by the current publication server, our research group focused on developing an interactive user application for direct access of White House publications. At the beginning, we considered building a simple graphical user interface for the subscription process. The user would use our application to select the special interest mailing lists to which she would like to subscribe. We analyzed this concept but later decided to give the user greater control of the document retrieval process. In addition to the ability to add oneself to a subscription list, a user should also be able to actively access and search for documents from the server. We decided to empower our users by providing them with a user-friendly front end, with which they could conduct active document searches on the database in real time. Developing the user interface for such an information retrieval system is the focus of this project.

## 1.2 Nature of This Project

This research project represents an attempt to explore the possibilities of user interface design. Over the course of the research period, new ideas emerged and original plans changed. We started the project with a mission to build a better interface model for the White House Paper browser. As we continued to look at different possibilities, the focus of the project evolved. At the beginning of the research period, we studied human factors in interface design and came up with ideas about design issues relevant to our browser interface. Several interface model designs were drafted as we tried to break away from the conventional user interface. As we studied these models, we gained more insight into the interface bandwidth that limited user control of information. We document our design effort in Chapter 2.

We finalized on a design for our browser interface and prepared a paper demonstration. A series of slides was used to depict the finished look of the interface. Details of the paper model included the appearance of the interface, user-interactive controls, and a sample usage scenario. Our original intention was to implement the interface on UNIX using the X library and Motif widgets. We realized that it was important that our browser interface be available in all major computer platforms, thus we should build the UNIX version as a prototype from which other versions would be based. Our focus then turned to the increasing popularity of the World Wide Web, a network consisting of many nodes, each representing a document. Users on the network could access all documents through hypertext links. We studied HTML, the markup language used by hypertext documents, and decided that our publication browser could be implemented with it. We could therefore serve our browser over the network as a document on the Web. Since a number of Web browsers running on different computing platforms were

available, the issue of platform-dependency was resolved. Implementation issues are discussed in Chapter 3.

Serving our browser interface on the Web had its limitations, however. Using the limited set of HTML functions, we could only build an interface that was form-based, which was different from that of our original proposal. As we experimented with the new prototype, we concluded that a number of modifications to the current HTML standard would help our implementation effort. We studied the features of the current HTML standard, as well as the proposed HTML+ extension under development. We further identified a set of extra features beyond those available or proposed elsewhere, and documented their specification details. A major portion of the research effort has been dedicated to the implementation of an HTML extension that runs on the current version of NCSA Mosaic, a WWW browser. We discuss the specification and use of our HTML extension in Chapter 4.

Having completed our study, we set our sight on more ambitious possibilities in hypermedia design. We studied the current development in hypermedia applications -- especially those applied to computer assisted instruction -- for clues in problems and issues facing developers and users today. We introduced the concept of a fully interactive document, as a cross between the traditional document and the object-oriented program. We further proposed that application-specific modules be used to handle different types of data within a document. This would render the document as the basis of information flow across the network, on top of which applications and user-interactive functions could reside. We present our ideas for hypermedia design in Chapter 5.



## 2 Search Model

### 2.1 Design Issues

Human factors play an important role in user interface design. We therefore began the research project by first thinking through a number of design issues relevant to our browser interface.

#### *Active vs Passive Interface*

Some types of user interface require active participation from the user. Examples are command-based interface, traditional programming, and keyword search. These input methods require that the user know a fair amount about what they want to see and how to get it. The system provides little guidance as to what is available to the user. The advantage of active interface is that users have the freedom to manipulate ideas and have full control over what they want the system to do for them. The disadvantage is that users who do not know what they want to see may feel helplessly immobilized. For example, a user may want to read some articles about a certain topic, but cannot think of the exact word to describe it, although he may recognize the word when he sees it. Situations like this are very common and have been the motivation behind more user-friendly interface models.

Modern user interfaces usually require only passive participation from the user. A menu bar, for example, allows the user to browse all the options and choose one from a list. It is much easier to have all the options laid out in front of the user so that he can simply pick from the list. The hierarchical tree structure is another example of a passive user

interface. Everything that is available to the user can be scanned by traversing the tree. The disadvantage of passive interfaces, if any, is that some users who know exactly what they want may feel constrained by the disability to directly choose the desired item. A user who knows that he is interested in reading documents about health care may prefer to supply that as the key word to do a search, instead of having to traverse levels within the tree.

The goal of our user interface is to support active manipulation by the user while making it easy enough that users with little idea of what they want can still see all the options available to them. A good interface should therefore present the user with a lot of information while leaving the active control in navigating through such data to the user.

### ***The Role of Intelligence***

Intelligence, in the context of user interface design, is not measured by how closely it mimics human behavior. Rather, an intelligent user interface should be an extension of the user's abstract mind, not a totally foreign mental being. It is easier to find information by clicking on icons and menus, than by using some fancy multimedia interfaces that lets the user talk to the computer to select a file. Similarly, a user may find it easier to express an abstract concept through the direct manipulation of graphical icons than by talking to a human being or a rule-based interviewer. For some applications, therefore, we do not need a system that provides the same functionality as that of a human being. An interface that asks users questions about what they are interested in, is probably not the most effective interface for our system.

### ***Familiarity***

There are some user interfaces that are known to be sub-optimal, but since people have been using them for a long time, it is better to stay with the old standard. The keyboard is an example. The standard QWERTY keyboard is proven to be ineffective since about 60% of the time we are using the right hand. Another configuration of keys, which results in a more even split of duty between the two hands, is proven to allow trained

typists to type much faster. The question is whether we shall adopt the new keyboard configuration. The QWERTY keyboard, although theoretically sub-optimal, is so well established in users' experience that to change their habits requires too much an effort to be worthwhile.

We can draw an analogy with the newspaper. Besides its role as a presentation of information, the newspaper also contains an indexing mechanism that people use to find information. Headlines on the front page allows the reader to browse through potentially interesting topics, and read on further for the actual content, which often requires the reader to flip to another page for the remainder of the article. We can argue that there are much better ways to index information than the primitive mechanism used in newspapers, but the act of scanning headlines and looking through different sections for different topics (e.g. section B for local news) is intuitive for us, simply because we have been reading newspapers for many years. The question is: how much does the newspaper analogy help users in making them feel comfortable with the interface? How far shall we deviate from the newspaper metaphor in order to capture more powerful techniques for indexing? These are important questions that we need to consider when building our user interface.

### ***Indexing and Presentation***

We have considered the interface model in which indexing and presentation of information is one seamless process. In the newspaper example above, the layout of the paper (i.e. its interface) is both a mechanism for the reader to find information and to read the actual content of articles. The reader does not, for example, explicitly go through an index to find out all the possible topics from a hierarchy tree, and then read the article that interests him. The tasks for searching and reading is *simultaneous* and the distinction between them is vague. When reading a newspaper, we use indices subconsciously. For example, we follow implicit indices when we scan headlines across the front page or flip to a particular section in the paper. An explicit indexing mechanism may not expedite our search, especially if we do not know in which topic we

are particularly interested. Our interface should therefore blur the distinction between hierarchies, topics, and the actual text of documents. The user can navigate in a seamless manner from one logical level to another, and back, without having to think about the actual physical distinctions.

## 2.2 New Ideas on Interface Design

With these design issues in mind, we set out to create several interface models.

### *Picture Action Model*

The picture action model consists of graphical icons representing several classes of concrete and abstract concepts, including:

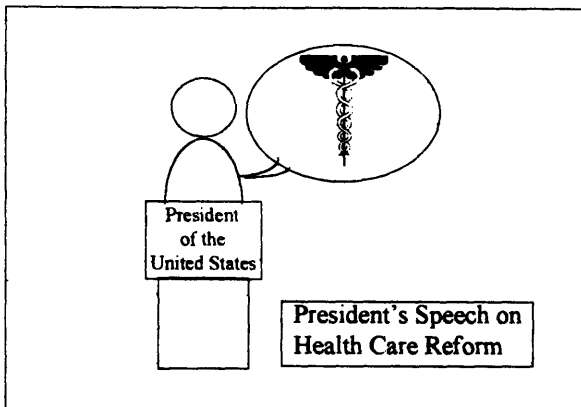
- Agents, such as the President, the Department of Defense, and the Treasury
- Actions, such as that of giving a speech, or of vetoing a legislation
- Subjects, such as the economy, welfare, and technology

The user manipulates these icons to form a picture representing a concept, which is similar in nature to the description given in the taxonomy of White House publications.

Examples of concepts are “The Present’s speech on the new health care plan”, “Homelessness”, or “Reorganization of the Executive Branch”. For example, a user who wishes to read about the President’s speech on the new health care plan can do the following sequence of manipulations:

- place the *President* icon on the sheet
- drag a *bubble* icon representing speech and put it next to the *President* icon
- put a *Medicare* symbol and put it inside the bubble
- put an *podium* icon to indicate that it is a formal speech to an audience

The system displays descriptive labels next to each icon, and shows a textual description of the concept as interpreted by the knowledge base. The user verifies the textual description as the concept in which she is interested, and tells the system to go search for the relevant articles.



Sample scenario of the picture action model

This model is actually a graphical version of the query model, which allows the user to type a text query to search for articles. The incremental advantage is that by manipulating graphical icons, the user can more quickly produce queries and experiment with different combinations of icons to find out new query possibilities that would otherwise escape her mind. The problem with this interface is that some concepts are inherently difficult to represent as graphical icons, and what may seem intuitive to one user may not be true with another. We shall also note that this interface requires active user participation in moving the icons. Nonetheless, by providing moving icons to the user as tools, the user may find it less intimidating than having to type up a text query herself. The feedback text description is an important means of making sure that the system understands what the user is thinking about.

### ***Information Block Model***

The information block model is a 3-D graphical interface that represents an implicit hierarchical function. The idea is to use an intuitive structure of organization to represent levels of hierarchy in a tree. The user can walk through the 3-D world to find the information they need. One intuitive model of organization is the concept of cities, streets, and buildings, which can be abstractly represented as blocks in the 3-D world.

The following mapping between the graphical world and the actual taxonomy is considered:

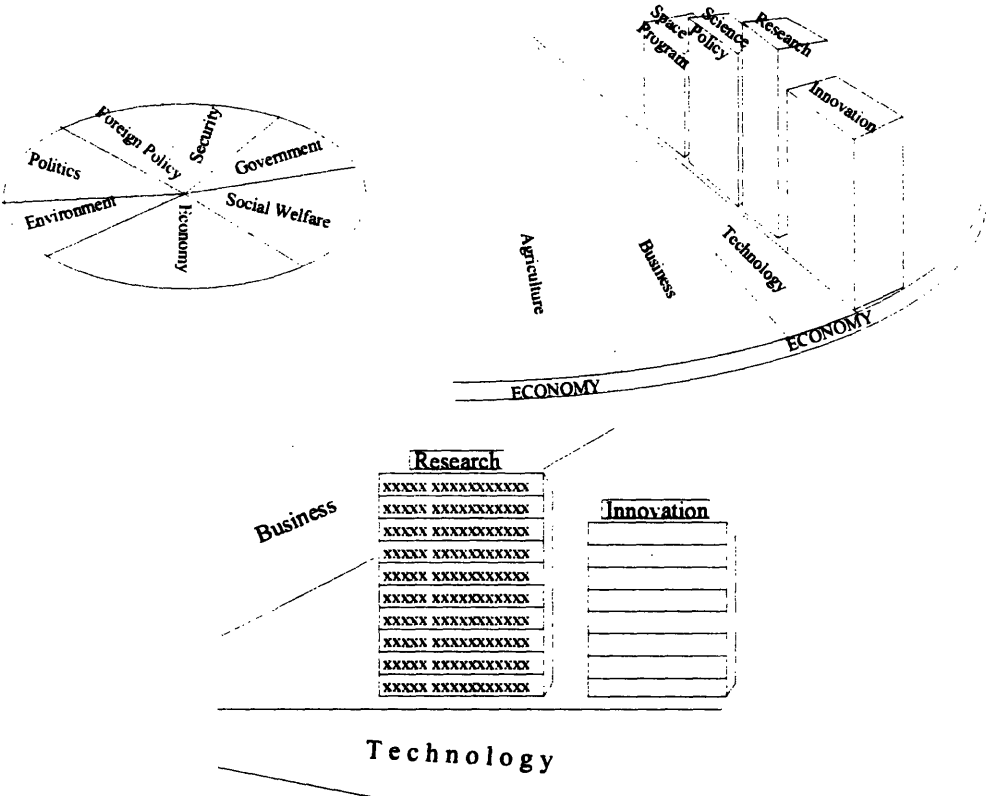
- ***cities*** represent the most general taxonomy level (e.g. city A = Economy, city B = Environment, city C = Foreign Affairs, and so on)
- ***streets*** represent the second taxonomy level (e.g. within city B, street 1 = Agriculture, street 2 = Business, street 3 = Energy ..... street 7 = Welfare)
- ***buildings*** represent the third taxonomy level (e.g. within street 6 of city B, building *a* = Innovation, building *b* = Research Funding, building *c* = Science Policy, building *d* = Space Program)
- ***levels*** in a building represent the actual document

Other visual attributes provide extra information on the documents:

- light intensity represents the age of the document: the newer the document, the brighter the displayed level
- height of a building represents the number of documents under the third-level category, since each document takes up one level of the building
- color of a building represents the second-level category (the street) to which a third-level category (a building) belongs: all buildings on the same street have the same color; for example, *Innovation*, *Research Funding* and *Science Policy* all have the same color

The user begins the session with an aerial view of all the cities, showing streets and buildings within each city. The user rotates the world to see a different perspective view, and chooses a city in which she is interested, by moving downward towards the city of interest (e.g. the *Economy* city). The user sees a closer view of the city, revealing more clearly the actual streets and buildings. All cities, streets and buildings are labeled with the name of the category and a graphical icon representing the category. At the city level, the user can again rotate the city to see different streets within the city. She may choose the street of interest (e.g. the *Technology* street), and move forward along the street to see the buildings on that street (e.g. the *Science Policy* building). By turning

around while walking through the street, the user can see more closely the name and category of a building, and the document headings displayed on each level of the building. To view a document, the user clicks on the heading, and a document pops up from the building. At any time, the user can move forward, backward, upward, downward, and turn left or right to obtain different perspectives of the virtual information world. After reading the document, the user can move backward from the level, the building, away from the street, and all the way up above the city. She may also move up and down a building to view documents under the same third-level category, or up and down a street to see a broad perspective of all categories under the same second-level category, and so on.



Sample scenarios of the Information Block Model

**Virtual Information World**

The proposed user interface requires high-speed graphics coupled with the control of movement through the virtual world, similar to that used in many virtual reality

applications. Although the field is relatively new and the technology immature, it is conceivable that a few years later PCs will be powerful enough to provide reasonable performance in virtual reality presentations. In the meantime, we may simplify the interface for less-powerful computers by requiring the user to point and click at cities, streets, and buildings, to move to the respective object, instead of allowing a virtual walk-through.

Not only can we use high-speed 3-D graphics to represent a virtual world, but we can also apply the technology to represent abstract information. The advantage of using 3-D graphics that gives the user control of the perspective view, movement, and focus, is that the user is granted with control over much more information. This is especially important to information access interfaces that aim at presenting as much information as possible within a relatively small 2-D window. The use of light intensity, color, and physical size further increases several dimensions of information that the user can intuitively comprehend. The idea is to give the user multiple dimensional information using a 3-D image projected on a 2-D screen.

A commercial interface product that uses analogous techniques is the virtual walk-through model of an art museum, where the user can move within the museum to view the art collections. Our proposed system pushes the idea further. We imagine that there are virtual cities, where information is printed on the walls of the buildings, and is organized similarly to the layout of a city. The reason for using a virtual world to represent hierarchical categories is that such a world allows the user the freedom to move from one level to another without being aware of crossing such boundaries. When a user focuses on the headings of several documents (on several levels) in a third-level category (a building), she can still see other neighboring buildings in the immediate background, a remote view of other streets within the city, and even other cities in the distant background. Thus the user's view is never taken away from any level of the taxonomy. This makes it easier for the user to move from one level to another, and conduct search



more efficiently. Ease of search within a multi-level taxonomy is the motivation for using the virtual information block model.

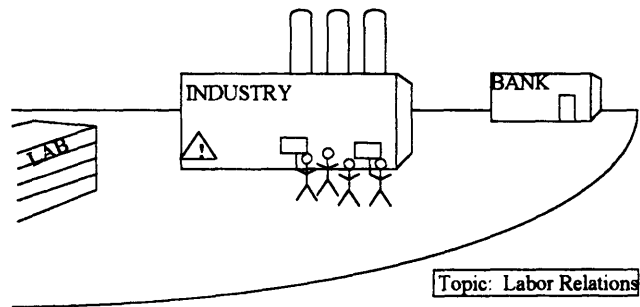
### ***World View Model***

The World View Model is an extension of the Information Block Model, in that high-speed graphics and movement in a virtual world are central to the interface. While the information block model proposes that abstract categories are simply printed on the walls of buildings, the world view model further promotes the intuition that such categorical concepts are represented as graphical objects in the virtual world, in which the shape and structure of such objects gives information on the category itself. That is, instead of showing monotonous cities, streets and blocks, that require the use of labels and icons to identify the underlying concept, the world view model shows graphical images representing the White House, the Government, the society, the country, and so on. The inherent hierarchical structure is de-emphasized, whereas the societal structure and people's image of the government plays a more important role in indexing the categories.

The interface session begins by showing the image of White House. The user can zoom into the White House to see different parts of the building. To see documents about the President's speech, the user can move to the conference room in which speeches are given. Once the internal view of the room is shown, the user can zoom into the presentation board to view the actual document. In another example, the user may search for a document about labor relations. Moving above and beyond White House, the user sees an aerial view of a city, with images of a power plant, bank, research lab, highways, factories, and other agents in the economy. The user can zoom into a factory, and sees a graphical image of a hazard warning, and another image showing a group of people on strike. Zooming into the first image reveals documents related to occupational safety. Zooming into the second image reveals labor relations articles, which is what she wants.

Zooming up and above the factory, and then above the city, the user sees an image of the entire country, showing all the states and some foreign countries across the oceans. To

read a document about California, she can move the virtual view across the nation to California, then move down to view documents concerning the state.



Sample scenario of the World View Model

The advantage of this interface model is that it provides a virtual view of the world, with respect to the society and the government, just as category concepts and the underlying documents provide an informative perspective of the society and of the government. This analogy is particularly powerful in building an intuitive user interface for indexing such information.

## 2.3 Indices For Document Search

Before we set out to build the user interface, we need to consider the different attributes that we can use to search for a document. A list of potential indices that may provide hints to describe a document is as follows:

- keyword --- a set of words that describes the content of a document
- key text --- statistical information on word occurrence over a specified section of a document
- key document --- a hypertext link to another document that relates to a highlighted topic
- subject area --- topic of interest as defined by the publications taxonomy

- source --- author, organization from which document is generated
- target audience --- for example, the general public, or special interest groups
- type of document --- one of article, speech, statement, legislation, etc.
- date of publication --- release date from the White House
- circulation --- number of readers who has requested the document to date
- endorsement --- recommendation of the document by recognized special interest groups

We shall let the user choose any combination of the above attributes when searching for articles. Each attribute would have some default value. The default value for *source*, for example, would be *any*, and the default value of *date of publication* would be *today*, and so on. The user can define a set of attributes and reuse in later searches by changing only the necessary attributes.

The user may deactivate an attribute by setting it back to its default value, thus not using the attribute as a qualifier in the search. Each of these attributes appears as a distinct search qualifier object. A conjunction or disjunction of such attributes forms the search criteria set used by the publication server to locate matching documents.

## 3 Implementation Issues

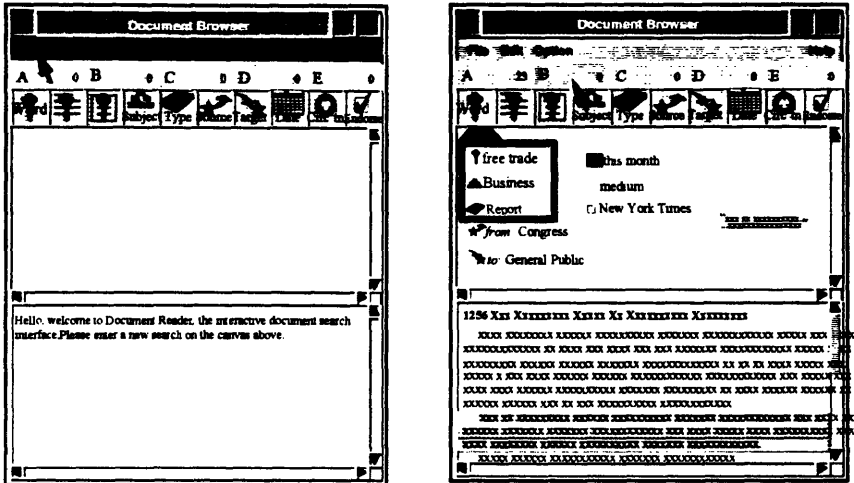
### 3.1 Choosing The Interface Model

Before we developed the first working prototype of the White House Paper browser, we drafted a mockup screen using a graphics editing program. The mockup screen depicts the final design of our user interface. Keeping in mind rendering limitations of graphical user interfaces, we chose to build an interface with a much simpler appearance, drawing our inspiration from window-based file operating systems. Our interface design consists of a window divided into five main areas:

- menu bar
- search attribute selection buttons
- search portfolio navigation buttons
- search portfolio canvas area
- document heading / text area

The menu bar contains typical pull-down menus for *File*, *Edit*, *Options* and *Help* functions. Below the menu bar is a set of search attribute selection buttons, each representing one of the search criteria described previously -- key word, key text, key document, subject area, source of document, target audience, document type, date of publication, circulation and endorsement. Between these buttons and the canvas area is a bar consisting of several search portfolio navigation buttons, used for switching from one search portfolio to another. The canvas area accommodates two types of graphical objects -- search folders, represented as large rectangular folders; and search attributes, represented as small graphical icons depicting the attribute type. The user can click and drag each graphical object. Below the canvas area is the document heading / text area,

which displays headings for retrieved documents or the actual text content of a selected document. A paper demonstration of the user interface design is shown in Appendix A.



Sample scenarios from the proposed user interface

The user starts a search session by clicking on several search attribute buttons. When one of these buttons is clicked, a dialog box appears to prompt the user for an attribute value. For example, when the user clicks on the subject area button, a dialog box with a pull-down menu appears, from which the user chooses the subject area. When the value is chosen, a graphical icon depicting the concept of a subject area appears on the canvas area, with a text label indicating the subject name (e.g. *Economy*) next to it. After a number of search attributes are selected, the user may drag a region surrounding the attribute icons to define a search portfolio. A search portfolio consists of a set of attributes used to conduct a document search. Only search attributes whose graphical icon appears on the portfolio folder are included in the portfolio. Search attributes whose graphical icon appears on top of blank space is not used. The user can drag an attribute icon across the canvas area, from one folder to another or from blank space to a folder. Multiple folders are permitted and each is identified by a user-definable name. Search

portfolio navigation buttons allow the user to quickly move the mouse pointer to a part of the canvas area containing the chosen portfolio folder, and makes the folder appear active. Double clicking on the chosen portfolio folder, or on the button corresponding to the search portfolio, causes the system to submit a search query based on the attributes defined on the folder. Searches involving multiple levels of specificity can be done by defining several search portfolios, each differing in one or two search attributes while keeping the others constant. Standard graphics editing functions such as *duplicate* and *delete* make it easy to conduct such searches.

If a search based on the submitted attributes is successful, the headings of the retrieved documents appear in the headings / text area. The user has the option to sort the incoming documents according to any attribute he chooses. To view a document, the user simply double-clicks on the heading of the desired document, and the same area displays the content of the document. The user can save search portfolios so that the same set of search attributes can be applied to future sessions.

## 3.2 Implementation Platform

We anticipate that users of the White House publication browser will run our application on different computer platforms, including UNIX, Windows and Macintosh. CPU performance, graphics capabilities and communication bandwidth also varies. We originally expected that we would need to port our application from one platform to another in order to accommodate users with different equipment. We would first build a prototype that runs on UNIX, written using X library functions and Motif widgets. The finished prototype would be ported to the PC and Mac platforms using ad-hoc graphical user interface development tools. Nonetheless, we later noticed the increasing popularity of hypertext documents, written in a standard language known as HyperText Markup Language, or HTML. These documents are delivered across the World Wide Web, a network connecting users around the world, in which each document has its unique

Uniform Resource Locator (URL). A URL is a name that uniquely identifies a document in the web, and consists of the server name and path name of the document file. A number of web browsers have been written as a front end to the web, delivering HTML documents through the use of hyperlinks. The user can explore the web by following a series of hyperlinks which links one document to another. Hyperlinks provide a powerful means of navigating through the web, and gives the user much control over which document he chooses to read.

### 3.3 HTML Features

HTML is a set of style commands added to plain text to define the presentation of the document on the World Wide Web (WWW). HTML is based on the Standard Generalized Markup Language (SGML), an international standard markup language that introduced the concept of using tags to define the logical elements of a document. Each class of documents is defined by an SGML Document Type Definition (DTD). For example, the HyTime architectural forms of SGML define hypermedia based documents, while the Text Encoding Initiative (TEI) forms of SGML define documents for the humanities. With the wide range of information that can possibly flow across the web, it becomes impractical to use a unique DTD for each class of documents. A general purpose document type definition that supports many document and display types using a simple set of presentation hints is in order. This motivated the original HTML design.

The concept of HTML is similar to that of LaTeX, except that HTML documents are intended to be viewed in an interactive setting on the WWW using graphical user interfaces such as NCSA Mosaic, whereas LaTeX documents are usually converted into PostScript files that are either printed as hardcopies or viewed using a non-interactive user interface. The most significant feature that motivates the development of HTML is hypertext links. Users can move from one point in a document to another document by following hypertext links that are defined with HTML commands. This capability has

significantly reduced the amount of time a user spends searching for references using traditional non-interactive approaches. Hypertext links are especially useful for reading documents from the internet. The user is abstracted from the task of remembering and typing addresses to download documents, as one would have to do when using a file transfer protocol (ftp). Using hypertext, the author can design its document to make it open to active exploration, and thus communicate ideas more effectively.

With the release of HTML 2.0, more features have been added. Fill-out forms and inline images are now standard features. Fill-out forms provide a simple and easy-to-use interface that allows the user to enter information and send it back to the server for further processing. In most cases such fill-out forms provide the user a means of selecting what kind of information he or she would like to see, and thus which document the server should send over the internet. In HTML 2.0, fill-out forms support a number of common input types, such as plain text, password, checkbox, radio buttons, and selection lists. These input styles are very similar to those found in typical graphical user interfaces, such as those provided by Motif widgets. The most important difference between an HTML form and a Motif widget is that HTML follows a document-flow style that is presented in a logical sequence, whereas Motif allows the user to arbitrarily define the location in which a widget appears. The document-oriented approach used by HTML requires a smaller overhead in text markup because location information of an object is implicit in the placement of that object within the text. This is what distinguishes a markup language such as HTML from a traditional GUI language such as X and Motif. One may therefore consider HTML as a hybrid language that supports both text-presentation and user interaction. HTML is used to define what we predict will become known as the *interactive document*.

HTML supports inline images for users having access to a graphics-capable terminal. HTML-based browsers can display X Bitmap or GIF format images inside documents. Inline images are usually used as static icons that enhance the appearance of a text document, but can also act as hyperlinks that lead to another document. For example, an



image that shows a dictionary may act as a hyperlink to an online dictionary function. The user can access the dictionary by simply clicking on the representative image icon. Furthermore, inline images can also serve as a hyperlink to a larger image, often the full-scale rendition of itself. Larger graphical images are usually viewed by an external viewer that pops up separately from the text window.

### 3.4 The Prototype Browser

We decided to build our interface application as an HTML file, to be delivered from our server site to all potential users who have access to the internet. Using the web as a media for delivering our application allows us to modify it periodically as we see fit. We choose to use a web browser application known as NCSA Mosaic, developed by the National Center for Supercomputer Applications at the University of Illinois. Mosaic is a browser that delivers HTML documents and presents them within a user-friendly window. Our application is implemented simply as an HTML document consisting of a fill-out form, that prompts the user for various attribute values and submits the completed form to the server. Our server processes the form and returns the retrieved document headings. The user then chooses from among those headings, which are themselves hyperlinks, to retrieve the content of the chosen document. This scheme provides us with a very cost-effective way to achieve our goal of reaching the broadest audience possible. A simple HTML form is also easy to maintain and modify. The drawback, as we later discovered, is that the amount of user interaction possible in our fully interactive GUI browser could not be implemented as an HTML file. This is because of the limited set of functions available in the current version of the document markup language. Our prototype interface application is a simple query-based form, which asks the user to supply one or more attribute values corresponding to our list of attribute types previously identified. Standard widgets such as input text boxes, radio buttons and pull-down menus are used to build the prototype. The resulting query string is tested with a standard server site which returns the values of all attributes submitted with the query.

The prototype White House publication browser consists of the following input areas, which closely corresponds with our proposed attribute list:

- Keyword --- a text area that allows the user to enter multiple words as search keys
- Subject area --- consists of a text area for entering multiple subject names, a pull-down menu for choosing a subject name from the available list, and a radio button for choosing whether the multiple subject names are used in conjunction or disjunction
- Source of document --- a pull-down menu for choosing the origination of the document: choices available include Office of the President, Vice President, First Lady, and so on. Default value is *Anywhere*, indicating the absence of constraint
- Target audience of document --- a pull-down menu for choosing the intended reader of the document: choices available include various types of special interest groups, professionals, educators, political groups, as well as the default value of *General Audience*
- Publication Date --- consists of a pull-down menu for selecting an approximate time frame: one of *Today, This Week, This Month, This Year*, and the default value of *Anytime*; a text area for entering the exact date in a *yymmdd-yymmdd* format; and a radio button for choosing which input method to use
- Type of document --- a pull-down menu for choosing the document type: one of *Speech, Statement, Article, Report, Draft*, and the default value of *Any type*
- Circulation --- a pull-down menu for choosing the minimum number of readers as a criteria for selecting a document; default value is *doesn't matter*
- Endorsement --- a pull-down menu for specifying the endorsement of a document from special interest groups or public opinion leaders, as a criteria for selecting the document; default value is *doesn't matter*
- Document retrieval limit --- a pull-down menu for specifying the maximum number of document headings to retrieve in one search; default value is *no limit*

The prototype browser interface represents our effort to put most of our proposed search attributes into a hypertext fill-out form. Nonetheless, we left out two attributes, namely key text and key document. We anticipate that the key text function will become

obsolete as hypertext links are used between documents. When our document server is in service, we will process each incoming publication and insert hypertext links within the documents, so that the user can follow hyperlinks to read related documents. For example, an article concerning the current state of health care reform would contain links to other documents mentioning events that led to the current state. The key document function, like the key text function, is originally intended as a search attribute for locating documents related to the current one. Hypertext links on the headings of a document would serve the intended purpose. In other words, in addition to adding hyperlinks from the text body of a document to another document, we will also insert links from section headings to related documents.

Once the user finishes modifying his chosen set of search attributes, he may push the submit button to see a list of all document headings retrieved. Each heading is presented as a hyperlink leading to the actual document content. The user simply clicks on one of these hyperlinks to read the publication. Since our project concerns itself primarily with the user interface issue, we have yet to implement a functional server to handle such search requests. Although there are a number of publications browser under development, such products do not provide the same set of search capabilities offered by our prototype interface. This makes it difficult to take advantage of existing servers for delivering document under our proposed scheme. As a continuation to our project, we expect future development to include the design and implementation of a publications server that is tailored specifically for our search model. Once an operable server has been completed, we may test-launch our publications browser interface. A demonstration of the browser interface is shown in Appendix A.

### 3.5 Limitations of the Prototype

The degree of user interaction possible using HTML documents is restricted to hypertext links, image map links, and fill-out forms. A fully interactive graphical user interface on

a canvas screen, similar to the one originally proposed, is not possible. User interface bandwidth -- the amount of information that can be presented to the user, or input collected from the user -- is limited using only fill-out forms. To this end, we considered various options, among them extending the current HTML specification. Several features will be added to improve the current markup language standard. These features are documented in Chapter 5.

Although an HTML extension would enhance the user interface bandwidth, it would still not be able to support full graphical interface capabilities that our model requires. We consider the possibility of merging the benefits of a Motif-based application's interactive capability, with those of an HTML-based application's generality and platform independence. We envision a new type of markup language, used on more advanced web browser interfaces, that will provide much more than the simple hypertext document we have today. Our effort leads to the proposal of a new hypermedia markup language standard, which we document in Chapter 6.

## 4 HTML Extension

### 4.1 Description of HTML functions

HTML functions are generally concerned with hypertext links, text presentation, fill-out form support, and image presentation. HTML uses tags to define the presentation style of the enclosed text. HTML tags consist of a left angular bracket ( < ), followed by the *directive*, and closed by a right angular bracket ( > ). The directive is the command word that defines the function of the tag. Most tags require a corresponding ending tag that is identical to the starting tag except that the directive is preceded by a slash ( / ). The following is an example of all directives provided by HTML 2.0, grouped by their general purpose:

#### Hypertext Links

##### ***Anchor***

An anchor highlights a region of text and establishes a link from that region to another document. The anchor

```
<A HREF='Example.html'>Here is an example</A>
```

highlights the text between the beginning and ending tags and links it to the document file named "Example.html".

##### ***Uniform Resource Locator***

A Uniform Resource Locator consists of the type of a resource being accessed as well as the path of the file, in the following format:

```
resource://host.domain[:port]/path/filename
```

where resource is one of:

file            a file on the local server or an anonymous ftp server  
http            a file on the WWW server  
gopher         a file on the gopher server  
WAIS           a file on the WAIS server

Furthermore, anchors from a region of text in one document can be made to specific parts of another document, thus increasing the usefulness of hyperlinks.

## Text Presentation

### ***Title***

<TITLE>       specifies the title of the document

### ***Header***

<H1>           a first level header  
<H2> . . . . . <H6> are available as well

### ***Paragraph***

<P>            specifies end-of-paragraph, no ending tag is needed

### ***Unnumbered List***

<UL>           <LI> tags precede each individual item (no ending tag needed)

### ***Ordered List***

<OL>           <LI> tags again precede each individual item (no ending tag needed)

### ***Description List***

<DL>           displays description titles followed by an indented description text  
<DT> tags precede a description title  
<DD> tags precede a description text

***Nested List***   lists can be arbitrarily nested into multiple layers

### ***Preformatted Text***

<PRE>         displays the enclosed text in a fixed-width font and respects all spaces, new lines and tabs; useful for program listing

### ***Extended Quote***

<BLOCKQUOTE> displays the enclosed text as a separate block

### ***Address***

<ADDRESS>     specifies the author and his/her contact information

### ***Character Formatting***

<I>            italic font  
<B>            bold font  
<TT>          fixed width text

Some character formats are not available on all WWW viewers, although the above three are supported by NCSA Mosaic.

## Image Presentation

### *Inline Image*

`<IMG SRC="filename.GIF">` displays a GIF (or Xbitmap) image on the WWW browsers; currently available only on NCSA Mosaic

An inline image can serve as a hyperlink to another document or to an enlarged external image of itself. This can be done by defining the image as an anchor region.

### *External Image*

An external image can be accessed by defining an anchor that links to a graphics file, in either GIF, TIFF, JPEG, RGB, or HDF format.

## 4.2 HTML+ Features

HTML+ is an extension to HTML, and is currently under development. Like HTML, HTML+ is designed for use in the WWW as a delivery format for wide-area hypertext. It stems from several years of experience with HTML from the WWW user community. Discussions held among users resulted in a number of proposals for extending the current implementation of HTML to beyond its current capabilities. New ideas are collected at WWW workshops held over the past year. The result is a revised document format proposal which become known as HTML+.

Since HTML+ is designed as an extension to HTML, it provides backward compability with the older standard. The new markup language is designed to serve as a gradual transition from the current one, adding features such as tables, captioned figures, fill-out forms for database queries, and support for mathematical formulae. HTML+ also allows large documents to be broken down into smaller ones, each defined by a unique node that is accessible from the beginning node of the document. A description of the additional features provided by HTML+ follows:

## Hypertext Link

Several attributes of the anchor element are changed.

|                |  |
|----------------|--|
| <b>ID</b>      | used in place of the <b>name</b> attribute to uniquely identify an anchor;   |
| <b>REL</b>     | defines the relationship between the linked document and the current one;<br>long document can be subdivided into shorter ones by specifying<br><b>REL=Subdocument</b> ;   |
| <b>EFFECT</b>  | specifies the presentation method of the document;<br><b>EFFECT=Replace</b> causes the current document to be replaced by the<br>linked document; <b>EFFECT=NEW</b> causes the linked document to appear in<br>a new window; <b>EFFECT=OVERLAY</b> causes a pop-up window to appear,<br>showing the linked document; |
| <b>PRINT</b>   | specifies the relevant parts of the current document to be printed   |
| <b>SIZE</b>    | size of the linked document; serves as a guide to show progress when<br>retrieving documents   |
| <b>METHODS</b> | list of HTTP methods for rendering the linked document   |
| <b>SHAPE</b>   | used in metamap links that defines the shape of buttons on an image  |

## Normal Text

### *Annotations*

|                 |   |
|-----------------|---|
| <b>FOOTNOTE</b> | the text enclosed within the <b>footnote</b> tag can be accessed by clicking<br>the hypertext button that immediately follows the text; a pop-up window<br>appears and shows the footnote text                |
| <b>MARGIN</b>   | the text enclosed within the <b>margin</b> tag appears on the side of the<br>paragraph to attract reader's attention  |
| <b>NOTE</b>     | the <b>note</b> tag causes the enclosed text to appear in an indented block and<br>supports the use of graphical icons to represent the role of the note, which<br>may include note, tip, warning, and error. |



### ***Document Amendments***

- CHANGED**     the `changed` tag causes the enclosed text to appear under a change bar; this signifies that the text has been modified
- REMOVED**    the `removed` tag causes the enclosed text to appear under a strike-through line; signifies that the text has been removed in the current version
- ADDED**        the `added` tag signifies that the text has been added

### ***Conditional Text***

- ONLINE**        the text enclosed within the `online` tag appears only on documents that are read on-line using a browser; it does not appear on the printed version
- PRINTED**      the text enclosed within the `printed` tag appears only on the printed document and does not appear when read with a browser

### **Figures**

- FIG**            the `fig` tag is similar to the image element and replaces it
- CAPTION**      the `caption` tag provides a figure element with a description text
- ISMAP**        the `ismap` attribute, to be used within the `fig` tag, specifies that the figure supports mouse click and mouse drag on specific areas of the figure; the click or drag information is sent back to the server along with the suffix “`?x=X&y=Y`” for mouse clicks and “`?x=X&y=Y&w=W&h=H`” for mouse drags

### **Tables**

- TABLE**        the `table` tag specifies a table format including the contents of each cell within the table; similar in style to that used in Latex documents
- TH**            specifies table header cell
- TD**            specifies table data cell
- TR**            specifies end of a table row

## Mathematical Functions

HTML+ provides a set of tags used for specifying the appearance of mathematical formulae. These tags include **SUB** for subscript, **SUP** for superscript, **OVER**, **BOX**, and **ARRAY**, all set within the beginning and ending pair of a **MATH** tag.

## Fill-Out Forms

In addition to the original fill-out form types supported by HTML 2.0, HTML+ further provides support for the following types of fields:

|                 |  |
|-----------------|--|
| <b>INT</b>      | allows integer numbers only  |
| <b>FLOAT</b>    | allows floating point numbers only   |
| <b>DATE</b>     | allows the input of a specified date format  |
| <b>RANGE</b>    | restricts an integer type input to within a specified range  |
| <b>IMAGE</b>    | specifies an image area which serves as an input, producing values corresponding to the x and y coordinates of the mouse click                         |
| <b>SCRIBBLE</b> | a blank canvas area that allows pen or mouse input, producing values corresponding to time, coordinates, and pressure data                             |
| <b>AUDIO</b>    | allows the user to enter spoken messages into the form, using a number of standard tape controls such as playback, forward, rewind, and stop functions |

## 4.3 Additional Features For The HTML Extension

### *The Adaptive Survey Form*

In addition to the standard features found in HTML 2.0 and the proposed features in the HTML+ initiative, several other functions would be useful for implementing a truly interactive interface for our document browser. If we consider the current scheme of using e-mail surveys to identify the user's subscription preference, we can see the

potential demand imposed upon the server. Under the current scheme, each user has to answer a number of questions in several surveys. After completing each survey, the user must send the reply over e-mail back to the server. The server relays the message to the appropriate handler which automatically reads the incoming survey and mails yet another survey to the user for further information. This process of sending e-mail back and forth can take up a lot of time from both the user and the server. Even if we implement the survey in HTML such that a user can use a WWW browser to read it, the series of survey must each reside in a different document and the user would still have to send in several form submissions back to the server for processing. This scheme, like the one used in the original e-mail model, would cause just as much traffic at the server site. While this is not a problem with the current usage pattern, we certainly foresee a potential influx of survey traffic in the near future, especially when more people have access to the web and use our services. An HTML feature that allows the server to download part of the survey processing task to the client site would greatly reduce the load to the server.

Each survey in the document browser interface is implemented as an HTML fill-out form. In the current version using Mosaic, the user must submit the form upon completion. This causes the form to be sent back to the server site and thus increase the server load. If there is an HTML feature that allows a completed form to be locally processed, a submitted form can lead to another form that has been downloaded to the client during the first transmission from the server. This can be accomplished, for example, using conditional text that checks a set of valid variables against some prerequisites in order to determine whether or not to display the body of the text. In the case of our document browser interface, conditional forms can be very useful for surveys targeted at our users. We may be interested in using a usage survey form to gauge the general usage pattern of our potential users. In this survey, the user is asked several questions related to the document source, usage pattern, social affiliation, cost of usage, distribution mode, and distribution audience. Responses to these questions are used to determine the user type. Users are generally considered either as customers or distributors. Customer users may either be non-commercial, commercial, or they may be

former customers or non-customers. Distributors are either commercial, educational, advocate, governmental, company, individual, or they may be non-electronic distributors. Several logic clauses are used to determine the user type, which may be any one of the above customer or distributor types. The browser then shows the next set of questions. Depending on the user's responses in the first survey page, user-type-specific questions are shown on the next page. While some general questions may appear in surveys for more than one type of user, other questions may be unique to a particular user type. Responses in each survey page, in turn, are used to determine other variables that lead to specific questions in subsequent pages. Once the user has reached the end of the survey, she may click on the Submit button to send the survey form to our server. This logical capability enables the browser to process the response locally at the client site and eliminates the need to send each survey response back to the server. All surveys are loaded to the client site as a single document at once, while the user submits the form back to the server only after all necessary survey pages have been completed. The format of the new features are as follows:

***Newpage Feature:***

```
<input type="nextpage" value="Go to the Next Page" >
<input type="firstpage" value="Go back to the First Page" >

<newpage>
```

***Conditional Text Feature:***

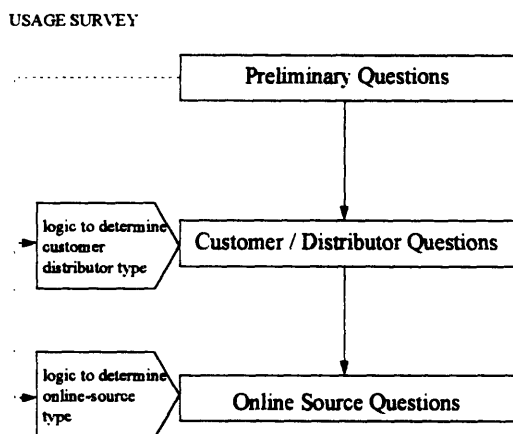
```
<condition [and / or] [not]
    name1="name of variable 1" value1="value of variable 1"
    name2="name of variable 2" value2="value of variable 2"
    name3="name of variable 3" value3="value of variable 3"
    .....
    >
    (body of conditional text)
</condition>
```

The newpage tag separates an HTML document into a number of pages. The user can only see one page at a time. By pressing the "nextpage" input button, the user can turn to the next page and see the text lines starting after the <newpage> tag and ending right

before the following <newpage> tag. Another button, the “firstpage” input button, allows the user to move back to the first page of the document. The browser, upon encountering a <newpage> tag, simply stops formatting the document elements that come after the tag and marks the first element following the tag as the starting point of the next page. When the user presses the “newpage” input button, the browser will start formatting the part of the document beginning at the mark. Besides its obvious function of separating a document into a number of pages, the newpage feature is essential to implementing adaptive survey forms. Since the browser uses responses from previous pages to determine which subsequent pages to show, a newpage function with an explicit “nextpage” input button allows the browser to collect all input values and check them against the predicates of the <condition> tags in the following page before actually displaying it.

A <condition> tag is used to mark up a text body such that it is displayed only if the predicates specified along with the tag are satisfied. Possible logic types include AND, OR, and NOT. By default, the AND logic is assumed. The NOT logic can be used with AND to form a NAND logic, or it can be used with OR to form a NOR logic. Besides the logic attributes, the <condition> tag also includes pairs of name and value attributes. Here, a predicate pair is specified by the equality pairs of *name1=' name of variable 1' value1=' value of variable 1'* where *name of variable 1* refers to the variable name specified with an input tag, and *value of variable 1* refers to the value of the input variable as determined by the user's selection or typed input. In the current implementation, the script programmer can specify up to 10 such name and value pairs within a <condition> tag. These attributes are referred to as name0 through name9 and value0 through value9, respectively. The browser checks for the validity of each name-value pairs and whether the content of the name attribute matches with any of the existing input variable names, and if so, whether the content of the value attribute matches with the corresponding value of that variable. If the matching pattern of all name-value pairs satisfy the specified logic type, then the body of the conditional text is displayed, otherwise it is hidden.

We have completed implementation of the <newpage> and <condition> features. This is done by writing an extension to the current version (2.x) of NCSA Mosaic. In order to test the resulting product, we have implemented a demonstration version of an existing survey form in use by the Intelligent Information Infrastructure group at the MIT AI Laboratory. The survey form consists of three logical groups of questions, which we can conveniently implement as three survey pages with the programming logic handled by our conditional text feature.



Usage Survey Form Logical Structure

The first survey page consists of preliminary questions for determining the user type, followed by the second page that contains specific questions targeted at each user group. The final page consists of questions related to the online distribution source from which the user obtains White House documents. We have successfully demonstrated that the inherent logic found in the original server-based survey form can be implemented using the newpage and conditional text features in our extended language. A demonstration of the actual user session is shown in Appendix B. A quick reference guide to the new extension, consisting of the language specifications, modified portions of the source code, compiling instructions and the sample survey script, is shown in Appendix C.

## 4.4 Graphics Extension

As we have mentioned earlier, in one of our original proposals the document browser uses graphical icons similar to those found in the Macintosh file operating environment. The interface consists of a canvas background, on which file icons and directory folders are displayed. The user may manipulate some of the file icons to represent the actual movement of files from one directory to another. These file icons also change when a file is created or deleted, opened or closed, and so on. HTML does not provide any support for graphical interaction, although users can click at certain points on an image map which act as anchor links to other documents. A fully interactive canvas map, supporting icons that can be clicked and dragged, would prove very useful to our icon-based interface model.

Our proposed interactive graphics feature consists of a canvas area tag:

```
<canvas name="Drawing_Board" width=300 height=200>
```

Within the canvas, several types of objects are allowed:

### ***Icon***

An icon is a graphical object on the canvas area that can be clicked and dragged. The shape and appearance of the icon is defined either by a GIF image or by simple attributes such as shapes and colors.

```
<icon type="image" name="Key" src="key.gif"  
      x=50 y=70 click="active" drag="active" >
```

This icon shows the image of a key from the source file `key.gif`, and appears with the top left corner of the image aligned at the x,y-coordinates of (50,70). The icon is defined as *click-active*, meaning that a mouse-click on top of the icon results in the creation of an event. When the user clicks on the icon, the browser processes the event according to a set of event-handling rules, defined at the end of the canvas area context. Similarly, the icon is also defined as *drag-active*, allowing the icon to be moved from one location to

another on the canvas area. When the drag is completed, an event is sent to the browser, which then checks for the appropriate event-handling rule.

An icon can be defined without a graphical image, using only geometric definitions to determine its appearance. For example, an icon defined as type="polygon" can take on certain attributes such as the shape of a circle, the color red, and a certain size.

```
<icon type='polygon' name='Red_Circle' x=120 y=70  
shape='circle' color='red' fill='yes' diameter=20 >
```

Another example would be:

```
<icon type='polygon' name='Black_Square' x=160 y=70  
shape='rectangle' color='black' fill='no' width=10 height=10 >
```

Another type of graphical object is the line. A line is defined by the starting and ending icons that it connects with. It represents a link between the two objects. The exact rendition of the line with respect to the icons is determined by the browser.

```
<line from='Red_Circle' to='Black_Square' style='dotted' >
```

In most graphical user interfaces, the button is an indispensable channel for user input. A simple graphical object that allows the creation of a button would be useful:

```
<button name='Hello_Button' label='Hello!'  
action='html://xxx.edu/hello.html' >
```

Here, the button tag produces a standard button shape with the label displayed on top of the button. The click-action is defined within the tag, since buttons are usually clicked and not dragged around.

At the end of each canvas area context, a set of interaction rules define the behavior produced by clicking and dragging the objects. For example, we can define the action resulting from putting a red circle on top of a black square.

```
<interaction type='overlap'  
top='Red_Circle' bottom='Black_Square'  
action='html://xxx.edu/red_on_black.html' >
```

This tag causes a link to the specified file whenever the red circle is dragged to overlap the black square. The following tag causes yet another link when the red circle is double-clicked.

```
<interaction type='doubleclick' object='Red_Circle'  
action='html://xxx.edu/click_red.html' >
```



If the canvas area is used as a special type of form, then we can insert a *submit* button at the end such that all the actions performed on the icons can be interpreted at once by the server. A URL extension that appends a list of all positions of all icons is sent back to the server for further processing. In the current example, after moving the red circle over the black square, the browser may send the following URL to the server when the *submit* button is clicked.

```
http://xxx.edu/post-query?Red_Circle.x=163&Red_Circle.y=71&Black_Square.x=160&BlackSquare.y=70
```

## 4.5 A Virtual-Reality Composition Language

A number of WWW users have visualized a new hypermedia markup language that can handle 3 dimensional compositions of graphical objects. Such a language would make feasible the exchange of 3-D graphical interaction found in virtual reality environments. Users can therefore access and explore a virtual cyberspace over the web, engaging in 3-D internet games and experience virtual representation of the internet world. 3-D graphics capability on the Web would also prove useful to education, medicine, manufacturing, engineering, or even as an intuitive interface for some consumer services such as shopping and banking. Our proposed 3-D *city-view interface* would certainly be feasible if such a hypermedia markup language exists.

The IEEE has already begun developing what is known as the Distribution Interactive Simulation (DIS) standard that can links 3-D graphical simulations at several sites to create a virtual environment for simulating interactive activities. Nonetheless, it will not be launched to the public domain until much later. Meanwhile, a number of web users are proposing to use CGM, a graphical object description language currently used by most people in the graphics business. We certainly anticipate that in a few years' time we can launch a 3-D interactive graphics version of the document browser. The information highway will not be the same when so many applications, including document browsers, enter the virtual reality world.

## 5 HyperMedia Design

Much effort has been devoted to the investigation and design of new hypermedia interfaces. The task identified is to anticipate the formats of future on-line documents and to design a standard markup language and corresponding web browser to handle them. We draw our inspiration primarily from existing computer-assisted instruction (CAI) applications. We studied the general model of on-line education and found that there are four main design issues: display, interaction, communication, and control (Chabay, 1992). Understandably, computer-assisted instruction is only one potential application of future on-line documents. Nonetheless, we believe that the lessons learned from CAI design can help us gain a deeper insight into the possibilities and limitations of interactive documents.

### 5.1 Document and Program Model

In the current HTML standard, a document is a file that consists of text and some markup tags giving hints about presentation styles, image files, and hyperlink anchors to other related documents. It uses the simple metaphor of a document -- which we commonly associate with papers, news articles, books, brochures and forms -- and extend the domain of the document to include audio and video capabilities as well. These items typically present information in a sequential order: readers are expected to read the document from the beginning to the end. The only exception is when the reader is referred to a footnote or an external reference, in which a hyperlink would make the task of moving between documents much easier. Nonetheless, the flow of the document is

generally sequential, thus preserving the style of documents as presented in traditional media.

On the other extreme is the application program. An application program, unlike a hypertext document, does not draw its metaphor from traditional documents. Rather, there is the analogy with traditional tools, instruments and equipment. For example, an engineering control application would provide a user interface that resembles the dials and switches on a real control panel. User interaction with the application typically does not follow a linear format. Rather, information is presented to the user in a number of states, in a pattern that can be described by a finite state diagram. Most application programs, therefore, are written in languages that support a more object-oriented approach. Whenever the user enters a command, by typing in some text or pointing with a mouse, the system generates an event and handles it according to a set of pre-defined event-handlers. Application program development therefore requires a programming model that is very different from that of a traditional document. Even if we consider the use of hyperlinks, the amount of interaction possible with a pure document format is still limited.

## 5.2 Interactive Document

If we consider the difference between a document and a program, we would find that the two represent a different approach to presenting information to the user. A document typically presents information in a sequential order, allowing only limited user interaction in terms of selecting what to see. A program, on the other hand, presents information in an order determined primarily by the user's action. At a very general level, a program takes in user input, processes the incoming information and presents the result back to the user. Which model is more appropriate for the development of a particular application product depends on the nature of the product itself. Clearly, if the product is an on-line version of a text book, the document model suffice. On the other

hand, if the product is a computer simulator for the elementary laws of physics, only the program model can handle the type of user interaction and information processing required to make the simulation work. Which model, then, shall we use for an on-line instructional application that teaches the user the fundamentals of physics? On one hand, the application resembles a document, in which instructional material is presented using text and graphics in sequential order. On the other hand, the application resembles a program, in which the user can provide some input and see the result of the simulation after that input has been processed. Computer assisted instructions therefore require an information model that is a hybrid of the traditional document model and the interactive programming model. This is where the concept of interactive documents comes into play.

The document model and the program model each have its advantages. While the document model is limited in terms of the user interaction it can support, writing a script for a document is easier than writing a program. A document written in a markup language is much shorter than the same document written using a programming language (using many *printf* commands in C, for example). A new markup language that supports the current lightweight format used to present text, but further provides additional functions that support user interaction, would prove very useful. What we are envisioning here is the highly interactive document, in which not only text and graphics, but also application programs in its traditional sense, are supported. That is, there is no longer the distinction between a text file and an application program. Both can be implemented in our new markup language and viewed (or used) with the same web browser.

### 5.3 Functions

In order to design an interactive markup language, we first considered the types of applications which we plan to support. In addition to the text and graphics capabilities

supported by current hypertext markup languages, our proposed language further supports the following standard application types:

- spreadsheet
- drawing
- presentation graphics
- animation graphics

### ***Spreadsheet***

Most document markup languages, including LaTeX and HTML+, support tables within the document. While it is possible to edit the content of a table, it is impossible to perform any calculations on its cell values. In order to make a document fully interactive, its tables must also support user interaction. This requires that the markup language support not only the presentation of values in a table format, but also the tabulation on those values according to user-definable formulae. In other words, the user can view the table as a self-contained spreadsheet application. This function is already available in commercial products such as Microsoft Office™, which with its Object Link Environment (OLE) standard has made it possible to include a spreadsheet application or a presentation graphics slide within a document. The user can edit the contents of the spreadsheet table or the graphics slide as if she were using an external application program from within the word processor. A similar idea can be applied to the design of our new markup language. The document may contain a table, which the user can choose to modify its value. A spreadsheet engine loaded from the library of application functions will handle the calculation. A document-based spreadsheet function, however, may support only a small subset of all spreadsheet functions found in a full-scale application. Simple mathematical, logic, and statistic functions will suffice for most application needs. With the spreadsheet support, documents such as the tax preparation form can be transmitted over the internet. The document may include a pre-formatted table that contains all cells found on the tax form, as well as pre-defined formulae for calculating the required taxes. The user simply inputs all relevant numbers and the document automatically calculates the total tax owed to the government. The user can

submit the completed tax form to the IRS server. This functionality would be impossible with either a document or a spreadsheet application alone.

### ***Drawing***

Current hypertext markup languages support image frames and limited interaction through mouse clicks within the frame. While it is feasible to view scanned-in images by loading in image files from an external source, simpler polygon-generated graphics may be loaded much more quickly if the markup language supports graphical descriptions as well. Graphical images represented as combinations of simple shapes and lines may be loaded as a graphical data structure rather than as a GIF image. Furthermore, users can manipulate the image by clicking and dragging graphical objects on top of it. It is therefore possible that users draw on top of an image canvas within a document, just as they would do with a drawing application program. The drawing function is in turn handled by a special drawing engine from the applications library.

A document that supports drawing functions may not only allow the user to use it as a virtual canvas, but also as an efficient means of user input. For example, a chess game between two players across the web is made possible by a simple interface application that displays a chess board and the chess pieces. Each player takes turn to drag a piece, sends a message to the chess server that handles the logistics of the game, and sees an update of the chess board while waiting for the other player to move. Interactive graphics is especially useful for instructional purposes, where virtual experiments are performed by manipulating graphical objects. Other applications include control panel applications, where special graphical objects represent the switches and dials of the actual equipment under control.

### ***Presentation Graphics***

Presentation graphics applications use the metaphor of slides. A typical presentation consists of a series of slides, each showing some text or chart to bring across an idea clearly without overwhelming the audience with too many words. A slide that further

allow user interaction by including buttons and active graphical objects can more efficiently convey an idea. Hypercards are an example of such interactive slides. Using the card stack metaphor, hypercards are especially powerful as a tool for computer assisted instruction applications. In order to incorporate the card model with our document model, it is necessary to provide the functionality of breaking a document into multiple pages, each representing a card of different size. Simple buttons allow the user to move between pages sequentially and through hyperlinks. A business proposal, for example, can be written with our markup language using the card metaphor, and delivered across the network to another party.

### ***Animation Graphics***

Besides static slides, some animation of the graphical objects can add interest and information to a presentation. This is especially true for instructional applications where demonstrations are often useful as an aid to help with the user's understanding. A tutoring application that teaches the fundamentals of dynamics, for example, may show the actual accelerating movement of a ball object upon collision. This movement can be represented as a series of time-consistent frames depicting the movement of the object. Another application example that uses animation graphics is the display of a weather map, showing series of graphs representing interval snapshots of predicted weather conditions. Animated graphics provide an extra dimension to static presentation graphics, in which a time component depicting an arbitrary time scale is possible.

## **5.4 Interface Features**

An interactive document markup language that supports the applications cited above must provide the necessary interface features:

- application-specific menu bar, tool bar, and information bar
- customized menu bar, tool bar, and information bar
- interface widgets within the document with real-time event handling

- text label boxes
- navigation buttons

Each application function (spreadsheet, drawing, presentation and animation graphics) comes with its standard set of features supported by the web browser. These features are presented in a standard menu bar with buttons and controls that change as the user moves from one type of application to another within a document. For example, when the user clicks on a text area, the default buttons for controlling a browser application applies.

The menu bar changes its buttons to those for controlling spreadsheet applications when the user clicks on a table, and those for controlling drawing applications when the user clicks on a canvas. The browser therefore acts as a standard graphical user interface platform that adapts itself so that different types of applications can run on it. With the document model, we make it possible that several application types coexist within a single document. The concept of an application is no longer distinguishable from that of a document. A document may contain plain text, graphics, tables, and other data types, each of which linked to its specific application feature module that modifies the user interface of the browser to accommodate user interaction with its content. The markup language also supports new types of applications by allowing user-modifiable menu bars and buttons. For example, we can build a chess game interface by modifying an existing drawing application interface, to include special buttons for changing the difficulty level and other relevant controls.

Each application function is defined by an external logistics module residing in the applications library. Whenever the user clicks on a special data type, such as a table area, the browser looks up the table tag from a list of standard application features and locate the logistics module from the library. It loads in the module and changes its interface features such as menu and information bars. All user interactions with the table result in an event that is handled by the logistics code.

To provide user interaction, the browser must support a number of different interface widgets. This includes buttons, slides, radio and check boxes, pull-down menus, text



input areas and dialog boxes. Each of these widgets, similar to those found in Motif and unlike those supported by HTML fill-out forms, produce events that are handled immediately according to a set of rules defined as part of the document. By downloading most of the interface logistics to the document level, the server is freed from handling simple requests resulting from user interface events. Variables associated with each input widget are valid across the context of the entire document. These variables can be used as predicates to determine the presentation of the current document or submitted to the server as queries.

Many applications, especially computer assisted instructions, require the use of demonstrations and diagrams to illustrate an idea. Text labels that are presented on top of a graphical image or alongside an animation graphics scene are especially useful as aides to help the user understand the message. Furthermore, conditional text labels that are shown only when the user moves the mouse pointer to certain areas of the document or the menu bar, are helpful for providing the user with information about the purpose of the pointed item. Finally, to help the user navigate within a page and between pages of the document, navigation buttons and indicators such as a document hierarchy map would prove helpful.

## 5.5 Authoring Tool

As the document markup language provides more features and supports more types of standard and user applications, the language itself would inevitably become more complex. While the development of simple hypertext documents requires only a few command tags, writing more complex user interfaces takes a considerably longer script. The author has to write some program code to handle interface events and application logistics. An authoring application that allows developers to write documents using a user-friendly, what-you-see-is-what-you-get environment will make the task of developing on-line documents much easier. In addition to the standard word-processing

capabilities, the authoring tool must further provide editing functions for other data types including images, drawings, spreadsheets, presentation graphics and animation graphics. Standard user interface widgets are also available by simply selecting them from a menu. The user can therefore type in the text, select an input widget, open a canvas map, draw in some graphics, and even prepare each frame of her animated presentation using the authoring tool. She can test-run the application as she develops it, all within the same interface.

The authoring tool, like any one of our standard application features, exists as a virtual application sitting on top of the browser interface platform. By choosing the editing mode, the user can begin modifying an existing document or create a new one, all of which is done within the same browser interface used for reading documents. To further support the development of programming logistics, we will further include a function that allows users to develop the logistics of new applications in a graphical environment. Not only will a user-friendly authoring tool encourage document development, but it will also give the reader a way to interact with existing documents in ways never before possible.

## 6 Conclusion

Since the beginning of the research project, we have focused on building a better user interface for accessing documents. We thought about computer-human interface issues, and explored the outer bounds of interface design. Using a paper demonstration, we were able to show how one may build a highly interactive interface using existing GUI platforms. Just when we were planning to build the interface application, we noticed the exciting opportunities that a World Wide Web document browser can bring to the user. A uniform markup language, HTML, supports a variety of document types, including the capabilities of video and audio output, images and fill-out forms, and most importantly, hypermedia links between documents. With user-friendly graphical browsers such as NCSA Mosaic, exploring all the information resources from the Internet has never been easier.

We noticed the widespread use of web browsers for accessing hyperlinked documents across the internet, and began to think that our own White House documents browser could be distributed as a WWW document as well. Exploring with HTML and Mosaic, we found that the unique qualities of a document is fundamentally different from those of an application program. While documents can be easily distributed using network browsers, they lack the logical and control capabilities characteristic of programs. We recognized a need to modify the markup language specifications so that it supports logic. Our implementation of the HTML extension helps us realize this goal. Adaptive survey forms that process user inputs at the client site are now a reality, as we have successfully demonstrated using the sample usage survey form.


An interesting question still remains. While we succeeded in introducing logic control into a document, we have yet to reconcile the fundamental differences between the document model and the programming model. Specifying logic control in the style of a markup language results in code that is inevitably verbose. This is complicated by the snapshot view of the document which limits user interaction. To achieve the effect of user interaction in an inherently sequential and static document remains a challenge. A current trend seen in HTML documents is to use external viewers to handle anything that cannot be presented within the traditional document model. This approach, however, does not provide the seamless document view that a truly universal markup language can provide. We are certainly interested in seeing a much more general language that is equally suitable for implementing text-based documents as it is for the multimedia and interactive varieties. We hope our effort in introducing logic and control to the HTML language would serve as a beginning to this pursuit.

# Appendix A

## Search Interface

File Options Navigate Annotate Help

---

Document Title:  

Document URL:

---

### White House Publication Browser

---

Please select at least one of the following properties to be searched for.  
Then press the **submit** button to search the database.

---

Keywords:

---

Subject Areas:

Search for **any**  or **all**  of the items typed above or selected from the list:

|                      |
|----------------------|
| ECONOMY              |
| Agriculture          |
| Business             |
| Energy               |
| Infrastructure       |
| Macroeconomic Policy |
| Technology           |
| Welfare              |

(use CTRL to select multiple items)

---

Source of Document:

---

Target Audience of Document:

---

Document Title:

Document URL:



Search for **any**  or **all**  of the items typed above or selected from the list:

- ECONOMY
- Agriculture
- Business
- Energy
- Infrastructure
- Macroeconomic Policy
- Technology
- Welfare

(use CTRL to select multiple items)

Source of Document:

Target Audience of Document:

Publication Date:  Choose from the default list:

OR  Specify as yymmdd-yymmdd, e.g. 940125-940211 :

Type of Document:


Circulation: The document must be read by at least:

# Appendix B

## Sample Usage Survey

File Options Navigate Annotate Help

---

Document Title:  

Document URL:

---

### Usage Survey

---

Thank you for completing this usage survey. This survey is intended to measure access to and use of the electronic publication of White House documents. The survey results can guide improvements in this new information resource and provide insights into how electronic access to government might enhance the democratic process.

This is a real-time adaptive survey form. You will see a number of survey pages, each consisting of several questions. After completing all the questions on a page, you may proceed to the next page by pressing the CONTINUE button. Your response to previous survey pages will determine which page you will see next. At the bottom of the final page there is a SUBMIT button, which you can press in order to submit the entire survey form.

Thank you again for your time.

---

• **How do you currently access White House documents?**

---

• **What routines do you follow for handling, saving and deleting the White House documents?**

---

Document Title: Conditional Usage Survey Form

Document URL: file:///localhost/afs/athena.mit.edu/user/f/r/



• How do you currently access White House documents?

*Email from a mailing list to my email address on Internet.*

• What routines do you follow for handling, saving and deleting the White House documents?

*Save the interesting documents.*

• Are you acting for or affiliated with one of the following types of institutions when you handle the documents?

*College or university.*

• On a scale of 1 to 5, how satisfied are you with the electronic publication of these documents?

Not satisfied  1  2  3  4  5 Very Satisfied

• Do you pay either to get the documents or to use the computer account through which you get the documents?

Yes  No



Document Title: Conditional Usage Survey Form

Document URL: file://localhost/afs/athena.mit.edu/user/f/r/



On a scale of 1 to 5, how satisfied are you with the electronic publication of these documents?

Not satisfied v 1 v 2 v 3 v 4 ^ 5 Very Satisfied

Do you pay either to get the documents or to use the computer account through which you get the documents?

v Yes ^ No

If you distribute the White House documents or their information, what means do you use?

*Distribution over a local or wide-area computer network.* \_|

Please estimate as best you can the number of people who routinely receive or access from you the White House documents or reports on their content.

*4 - 7 people.* \_|

Press here to continue to the next page... CONTINUE

Press here to reset the survey form... RESET

Document Title: Conditional Usage Survey Form

Document URL: file:///localhost/afs/athena.mit.edu/user/f/r/



● **Non-commercial Customer Survey**

● From what source did you discover how to access White House documents via a computer network?

● How many hours per week on average do you spend viewing, listening to, or reading offline news or other material on politics?

● How politically active are you?

● Which computer site did you use to retrieve White House documents?

Please indicate the host name, e.g. clinton.ai.mit.edu :

Press here to submit the survey form...

Press here to go back to the first page...

Document Title:

Document URL:



☛ Are you acting for or affiliated with one of the following types of institutions when you handle the documents?

☛ On a scale of 1 to 5, how satisfied are you with the electronic publication of these documents?

Not satisfied  1  2  3  4  5 Very Satisfied

☛ Do you pay either to get the documents or to use the computer account through which you get the documents?

Yes  No

☛ If you distribute the White House documents or their information, what means do you use?

☛ Please estimate as best you can the number of people who routinely receive or access from you the White House documents or reports on their content.

Document Title: Conditional Usage Survey Form

Document URL: file:///localhost/afs/athena.mit.edu/user/f/r/



● **Commercial Customer Survey**

● From what source did you discover how to access White House documents via a computer network?

● How many hours per week on average do you spend viewing, listening to, or reading offline news or other material on politics?

● How politically active are you?

● Please type the name of the commercial network or information service through which you get the documents:

● Approximately how much do you pay in the average month to receive or access the White House documents?

[Back](#) [Forward](#) [Home](#) [Reload](#) [Open...](#) [Save As...](#) [Clone](#) [New Window](#) [Close Window](#)

Document Title: Conditional Usage Survey Form

Document URL: file://localhost/afs/athena.mit.edu/user/f/r/



• Are you acting for or affiliated with one of the following types of institutions when you handle the documents?

*College or university.*

• On a scale of 1 to 5, how satisfied are you with the electronic publication of these documents?

Not satisfied v 1 v 2 v 3 v 4 v 5 Very Satisfied

• Do you pay either to get the documents or to use the computer account through which you get the documents?

v Yes v No

• If you distribute the White House documents or their information, what means do you use?

*Distribution over a local or wide-area computer network.*

• Please estimate as best you can the number of people who routinely receive or access from you the White House documents or reports on their content.

*101 - 500 people.*

Document Title: Conditional Usage Survey Form

Document URL: file:///localhost/afs/athena.mit.edu/user/f/r/



**● Educational Distributor Survey**

☛ **By what means do you distribute or provide access to the White House documents?**

☛ **Do you distribute all the electronic documents released by the White House each day, or only more specific documents?**

☛ **How many people per day, on average, receive or access the White House documents directly from your service?**

☛ **Are the White House documents you distribute used as texts or sources in classroom discussions?**

**Press here to submit the survey form... Submit**

Document Title: Conditional Usage Survey Form

Document URL: file://localhost/afs/athena.mit.edu/user/f/r/



● **Advocate Distributor Survey**

● **By what means do you distribute or provide access to the White House documents?**

● **Do you distribute all the electronic documents released by the White House each day, or only more specific documents?**

● **How many people per day, on average, receive or access the White House documents directly from your service?**

● **What percent of the people whom you would like to reach are already on-line and able to receive the information you distribute?**

Press here to submit the survey form... Submit

Document Title: Conditional Usage Survey Form

Document URL: file:///localhost/afs/athena.mit.edu/user/f/r/



● **Commercial Distributor Survey**

● **By what means do you distribute or provide access to the White House documents?**

● **Do you distribute all the electronic documents released by the White House each day, or only more specific documents?**

● **How many people per day, on average, receive or access the White House documents directly from your service?**

● **How are customers charged for accessing or receiving daily White House documents?**

Press here to submit the survey form... Submit



# Appendix C

## Quick Reference Guide to the Extension

The new extension consists of two new HTML mark types:

### *Newpage*

The newpage tag marks the end of a page and the beginning of the next page. Text elements after a newpage tag are not displayed within the same window with those before the newpage tag. Explicit input tags with special types “nextpage” or “firstpage” are used to create the input buttons, which the user can click to move to the next page or the first page of the document, respectively. These special input tags are the only means that the user have for moving through pages. Similar to any input tag, nextpage or firstpage input tags must exist within a fill-out form. Notice that there are no ending tags for the newpage tag, which marks the location of a page break.

```
<form action="http://xx.xx.xx/server.html" method="post">  
  (text of page 1)  
  <input type="nextpage" value="Goto Page 2">  
  <newpage>  
  (text of page 2)  
  <input type="nextpage" value="Goto Page 3">  
  <input type="firstpage" value="Back to Page 1">  
  <newpage>  
  (text of page 3)  
</form>
```

### *Condition*

The condition tag specifies that the text body enclosed by the beginning and ending tags is displayed only if a certain set of predicates are met. Name and value attribute pairs are matched against input variables from previous pages. Logical operators such as AND and OR are permitted. The NOT operator can be used in addition to AND or OR to form the NAND or NOR logic functions as well. Condition tags cannot be nested. Therefore, multilevel logic must be implemented using internal variables. Internal variables can be specified using hidden input variables of which the value is set in the script. Predicates

of a condition tag in a particular page are tested the moment when the user clicks a button to move into the page.

```
<condition and name1="color" value1="red"
              name2="shape" value2="round">
<input type="hidden" name="object" value="red-circle">
</condition>
```

```
<condition and name1="color" value1="blue"
              name2="shape" value2="square">
<input type="hidden" name="object" value="blue-square">
</condition>
```

```
<condition or name1="object" value1="red-circle"
              name2="object" value2="blue-square">
You have selected either a red circle or a blue square.
</condition>
```

```
<condition not and name1="object" value1="red-circle"
                  name2="object" value2="blue-square">
You did not select both the red circle and the blue square.
</condition>
```

In this example, the first two condition tags each encloses between its beginning and ending tags a hidden input tag. This tag defines the value of an internal variable which in turn is used as a predicate in the last two condition tags. The name and value attributes are matched such that the input variable specified by *name1* is matched against the input value specified by *value1*, and so on. Name attributes can be specified from *name0* through *name9*, and value attributes from *value0* through *value9*. Either the AND or the OR operators, but not both, can be specified. By default, the logic chosen is AND. The NOT operator is optional. Conditional tags require both starting and ending tags. A condition tag with no attributes at all always hides the enclosed text body.

## ***Installation Instructions***

The Mosaic source code may be loaded directly from NCSA by ftp at the site  
ftp.ncsa.uiuc.edu  
under the Mosaic directory.

The Mosaic directory contains several Makefiles for specific platforms, as well as a number of subdirectories. One of the subdirectories is called “libhtmlw”, in which there are a number of files with the name beginning with the letters “HTML”. Several of these files, namely HTML.h, HTMLP.h, HTML.c, HTMLformat.c, HTMLlists.c, HTMLparse.c and HTMLwidgets.c, have been modified. All modified parts of the source code are marked by the comment line “/\*\*\* NEW\_EXTENSION \*\*\*/” at the beginning and the line “/\*\*\* END OF NEW\_EXTENSION \*\*\*/”, with the nature of the change clearly commented and explained. None of the other files from this or any other subdirectories in Mosaic have been modified.

To install the extended version of Mosaic, download all files from the Mosaic directory and replace all the files from the libhtmlw subdirectory with all the files from the developmental directory. Choose a Makefile that matches the relevant platform and compile the code using that Makefile. The resulting executable Mosaic, which can be found within the subdirectory “src”, will support the new extended features.

## ***Source Code Extension***

All modified parts of the original source code within the directory “libhtmlw” are shown on the following pages.

In HTML.h:

```
/** NEW_EXTENSION : logic types */
#define OR      0
#define AND    1
/** END OF NEW_EXTENSION */
```

In HTMLP.h:

(within typedef struct \_HTMLPart)

```
/** NEW_EXTENSION */
struct mark_up      *html_next_object;
int                 newpage_flag;
int                 nextpage_reformat;
/** END OF NEW_EXTENSION */
```

In HTML.c:

```
/** NEW_EXTENSION : Force vertical scrollbar */
/** We want to avoid calling FormatAll more than once per event */
/* if (temp > hw->core.height - sheight) */
if (TRUE)
{
hw->html.use_vbar = True;
if (hw->html.vbar_right == True)
{
ItMoveWidget(hw->html.vbar,
(hw->core.width - swidth), 0);
}
else
{
ItMoveWidget(hw->html.vbar, 0, 0);
}
ItManageChild(hw->html.vbar);
hw->html.view_width = hw->core.width - swidth - (2 * st);
}
/** END OF NEW_EXTENSION */
```

(within Initialize)

```
/** NEW_EXTENSION */
new->html.html_next_object = new->html.html_objects;
/** END OF NEW_EXTENSION */
```

(within SetValues)

```
/** NEW_EXTENSION */
new->html.html_next_object = new->html.html_objects;
/** END OF NEW_EXTENSION */
```

```

(within HTMLSetText)
/** NEW_EXTENSION ***/
hw->html.html_next_object = hw->html.html_objects;
/** END OF NEW_EXTENSION ***/

```

In HTMLformat.c:

```

/** NEW_EXTENSION ***/
extern int CollectSubmitInfo();  /** needed for conditions ***/
/** END OF NEW_EXTENSION ***/

static int HideText;  /** NEW_EXTENSION ***/

(within TriggerMarkChanges)
/** NEW_EXTENSION ***/
/** if HideText then ignore until we have reached end of CONDITION tag ***/
if ((HideText)&&(type != M_CONDITION))
{
    if (!mark->is_end)
    {
if ((type == M_INPUT)|| (type == M_SELECT)|| (type == M_TEXTAREA))
        WidgetId++;
else if (type == M_INDEX)
        WidgetId = WidgetId + 2;
    }
return;
}
/** END OF NEW_EXTENSION ***/

/** NEW_EXTENSION ***/
/** To handle a NEWPAGE, set the html_next_object pointer to the next
*** object in the list so that next time we know where to start a page
***/
case M_NEWPAGE:
    ConditionalLineFeed(hw, x, y, 1);
    if (mptr->next != NULL)
        hw->html.html_next_object = mptr->next;
    hw->html.newpage_flag = 1;  /** set the flag **/
    temp = mptr->next;
    break;
/** END OF NEW_EXTENSION ***/

/** NEW_EXTENSION ***/
/** For a CONDITION tag, check to see if the predicate is satisfied
*** If so, then HideText flag is 0 (default)
*** If not, then set HideText flag to 1 so that text is not shown
***/
case M_CONDITION:
    if (mark->is_end)
    {

```

```

    HideText = 0;
}
else
{
    char **name_list;
    char **value_list;
    int cnt, i;

    char *name0, *name1, *name2, *name3, *name4;
    char *name5, *name6, *name7, *name8, *name9;
    char *value0, *value1, *value2, *value3, *value4;
    char *value5, *value6, *value7, *value8, *value9;
    int logic, negate, cond_cnt, match_cnt;

    name0 = ParseMarkTag(mptr->start, MT_CONDITION, "name0");
    name1 = ParseMarkTag(mptr->start, MT_CONDITION, "name1");
    name2 = ParseMarkTag(mptr->start, MT_CONDITION, "name2");
    name3 = ParseMarkTag(mptr->start, MT_CONDITION, "name3");
    name4 = ParseMarkTag(mptr->start, MT_CONDITION, "name4");
    name5 = ParseMarkTag(mptr->start, MT_CONDITION, "name5");
    name6 = ParseMarkTag(mptr->start, MT_CONDITION, "name6");
    name7 = ParseMarkTag(mptr->start, MT_CONDITION, "name7");
    name8 = ParseMarkTag(mptr->start, MT_CONDITION, "name8");
    name9 = ParseMarkTag(mptr->start, MT_CONDITION, "name9");

    value0 = ParseMarkTag(mptr->start, MT_CONDITION, "value0");
    value1 = ParseMarkTag(mptr->start, MT_CONDITION, "value1");
    value2 = ParseMarkTag(mptr->start, MT_CONDITION, "value2");
    value3 = ParseMarkTag(mptr->start, MT_CONDITION, "value3");
    value4 = ParseMarkTag(mptr->start, MT_CONDITION, "value4");
    value5 = ParseMarkTag(mptr->start, MT_CONDITION, "value5");
    value6 = ParseMarkTag(mptr->start, MT_CONDITION, "value6");
    value7 = ParseMarkTag(mptr->start, MT_CONDITION, "value7");
    value8 = ParseMarkTag(mptr->start, MT_CONDITION, "value8");
    value9 = ParseMarkTag(mptr->start, MT_CONDITION, "value9");

    if (ParseMarkTag(mptr->start, MT_CONDITION, "or"))
    logic = OR;
    else
    logic = AND;    /** default logic is AND **/
    if (ParseMarkTag(mptr->start, MT_CONDITION, "not"))
    negate = TRUE;
    else
    negate = FALSE;    /** negate only if not tag exists **/

    if (CurrentForm != NULL)
    {
        name_list = NULL;
        value_list = NULL;
        cnt = CollectSubmitInfo(CurrentForm, &name_list, &value_list);
        cond_cnt = 0;
    }
}

```

```

match_cnt = 0;

if ((name0)&&(value0))
{
    cond_cnt++;
    for (i=0; i<cnt; i++)
{
    if ((name_list[i])&&(value_list[i])&&
        (strcmp(name0, name_list[i])==0)&&
        (strcmp(value0, value_list[i])==0))
        match_cnt++;
    }
}
if ((name1)&&(value1))
{
    cond_cnt++;
    for (i=0; i<cnt; i++)
{
    if ((name_list[i])&&(value_list[i])&&
        (strcmp(name1, name_list[i])==0)&&
        (strcmp(value1, value_list[i])==0))
        match_cnt++;
    }
}
if ((name2)&&(value2))
{
    cond_cnt++;
    for (i=0; i<cnt; i++)
{
    if ((name_list[i])&&(value_list[i])&&
        (strcmp(name2, name_list[i])==0)&&
        (strcmp(value2, value_list[i])==0))
        match_cnt++;
    }
}
if ((name3)&&(value3))
{
    cond_cnt++;
    for (i=0; i<cnt; i++)
{
    if ((name_list[i])&&(value_list[i])&&
        (strcmp(name3, name_list[i])==0)&&
        (strcmp(value3, value_list[i])==0))
        match_cnt++;
    }
}
if ((name4)&&(value4))
{
    cond_cnt++;
    for (i=0; i<cnt; i++)
{

```

```

if ((name_list[i])&&(value_list[i])&&
    (strcmp(name4, name_list[i])==0)&&
    (strcmp(value4, value_list[i])==0))
    match_cnt++;
}
}
if ((name5)&&(value5))
{
    cond_cnt++;
    for (i=0; i<cnt; i++)
{
    if ((name_list[i])&&(value_list[i])&&
        (strcmp(name5, name_list[i])==0)&&
        (strcmp(value5, value_list[i])==0))
        match_cnt++;
}
}
if ((name6)&&(value6))
{
    cond_cnt++;
    for (i=0; i<cnt; i++)
{
    if ((name_list[i])&&(value_list[i])&&
        (strcmp(name6, name_list[i])==0)&&
        (strcmp(value6, value_list[i])==0))
        match_cnt++;
}
}
if ((name7)&&(value7))
{
    cond_cnt++;
    for (i=0; i<cnt; i++)
{
    if ((name_list[i])&&(value_list[i])&&
        (strcmp(name7, name_list[i])==0)&&
        (strcmp(value7, value_list[i])==0))
        match_cnt++;
}
}
if ((name8)&&(value8))
{
    cond_cnt++;
    for (i=0; i<cnt; i++)
{
    if ((name_list[i])&&(value_list[i])&&
        (strcmp(name8, name_list[i])==0)&&
        (strcmp(value8, value_list[i])==0))
        match_cnt++;
}
}
if ((name9)&&(value9))

```



```

    {
        cond_cnt++;
        for (i=0; i<cnt; i++)
    {
        if ((name_list[i])&&(value_list[i])&&
            (strcmp(name9, name_list[i])==0)&&
            (strcmp(value9, value_list[i])==0))
            match_cnt++;
        }
    }
    if ((logic == AND)&&(cond_cnt > 0)&&
        (cond_cnt == match_cnt))
HideText = 0;
    else if ((logic == OR)&&(cond_cnt > 0)&&
            (match_cnt > 0))
HideText = 0;
    else
HideText = 1;
    if ((negate)&&(cond_cnt > 0))
    {
        if (HideText == 0)
            HideText = 1;
        else
            HideText = 0;
    }
    }
    break;
/** END OF NEW EXTENSION ***/

(within FormatChunk)
/** NEW_EXTENSION ***/
/** Checks to see if we are in the first page. If so, then
*** set mptr to html.html_objects as before. Otherwise,
*** set mptr to html.next_object which is the beginning of a
*** new page.
***/
if ((hw->html.html_next_object != hw->html.html_objects)&&
    (hw->html.nextpage_reformat == 1))
    {
        mptr = hw->html.html_next_object;
    }
else
    {
        mptr = hw->html.html_objects;
    }
/** END OF NEW EXTENSION ***/

/** NEW_EXTENSION **/
while ((mptr != NULL)&&(hw->html.newpage_flag == 0))
{

```

```

TriggerMarkChanges(hw, mptr, x, y);
/*
 * Save last non-text mark
 */
if (mptr->type != M_NONE)
{
Last = mptr;
}
mptr = mptr->next;
}
hw->html.newpage_flag = 0;    /** reset the flag **/
}
/** END OF NEW EXTENSION **/

(within FormatAll)
/** NEW_EXTENSION **/
/** Clear WidgetId only if this is not a new page **/
if (!hw->html.nextpage_reformat)
    WidgetId = 0;
HideText = 0;    /** initialize HideText flag **/
/** END OF NEW EXTENSION **/

/** NEW_EXTENSION: Do not set Form to NULL unless there is </form> tag **/
/** CurrentForm = NULL;    ***/
/** END OF NEW EXTENSION **/

```

In HTMLparse.c:

```

(within HTMLParse)
/** NEW_EXTENSION **/
else if ((mark != NULL)&&    /** hidden text **/
        (mark->type == M_HIDDEN)&&
        (!mark->is_end))
{
    text = get_text(start, &end);
    strcpy (text, "");
}
/** END OF NEW EXTENSION **/

(within ParseMarkType)
/** NEW_EXTENSION **/
else if (caseless_equal(str, MT_HIDDEN))
{
type = M_HIDDEN;
}
else if (caseless_equal(str, MT_NEWPAGE))
{
type = M_NEWPAGE;
}
else if (caseless_equal(str, MT_CONDITION))

```

```

{
type = M_CONDITION;
}
/** END OF NEW EXTENSION ***/

```

In HTMLwidgets.c:

```

/** NEW_EXTENSION ***/
extern void HideWidgets();
extern void ReformatWindow();
extern void ViewClearAndRefresh();
extern void HTMLGoToId();
/** END OF NEW EXTENSION ***/

```

```

/** NEW_EXTENSION ***/
/** Callback function for the Next Page button ***/
void
CBNextPage(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
    FormInfo *fptr = (FormInfo *)client_data;
    HTMLWidget hw = (HTMLWidget)(fptr->hw);
    WbFormCallbackData cbdata;
    int val, size, inc, pageinc;

```

```

#ifdef MOTIF
    ImPushButtonCallbackStruct *pb =
        (ImPushButtonCallbackStruct *)call_data;
#endif /* MOTIF */
#ifdef MOTIF
    cbdata.event = pb->event;
#else
    /** WE HAVE NO EVENT in ATHENA ***/
    cbdata.event = NULL;
#endif /* MOTIF */
    hw->html.nextpage_reformat = 1;    /** Set flag **/
    HideWidgets(hw);
    ReformatWindow(hw);
    ViewClearAndRefresh(hw);
#ifdef MOTIF
    ImScrollBarGetValues(hw->html.vbar, &val, &size, &inc, &pageinc);
    ImScrollBarSetValues(hw->html.vbar, 0, size, inc, pageinc, True);
    ImScrollBarGetValues(hw->html.hbar, &val, &size, &inc, &pageinc);
    ImScrollBarSetValues(hw->html.hbar, 0, size, inc, pageinc, True);
#else
    ScrollToPos(hw->html.vbar, hw, newy);
    ScrollToPos(hw->html.hbar, hw, 0);
    setScrollBar(hw->html.vbar, newy,

```

```

        hw->html.doc_height,
        hw->html.view_height);
#endif
hw->html.nextpage_reformat = 0;  /** Reset the flag **/
}

CBFirstPage(w, client_data, call_data)
Widget w;
caddr_t client_data;
caddr_t call_data;
{
    FormInfo *fptr = (FormInfo *)client_data;
    HTMLWidget hw = (HTMLWidget)(fptr->hw);
    WbFormCallbackData cbdata;
    int val, size, inc, pageinc;

#ifdef MOTIF
    ImPushButtonCallbackStruct *pb =
        (ImPushButtonCallbackStruct *)call_data;
#endif /* MOTIF */
#ifdef MOTIF
    cbdata.event = pb->event;
#else
    /***** WE HAVE NO EVENT in ATHENA *****/
    cbdata.event = NULL;
#endif /* MOTIF */
    HideWidgets(hw);
    ReformatWindow(hw);
    ViewClearAndRefresh(hw);
#ifdef MOTIF
    ImScrollBarGetValues(hw->html.vbar, &val, &size, &inc, &pageinc);
    ImScrollBarSetValues(hw->html.vbar, 0, size, inc, pageinc, True);
    ImScrollBarGetValues(hw->html.hbar, &val, &size, &inc, &pageinc);
    ImScrollBarSetValues(hw->html.hbar, 0, size, inc, pageinc, True);
#else
    ScrollToPos(hw->html.vbar, hw, newy);
    ScrollToPos(hw->html.hbar, hw, 0);
    setScrollBar(hw->html.vbar, newy,
        hw->html.doc_height,
        hw->html.view_height);
#endif
}
/** END OF NEW EXTENSION ***/

/** NEW_EXTENSION ***/
/** Next Page callback function ***/
void
PrepareNextPage(hw, w, fptr)
HTMLWidget hw;
Widget w;
FormInfo *fptr;

```

```

{
#ifdef MOTIF
    XtAddCallback(w, XmNactivateCallback,
(XtCallbackProc)CBNextPage, (caddr_t)fptr);
#else
    XtAddCallback(w, XtNcallback,
(XtCallbackProc)CBNextPage, (caddr_t)fptr);
#endif /* MOTIF */
}

/**/ First Page callback function ***/
void
PrepareFirstPage(hw, w, fptr)
HTMLWidget hw;
Widget w;
FormInfo *fptr;
{
#ifdef MOTIF
    XtAddCallback(w, XmNactivateCallback,
(XtCallbackProc)CBFirstPage, (caddr_t)fptr);
#else
    XtAddCallback(w, XtNcallback,
(XtCallbackProc)CBFirstPage, (caddr_t)fptr);
#endif /* MOTIF */
}
/**/ END OF NEW EXTENSION ***/

/**/ NEW_EXTENSION ***/
/**/ Next page input button ***/
else if ((type_str != NULL)&&(strcmp(type_str, "nextpage") == 0))
{
XmString label;
type = W_PUSHBUTTON;
label = NULL;
value = ParseMarkTag(text, MT_INPUT, "VALUE");
if ((value == NULL)||(*value == '\0'))
{
value = (char *)malloc(strlen("Next Page") + 1);
strcpy(value, "Next Page");
}
argcnt = 0;
XtSetArg(arg[argcnt], XmNx, x); argcnt++;
XtSetArg(arg[argcnt], XmNy, y); argcnt++;
if (value != NULL)
{
label = XmStringCreateSimple(value);
XtSetArg(arg[argcnt], XmNlabelString, label);
argcnt++;
}
w = XmCreatePushButton(hw->html.view, "Button",
arg, argcnt);

```

```

ItSetMappedWhenManaged(w, False);
ItManageChild(w);
if (label != NULL)
{
ImStringFree(label);
}
PrepareNextPage(hw, w, fptr);
}

/** First page input button */
else if ((type_str != NULL)&&(strcmp(type_str, "firstpage") == 0))
{
ImString label;
type = W_PUSHBUTTON;
label = NULL;
value = ParseMarkTag(text, MT_INPUT, "VALUE");
if ((value == NULL)||(*value == '\0'))
{
value = (char *)malloc(strlen("First Page") + 1);
strcpy(value, "First Page");
}
argcnt = 0;
ItSetArg(arg[argcnt], XmNx, x); argcnt++;
ItSetArg(arg[argcnt], XmNy, y); argcnt++;
if (value != NULL)
{
label = ImStringCreateSimple(value);
ItSetArg(arg[argcnt], XmNlabelString, label);
argcnt++;
}
w = ImCreatePushButton(hw->html.view, "Button",
arg, argcnt);
ItSetMappedWhenManaged(w, False);
ItManageChild(w);
if (label != NULL)
{
ImStringFree(label);
}
PrepareFirstPage(hw, w, fptr);
}
/** END OF NEW EXTENSION */

```

<title>Conditional Usage Survey Form</title>

<h1>Usage Survey</h1>

<hr>

Thank you for completing this usage survey.  
This survey is intended to measure access to and use of the electronic publication of White House documents. The survey results can guide improvements in this new information resource and provide insights into how electronic access to government might enhance the democratic process.

<p>

This is a real-time adaptive survey form. You will see a number of survey pages, each consisting of several questions. After completing all the questions on a page, you may proceed to the next page by pressing the CONTINUE button. Your response to previous survey pages will determine which page you will see next. At the bottom of the final page there is a SUBMIT button, which you can press in order to submit the entire survey form.

<p>

Thank you again for your time.

<hr>

<form action = "http://hoohoo.ncsa.uiuc.edu/htbin-post/post-query" method="POST">



<b>How do you currently access White House documents?</b><br>

<select name="online-source">

<option selected>

<option value="clinton-info">Email directly from Clinton-Info to my email address.

<option value="internet-mailing-list">Email from a mailing list to my email address on Internet.

<option value="non-internet-mailing-list">Email from a wide-area non-Internet mailing list.

<option value="local-distribution">Indirectly from Clinton-Info through a local distribution point.

<option value="usenet">From a newsgroup on Usenet/Netnews.

<option value="commercial-email">Through email on a commercial computer network.

<option value="commercial-forum">From a forum on a commercial information service, e.g., CompuServe.

<option value="ftp-sites">From online archives by FTP, Gopher, WAIS or World-Wide-Web.

<option value="bulletin-board">From a dial-up computer bulletin board.

<option value="phone-network">Email received over a telephone-based network, e.g., Fidonet.

<option value="hardcopy">In hardcopy from another person or an organization.

Once did but not currently receive these documents.  
Never accessed or received these documents.  
</select>  
<p>  
<hr>

<b>What routines do you follow for handling, saving and deleting the White House documents?</b><br>  
<select name="retention">  
<option selected>  
<option value="delete-immediately">Delete the documents after reading them.  
<option value="delete-without-reading">Delete the documents without reading them.  
<option value="scan-and-choose">Scan documents and save some for later reading.  
<option value="keep-interesting">Save the interesting documents.  
<option value="personal-library">Save in a private document collection.  
<option value="electronic-librarian">Put in a library's electronic document collection.  
<option value="hardcopy-librarian">Put hardcopy in a library's document collection.  
<option value="archivist">Make available for retrieval by others via a computer network.  
<option value="other">Other.  
<option value="not-applicable">I don't receive the documents.  
</select>  
<p>  
<hr>

<b>Are you acting for or affiliated with one of the following types of institutions when you handle the documents?</b><br>  
<select name="affiliation">  
<option selected>  
<option value="university">College or university.  
<option value="info-service">Computer-based, commercial information-service.  
<option value="school">Elementary, middle or high school.  
<option value="federal-government">Federal government.  
<option value="independent">Independent, i.e., no institutional affiliation.  
<option value="issue-advocacy">Issue-oriented advocacy group.  
<option value="media-sector-company">Media sector company.  
<option value="for-profit-company">Other for-profit company.  
<option value="political-or-civic">Political party or civic association.  
<option value="economic-interest-group">Professional, business or labor group.  
<option value="public-sector">Public sector organization.  
<option value="state-or-local-government">State or local government.  
<option value="not-applicable">Not applicable, I don't receive the documents.  
</select>  
<p>  
<hr>



**  
<b>On a scale of 1 to 5, how satisfied are you with the electronic  
publication of these documents?</b><br>**

**Not satisfied**

**<input type="radio" name="satisfaction" value="1">1  
<input type="radio" name="satisfaction" value="2">2  
<input type="radio" name="satisfaction" value="3">3  
<input type="radio" name="satisfaction" value="4">4  
<input type="radio" name="satisfaction" value="5">5 Very Satisfied**

**<p>  
<hr>**

**  
<b>Do you pay either to get the documents or to use the computer account  
through which you get the documents?</b><br>**

**<input type="radio" name="cost" value="yes">Yes  
<input type="radio" name="cost" value="no">No**

**<p>  
<hr>**

**  
<b>If you distribute the White House documents or their information,  
what means do you use?</b><br>**

**<select name="distribution">  
<option selected value="electronic">Distribution over a local or  
wide-area computer network.  
<option value="discussion">Face to face discussion with others.  
<option value="telephone">Telephone conversations.  
<option value="online-discussion">Mention or quote in online  
discussions, e.g. email, IRC.  
<option value="offline-writing">Mention or quote in offline writing.  
<option value="display">Hardcopy distributed in workplace, school, club  
or church.  
<option value="live-audience">Classroom or lectual hall presentation.  
<option value="mailing">Hardcopy mailed through the Postal System.  
<option value="newspaper">Items in a printed newspaper.  
<option value="tv-or-radio">Items on TV or radio programs.  
<option value="not-applicable">Not applicable.**

**</select>  
<p>  
<hr>**

**  
<b>Please estimate as best you can the number of people who routinely  
receive or access from you the White House documents or reports on their  
content.</b><br>**

**<select name="distribution-audience">  
<option selected value="0">No one.  
<option value="1-3">1 - 3 people.  
<option value="4-7">4 - 7 people.  
<option value="8-15">8 - 15 people.**

```

<option value="16-25">16 - 25 people.
<option value="26-100">26 - 100 people.
<option value="101-500">101 - 500 people.
<option value="501-1000">501 - 1000 people.
<option value="1001-10000">1001 - 10000 people.
<option value="over-10000">Over 10000 people.
</select>
<p>
<hr>
<b>Press here to continue to the next page... </b>
<input type="nextpage" value="CONTINUE">
<p>
<b>Press here to reset the survey form... </b>
<input type="reset" value="RESET">

<hr>
<newpage>

<condition or name1="distribution-audience" value1="0"
name2="distribution-audience" value2="1-3" name3="distribution-audience"
value3="4-7" name4="distribution-audience" value4="8-16"
name5="distribution-audience" value5="16-25">
<input type="hidden" name="personal-distribution" value="yes">
</condition>

<condition not name1="personal-distribution" value1="yes">
<input type="hidden" name="personal-distribution" value="no">
</condition>

<condition not and name1="online-source" value1="not-current"
name2="online-source" value2="never-received">
<input type="hidden" name="still-receiving" value="yes">
</condition>

<condition or name1="cost" value1="yes" name2="online-source"
value2="commercial-email" name3="online-source" value3="commercial-forum">
<input type="hidden" name="pay-for-the-privilege" value="yes">
</condition>

<condition not and name1="pay-for-the-privilege" value1="yes">
<input type="hidden" name="pay-for-the-privilege" value="no">
</condition>

<condition and name1="personal-distribution" value1="yes"
name2="still-receiving" value2="yes" name3="pay-for-the-privilege"
value3="no">
<input type="hidden" name="user-type" value="non-commercial-customer">
<h2> Non-commercial Customer Survey</h2>
</condition>

<condition and name1="personal-distribution" value1="yes"

```

```

name2="still-receiving" value2="yes" name3="pay-for-the-privilege"
value3="yes">
<input type="hidden" name="user-type" value="commercial-customer">
<h2> Commercial Customer Survey</h2>
</condition>

<condition and name1="online-source" value1="not-current">
<input type="hidden" name="user-type" value="former-customer">
<h2> Former Customer Survey</h2>
</condition>

<condition and name1="online-source" value1="never-received">
<input type="hidden" name="user-type" value="non-customer">
<h2> Non-Customer Survey</h2>
</condition>

<condition or name1="distribution" value1="electronic" name2="retention"
value2="electronic-librarian" name3="retention" value3="archivist">
<input type="hidden" name="archiving" value="yes">
</condition>

<condition or name1="affiliation" value1="media-sector-company"
name2="affiliation" value2="info-service">
<input type="hidden" name="affiliation-type" value="commercial">
</condition>

<condition or name1="affiliation" value1="school" name2="affiliation"
value2="university">
<input type="hidden" name="affiliation-type" value="educational">
</condition>

<condition or name1="affiliation" value1="economic-interest-group"
name2="affiliation" value2="issue-advocacy" name3="affiliation"
value3="political-or-civic">
<input type="hidden" name="affiliation-type" value="advocate">
</condition>

<condition or name1="affiliation" value1="federal-government"
name2="affiliation" value2="public-sector" name3="affiliation"
value3="state-or-local-government">
<input type="hidden" name="affiliation-type" value="governmental">
</condition>

<condition or name1="affiliation" value1="for-profit-company">
<input type="hidden" name="affiliation-type" value="company">
</condition>

<condition or name1="affiliation" value1="independent">
<input type="hidden" name="affiliation-type" value="independent">
</condition>

```

```
<condition and name1="affiliation-type" value1="commercial"
name2="archiving" value2="yes" name3="personal-distribution" value3="no"
name4="still-receiving" value4="yes">
<input type="hidden" name="user-type" value="commercial-distributor">
<h2> Commercial Distributor Survey</h2>
</condition>
```

```
<condition and name1="affiliation-type" value1="educational"
name2="archiving" value2="yes" name3="personal-distribution" value3="no"
name4="still-receiving" value4="yes">
<input type="hidden" name="user-type" value="educational-distributor">
<h2> Educational Distributor Survey</h2>
</condition>
```

```
<condition and name1="affiliation-type" value1="advocate"
name2="archiving" value2="yes" name3="personal-distribution" value3="no"
name4="still-receiving" value4="yes">
<input type="hidden" name="user-type" value="advocate-distributor">
<h2> Advocate Distributor Survey</h2>
</condition>
```

```
<condition and name1="affiliation-type" value1="governmental"
name2="archiving" value2="yes" name3="personal-distribution" value3="no"
name4="still-receiving" value4="yes">
<input type="hidden" name="user-type" value="governmental-distributor">
<h2> Governmental Distributor Survey</h2>
</condition>
```

```
<condition and name1="affiliation-type" value1="company"
name2="archiving" value2="yes" name3="personal-distribution"
value3="no" name4="still-receiving" value4="yes">
<input type="hidden" name="user-type" value="company-distributor">
<h2> Company Distributor Survey</h2>
</condition>
```

```
<condition and name1="affiliation-type" value1="independent"
name2="archiving" value2="yes" name3="personal-distribution" value3="no"
name4="still-receiving" value4="yes">
<input type="hidden" name="user-type" value="independent-distributor">
<h2> Independent Distributor Survey</h2>
</condition>
```

```
<condition or name1="distribution" value1="display" name2="distribution"
value2="discussion" name3="distribution" value3="offline-writing"
name4="distribution" value4="mailing" name5="distribution"
value5="newspaper" name6="distribution" value6="telephone"
name7="distribution" value7="tv-or-radio" name8="retention"
value8="hardcopy-librarian">
<input type="hidden" name="non-electronic-media1" value="yes">
</condition>
```

```
<condition not or name1="distribution" value1="electronic"
name2="archiving" value="yes">
<input type="hidden" name="non-electronic-media2" value="yes">
</condition>
```

```
<condition and name1="personal-distribution" value1="no"
name2="non-electronic-media1" value2="yes" name3="non-electronic-media2"
value3="yes" name4="still-receiving" value4="yes">
<input type="hidden" name="user-type" value="non-electronic-distributor">
<h2> Non-electronic Distributor Survey</h2>
</condition>
```

```
<hr>
```

```
<condition or name1="user-type" value1="non-commercial-customer"
name2="user-type" value2="commercial-customer" name3="user-type"
value3="former-customer" name4="user-type" value4="non-customer">
<input type="hidden" name="user-group" value="customer">
</condition>
```

```
<condition not name1="user-group" value1="customer">
<input type="hidden" name="user-group" value="distributor">
</condition>
```

```
<condition name1="user-group" value1="customer">
```

```

<b>From what source did you discover how to access White House documents
via a computer network?</b><br>
<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>
```

```

<b>How many hours per week on average do you spend viewing, listening
to, or reading offline news or other material on politics?</b><br>
<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>
```

```

<b>How politically active are you?</b><br>
<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>
```

</condition>

```
<condition name1="user-type" value1="non-commercial-customer">

<b>Which computer site did you use to retrieve White House documents?</b><br>
Please indicate the host name, e.g. clinton.ai.mit.edu :<br>
<input name="" size=30>
<hr>
</condition>
```

```
<condition name1="user-type" value1="commercial-customer">
```

```

<b>Please type the name of the commercial network or information service
through which you get the documents: </b><br>
<input name="" size=30>
<hr>
```

```

<b>Approximately how much do you pay in the average month to receive or
access the White House documents?</b><br>
<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>
```

</condition>

```
<condition name1="user-group" value1="distributor">
```

```

<b>By what means do you distribute or provide access to the White House
documents?</b><br>
<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>
```

```

<b>Do you distribute all the electronic documents released by the White
House each day, or only more specific documents?</b><br>
<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>
```

```

<b>How many people per day, on average, receive or access the White
House documents directly from your service?</b><br>
<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>

</condition>
```

```
<condition name1="user-type" value1="commercial-distributor">

<b>How are customers charged for accessing or receiving daily White
House documents?</b><br>
<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>
</condition>
```

```
<condition or name1="user-type" value1="advocate-distributor"
name2="user-type" value2="non-electronic-distributor">

<b>What percent of the people whom you would like to reach are already
on-line and able to receive the information you distribute?</b><br>
<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>
</condition>
```

```
<condition name1="user-type" value1="educational-distributor">

<b>Are the White House documents you distribute used as texts or sources
in classroom discussions?</b><br>
<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>
</condition>
```

```
<condition name1="user-type" value1="governmental-distributor">

<b>Please characterize the audience to which you make available or
distribute the documents.</b><br>
```

```

<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>
</condition>

<condition or name1="user-type" value1="company-distributor"
name2="user-type" value2="independent-distributor" name3="user-type"
value3="non-electronic-distributor">

<b>Why are your recipients interested in getting the documents?</b><br>
<select>
<option selected>
<option>Hi there! This is just a test!
</select>
<p><hr>
</condition>

<b>Press here to submit the survey form... </b>
<input type="submit" value="Submit">
<p>
<b>Press here to go back to the first page... </b>
<input type="firstpage" value="Go Back">
</form>

```



## References

- [barfield91] Barfield, Woodrow. Lim, Rafael. Evaluation of Computer Graphics Techniques for the Design of Images for Human-Computer Interaction. *Human Aspects in Computing*. Elsevier, New York, 1991.
- [berners-lee93a] Berners-Lee, Tim. World Wide Web Seminar. *WWW Hypertext Document*. 1993.
- [berners-lee93b] Berners-Lee, Tim. Graphical Composition Languages and Virtual Reality. *WWW Hypertext Document*. 1993.
- [chabay92] Chabay, Ruth W. Sherwood, Bruce A. A Practical Guide for the Creation of Educational Software. *Computer-Assisted Instruction and Intelligent Tutoring Systems*. Lawrence Erlbaum Associates, New Jersey, 1992.
- [chatty92] Chatty, Stephane. Defining the Dynamic Behavior of Animated Interfaces. *Engineering for Human-Computer Interaction. IFIP Transactions A-18*. North Holland, Amsterdam, 1992.
- [dix92] Dix, Alan. Beyond the Interface. *Engineering for Human-Computer Interaction. IFIP Transactions A-18*. North Holland, Amsterdam, 1992.
- [felix91] Felix, D. Graf, W. Krueger, H. User Interfaces for Public Information Systems. *Human Aspects in Computing*. Elsevier, New York, 1991.
- [heinich93] Heinich, Robert. Molenda, Michael. Russell, James D. *Instructional Media and the New Technologies of Instruction*. Macmillan, New York, 1993.
- [hoppe92] Hoppe, H.Ulrich. Towards Task Models for Embedded Information Retrieval. *Human Factors in Computing Systems – CHI '92 Conference Proceedings*. ACM Press, Reading, 1992.
- [howes90] Howes, A. Payne, S.J. Display-based competence: towards user models for menu-driven interfaces. *International Journal of Man-Machine Studies, Volume 33*.
- [hutchins86] Hutchins, E.L. Hollan, J.D. Norman, D.A. Direct Manipulation Interfaces. *User Centered System Design: New Perspectives on Human-Computer Interaction*. Lawrence Erlbaum, New Jersey, 1986.
- [kaster91] Kaster, Annette. User Interface Design and Evaluation. *Human Aspects in Computing*. Elsevier, New York, 1991.
- [kwasnik92] Kwasnik, Barbara H. A Descriptive Study of the Functional Components of Browsing. *Engineering for Human-Computer Interaction. IFIP Transactions A-18*. North Holland, Amsterdam, 1992.
- [ncsa93a] National Center for Supercomputer Applications. Mosaic for X version 2.0 Fill-Out Form Support. *WWW Hypertext Document*. 1993.

- [ncsa93b] National Center for Supercomputer Applications. *A Beginner's Guide to HTML. WWW Hypertext Document*. 1993.
- [raggett93] Raggett, David. *HTML+ Document Format -- Internet Draft. WWW Hypertext Document*. 1993.
- [sengupta91] Sengupta, Kishore. Te'eni, Dov. *Direct Manipulation and Command Language Interfaces: a Comparison of Users' Mental Models. Human Aspects in Computing*. Elsevier, New York, 1991.
- [shneiderman89] Shneiderman, Ben. Kearsley, Greg. *Hypertext Hands-On -- An Introduction to a New Way of Organizing and Accessing Information*. Addison-Wesley, Reading, 1989.
- [springett92] Springett, M.V. *The Utility of User Action Models for Direct Manipulation Design. Engineering for Human-Computer Interaction. IFIP Transactions A-18*. North Holland, Amsterdam, 1992.
- [ukelson92] Ukelson, Jacob P. Gould, John D. Boies, Stephen J. *User Navigation in Computer Applications. Engineering for Human-Computer Interaction. IFIP Transactions A-18*. North Holland, Amsterdam, 1992.
- [wastell91] Wastell, David. *Physiological measurement of cognitive load during interaction with process control displays. Human Aspects in Computing*. Elsevier, New York, 1991.
- [whiteside88] Whiteside, John. Wixon, Dennis. Jones, Sandy. *User Performance with Command, Menu, and Iconic Interfaces. Advances in Human-Computer Interaction, Volume 2*. Albex, Norwood, 1988.
- [woolf91] Woolf, Beverly. *Representing, Acquiring, and Reasoning about Tutoring Knowledge. Advanced Research on Computers in Education*. Elsevier, Amsterdam, 1991.