Development and investigation of a non-visual WWW browser.

Tarek. El-Haddad
University of Windsor

Development and Investigation of a Non-Visual WWW Browser

By

Tarek El-Haddad

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science
in Partial Fulfillment of the Requirements
for the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada
1997

0-612-30940-1

Abstract

The World-Wide-Web (WWW), is a massive collection of information dispersed over the Internet, that is primarily accessed using a graphical web browser such as Netscape or Microsoft Explorer. However for many people and for many applications a graphical web browser is useless and/or inappropriate.

In this thesis a prototype of a Non-Visual Web Browser was developed. this developed browser uses speech recognition as input and a speech synthesizer as output and its navigation method is based on document content rather than its layout. An investigation of the developed browser was carried out and the findings from this investigation were discussed. Other features such as auditory clues and spatial layout, were also discussed as factors in the development of an ideal non-visual web browser.

To My Parents

Acknowledgments

I would like to express my deepest thanks. appreciation and respect to my supervisor Dr. Richard Frost. His guidance and enthusiasm made this thesis a very enjoyable endeavor for me. Many thanks to Dr. Tjandra and Dr. Mogyorody for their valuable input and constructive comments of this thesis work. Also I would like to thank Dr. Joan Morrissey for her continuous support throughout my university life and for chairing my defense presentation.

I wish to thank Mr. Walid Mnaymneh for always being there when technical help is needed. Also many thanks to the wonderful secretaries of the School of Computer Science Mary. Margaret. and Josette for their help.

Last but not least I would like to thank the graduate students in the School of Computer Science especially Robert Yang. Zeina Al-Janabi. Mazen Manfoukh and Halima El-Katib.

TABLE OF CONTENTS

List of Figures

1 Introduction

The World Wide Web consists of an enormous amount of interconnected data, images, and other sources of knowledge that are held at millions of sites connected through the Internet. This vast collection of knowledge is accessible through web "browsers" which enable users to retrieve and display data from remote sites and to follow hypertext links from one document to other documents possibly located at different sites.

The majority of web browsers are based on a graphics interface. A mouse is used to move a cursor over a graphics display of a web page. Hypertext links are activated through mouse button clicks when the cursor is located over the representation of the hypertext link. The representation can be text or a graphical image.

The majority of the web browsers are of no value to users who cannot see the graphical representation of the web page. Such users require "non-visual" web browsers. A number of non-visual browsers are available. Nearly all are based on screen-reader technology. Users still navigate the page using the keyboard. However, instead of seeing the web page, screen-reader software reads out portions of the page that are related to the position of the cursor. The original layout of the web page is preserved and is accessible to the user through the relationship between the cursor position and the output from the screen-reader software.

Non-visual browsers that are based on the screen-reader paradigm are of value to visually challenged users but they do have a number of shortcomings. In particular, users have to invest a considerable amount of effort in gaining a feel for the physical layout of the page before accessing the content of the document. In addition. browsing via screen-reader output can be tedious if long paragraphs are read in entirety.

An ideal non-visual browser would provide a number of features to facilitate access to the content of web pages:

1. "Non-prompted" auditory clues to the document content. For example. when a page is being loaded into the browser. auditory output could indicate how "busy" the page is in terms of the density of hypertext links. Other auditory output could indicate when a list is being loaded, etc. The effect would be to give the user a feel for the content of the page before browsing or navigation commands are issued.

2. Access to some spatial "layout" of the page. This layout could be:

    a) The physical layout defined by the original hypertext markup tags (i.e. the layout that would be displayed by a conventional browser). This layout could be accessed through screen-reader technology as discussed above.

Or

    b) A "virtual" spatial layout that is generated from the content of the web page. For example, a generic virtual layout could be used for all pages, in which all hypertext links are collected in a list that is placed at, say, the top right-hand corner of a virtual spatial layout of the page.

In eithér case. the effect would be to provide the user with a spatial context in which to browse the page. The advantage of the first approach is that it would preserve the original intended spatial layout of the page. The advantage of the second approach is that the spatial layout of all pages would be the same. thereby simplifying navigation "around" the page.

3. A "content-based" browsing facility. In addition to providing users with methods to navigate around a page according to layout. an ideal non-visual browser would also allow users to navigate according to the semantic content of the document. For example. users would be able to ask for the title of a graphics image, for a list of technical words in the page, for the number of hypertext links. etc.

4. An ideal non-visual browser should accept natural language spoken commands and should respond with synthesized voice output. The browser should support a robust user-friendly dialogue that can confirm user commands and respond in a timely manner.

No browser has yet been built that contains all of the features above, and very little work has been done to investigate the relationship between the features and their relative value in creating a user-friendly robust non-visual browser. In this report, we describe the design and implementation of a system NVWB that will contribute to the longer-term goal of creating an ideal non-visual browser. NVWB is a "content-based" voice-in/voice-out browser and is the first such browser to be built anywhere. NVWB is not an ideal non-visual browser. It only supports content-based browsing and provides neither non-prompted auditory clues nor spatial-layout browsing. The objective of the work is to create a content-based browser that can be used by other researchers in the future to investigate the advantages and shortcomings of content-based browsing and its relationship to non-prompted auditory clues and representation of spatial layout.

NVWB is a complex package of software consisting of a number of components:

a) A network-interface module that allows users to connect to the Internet and to download web pages.

b) An HTML-document parser. which accepts web pages as input. processes them to extract useful information. and converts them into an appropriate form for command-driven browsing.

c) A speech-recognition module that accepts user-independent continuous speech commands and which provides a dialogue with the user confirming commends and activating the browser.

d) A content-based browser that accepts commands from the speech-recognizer and, processes those commands according to the web-page contents.

e) An interface module that connects the above components together and which directs output to the speech-synthesizer.

Various design decisions were made during the construction of NVWB:

a) Some decisions involved choice of data structures and algorithms necessary to provide "real-time" response to browser commends.

b) Some decisions involved the definition of an appropriate voice input command language and finding the right trade-off between recognition accuracy and expressiblity.

c) Some decisions involved the development of an appropriate input/output dialogue and finding the right trade-off between robustness (e.g. confirming all user input before acting on it) and naturalness.

d) Some decisions involved the extent to which the incoming web pages could be assumed to conform to guidelines that facilitate non-visual web browsing. Various bodies have proposed several guidelines, and an assumption of their use can simplify the construction of a non-visual browser.

Although it was beyond the scope of this thesis work to undertake a thorough investigation of the value of content-based non-visual browsing, it was possible to collect some data from a number of trial sessions with NVWB involving several users. A brief analysis of this data was carried out and an attempt was made to relate it to work that is being carried out elsewhere on non-visual WWW browsing.

2 EXISTING NON-VISUAL BROWSERS

Before describing the non-visual web browser. NVWB. that we have built. we describe some existing non-visual browsers.

2.1 Non-visual WWW browsers based on layout

Non-visual browsers that navigate through a document according to its visual layout typically consist of a simple interface to an ASCII text-based browser (such as LYNX. W3 or Cern Line Mode Browser)[2] that sends the ASCII text in the web browser to an output device. This output device can be a Braille output device (Braille display), which converts text that is displayed on the screen into Braille characters on a touch-pad. The user then feels what is on the screen.

Another mode of output can be through text-to-speech synthesisers. These devices. working together with screen access software packages (screen readers). appear to be the more common and economical approach.

Even though screen-reading software outputs the information. it is up to the user to interpret command menus, text, and status displays, which are either spoken with a synthesised voice or are displayed on the Braille display.

2.1.1 Screen Readers

Screen readers, or speech-access software packages, are used with speech-synthesis hardware to convert standard computers into talking computers. The speech-synthesis hardware makes the sounds, and the screen-reading software drives the hardware and enables the user to use the synthesised speech with the text-based screen or in our case with a text-based WWW browser. The whole purpose of the screen-reading software is to enable the visually challenged user to operate the WWW browser independently without any assistance from a sighted user. This is a significant challenge for both the technology and the user alike.

2.1.2 Braille Displays

Braille is a technique for enabling visually challenged people to read and write. Braille characters are simply a group of raised dots in a 2-column by 3-row array, allowing for 64 different character combinations. The U.S. Braille standard is 6 dots, however computer access created the need for 2 extra dots which are used for special characters and to indicate the cursor.

Refreshable Braille displays are electronic devices that connect to the computer by a serial cable, along with this device comes software that will produce Braille output on the display, corresponding to the text displayed on the monitor. The refreshable Braille display is located at the lower edge of the standard computer keyboard, these displays usually come in 20, 40 or 80 Braille cells. Refreshable Braille displays lift small retractable pins as needed to form Braille characters with one line being displayed at a time, after a line is read, the user can "refresh" the display in order to read additional lines. Refreshable Braille displays typically read one line of text at a time, these displays generally include directional keys which allow the user to navigate through a document.

Braille displays work very much similar to screen readers, but instead of the text being read out it is displayed as Braille characters. It also is an appropriate device for users who are deaf as well as visually challenged.

2.2 Non-visual WWW browsers based on content

Non-visual browsers that navigate through a document according to its documents content are more complex than those based on visual layout. The browser retrieves the HTML text of the document and allows the user to navigate the document according to the content of paragraphs, sentences, lists etc., rather than with scrolling and interpreting a structured screen display. Since the browser retrieves the HTML code, it is able to directly interpret the content of the document by interpreting the HTML tags in it. For example the browser can determine when a table starts or ends by searching for the *<TABLE>* and *</TABLE>* tags.

Only one such browser exists commercially today, *PwWebSpeak*[4]. This browser is designed specifically to interact with the information on Web pages with keyboard input accepted from the user. PwWebSpeak recognises various HTML constructs and automatically bypasses those constructs that have no relation to the information content of the document. Output from PwWebSpeak is translated into speech output. It works directly with three different synthesisers: SoftVoice, InfoVox 220 and Creative Labs Text Assist. PwWebSpeak also supports a wide range of synthesisers through the Arkenstone SSIL interface. The user controls PwWebSpeak through a combination of keystroke input, with each key representing a different function.

3 GUIDELINES FOR THE CONSTRUCTION OF WEB PAGES AND WEB BROWSERS

3.1 Existing guidelines for the construction of web pages

The structured nature of HTML provides tremendous power and flexibility in presenting information in multiple formats (text, audio, video, graphic, etc.). However, the features that make the WWW so popular and powerful present potential barriers for users such as those who are visually challenged. One solution to this problem would be to carefully design and code information in a way that would alleviate the access barriers. Various HTML design guidelines that are being developed by a number of groups are intended to make the web more accessible to all kinds of users. However, most of these guidelines address problems that people who are using screen readers (or Braille displays) encounter in using the web. Hardly any guidelines for users of non-visual content browsers have been developed.

A document author could create a widely accessible document by creating one that only has text and hypertext links (no images, sounds, tables, frames, JavaScript, etc.). However, web page authors do not need to avoid all the aspects that make the web so powerful. Some of the problems that non-visual browsers face are discussed below along with some guidelines to solve them.

GIF and other Inline Graphics

Inline graphics in web documents cause an accessibility problem. A person using a non-visual browser would be informed of the inline graphics, but would not know what the image represents. This problem is particularly significant if the inline image is a text anchor.

Example:

The psychoanalytical theory developed by Sigmund Freud

<a href="/images/raw/f.gif">

<img src="/images/small/f.gif"> </a>

refers to the unconscious and the need to repress traumatic memories and taboo desires.

In this example, the graphics anchor conveys no information about the image. The user of a non-visual browser would not know what the image is. An easy solution to this problem would be to use the ALT attribute in an image-reference anchor and include selection text within the anchor.

Example of accessible code:

The psychoanalytical theory developed by

<a href="/images/raw/f.gif">

<img src="/images/small/f.gif" alt="picture of Sigmund Freud.">

Sigmund Freud </a>

refers to the unconscious and the need to repress traumatic memories and taboo desires.

This example adds both the "alt" image expression and a text description within the anchor making it more accessible by a user using a non-visual browser.

Text Layout

Text that is presented in a tabular fashion causes problems for people using screen reading software because text is read row by row rather than by column. Since screen reading software reads the words on the page to the user one line at a time, it would start from the left margin and read straight across to the right margin. If there are two columns of text, it will read the words in the left column and then continue to read the words in the right column.

Example:

Canada contains great reserves	***Most of Canada's inhabitants live in***
of natural resources, notably	***the southern part of the country, and***
timber, petroleum, natural gas,	***vast areas of the north are sparsely***
metallic minerals, and fish.	***inhabited.***

The above example would be read as:

Canada contains great reserves most of Canada's inhabitants live in of natural resources, notably the southern part of the country, and timber, petroleum, natural gas, vast areas of the north are sparsely metallic minerals, and fish. Inhabited.

A solution to this problem is to avoid arranging text in columns. or to provide a text only version with text laid out in a single column. In some cases providing text in a single column is not possible. such as in cases of budget information displayed in several tables. No real solution has been provided to such a problem except to provide a contact number or e-mail that someone could call to obtain help with the table.

Image Maps

Image maps hold a great deal of information. The user of a conventional browser views an image map and clicks on part of the image that the user wants to link to. For a person using a conventional browser an image map can simplify the layout of a document, however for a person using a non-visual browser, an image map is impossible to access.

Example:

```
<Title>Map Of Canada </Title> <H1>Click on the Province Below </H1>
<A HREF="http://www.canada.gov/img/canadamap">
<IMG SRC="http://www.canada.gov/images/canada.gif" ISMAP> </A>
```

In this example. the entire page is lost for non-visual browser user. An alternative way to handle this would be to present an option for a list of provinces. This would enable a user of a non-visual browser to access all of the information in the image map.

Example of accessible code 1:

```
<Title>Map Of Canada </Title> <H1>Click on the Province Below or select from the
<A HREF="http://www.canada.gov/provinces.htm">list of provinces</A> </H1>
<A HREF="http://www.canada.gov/img/canadamap">
<IMG SRC="http://www.canada.gov/images/canada.gif" ISMAP> </A>
```

In this example the user is given the choice of an alternative page "*provinces.htm*" which might look something like the following:

```
<Title> Provinces in Canada </Title>
<H1> Select from the list of provinces below </H1> <ul>
<li><A href="http://www.canada.gov/prov/ontario.htm">Ontario</A>
<li> <A href="http://www.canada.gov/prov/quebec.htm">Quebec.</A>
<li><A href="http://www.canada.gov/prov/alberta.htm">Alberta.</A>
.......
<li> ......
</ul>
```

This approach would resolve the problem that users of non-visual browsers face when navigating a web page with image maps.

Another solution that has been suggested is to create a client-side image map using alt-text for each of the links. Client-side image maps are similar to server-side image maps except that the information regarding all of the links on the image are sent to the browser along with the image map picture. If ALT-TEXT for the image map links is provided with the URLs. then the browser can display the text associated with the links of the image map. For the example above. the HTML code that would include a client-side image map with ALT-TEXT could look like the following:

Example of accessible code 2:

```
<Title>Map Of Canada </Title> <H1>Click on the Province Below </H1>
<IMG SRC="http://www.canada.gov/images/canada.gif" ALT="Image map of Canadian provinces" USEMAP="map1" BORDER=0>
<MAP NAME="map1">
<area coords="25,22,29,26" href="prov/ontario.htm" alt="Ontario">
<area coords="25,22,29,26" href="prov/quebec.htm" alt="Quebec">
<area coords="25,22,29,26" href="prov/alberta.htm" alt="Alberta">
.......
</MAP>
```

Lists

Lists are widely used in web pages. To a person who has visual access to a document it is easy to interpret lists. However, a person using a non-visual browser often does not know where a list begins or ends, or where an entry begins or ends. For people using screen readers (or Braille displays), if an entry takes up two lines, it may appear to be two separate items.

Example:

The types are:

> ***Luxury Cars...***
>
> ...
>
> ***Sports cars...***
>
> ...
>
> > ***Corvette...***
> >
> > ***Porsche...***
>
> ***Sedans...***
>
> ...

If a non-visual browser does not inform the user when an item starts or ends it is up to the user to figure it out. In the above example it may not be a problem for the user to figure out when a new item starts, however in general for lists with items that cover many lines it could be a problem. One solution would be to number the items in the list and state how many items are in the upcoming list as follows:

The 3 types of cars are:

> ***1. Luxury Cars...***
>
> ...
>
> ***2. Sports cars...***
>
> ***... Two sports cars are:***
>
> > ***a. Corvette...***
> >
> > ***b. Porsche...***
>
> ***3. Sedans...***
>
> ...

Using this approach, lists would become more accessible to a user using a non-visual browser and the information in the list would be interpreted the way it was intended.

The example guidelines above are typical in that they are intended to make non-visual browsers based on screen-reader technology more effective. However, there are still a number of problems for these browsers that have not been addressed by any proposed guidelines. For example:

Text that changes or moves:

Screen reading software may freeze if it encounters blinking or marquee text, or may become confused if the screen reading software attempts to read the blinking text while it is not appearing.

Frames:

It is easy to read frames in a document and to move from one to another when using a regular browser. However, for a person using a non-visual browser frames can very confusing. One easy solution would be to include a text-only version of the document for web pages with complex frames.

Forms:

Forms include several types of components such as buttons, edit boxes, pop-up lists and radio buttons. It is difficult for a non-visual browser user to edit and manipulate form components.

JavaScript:

JavaScript typically helps make web pages more dynamic. It is mainly used to enhance forms, and therefore it has similar set of problems to those encountered in forms. One accessibility problem is the ability to have user action's in one frame stimulating change in another. No guidelines have been developed yet. However research is underway to address these problems.

Java:

Since Java is a full-fledged programming language with several issues of its own, a simple solution to make Java applets accessible through non-visual browsers is not possible. If any guidelines were to be developed to make Java applets more accessible by non-visual browsers, they would have to be implemented during the programming of the applet itself.

3.2 Existing guidelines for the construction of non-visual web browsers

The guidelines discussed so far are intended for authors of web pages. We now discuss guidelines that may be appropriate for people who are building non-visual web browsers. It should be noted that most of the guidelines refer to non-visual web browsers that are based on screen-reading technology. For Example:

- The user needs to know when a link has been crossed as a document is being read to him/her. As the non-visual browser reads out a web document, it should be able, in some manner, to inform the user when a hyper link has been crossed.

- Non-visual browsers should allow a user to navigate through a web document according to its grammatical structure, such as by title, sentence, line, paragraph and word. Since a document can typically be divided into separate grammatical components, a user should be able to access a specific component, making it easier for them to get a mental model of the document.

- The browser should allow the user to search the whole document for a specific word. Using a conventional visual web browser a user may quickly scan through a document searching for a keyword. A user using a non-visual browser should be able to do the same thing.

- Because a user of a non-visual browser is unable to view the web document, and therefore is unable to view hyper links in the document, the user should be able to search and list all the hyper links in the document.

- Tables, as discussed earlier, are a major obstacle for users of non-visual browsers. The user should be able to read each cell of a table along with the row and column title of that cell.

- The browser should allow the user to fill out and navigate between forms on a web document. Forms are a very important part in a web document, with a non-visual browser the user is unable to view them and therefore unable to use them. A non-visual browser should allow the user to fill out forms by allowing the user to navigate between the different parts of the form such as buttons, edit boxes, pop-up lists and radio buttons.

- The user should be able to navigate between screen windows such as the URL location window and different frame windows. A typical web browser contains two different windows. a URL location window and a main text window. A text window may also contain more than one window (because of frames). A user should be able to navigate between these different windows using a non-visual browser just as a user would do using a regular web browser.

- The browser should alert the user to a new font or change in font size. As a document is being read out to the user, the user should be informed when a new font or change in font size has occurred. this information may include a meaning that may help the user better interpret the document. In particular. the browser should inform the user when an existence of bold face. italics. etc. occurs. The use of italics and bold face is to emphasise the importance of a specific word. with a non-visual browser the user will have no way of knowing if a word is italicised or bolded unless the browser itself informs the user.

It should be noted that the guidelines discussed above are merely proposed guidelines. No investigation of their relative merits has yet been carried out. and not all were followed in the construction of NVWB. Some were deliberately omitted as it was felt that they would confuse the user rather than help. Others were omitted owing to lack of time.

4 INPUT/OUTPUT TECHNOLOGY FOR NON-VISUAL BROWSERS

4.1 Speech recognition technology for input

When we think of human-computer interaction. we typically think of the keyboard and the mouse. However the ideal natural computer interface should be through speech-in and speech-out.

Being able to speak to your personal computer and have it recognise and understand what you say would provide a comfortable and natural form of communication. It would reduce the amount of typing the user has to do, leaving his/her hands free allowing the user to move away from the computer [6].

Speech recognition technology has actually been around since the 1980's, but until recently it wasn't really an option for most PC users. The products were inaccurate and hard to master. very expensive, and required specialised systems. Given these limitations, speech recognition systems were used primarily by disabled workers and by medical and manufacturing professionals who were willing to pay the high price that allowed them to do their jobs [10]. Only recently has speech recognition advanced enough to become widespread in use. Speech recognition offers real hope to those who cannot use a traditional keyboard and mouse.

New speech recognition products are now available working under standard PCs and Macs, making them easier to use with everyday applications. Improved recognition algorithms and today's powerful CPUs and digital signal processors, have made voice input faster and more accurate [13],[9].

Speech recognition applications are designed in different ways depending on a number of factors:

Continuous versus discrete speech recognition.

Speech consisting of discrete words (short silences between the words) is much easier for the technology to recognise than continuous speech because word boundaries are difficult to find in continuous speech. Pronunciation of a word may sound different when using continuous speech and some letters may be dropped. Error rates can be reduced by requiring the user to pause between each word. however this type of restriction places a burden on the user and reduces the speed with which information can be input to the system.

Speaker-dependent versus speaker -independent speech recognition.

Many systems are speaker dependent, trained for use with each different operator. The general public can use relatively few speech recognition systems. Speaker-dependent systems have error rates roughly three to five times less than speaker-independent ones. Speaker-dependent systems must also be trained to recognise the patterns and inflection of each user's speech. This training can be tedious and time consuming and inappropriate for certain types of applications.

Large versus small vocabulary

The size of the vocabulary of words to be recognised has considerable affect on recognition accuracy. In addition large vocabularies are more likely to contain ambiguous words than small vocabularies and therefore have even lower accuracy rates. Vocabulary size also affects the amount of time it takes to search the speech-model database.

Grammar Constraint

The grammar of the recognition domain defines the allowable sequences of words. A tightly constrained grammar is one in which the number of words that can legally follow any given word is small. Systems that are tightly constrained are more accurate than those that give the user more freedom because the system can limit the effective vocabulary (and search space) to those words that can occur in the current input context.

Natural Speech Input

This is the ultimate goal in speech recognition technology. To be able to talk to your computer in a natural manner and have the computer understand what the user wants. then apply the commands or words (i.e. "create a new letter to Dr. Frost from University of Windsor"). The computer would create a new letter to Dr. Frost and insert the address of the University of Windsor.

Environment

Many recognition systems are capable of very low error rates as long as the environmental conditions remain quiet and controlled. However, performance degrades when noise is introduced or when conditions differ from the training session in user-dependent systems. To compensate, the user may have to wear a head-mounted, noiselimiting microphone (with the same response characteristics as the microphone used during training for user-dependent systems).

4.2 Synthesised voice output

Speech-synthesis systems convert text to spoken output by automatically generating synthetic speech. Speech synthesis is typically referred to as "Text-to-Speech" conversion (TTS). this technology takes text as input and produces. as output. sounds that resemble human speech.

Although speech synthesisers cannot imitate the human sound exactly. they can read text files and output them in a dull but intelligible voice. Some systems also allow the user to choose a voice from a pool of different voices. (child. male. female etc.). Since a speech synthesiser can read out any text file. people with impaired vision find it to be a very valuable device.

Most of the speech systems available are full text-to-speech systems for English. These systems are able to produce high quality speech in both male and female voices. These systems are phonetically driven. these do not contain a dictionary with word descriptions or pre-recorded human "readings" of words. That is the system generates voice output by analysing the structure and context of words. they also have good text-analysis capabilities. which allow for good word pronunciation as well as name pronunciation.

5 Design principles for the construction of speech-driven non-visual web browsers based on content rather than layout

Some of the guidelines described earlier. in section 3.2. are applicable to the construction of non-visual browsers based on content rather that layout, and can be rephrased as design principles for this type of browser:

1. The non-visual browser should allow the user to navigate through a document according to its grammatical structure e.g. word, sentence, or paragraph. Therefore the browser should be able to recognise when a new word, sentence or paragraph starts and where it ends.

2. The user should not have to read through the whole document to be able to realise which links it contains. A sighted user would generally skim through a document looking for a specific link without reading the whole document, a visually impaired user should also be able to do something similar. On the other hand the user should also be informed of a hypertext link as he/she navigates through the document, this would allow the user to better understand what the hypertext link corresponds to. There should also be a method that allows the user to choose a hypertext link to follow.

3. The browser should be able to understand the structure of a form, and should represent the form to the user in a manner in which the user would be able to

fill it out and submit it. The manner in which the form is to be represented should allow the user to fill out different portions of the form, moving from one part to another, checking the information typed so far, and correcting errors as desired.

4. Similar to forms, a method in which a user can access a table in the document also should be developed. The user should be able to read each cell of a table along with the row and column of that cell. The browser should give the user a comprehensive understanding of the table.

Other guidelines can be added which take into account the relationship between physical structure and the semantic content of documents. For example:

Design Principle 5:

The input command language has to be sufficiently rich to allow the user to obtain an "overview" of the document as well as having the ability to home in on specific components. Since the only interface the user has is a speech interface, the commands accepted by the browser must be rich enough to make it possible for the user to browse any component of the document without constraint. That is, the user should be able to move from one paragraph to another, request to read out a paragraph, sentence or word, move to a list, form or link etc. Making the command grammar richer allows the user to navigate through a document in an easier manner obtaining a better overview of the document. it would also enable a user to navigate a document in the manner the user prefers or feels more comfortable with.

Design Principle 6:

A method of informing the user of the existence of different objects such as tables, lists, links, forms etc. should be developed. One idea would be to change the voice of the speech synthesiser as hypertext links are read out, another would be to inform the user of the existence of a table by stating "*The following is a table*". However, when developing such ideas, the fact that a user's trail of thought can be disturbed must be considered. As a user is listening to the web document being read out, hearing tones, beeps, voice changes and messages for every object in the document can distract and sometimes disturb the user. Therefore, conveying information about the document is important, however keeping it simple and as clear as possible must be considered.

Additional design principles are necessary for those non-visual browsers that have speech input and synthesised voice output. For example:

Design Principle 7:

The characteristics of the speech recogniser should be considered when developing such a non-visual browser. The ultimate speech recogniser would be one that recognises continuous, speaker-independent speech, has a large vocabulary and allows natural speech input. However with such features, the accuracy of the speech recogniser would be very low. To increase the recogniser's accuracy some features such as large grammar or natural speech input have to be removed. To develop a speech interface to a non-visual browser, removing some features would only mean more constraints on the user. Therefore the proper features should be chosen to allow a high level of accuracy, but at the same time not to constrain the user too much, allowing the interface to be as user friendly as possible. For the web-browsing application the speech interface is mainly a command interface, that is an interface that accepts specific commands by the user and executes the necessary action according to the command. Because of that, the grammar of the speech recogniser can be constrained, containing only specific commands that the non-visual browser will accept such as ***"read next sentence", "go to next paragraph"*** or ***"go to paragraph two"*** etc.. By using a small grammar we can also afford to use a speaker-independent, continuous-speech system and still maintain high accuracy. With a continuous-speech system, the user does not have to pause between each word in the uttered command, making it more user friendly. With a speaker-independent system the non-visual browser may be used by any user without any training required, making it more accessible with less work in setting it up.

Design Principle 8:

Because the input language has to be sufficiently constrained in order to have high recognition accuracy, this precludes use of "document-dependent" words or phrases, such as "***follow the link to University Of Windsor***", when user-independent, continuous-speech recognition technology is being used. The above command would have to be re-phrased as "***follow link number three***". Consequently, the command language should include commands, such as "***list all links in the last paragraph***", which can be used to remind the user of the link numbers associated with link text.

Design Principle 9:

To improve speech accuracy even more, the browser should be able to change the working grammar "on the fly" so that the user would be constrained even more to the type of commands the user can ask. For example, if the user wants to reload the previous document, the browser can ask to confirm the action by stating "Do you want to reload the previous document?" the system at this point would then load in another grammar, one that only accepts the words "yes" and "no". At this point the user is expected to respond with a yes or no answer, therefore there is no need to have a larger grammar with words or phrases that the user would not be expected to utter. Once the user has responded with a yes or no, the browser then reloads in the original larger grammar. This approach can dramatically improve the accuracy of the speech recogniser considerably.

Other design principles derive from the assumption that content rather than layout should underlay the navigational process:

Design Principle 10:

Owing to the fact that the browser is based on content, no commands should refer to the visual layout of the document. For example, the user should not be able to ask to move to the next "line" or ask about the size of the font, since these questions or commands relate directly to the actual visual layout of the document. The only questions or commands allowed should be ones that are related directly to the semantic content of the document.

Design Principle 11:

Owing to the fact that the browser is based on content. output should ignore layout HTML tags unless they convey meaning. In those cases where meaning is conveyed by layout tags the output from the browser should be based on the "direct" conveyance of meaning rather than reference to some visual clue. For example. text which is enclosed by the bold font tag should not be output as "in bold font. an important message follows". but rather by a change in tone of the synthesised voice output. "An important message follows!" pronounced in a "*commanding*" voice.

However. these latter principles are based on assumptions that may not be justified. Perhaps physical layout. and font type. do serve as useful additional prompts to understanding and creating a mental model of document content. However. as this work is concerned with the construction of a "content-only" browser. guidelines 10 & 11 are have been followed in the construction of NVWB.

Other design principles depend on external factors. For example, if one can assume that all of the documents to be browsed conform to a certain set of guidelines then the browser can be specifically tailored to those documents. This simplifies design but constrains use of the browser.

One of the major larger-term objectives of the project, of which the work described in this report is a part, is to identify appropriate design principles through investigation and modification of the speech-driven non-visual browser constructed specifically for this purpose

6 The speech-driven non-visual web browser - NVWB

6.1 Components of NVWB

The principle components of our implementation of a non-visual user-independent continuous-speech, browser based on document content, include a page processor, an SSI speech recognizer (as an input device) and Dectalk (as an output device). The software was built entirely using Visual Basic, mainly because of the fact that the s/w package with the SSI speech recognizer contains tools for developing speech enabled applications in Visual Basic.

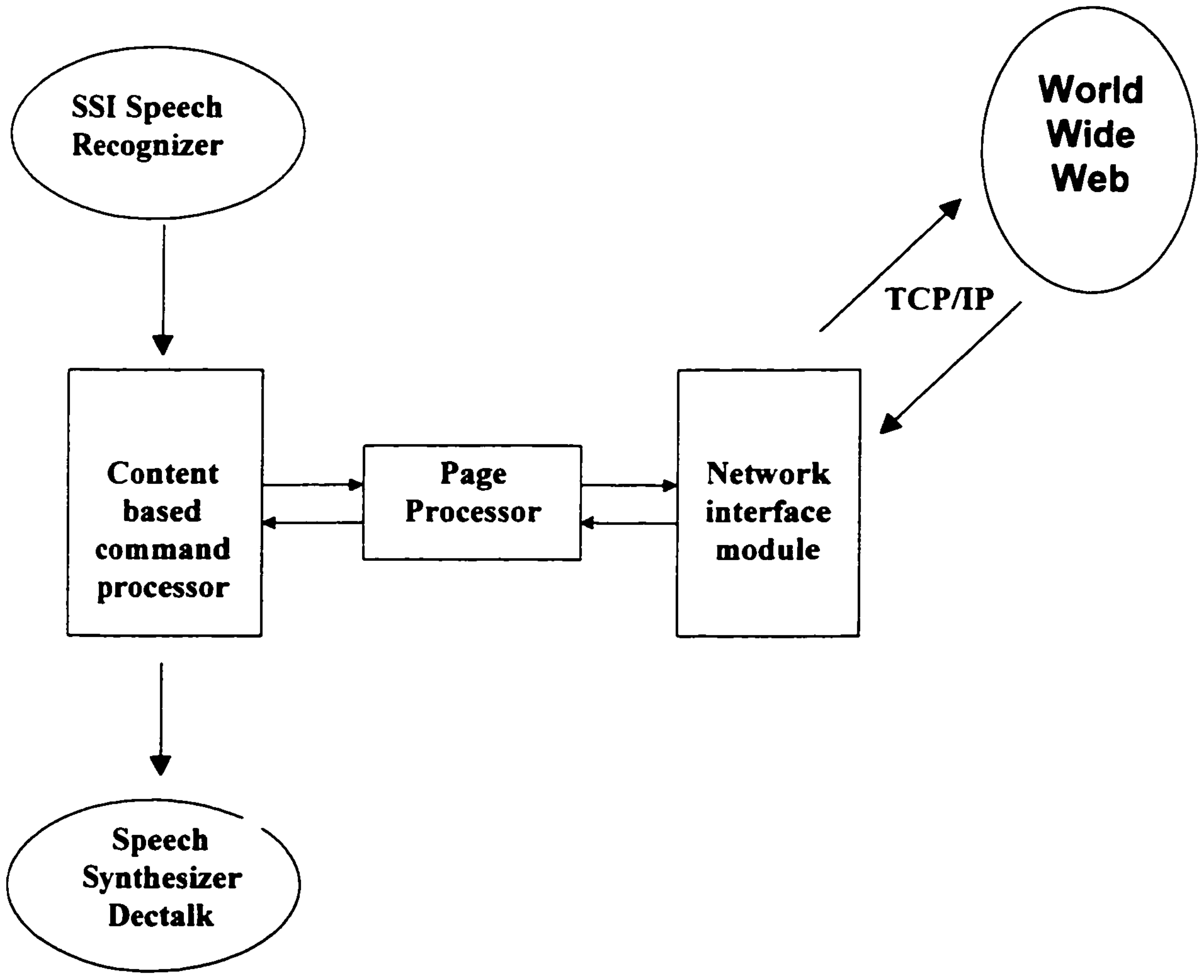


Figure 1. An illustration of the speech driven non-visual web browser

- Network Interface module

  This component retrieves a web document from a specified web site. The network interface makes an Internet connection using the TCP/IP protocol and retrieves from the specified server the specified HTML document. Below are the steps the interface module takes to retrieve a document.

  1) The network interface retrieves the web server name from the given address and a connection is established. For example if the user wants to retrieve a document from http://www.cs.uwindsor.ca/users/h/haddad/, the browser network interface would connect to the server www.cs.uwindsor.ca.
  2) The network interface issues a "*get*" command, which asks the web server to send a specific file. In this example the get command would appear as "*get /users/h/haddad/*". At this point the server would respond by sending back the requested file.
  3) The network would keep on retrieving the requested document, storing it into a string. Once the whole file has been retrieved the connection is closed.

- The Page Processor

  This component processes an HTML document gathering information about the page. A data structure is created containing relevant information about the page's links, images, lists etc. At the same time the page processor also reconstructs the document by removing HTML tags and adding some new tags and information to it.

The page processor does this by parsing through the whole document. As each HTML tag is visited the page processor reconstructs the document according to the tag. the processor does this by adding and removing text and tags from the document. The page processor also retrieves and stores information in different tables for different HTML constructs such as lists and links.

Because this is only a prototype of a NVWB. not all HTML constructs are covered. mainly because of the fact that HTML is a versatile scripting language and therefore it does not restrict the user to a rigid syntax.

- Content based command processor

  The content-based command processor accepts input from the SSI speech recognizer processes the commands and sends the output to Dectalk. It uses the document that the page processor reconstructed to respond to the user's commands and actions.

  As a user utters a command, the command processor responds to the command by gathering the necessary response from the newly constructed document and tables. or taking a necessary action. This response is than sent to the speech synthesizer for output.

- The SSI speech recognizer

  As the non-visual browser starts up, it initializes the SSI speech recognizer and loads the syntax for the command language. Only commands in a specific command language are accepted by the SSI speech recognizer and passed to the content-based command processor. The speech recognizer used is one that accepts continuous speech, is user independent and which has been tailored through the specification of a small grammar.

- Dectalk

  This component is the output device for the non-visual browser. All output from the command processor is sent to Dectalk, which converts the text. Dectalk is phonetically driven, that is it does not contain a dictionary for the pronunciation of each word, instead the system generates voice output by analyzing the structure and context of the word.

6.2 The Input command language and output from NVWB

The following is the initial grammar that the browser loads.

```
S -> title
     |list{links |images |bolded obj1 |italicized obj1 |headings}
     | how many OBJECT | headings
     | link  { Numbers1 | Numbers1 Numbers}
     | reed
     | set mode
     | forward
     | backwards
     | previous
     | next
     | go to { Numbers1 | Numbers1 Numbers}
     | skip { Numbers1 | Numbers1 Numbers}
     | spell
     | enter location
     | fill out form { Numbers1 | Numbers1 Numbers }

OBJECT ==links | lists | images | bolded words | italicized words

Obj1== words | phrases
```

```
NUMBERS1 -> one_to_9

one_to_9  == 1 2 3 4 5 6 7 8 9

NUMBERS -> 0_to_9

0_to_9  == 0 1 2 3 4 5 6 7 8 9
```

The following are examples of the commands in this grammar, together with the corresponding output from the browser. For a more detailed description of the input/output of the NVWB refer to appendix A.

Title

Requests from the browser the title of the document. The browser as a response would return the following:

"The title of this page is, TITLE"

How many links

Requests from the browser the number of links found in the document. The browser would respond with the number of links in the document, as follows:

"There are <NUMBER OF LINKS> ***Links."***

How many lists

Requests from the browser the number of lists found in the document. The browser would respond with the number of lists in the document, as follows:

"There are <NUMBER OF LISTS> ***Lists."***

How many images

Requests from the browser the number of images found in the document. The browser would respond with the number of images in the document. as follows:

"There are <NUMBER OF IMAGES> ***images."***

How many bolded phrases/words

Requests from the browser the number of bolded phrases found in the document. The browser would respond with the number of bolded phrases in the document. as follows:

"There are <NUMBER OF BOLDED PHRASES> ***bolded phrases.***

How many italicized phrase/words

Requests from the browser the number of italicized phrases found in the document. The browser would respond with the number of italicized phrases in the document, as follows:

"There are <NUMBER OF ITALICIZED PHRASES> ***italicized phrases."***

How many headings

Requests from the browser the number of headings found in the document. The browser would respond with the number of headings in the document, as follows:

"There are <NUMBER OF HEADINGS> ***headings."***

Even though the above three commands are closely related to the visual layout of the document. it should be noted that bolded and italicized words. as well as headings. do convey some meaning to the user. Bolded and italicized words can be considered as the important words in a document, and just as a sighted user can skim through a document looking at headings or bolded and italicized words. a visually impaired person should be also be able. in a similar manner. to obtain that information.

How many paragraphs

Requests from the browser the number of paragraphs found in the document. The browser would respond with the number of paragraphs in the document. as follows:

"There are <NUMBER OF PARAGRAPHS> ***paragraphs."***

How many sentences

Requests from the browser the number of sentences found in the document. The browser would respond with the number of sentences in the document, as follows:

"There are <NUMBER OF SENTENCES> ***sentences."***

List links

Requests a list of all the hypertext anchors in the document. As a response each hypertext anchor in the document is preceding each text anchor a number is attached. The user can use this number to reference that specific anchor, the response would be as follows:

"There are <NUMBER OF LINKS (N)> ***Links in this page, They are:***

1. <first hypertext anchor>

2. <second hypertext anchor>

....

N. <nth hypertext anchor>***"***

List images

Requests a list of all the images in the document. As a response each alternative text of each image in the document is returned. The response would be as follows:

"There are <NUMBER OF IMAGES (N)> ***images in this page, They are:***

1. <first image's alt text>

2. <second image's alt text>

....

N. <nth image's alt text>***"***

List italicized phrases/words

Requests a list of all the italicized phrases in the document. As a response each italicized phrase in the document is returned. The response would be as follows:

"There are <NUMBER OF ITALICIZED PHRASES (N)> ***italicized phrases,***

They are:

1. <first italicized phrase>

2. <second italicized phrase>

....

N. <nth italicized phrase>***"***

List bolded phrases/words

Requests a list of all the bolded phrases in the document. As a response each bolded phrase in the document is. The response would be as follows:

"There are <NUMBER OF BOLDED PHRASES (N)> ***bolded phrases, They are:***

1. <first bolded phrase>

2. <second bolded phrase>

....

N. <nth bolded phrase>***"***

List headings

Requests a list of all the headings in the document. As a response each heading in the document is returned. The response would be as follows:

"There are <NUMBER OF HEADINGS (N)> ***headings, They are:***

1. <first heading>

2. <second heading>

....

N. **<nth heading>"**

The next few commands (set mode, reed *(note: not misspelled, see later)*, next, previous, go to (n), skip (n) and spell) all depend on what mode the browser is currently set at. The browser can be set to word, sentence or paragraph mode. Word mode reads a word at a time, sentence mode reads a sentence at a time, and paragraph mode reads a paragraph at a time.

Set mode

Allows the user to choose the mode in which the browser would navigate through the document. Once the user issues this command the following grammar is loaded in:

```
S -> paragraph
        | sentence
        | word
```

and the following message is output :

"Please specify what mode you would prefer, paragraph, sentence, or word?"

At this point the user needs to respond with "*word*", "*sentence*" or "*paragraph*". The mode is then set to the choice that the user uttered. Once the user has made a choice and the proper mode has been set the browser responds with:

"The mode has been set to <CHOSEN MODE> "

Setting the mode to "*word*" would allow the user to clearly read out one word at a time, and if the user at any point is unable to understand the word, he/she would be able to spell it out.

Reed

This word is not misspelled, the reason for it being spelled this way is because the speech recognizer recognizes the word ***reed*** better than it does the word ***read.*** This command simply reads out text from the document according to the mode that is set, word mode would read out the current word, sentence mode would read out the current sentence and paragraph mode would read out the current paragraph.

Next

Works much like the ***Reed*** command with one difference, the browser moves to the next block of text (depending on the mode set, word sentence or paragraph) and reads that out.

Previous

Similar to the *next* command, but instead of reading the next block of text the browser is at, the browser reads the previous block of text (word, sentence or paragraph).

Spell

This command works if the user is set to word mode. the browser simply responds by spelling out the current word at which it is.

Backwards

Just like most other browsers this is an option for the user to go back to the previous document that the user has already visited. as a result the previous document is loaded into the browser.

Forward

Just like the **backwards** option this is an option for the user to go forward to the next document that the user has already visited. as a result the next document is loaded into the browser.

Enter Location

This command allows the user to type in a URL. As in any regular browser. the user can type in a URL location of a document to be retrieved. As a result the browser accepts keyboard input from the user. once the user presses return the document referred to is retrieved by the browser. The browser reads out each letter as it is typed, thereby enabling the user to confirm that the correct keys have been pressed. A possible enhancement to NVWB would be to allow the user to "speak" the URL one letter at a time, rather than use a keyboard.

For the following commands the user is expected to utter a value between 1 to 99 following each command. However, to make the system more accurate, the user is expected to spell out the number. That is if the user wants to enter the number 13, the user would be required to spell it out "one three".

Skip [1..99]

Informs the browser to skip a specified number of blocks (word, sentence or paragraph) of text.

Go to [1..99]

Informs the browser to go to a specified block (word, sentence or paragraph) of text in the document.

Link [1..99]

Informs the browser to link to a specific text anchor by simply referencing it according to its numerical occurrence in the document. The browser would retrieve the document that the specified hypertext link refers to.

Fill out form [1..99]

For a user to fill out a specific form in the document the following command has to be issued. the number given represents which form to fill out. As a result the following message is given:

"This form has <N> text boxes, would you like to edit a text box, clear text boxes, submit the form, or exit form?"

Where <N> is the number of text boxes available to be filled out by the user. At this point the following grammar is loaded:

```
s -> edit text box { Numbers1 | Numbers1 Numbers}
       | submit (form)
       | clear (text)  (boxes)
       | reset
       | exit form

NUMBERS1 -> one_to_9

one_to_9  == 1 2 3 4 5 6 7 8 9

NUMBERS -> 0_to_9

0_to_9  == 0 1 2 3 4 5 6 7 8 9
```

At this point the user can issue only commands that are related to the form, below

Edit text box [1..99]

This option allows the user to fill out one of the available text boxes in the form by specifying a number according to its occurrence in the form. The user is expected to type in, using the keyboard, the input the user wants to be inserted in this text box followed by the return key.

Clear/Reset

Clear or ***Reset*** simply clears all the input the user has entered into the text boxes so far, just as a regular reset button would work on a typical visual browser.

Submit

Once the user utters this command, the browser would execute the specified action by the form, sending as the input the input the user typed.

Exit form

This command simply exits the form the user is presently at and loads up the main grammar.

Filling out forms allows a user to access search engines, this a very important part of using the WWW, and users using NVWB should have access to all search engines just as users using a visual browser do.

A more detailed discussion of the command language is given in the Appendix.

6.3 Internal representation of HTML documents in NVWB

As each web page is retrieved specific information that is considered relevant to non-visual browsing is extracted and placed in a newly constructed page. In effect, tags that were provided for visual layout are converted or discarded and additional tags are added to indicate, for example, the number of elements in a list. Following is a discussion of the information extracted and the information added to the web document.

Title of a Web Document

The title of a document can be found between the following tags ***<title>.....</title>*** as in the following example:

<title> This is an example of a title. </title>

The text between the two tags is considered to be the title of the document. Therefore in a global variable called **TITLE**, the text between the title tags in the HTML is stored, and the title tags as well as the text between them are removed from the newly constructed page. When a user queries the browser about the title, the command processor retrieves the title from the data structure and sends a response to Dectalk.

Headers

A header tag is expected to contain text that is considered to be a header. An example of such a header tag would be:

<H1>A non-visual browser.</H1>

The value 1 in the above example represents the format of the header, the smaller the value the larger the header appears on a visual web browser.

The following tag would replace this tag:

<HX> A non-visual browser.</HX>

where *X* is a counter numbering each header according to its occurrence in the document, e.g. the first header would be replaced with *<H1>*, second with *<H2>* etc. As a sentence is being read out by Dectalk, all headers in the sentence would be read out just as regular text, a user should be able to distinguish a title from regular text by its content.

Bolded Phrases

A bolding tag is expected to contain text that is to be shown as bolded. An example of such a header tag would be:

<b>A non-visual browser.</b>

The following tag would replace this tag:

<bX> A non-visual browser.</bX>

where *X* is a counter numbering each bolded phrase according to its occurrence in the document, e.g. the first bolded phrase would be replaced with *<b1>*, second with *<b2>* etc. As a sentence is being read out by Dectalk, all bolded phrases in the sentence would be read out just as regular text, a user would be able to retrieve all bolded phrases in the document by listing them.

Italicized Phrases

An italic tag is expected to contain text that is to be shown as italicized. An example of such a header tag would be:

<i>A non-visual browser.</i>

The following tag is replaced with this tag:

<iX> A non-visual browser.</iX>

where *X* is a counter numbering each italicized phrase according to its occurrence in the document, e.g. the first italicized phrase would be replaced with *<i1>*, second with *<i2>* etc. As a sentence is being read out by Dectalk, all italicized phrases in the sentence would be read out just as regular text, a user would be able to retrieve all italicized phrases in the document by listing them.

The reason for not informing the user of the presence of a heading, bolded phrase or italicized phrase, was because of doing so, the user would be easily distracted from the information being outputted by Dectalk. During the design of this browser, it was discovered that informing the user of specific information while reading out text from the document should be very minimal in order not to distract the user.

Forms

A form tag is expected to contain the description of a form to appear on the web document. An example of such a form tag would be:

```
<form>
Please enter your name: <input type=text>
<p>
Please enter your email address: <input type = text>
<p>
<input type=submit>
<input type=reset>
</form>
```

The following tag replaces this tag:

```
<formX>
Please enter your name: <input type=text>
<p>
```

```
Please enter your email address: <input type = text>
<p>
<input type=submit>
<input type=reset>
</formX>
```

where *X* is a counter numbering each form according to its occurrence in the document. As a sentence is being read out by Dectalk, all forms in the document would be read out just as regular text, however as the browser crosses over a text box a beep is sounded to inform the user of the text box.

Images

An image tag is expected to contain the image's path along with some alternative text describing the image. An example of such an image tag is:

```
<img src="http://www.host.ext/path/imagefile" alt="description of image">
```

A simple tag would replace this tag:

```
<imageX>
```

where *X* is a counter numbering each image according to its occurrence in the document, e.g. the first image would be replaced with *<image1>*, second with *<image2>* etc. As for both the image's filename and alternative text, they are stored in a table corresponding to the image's *X* value. As Dectalk is reading out a sentence, each image in the sentence is replaced with its alternative text. Since the image's alternative text replaces the image tag, the user won't be able to distinguish between the regular text, and the alternative text. As a solution to this, the method we used to inform the user was to change the voice of the Dectalk when the alternative text is uttered. That is the voice may change from a regular male voice to a kid's voice as a hypertext link is read out. This approach while informing the user of the alternative text, it does not disturb the user's concentration.

Links

A link in an HTML document has a format similar to the following:

<a href="http://www.host.ext/path/filename.htm"> ....link description.......</a>

Such a link is replaced in NVWB with a tag with the following format:

<linkW>.........link description.....</linkW>

where ***W*** is a counter numbering each link according to it's occurrence in the document. As for the links URL, it is stored in a table corresponding to the link's ***W*** value.
When a sentence is being read out to the user, all links are uttered in a different voice, this is for the user to distinguish regular text from a link. When a user links to a specific link he/she would be required to state the ***W*** value of the link, e.g. to link to ***link4*** the user would utter ***"Link four"***. The page processor at this point would ask the user if he/she wanted to retrieve link *"link description"*, and if the response of the user was *"yes"* the browser would then retrieve the URL location of ***link4*** and would retrieve the document.

Lists

There are two kinds of lists handled by the page processor. ordered and unordered. Even though the two types are different. they are both handled the same way by the page processor. The HTML tags for ordered and unordered lists are ***<ol>...</ol>***. ***<ul>....</ul>*** respectively. A simple tag replaces both of these tags:

<listY>........</listY>

where ***Y*** is a counter numbering each list (ordered or unordered) according to its occurrence in the document. In both kinds of lists, the individual items are designated with a <li> tag, this tag is replaced by a tag <li***Z***> were ***Z*** is a variable numbering each item according to its occurrence in between the list tags. Nested lists are also handled, below is an example of an HTML code of a nested list and how the page processor would modify it:

HTML Code

```
<ul>
        <li> item1
        <li> item2
 <ul>
          <li>item1
        <li>item2
        <li>item3
</ul>
        <li> item3
</ul>
```

Processed HTML Code

```
<list1>
         <li1> item1
         <li2> item2
 <list2>
              <lia> item1
              <lib> item2
              <lic> item3
</list2>
       <li3> item3
</list1>
```

From the above example it can be seen that characters instead of numbers identify items in nested lists. When a list is read out to the user, before each item the item's number (or character) is first announced in a different voice, this is so that the user will be able to recognize when a new item is visited and when a nested list is visited.

Sentence Tag

As the HTML document is being retrieved, the page processor breaks up the document into sentences by inserting a sentence tag at the start of each new sentence. A sentence tag <s*W*>, ***were W*** is a counter numbering each sentence according to its occurrence in the document, is inserted in the following cases:

- If the sentence is the first in the document.
- After a sentence that just ended with a period, and the period is not a decimal point used in a number.
- Following a <br> tag.
- Following a <p> tag.
- Following a table.
- Before a <ul>,<ol> or <li> tag.
- Following a </ul> or </ol> tag.

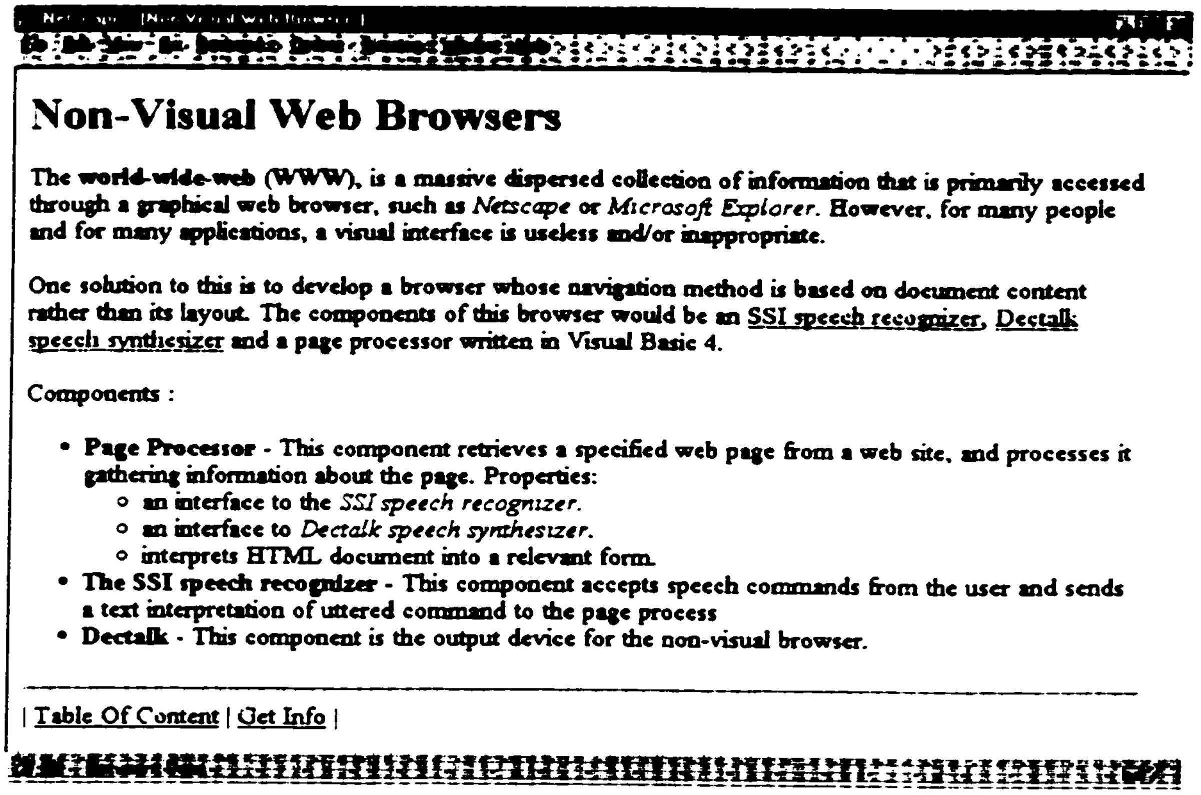
Paragraph Tag

As the HTML document is being retrieved, the page processor breaks up the document into paragraphs by inserting a paragraph tag at the start of each new block of text. A paragraph tag ***<pW>*, *were W*** is a counter numbering each paragraph according to it's occurrence in the document, is inserted in the following cases:

- If the paragraph is the first in the document.
- After two line breaks. A line break is considered any tag that causes the browser to break to the next line (including <br>, </table>, </ul> tags).
- Following a <p> tag.

6.4 Sample Session Using NVWB

When the non-visual browser is started, a default web document is retrieved. At this point in time the user is able to issue commands, gathering information about the web document. The non-visual browser does not display any part of the web document, instead the way it works is by responding to commands issued by the user, giving the user a comprehensive understanding of the document. The fundamental idea behind the design of the browser is to imagine a friend holding a document, which you are not able to see. The only thing your friend can do is to respond to questions about the document. Such questions may be to ask about any titles or headings, read out a paragraph or sentence or give a description of images or pictures. By asking your friend a series of questions you would be able to have a very good understanding of what the document contains. This is exactly how NVWB works, the only difference is that the user is constrained in the questions he/she can ask, only questions from the command language are accepted. All responses from the browser are sent to Dectalk and converted from text to speech. The following is a sample session with NVWB, when the following web document is loaded:


Non-Visual Web Browsers

The **world-wide-web (WWW)**, is a massive dispersed collection of information that is primarily accessed through a graphical web browser, such as *Netscape* or *Microsoft Explorer*. However, for many people and for many applications, a visual interface is useless and/or inappropriate.

One solution to this is to develop a browser whose navigation method is based on document content rather than its layout. The components of this browser would be an SSI speech recognizer, Dectalk speech synthesizer and a page processor written in Visual Basic 4.

Components :

- **Page Processor** - This component retrieves a specified web page from a web site, and processes it gathering information about the page. Properties:
  - an interface to the *SSI speech recognizer.*
  - an interface to *Dectalk speech synthesizer.*
  - interprets HTML document into a relevant form.
- **The SSI speech recognizer** - This component accepts speech commands from the user and sends a text interpretation of uttered command to the page process
- **Dectalk** - This component is the output device for the non-visual browser.

| Table Of Content | Get Info |

Figure 2. A snap shot of the web page being browsed.

The HTML code of this document would be:

```
<HTML>
<title>Non-Visual Web Browsers</Title>
<body>
<font size=4>
<H1>Non-Visual Web Browsers</H1>
The <b>world-wide-web (WWW)</b>, is a massive dispersed
collection of information that is primarily accessed
through a graphical web browser, such as
<i>Netscape</i> or <i>Microsoft Explorer</i>. However,
for many people and for many applications, a visual
interface is useless and/or inappropriate. <p>
One solution to this is to develop a browser whose
navigation method is based on document content rather
than its layout. The components of this browser would
be an <a href=http://www.speechsys.com>SSI speech
recognizer</a>, <a href=http://www.dectalk.com>Dectalk
```

```
speech synthesizer</a> and a page processor written in
Visual Basic 4. <p>
Components :
<ul>
<li><b>Page Processor</b> - This component retrieves a
specified web page from a web site, and processes it
gathering information about the page.
Properties:
<ul>
<li>an interface to the <i>SSI speech recognizer</i>.
<li>an interface to <i>Dectalk speech synthesizer</i>.
<li>interprets HTML document into a relevant form.
</ul>
<li><b>The SSI speech recognizer</b> - This component
accepts speech commands from the user and sends a text
interpretation of uttered command to the page process
<li><b>Dectalk</b> - This component is the output
device for the non-visual browser.
</ul>
<p><HR>| <a href=index.htm>Table Of Content</a> | <a
href=form.htm>Get Info</a> |
</font></body>
</HTML>
```

Once the page processor has processed the document, the web document would look something like the following.

```
<p1><s1><H>Non-Visual Web Browsers</H>
<p2><s2>The <b>world-wide-web (WWW)</b>, is a massive
dispersed collection of information that is primarily
accessed through a graphical web browser, such as
<i>Netscape</i> or <i>Microsoft Explorer</i>.
<s3> However, for many people and for many applications, a
visual interface is useless and/or inappropriate.
<p3><s4> One solution to this is to develop a browser whose
navigation method is based on document content rather than
its layout.
<s5> The components of this browser would be
an <link1>SSI speech recognizer</link1>, <link2>Dectalk
speech synthesizer</link2> and a page processor
written in Visual Basic 4.
```

```
<p4><s6>Components :
<list1>
<s7><li1><b>Page Processor</b> - This component retrieves a
specified web page from a web site, and processes it
gathering information about the page.
Properties:
<list2>
<s8><lia>an interface to the <i>SSI speech recognizer</i>.
<s9><lib>an interface to <i>Dectalk speech synthesizer</i>.
<s10><lic>interprets HTML document into a relevant form.
</list2>
<p5><s11><li2><b>The SSI speech recognizer</b> -
This component accepts speech commands from the user and
sends a text interpretation of uttered command to the page
process
<s12><li3><b>Dectalk</b> - This component is the output
device for the non-visual browser.
</list1>
<p6><s13>| <link3>Table Of Content</link3> | <link4>Get
Info</link4> |
```

The following is a sample of commands uttered by the user and the response from the non-visual browser.

Command	[illegible]
Title.	The title of this page is, non-visual web browsers.
How many links?	There are four links.
How many lists?	There are two lists.
How many images?	There are no images.
How many sentences?	There are thirteen sentences.

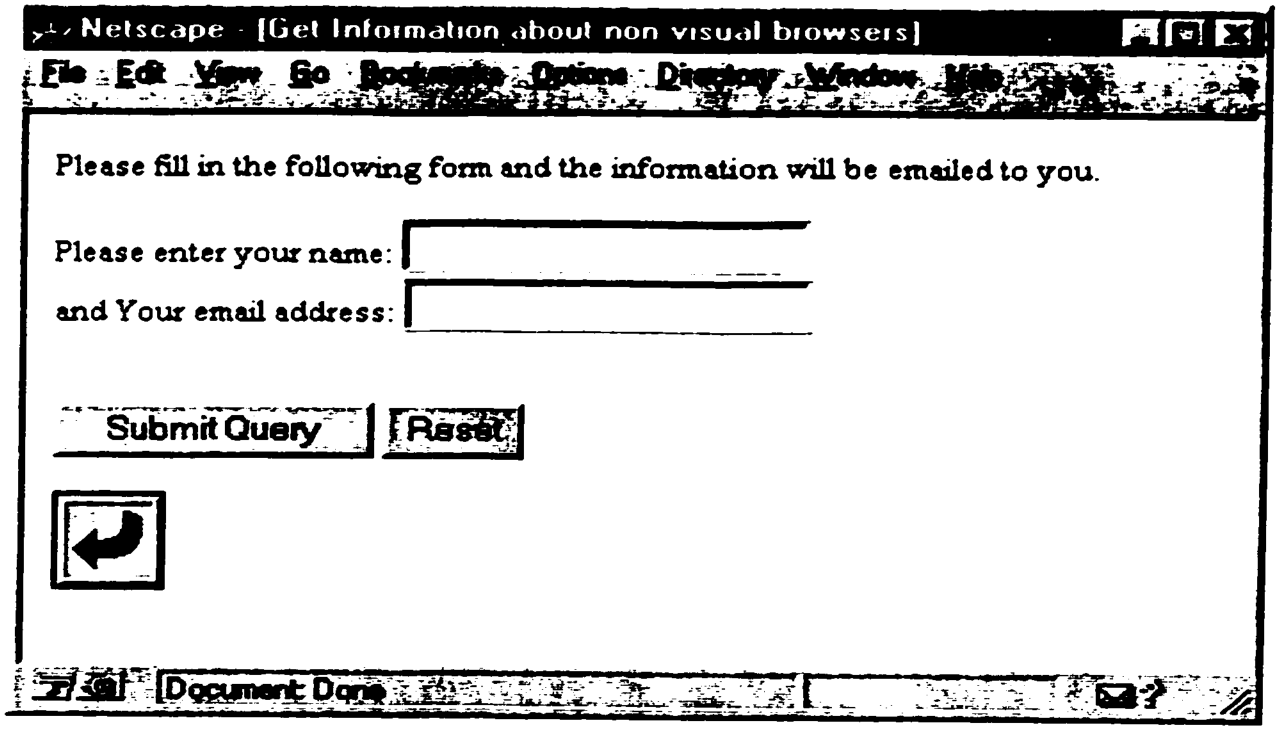
How many paragraphs?	**There are six paragraphs.**
Reed.	Non-Visual Web Browsers.
Next.	The world-wide-web (WWW), is a massive dispersed collection of information that is primarily accessed through a graphical web browser, such as Netscape or Microsoft Explorer.
Next.	However, for many people and for many applications, a visual interface is useless and/or inappropriate.
Set mode.	Please specify what mode you would prefer, paragraph, sentence, or word?
Paragraph.	The mode has been set to paragraph.
Reed.	The world-wide-web (WWW), is a massive dispersed collection of information that is primarily accessed through a graphical web browser, such as Netscape or Microsoft Explorer. However, for many people and for many applications, a visual interface is useless and/or inappropriate.
Next.	One solution to this is to develop a browser whose navigation method is based on document content rather than its layout. The components of this browser would be an SSI speech recognizer, Dectalk speech synthesizer and a page processor written in Visual Basic 4.
Next.	Components : 1. Page Processor - This component retrieves a specified web page from a web site, and processes it gathering information about the page. Properties:

	a. an interface to the SSI speech recognizer. **b. an interface to Dectalk speech synthesizer.** **c. Interprets HTML document into a relevant form.**
Next.	2. The SSI speech recognizer - This component accepts speech commands from the user and sends a text interpretation of uttered command to the page process 3. Dectalk - This component is the output device for the non-visual browser.
List Links.	There are four links in this page, they are: 1. **SSI speech recognizer.** 2. **Dectalk speech synthesizer.** 3. **Table Of Content.** 4. **Get Info.**
List italicized phrases.	There are four italicized phrases, they are: 1. Netscape. 2. Microsoft Explorer. 3. SSI Speech recognizer. 4. Dectalk speech synthesizer.
List bolded phrases.	There are four bolded phrases, they are: 1. The World-Wide-Web (WWW). 2. Page Processor. 3. The SSI speech recognizer. 4. Dectalk.
Set mode	Please specify what mode you would prefer, paragraph, sentence, or word?
Sentence	The mode has been set to sentence.
Go to 2	Do you want to go to sentence 2?
Yes	At 2.

Reed	The world-wide-web (WWW), is a massive dispersed collection of information that is primarily accessed through a graphical web browser, such as Netscape or Microsoft Explorer.
Set mode	Please specify what mode you would prefer, paragraph, sentence, or word?
Word	The mode has been set to word.
Skip 2 2	Do you want to skip twenty-two words?
Yes	Skipped twenty-one.
Reed	As
Next	Netscape
Spell	N E T S C A P E
link 2 3	Do you want to link to twenty-three?
Yes	Hypertext link twenty-three does not exist, there are four links in this page.
Link 4	Do you want to Get Info?
Yes	Linking to hypertext link 4.

For the last command uttered, the browser loads in the document called

forms.htm this file is as follows:


Netscape - [Get Information about non visual browsers]
File Edit View Go Bookmarks Options Directory Window Help
Please fill in the following form and the information will be emailed to you.

Please enter your name:

and Your email address:

Submit Query
Reset
Document: Done

Figure 3. A Snap shot of the fill out form web page.

```
<HTML>
<HEAD>
  <TITLE>Get Information about non-visual browsers</TITLE>
</HEAD>
<BODY>
<P>
Please fill in the following form and the information will
be emailed to
you.
<FORM method="get" action="http://www.cs.uwindsor.ca/cgi-
bin/haddad/test.tcl">
  Please enter your name:
  <INPUT type="text" name="name"><BR>
  and Your email address:
  <INPUT type="text" name="addr">
  <P>
  <BR>
```

```
   <INPUT TYPE=submit>
   <INPUT TYPE=reset>
   <P>
   <A href=test1.htm><IMG ALIGN="Middle" SRC="go_pre-1.gif"
ALT="Previous Document"
       WIDTH="40" HEIGHT="40">Previous</A>
</FORM>
</BODY></HTML>
```

Once the page processor has processed the document, the web document would look something like the following.

```
<p1><s1>Please fill in the following form and the
information will be emailed to you.
<FORM1>
Please enter your name:
<INPUT type="text" name="name"><BR>
and Your email address:
<INPUT type="text" name="addr">
<P>
<BR>
<INPUT TYPE=submit>
<INPUT TYPE=reset>
<P>
<?FORM1>
<link1><Image1>Previous</link1>
```

The browser at this point waits for the user to utter a command, below is a sample session, giving examples of commands the user may ask along with the response from the browser:

Command	Response
Title	The title of this page is, Get Information about non-visual browsers.
List Links	There is one link, it is: 1. Previous document

Set mode	Please specify what mode you would prefer, paragraph, sentence or word?
Sentence	The mode has been set to sentence.
Reed	Please fill in the following form and the information will be emailed to you.
Fill out form 1	Do you want to fill out form 1?
Yes	This form has two text boxes, would you like to edit a text box, clear textboxes, submit the form, or exit form?
Edit text box 1	<beep> Please enter your name
Tarek Haddad (user input from keyboard)	T A R E K H A D D A D. Input has been accepted.
Edit text box 2	<beep> And your email address
Haddad@uwindsor.ca (user input from keyboard)	H A D D A D @ U W I N D S O R Input has been accepted.
Submit	Do you want to submit this form?
No	Submit form command canceled.
Link 1	Do you want to link to hypertext link 1?
No	Link command canceled.

7 SOME COMMENTS ON THE USE OF THE SYSTEM

Seven users have experimented with NVWB. All the users were computer science students or faculty members. they were also all sighted users familiar with the WWW however. none were familiar with the NVWB or any other tool used for non-visual browsing of a web page. Each user was given the web page discussed in the example usage section. which was designed to be universally accessible. and later each user was asked to navigate through the document gathering as much information as they could. Once they were done. each user was asked to explain what they thought the web document contained and their responses were noted. The users were than given a web page that was arbitrarily picked from the NASA web site. the users were then asked to navigate through that document and were later asked what information they gathered from that site. Below are some comments on the user of the NVWB:

- Most users easily identified hypertext links while navigating the document, especially since they were also able to list them all out. One problem however was that in some cases the document may contain many links, and for the user to list all of the links could be very time consuming, and in some cases annoying. Therefore some users suggested that giving the user the option of listing a specific number of hypertext links, starting from one point and ending at another, would be beneficial.

- Most users found the system hard to control at the beginning since they were still unfamiliar with both the command language as well as what response to expect. However around 15-30 minutes later, once the users became familiar with the system, they found the speech recognition to be very accurate and became more familiar with the kinds of responses the system generates.

- In some cases the users found the Dectalk speech synthesizer to be unclear and with low volume. In response to that, the Dectalk was connected to amplified speakers and the speech synthesizer was set to read at a slower pace (initially at 180 words/min to 150 words/min). After these changes the users found Dectalk to be much clearer.

- Even though the images in the NASA web page did not have any alternative text, the users did not appear to have lost any information with respect to the topic discussed in the document.

- In general the users were able to understand the speech synthesizer, however in some cases specific phrases were read out in a different way than a person would read it out. For example the phrase "*World WAR II*", which should be read out as "*World War 2*" was read out by the speech synthesizer as "*World War I I*". Other formats such as email addresses and initials such as SSI may be read out in different ways. It would require the user in this case to interpret the real meaning of the phrase, however in some cases the user became confused and in some cases the user had to read out the phrase more than once to interpret the real meaning of the phrase.

In general it was found that each user was able to obtain a good understanding of the information that was on both web documents. Even though each user took a while to become familiar with the NVWB, once they became familiar with it and with the command language, their understanding of the information on the web document improved. All the users were able to identify the topics in the web pages, in some cases the users were also able to create a mental model of the document that was similar to the actual layout of the document.

It was also noticed that most users did not remember most of the links even though they were read out in a different voice, however this did not affect their navigation, and when they were interested in finding what hypertext links exist in the document they listed them again. It seems that most users only listed hypertext links, bolded/italicized phrases, headers and images only to gather a feeling of what kind of information can be found in the document, and non of the users actually remembered the listings. This demonstrates that all the user needs to have is a mental understanding of what kind of information is in the document, whereas the actual format and layout of it is irrelevant.

8 ADDITIONAL DESIGN PRINCIPLES AND POSSIBLE FUTURE MODIFICATIONS TO NVWB

From the investigation that was carried out. we have noted a few design principles and possible future modification that may improve NVWB.

1. The text sent to the speech synthesizer should be formatted. The speech synthesizer reads out some words in the manner they are spelled instead of in the manner it suppose to be read. for example *"World War II"* is read as *"World War I I"* instead of *"World War 2"*. It is necessary for the NVWB to format those phrases into what they should sound like before it is sent to the speech synthesizer. This. however. would require the NVWB to have some semantic understanding of the document to be able to understand how a phrase should be read out. For the time being this is very hard to accomplish. therefore the user would have to be left with the burden of trying to make sense of unformatted phrases.
2. Some new commands that would allow the user to output only a portion of a list should be added. For example. instead of listing all the links in the document the user should be able to say *"list links 12 until 20"*.
3. The user should be given control over the pace of which the speech synthesizer speaks. It seems that at the start of the testing some users found it to be fast. however with time. as they became familiar with it, the speed stopped bothering them.

9 OTHER TECHNIQUES THAT CAN BE USED IN NON-VISUAL WEB BROWSING

The primary objective of the thesis work described in this report was to construct a content-based non-visual browser that could be used in a subsequent project to investigate the relative merits of content-based browsing and its relationship to other techniques used in non-visual browsing.

Although such an investigation is beyond the scope of this thesis work, it is appropriate to briefly discuss other techniques that can be used in a non-visual interface to the web.

Many of the techniques, other than content-based browsing, that can be used in a non-visual interface to the web fall into two categories: 1) use of non-prompted auditory clues, and 2) creation of a spatial representation of the web page. The relationship of these techniques to the psychological aspects of creating a mental model of a web page is being investigated by Frankie James as part of her doctoral studies at Stanford University [1].

In this section we discuss the use of non-prompted auditory clues in helping the user gain a "feel" for a web page, we then discuss how users can develop a "spatial representation" of a web page. Finally, we summarise some of Frankie James' ideas.

9.1 Non-prompted auditory clues

As a sighted user loads up a web document. some information about the document is provided. For example information on the size of the document is given by viewing the handle of the scroll bar of the browser on the side. the smaller it is the larger the document is, and the larger it is the smaller the document is. This does not inform the user of the exact size of the document. however. it gives the user an idea of how big it is. Another clue that the user gathers is if there are many images in the document or not. Typically a browser would load up the text in the document first. and then move on to load up the images. The longer it takes the browser to load up the images the more (or larger) the images are. Once again, the user is not informed of exactly how many images exist or how large they are, however the user would be able to gather some idea of the richness of images in the document.

These clues however are not available for a visually impaired user. and therefore some method should be adopted to inform the user of clues about the document. With these clues a user would be able to gather an idea about specific characteristics in the document once it has been loaded. Below are a number of suggestions on how auditory clues can enhance a user's mental understanding of a document:

- Once a document has been loaded a sound could be generated to represent the size of the document to the user. The larger the document is the "bigger" the sound, the smaller the document the "smaller" the sound generated. This in result would give a user the ability to recognize approximately how large the document just loaded is.

- Once the document has been loaded. a noise of jingling bells could be generated to represent the density of hypertext links in the document. the more dense they are the more dense the jingling is. the less dense they are the less dense the jingling is.

- As a document is being loaded a tick-tock sound could be generated proportional to the transfer rate of the document. That is as the document is being loaded, the faster the transfer time is the faster the tick-tock sound is. the slower the transfer time is the slower the tick-tock sound is.

Other auditory clues could be provided such as generating sounds for the density of images. lists, tables etc. However a balance between clarity and information should be considered. If too many clues are generated this may confuse the user

When developing a non-visual browser, cognitive factors should always be considered. The information a sighted user is able to gather by viewing or skimming through a document, a visually impaired user is unable to acquire. A combination between spatial organization and auditory clues would improve a user's ability to navigate and better interpret the document.

9.2 Creating a spatial representation of a web page

Visually impaired computer users, in practice, have been found to rely even more heavily on the physical properties of objects than do sighted users [14]. Visually impaired users were found to define a place for everything, order their diskettes, put recognisable cues into documents, etc. A spatial understanding of a document can be done in two different ways:

- One approach would be by using the structured nature of the web document and translating the structure of the document into meaningful earcons. That is, the user using a pointing device (a mouse) can move around the web document. As the pointer crosses specific objects in the web document (for example a table, image or block of text), a specific sound is generated that informs the user of the existence of that object at that location. By moving the mouse around the user would be able to interpret the spatial organisation of the document, therefore retrieving the desired mental model of the actual web document as a sighted user would view it. The advantage of this would be that the user would be able to get an exact mental image of the actual web page. The problem however would be that the user would have to distinguish between the many different sounds that the browser would generate, and the interpretation of each one. It is also more difficult for the user to build up the spatial representation of the document since the user has no prior knowledge of what the document contains.

- Another approach is for the web browser to organise specific objects that are to be found in every web document, such as bookmarks and hypertext links, in a spatial environment [8]. That is the web browser generates, from the content of the web page, a virtual spatial environment, one for example, were the hypertext links can be found to the left of the user, and the list of bookmarks can be found to the right. Below is an illustration of how this virtual environment might be presented.

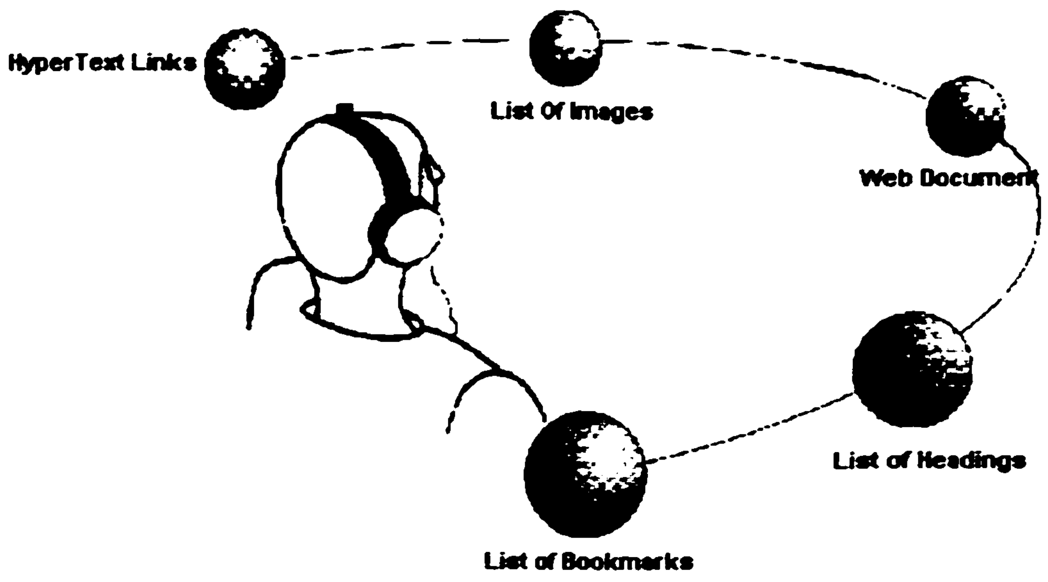


Figure 4 An illustration of the spatial environment of a web document.

The advantage of this approach is that as a user loads up a new document, specific objects can always be found at a known location in a virtual spatial environment, giving the user knowledge of where some information can be retrieved about the document. The disadvantage however is that the user would be unable to construct a mental model of the actual document, and only a virtual spatial environment would be constructed, leaving it up to the user to develop his/her own mental model of the document.

To pick the best approach for including a spatial environment in a web document is difficult, each approach has its advantages and disadvantages. However, it is necessary for a spatial environment to exist, one that would allow a visually impaired user to navigate the document in a better manner.

9.3 Psychological Factors related to non-visual web browsing

When using a non-visual web browser, the pages are presented in audio output. However presenting a document in audio format is not as simple as it sounds, and underway is a study by a Doctoral student, Frankie James at Stanford University (mailto: fjames@cs.stanford.edu , http://www.pcd.stanford/~fjames/frankie.html), to develop a set of guidelines for designing audio interfaces to HTML.

This study compared several different audio presentation styles on twenty-four subjects (twelve blind and twelve sighted). The interfaces for this experiment marked HTML structures with both non-speech sound effects and speaker changes. The interfaces that were used in this experiment were based on four general formats:

- one speaker, few sound effects (OS/V)
- one speaker, many sound effects (OS/MS)
- multiple speakers, few sound effects (MS/V)
- multiple speakers, many sound effects (MS/MS)

Sounds that seemed intuitively related to the structural element that they were meant to represent were chosen. However in the case where there is no obvious sound, a short abstract sound was used. For example, link points were marked with different overlaid sounds indicating whether the link was pointing within the same document (sound of footsteps), to another document (sound of phone ring), or a mailto link (sound of doorbell). Also simple tones that varied in pitch was used to indicate heading level. As for changes in sound, the analogy of a sports broadcast was used in which there is more than one announcer, each having specific role and presenting only certain information.

The following are some of the findings that Frankie James developed from this experiment:

Headings

The users had trouble distinguishing heading levels that were differentiated by pitch even when two headings sounded in succession. Some users also said that the explicit tag was too long or cluttered the presentation.

Link Points

The users found the interfaces that marked links with natural sounds (OS/MS and MS/MS) were significantly more effective than those that used simple tones (OS/V and MS/V). The users stated that they had difficulty discerning the extent of the link text in the verbose protocols because the tone in the verbose protocols sounded either at the beginning or the end of the link text.

Lists

Lists produced different results for the blind and sighted users. Blind users had more correct answers when using OS/MS even though it was rated as both too loud and too slow. Sighted users, however, had more correct answers using MS/V. MS/V both separated list levels by speaker and marked list items with a short audio bullet.

Pauses

Pauses also generated different results with the blind and sighted users. The blind users found the pauses to be to long and the presentation too slow, probably because they were accustomed to using audio to retrieve information. Sighted users on the other hand found the pauses too short and the presentation too fast.

Volume

The users were given full control over the volume, and most of them found that OS/MS was significantly louder than the other interfaces. This could be because certain sound effects were too loud in comparison with the rest of the interface and, thus, distracting.

Even though the totals were not significant, ten users found one speaker with minimal sound effects (OS/V) to be most effective. Second choice was multiple speakers with minimal sound effects (MS/V) receiving five votes, whereas the two interfaces using multiple sound effects each got three votes.

9.4 Integrating different approaches in order to construct an ideal non-visual web browser

NVWB is a content-based browser. It does not use non-prompted auditory clues, nor does it provide the user with a spatial representation of web pages. It seems reasonable that the capability of NVWB would be enhanced if these features were added. However, it is not obvious how these features should be integrated with content-based browsing. The small investigation of the use of NVWB has identified a number of areas for improvement, which have already been discussed. However, this investigation has not really helped determine how best to integrate auditory clues and/or spatial representation into a content-based non-visual browser.

Although a detailed investigation of these issues is beyond the scope of this thesis work, it is appropriate to make a few suggestions that may form the basis of a subsequent study.

Suggestions for the design of an ideal non-visual browser:

1) Content-based browsing should form the "backbone" of the non-visual browser. Other features should be regarded as augmenting content-based browsing. This suggestion is based on the assumption that in most cases users are primarily interested in the information stored in a web page rather than in the way that information is presented. The presentation can help a user interpret the content of a page (e.g. putting text in bold font indicates an emphasized importance of the content of that text). However, a sophisticated content-based browser should be capable of carrying out the same

interpretation. Rather than telling the user that text is in bold font, the browser should be able too indicate that the emboldened text is of particular importance.

2) The non-visual browser should be capable of providing the user with a feel for a web page without any prompts from the user. When the page is being loaded, there should be some method to let the user know what the title is, how big the page is, how many links there are, etc. Ideally, the browser should be able to tell the user what the page "is about". This last feature would require that the browser be capable of interpreting and summarizing the content of the web page. Techniques from text retrieval, natural language processing and artificial intelligence would have to be used to provide this capability.

3) The non-visual browser should provide a mechanism by which the user can get a feel for the intended spatial layout of the page, as it would appear when presented by a conventional browser. This could involve the use of a haptic device, which allows the user to "feel" the structure of the page, for example the haptic device may exert pressure on the user's hand as the user crosses over the boarder of a window. Or it could involve something similar to screen-reader technology, or alternatively it could involve spoken output describing the layout of the page.

4) In addition to providing the user with a feel for the intended spatial layout of the page, the browser should also provide the user with a generic spatial representation. That is, the content of all pages should be organized in some generic way that can help the

user develop a mental model of the content of the page. For example, the generic spatial model might be such that all hyper-links are "situated" at the top-right hand corner of a virtual spatial layout. This suggestion is based on an assumption that may be incorrect, that a spatial representation of a page can help users organize and understand that material on a page. Various people have suggested this, however, there is as this time no strong supporting evidence.

5) It should be possible to turn features on and off to cater for different users and different applications. For example, the generic virtual spatial representation would not be required (and would probably be confusing) if a representation of the intended spatial layout were available. As another example, in some applications it may be appropriate for the user to interact with the computer only through voice commands. In this case, a haptic device would not be appropriate, neither would the use of screen reading technology that was mouse-based.

The construction of an ideal non-visual web browser is a non-trivial task and will involve the development and investigation of various combinations of features. A number of projects that might ultimately result in an ideal browser are listed in the next section.

10 CONCLUDING COMMENTS

A comprehensive survey of speech-recognition technology and its applications [5] has been carried out in addition to a review of existing non-visual web browsers. It was found that no ideal non-visual web browser has yet been constructed. In particular, little work has been carried out on the development of speech-in/speech-out content-based non-visual browsers. Consequently, it was decided to construct such a browser in order to provide an environment for further investigation in future projects.

A non-visual web browser NVWB has been built. It can handle most HTML constructs with the exception of tables, frames and JavaScript. The browser runs in real time providing users with fast response to browser commands. These commands are issued in a sub-set of natural language through a user-independent continuous-speech recognition interface. Output is provided through realistic voice output. The system runs on a "standard" PC equipped with a speech card and a DECTALK speech synthesizer.

Design of NVWB took into consideration various guidelines that have been suggested by various groups. Not all guidelines were followed as some proved to be inappropriate in practice.

Although it was beyond the scope of the thesis work to conduct a comprehensive investigation of the use of NVWB and the relative advantages/disadvantages of content-based browsing, a small investigation of the use of NVWB was conducted. A number of shortcomings were identified and proposed modification of the guidelines for constructing non-visual browsers have been suggested.

The following future research projects are suggested as continuation of this project:

1) A thorough investigation of the use of NVWB and content-based browsing should be performed involving non-sighted users. A graduate Social Science, Communications Studies or Psychology student might carry this out. A Psychology student would be able to discover how a user can better perceive information represented to the user by using different sounds, voices etc. A Communication Studies student would be able to study how to better communicate information to the user to allow the user to better understand the document.

2) Extension of NVWB to include non-prompted auditory clues and spatial representation. A graduate Computer Science student might carry out this project.

3) An investigation of the relationship between content-based browsing, non-prompted auditory clues, and spatial representation. This could be a combined project involving a student from each of Computer Science and Communication Studies (or Psychology). The extended NVWB could be modified so that it can function in various modes, with features being turned on and off during the investigation.

4) The construction of a robust "ideal" non-visual web browser.

APPENDIX A - NVWB INPUT/OUTPUT

The following is the initial grammar that the browser loads up.

```
S -> title
     |list{links |images |bolded obj1 |italicized obj1 |headings}
     | how many OBJECT | headings
     | link  { Numbers1 | Numbers1 Numbers}
     | reed
     | set mode
     | forward
     | backwards
     | previous
     | next
     | go to { Numbers1 | Numbers1 Numbers}
     | skip { Numbers1 | Numbers1 Numbers}
     | spell
     | enter location
     | fill out form { Numbers1 | Numbers1 Numbers }

OBJECT ==links | lists | images | bolded words | italicized words

Obj1== words | phrases

NUMBERS1 -> one_to_9

one_to_9  == 1 2 3 4 5 6 7 8 9

NUMBERS -> 0_to_9

0_to_9  == 0 1 2 3 4 5 6 7 8 9
```

The following are examples of the commands in this grammar, together with the corresponding output from the browser:

Title

This command requests from the browser the title of the document. The title is the text that is found between the **<title>...</title>** tags in the HTML document. The browser as a response would return the following:

"The title of this page is, TITLE"

In the case the document has no title the response would be:

"This page has no title."

How many links

This command requests from the browser the number of links found in the document. A link in an HTML document is the text that is found between the **<A >.... </A>**. The browser would respond with the number of links in the document, as follows:

"There are <NUMBER OF LINKS> ***Links."***

Or in the case were there is only one link the response would be:

"There is one Link."

Or in the case were there are no links the response would be:

"There are no Links."

How many lists

This command requests from the browser the number of lists found in the document. A list in an HTML document is the text that is found between the **<UL> ... </UL>** tags. The browser would respond with the number of lists in the document, as follows:

"There are <NUMBER OF LISTS> ***Lists."***

Or in the case were there is only one list the response would be:

"There is one List."

Or in the case were there are no lists the response would be:

"There are no Lists."

How many images

This command requests from the browser the number of images found in the document. An image in an HTML document can be found in <IMG..> tag. The browser would respond with the number of images in the document, as follows:

"There are <NUMBER OF IMAGES> ***images."***

Or in the case were there is only one image the response would be:

"There is one image."

Or in the case were there are no images the response would be:

"There are no images."

How many bolded phrases/words

This command requests from the browser the number of bolded phrases found in the document. A bolded phrase in an HTML document is the text that is found between the <B> ... </B> tags. The browser would respond with the number of bolded phrases in the document, as follows:

"There are <NUMBER OF BOLDED PHRASES> ***bolded phrases."***

Or in the case were there is only one bolded phrases, the response would be:

"There is one bolded phrase."

Or in the case were there are no bolded phrases, the response would be:

"There are no bolded phrases."

How many italicized phrase/words

This command requests from the browser the number of italicized phrases found in the document. An italicized phrase in an HTML document is a phrase that is found between the <i> ... </i> tags. The browser would respond with the number of italicized phrases in the document, as follows:

"There are <NUMBER OF ITALICIZED PHRASES> ***italicized phrases."***

Or in the case were there is only one italicized phrase the response would be:

"There is one italicized phrase."

Or in the case were there are no italicized phrases the response would be:

"There are no italicized phrases."

How many headings

This command requests from the browser the number of headings found in the document. A heading in an HTML document is a phrase that is found between the **<Hx> ... </Hx>** tags, were **x** is a value representing the type of the heading. The browser would respond with the number of headings in the document, as follows:

"There are <NUMBER OF HEADINGS> ***headings."***

Or in the case were there is only one heading the response would be:

"There is one heading."

Or in the case were there are no headings the response would be:

"There are no headings."

How many paragraphs

This command requests from the browser the number of paragraphs found in the document. Text in an HTML document is considered a new paragraph based on a couple of different situations (discussed in section ***7.3***). The browser would respond with the number of paragraphs in the document, as follows:

"There are <NUMBER OF PARAGRAPHS> ***paragraphs."***

Or in the case were there is only one paragraph the response would be:

"There is one paragraph."

Or in the case were there are no paragraphs the response would be:

"There are no paragraphs."

How many sentences

This command requests from the browser the number of sentences found in the document. Text in an HTML document is considered a new sentence based on a couple of different situations (discussed in section ***7.3***). The browser would respond with the number of sentences in the document, as follows:

"There are <NUMBER OF SENTENCES> ***sentences."***

Or in the case were there is only one sentence the response would be:

"There is one sentence."

Or in the case were there are no sentences the response would be:

"There are no sentences."

List links

This command requests a list of all the hypertext anchors in the document. As a response each hypertext anchor in the document is returned (that is the text that is between the **<a..>...</a>** tags), preceding each text anchor a number is attached (this number is uttered by Dectalk in a different voice then the rest of the text). The user can use this number to reference that specific anchor., the response would be as follows:

"There are <NUMBER OF LINKS (N)> ***Links in this page, They are:***

3. <first hypertext anchor>

4. <second hypertext anchor>

....

O. <nth hypertext anchor>*"*

Or in the case were there is only one hypertext anchor in the document, the response would be:

"There is one Links in this page, it is:

1. <first hypertext anchor>

Or in the case were there are no hypertext anchors in the document, the response would be:

"There are no links in this page."

List images

This command requests a list of all the images in the document. As a response each alternative text of each image in the document is returned (that is the text that is found in the alt attribute in the **<img...>** tag). The response would be as follows:

"There are <NUMBER OF IMAGES (N)> ***images in this page, They are:***

3. <first image's alt text>

4. <second image's alt text>

....

O. <nth image's alt text>*"*

Or in the case were there is only one image in the document, the response would be:

"There is one images in this page, it is:

1. <first image's alt text>

Or in the case were there are no images in the document, the response would be:

"There are no images in this page."

List italicized phrases/words

This command requests a list of all the italicized phrases in the document. As a response each italicized phrase in the document is returned (that is the text that is found is between the **<I>...</I>** tags). The response would be as follows:

"There are <NUMBER OF ITALICIZED PHRASES (N)> ***italicized phrases,***

They are:

3. <first italicized phrase>

4. <second italicized phrase>

....

O. <nth italicized phrase>*"*

Or in the case were there is only one italicized phrase in the document, the response would be:

"There is one italicized phrase, it is:

1. <first italicized phrase>

Or in the case were there are no italicized phrases in the document, the response would be:

"There are no italicized phrases."

List bolded phrases/words

This command requests a list of all the bolded phrases in the document. As a response each bolded phrase in the document is returned (that is the text that is found is between the <b>...</b> tags). The response would be as follows:

"There are <NUMBER OF BOLDED PHRASES (N)> ***bolded phrases, They are:***

3. <first bolded phrase>

4. <second bolded phrase>

....

O. <nth bolded phrase>"

Or in the case were there is only one bolded phrase in the document, the response would be:

"There is one bolded phrase, it is:

1. <first bolded phrase>

Or in the case were there are no bolded phrases in the document, the response would be:

"There are no bolded phrases."

List headings

This command requests a list of all the headings in the document. As a response each heading in the document is returned (that is the text that is found is between the <H>...</H> tags). The response would be as follows:

"There are <NUMBER OF HEADINGS (N)> ***headings, They are:***

3. <first heading>

4. <second heading>

....

O. <nth heading>"

Or in the case were there is only one heading in the document, the response would be:

"There is one heading, it is:

1. <first heading>

Or in the case were there are no headings in the document. the response would be:

"There are no headings."

The next few commands (set mode, reed, next, previous, go to (n), skip (n) and spell) all depend on what mode the browser is currently set at. The browser can be set to either word, sentence or paragraph mode. By default it is set to sentence at startup. Word mode reads a word at a time, sentence mode reads a sentence at a time, and paragraph mode reads a paragraph at a time. The boundaries, however, of word mode is the sentence the browser is currently at, and therefore the user is not able to go through the whole document with the mode set to word.

Set mode

This command allows the user to choose the mode in which the browser would move navigate through the document. Once the user issues this command the following grammar is loaded in:

```
S -> paragraph
        | sentence
        | word
```

and the following message is outputted :

"Please specify what mode you would prefer, paragraph, sentence, or word?"

At this point the user is able to utter one of the following commands:

1. Paragraph

This command sets the mode of the document to paragraph. and the paragraph of which the browser sets to is the one that contains the sentence the user is currently at. That is, if the user was at *sentence 3* and *sentence 3* is in *paragraph 2* then the browser would set to *paragraph 2*.

2. Sentence

This command sets the mode of the document to sentence. If the previous mode was paragraph then the browser sets the sentence to the first sentence in the paragraph that the user was currently set at. That is, if the user was set at *paragraph 3*, and *sentence 7* is the first sentence in *paragraph 3* then the browser would set to *sentence 7*. Otherwise if the previous mode was word, then no changes occur.

3. Word

This command sets the mode of the document to word. If the previous mode was paragraph then the browser sets the sentence to the first sentence in the paragraph that the user was currently set at and sets the word to the first word in that sentence. Otherwise if the previous mode was sentence, then the browser simple sets the word to the first word in the current sentence.

Once the user has made a choice and the proper mode has been set the browser responds with:

"The mode has been set to <CHOSEN MODE> "

and the grammar for the main menu is reloaded.

Reed

This word is not misspelled, the reason for it being spelled that way is because the speech recognizer recognizes the word ***reed*** better than it does the word ***read.*** This command simply reads out text from the document. If the mode were set to word, the word the browser is currently pointing at would be read out. If the mode were set to sentence, the whole sentence the browser is pointing at would be read out. If the mode is set to paragraph, the whole paragraph that the browser is pointing at is read out. As the text is being read out it at the same time is formatted (discussed in section PPP), this formatting helps the user distinguish between different objects in the document, for example a hypertext link, an image, a list etc..

Next

This command is much like the ***Reed*** command with one difference, the browser moves to the next block of text and reads that out. For example if the mode was set to sentence and the browser was currently at *sentence 2*, a *Next* command would issue the browser to move to *sentence 3* and read out *sentence 3*. The following is an explanation of what occurs at each different mode:

Word

The boundaries for the mode word are the sentence that the browser is currently at. If a user issues a Next command the response wouid simply be the next word at which the browser is at. In the case were the user is at the last word in the sentence and the *Next* command is issued, the following response is returned:

"You are currently at the last word in the sentence."

Sentence

The boundaries for the mode sentence are the document. If a user issues a Next command the response would simply be the next sentence at which the browser is at. In the case were the user is at the last sentence in the document and the *Next* command is issued, the following response is returned:

"You are currently at the last sentence in the document."

Paragraph

The boundaries for the mode paragraph are the document. If a user issues a Next command the response would simply be the next paragraph at which the browser is. In the case were the user is at the last paragraph in the document and the *Next* command is issued, the following response is returned:

"You are currently at the last paragraph in the document."

Previous

This command is similar to the next command, but instead of reading th next block of text the browser is at, the browser reads the previous block of text (word, sentence or paragraph). The boundaries for the word mode is a sentence and in the case the user is at the first word in the sentence when this command is issued, the following response is returned:

"You are currently at the first word in the sentence."

The boundaries for the mode sentence is the document, and in the case this command was issued while the browser is at the first sentence in the document the following response is returned:

"You are currently at the first sentence in the document."

The boundaries for the mode paragraph is the document, and in the case this command was issued while the browser is at the first paragraph in the document the following response is returned:

"You are currently at the first paragraph in the document."

Spell

This command will not work if the user is currently at mode sentence or paragraph, the user needs to be in mode word, at this point the browser simply responds with the spelling of the word at which it is currently at. For example if the browser is currently at the word ***"the"*** the response would be ***"TH E"***. If the user was set to mode sentence or paragraph the response would be:

"You are currently on mode sentence"

Or ***"You are currently on mode paragraph"***

These next commands (skip, go to, back, forward, link (N), enter location, fill out form (N)) all require the user to confirm the action to be taken. For example if a user utters the command **back** the browser would ask the user: **"Do you want to go back to the previous document?"** and at the same time the following grammar loads up:

```
s -> yes | no
```

This grammar constrains the user to answering **"yes"** or **"no"**.

Backwards

Just like most other browsers this is an option for the user to go back to the previous document that the user has already visited. Once the user utters this command the browser confirms the action by asking:

"Do you want to go back to the previous document?"

At this point the user is expected to answer **"yes"** or **"no"**, if the user responds with a **"no"** then no change occurs, if the user responds with a **"yes"** then the previous document is loaded into the browser, if however there is no previous document the browser responds with the following message:

"This is the first document."

Forward

Just like the **back** option this is an option for the user to go forward to the next document that the user has already visited. Once the user utters this command the browser confirms the action by asking:

"Do you want to go forward to the next document?"

At this point the user is expected to answer **"yes"** or **"no"**, if the user responds with a **"no"** then no change occurs, if the user responds with a **"yes"** then the next document is loaded into the browser, if however there is no document following the document the user is at, the browser responds with the following message:

"This is the last document."

Enter Location

For the user to type in a URL this would be the command to issue. As in any regular browser, the user can type in a URL location of a document to be retrieved. Once the user utters this command, the browser confirms the action by asking:

"Do you want to enter a U R L Location?"

If the user responds with a **"no"**, no change occurs. If the user however responds with a "yes" the browser sets focus to a text area were the user is to type in the new URL and the following message is given:

"Type in your U R L Location then press return."

As the user presses a key on the keyboard, the speech synthesizer utters each key, this is to inform the user of what keys are being pressed in case a mistake occurs. In general most visually impaired users are able to use the keyboard, therefore even though some limited input is required by the user while using this browser, it should not effect the user friendliness of the system. Once the user has typed in the URL the user is expected to press return, at this point the document referred to is retrieved by the browser.

Skip [1..99]

This command informs the browser to skip a specified number of blocks (word, sentence or paragraph) of text. When a user issues a ***"skip n"*** command, the browser first confirms this action by asking:

"Do you want to skip N <BLOCK>***s"***

Were N is the number the user utters to skip, and <BLOCK> is either sentence, word or paragraph. If the user responds with a **"no"**, no change occurs. If the user responds with a **"yes"** then according to the state the mode is set at a different action occurs.

If the mode is set to word, a **"skip n"** command would skip **n** number of words from the word the browser is currently set at. If the number of words to skip points to a word out of the sentence the browser is currently set at, the following message is resulted:

"There are only m words in this sentence."

Were ***m*** represents the number of words in the sentence that the browser is currently set at.

If the mode is set to sentence, a "**skip n**" command would skip **n** number of sentences from the sentence the browser is currently set at. If the number of sentences to skip points to a sentence that exceeds the last sentence in the document, the following message is resulted:

"There are only m sentences in this document."

Were ***m*** represents the number of sentences in the document.

If the mode is set to paragraph, a "**skip n**" command would skip **n** number of paragraphs from the paragraph the browser is currently set at. If the number of paragraphs to skip points to a paragraph that exceeds the last paragraph in the document, the following message is resulted:

"There are only m paragraphs in this document."

Were ***m*** represents the number of paragraphs in the document.

Once the skip command has been executed the browser than reloads in the main grammar.

Go to [1..99]

This command informs the browser to go to a specified block (word, sentence or paragraph) of text in the document. When a user issues a ***"go to n"*** command, the browser first confirms this action by asking:

"Do you want to go to <BLOCK> N***"***

Were **N** is the number the user utters to go to. and <BLOCK> is sentence. word or paragraph. If the user responds with a "**no**". no change occurs. If the user responds with a "**yes**" then according to the state the mode is set at. a different action occurs.

If the mode is set to word, a "**go to n**" command would go to word **n** in the sentence n the browser is currently set at. If the number of word to go to points to a word out of the sentence the browser is currently set at, the following message is resulted:

"There are only m words in this sentence."

Were ***m*** represents the number of words in the sentence that the browser is currently set at.

If the mode is set to sentence, a "**go to n**" command would go to sentence **n** in the document. If the sentence to go to points to a sentence that exceeds the last sentence in the document, the following message is resulted:

"There are only m sentences in this document."

Were ***m*** represents the number of sentences in the document.

If the mode is set to paragraph, a "**go to n**" command would go to paragraph **n** in the document. If the paragraph to go to points to a paragraph that exceeds the last paragraph in the document, the following message is resulted:

"There are only m paragraphs in this document."

Were ***m*** represents the number of paragraphs in the document.

Once the go to command has been executed the browser than reloads in the main grammar.

Link [1..99]

This command informs the browser to link to a specific text anchor by simply referencing it according to its occurrence in the document. For example the command **"Link 3"**, would force the browser to load up the HTML document referred to by the third text anchor in the document. The document that the browser links to is the one referenced by the specified text anchor in the **<a..>...</a>** under the **href** attribute. When a user utters this command, the browser asks the user to confirm the action by asking:

"Do you want to link to <N>?"

Where ***<N>*** is the specific text anchor to link to. If the user answers ***"no"*** then no change occurs. If the user answers ***"yes"***. The browser would retrieve the document from that text anchor by looking up it's URL e.g. text anchor three would refer to the third text anchor found in the document, the browser would lookup the URL of the anchor and would retrieve the document.

Fill out form [1..99]

For a user to fill out a specific form in the document, the following command has to be issued. The number given represents which form to fill out. For the sake of testing forms, not all items that can be inserted into a form have been covered, only text boxes (were the input attribute is defined as text) is considered. Once the user issues a fill out form command a confirmation message is given:

"Do you want to fill out form <N>?"

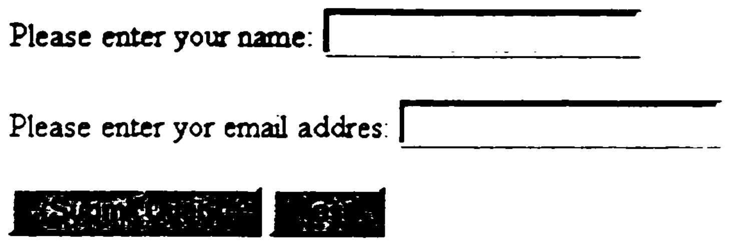
If the user responds "no" then no change occurs. However if the user responds "yes" then the following message is given:

"This form has <N> text boxes, would you like to edit a text box, clear text boxes, submit the form, or exit form?"

Were <N> is the number of text boxes available to be filled out by the user. At this point the following grammar is loaded:

```
s -> edit text box { Numbers1 | Numbers1 Numbers}
          | submit (form)
          | clear (text)  (boxes)
          | reset
          | exit form

NUMBERS1 -> one_to_9

one_to_9  == 1 2 3 4 5 6 7 8 9

NUMBERS -> 0_to_9

0_to_9  == 0 1 2 3 4 5 6 7 8 9
```

At this point the user can issue only commands that are related to the form, below is an example of a form that may appear in a web document and a explanation of the commands that may be used to access a form.



Please enter your name:

Please enter yor email addres:

Figure 5. A snap shot of a HTML form.

Edit text box [1..99]

This option allows the user to fill out one of the available text boxes in the form. The user specifies what text box is to be filled. for example in the case of the form above a user may utter ***"edit text box 2"***. the browser would than respond by saying ***"Please enter your email address: "*** followed by a beep. At this point the user is expected to type in. using the keyboard. the input the user wants to be inserted in this text box followed by the return key. Once the user has pressed the return key the browser responds with ***"Input has been accepted"*** and the value of this text box would be stored as the input the user gave.

Clear/Reset

Clear or ***Reset*** simply clears all the input the user has entered into the text boxes so far. that is if the user entered some text into text box 2. and the user later issues a clear command, the value of text box 2 would be deleted. This is similar to the reset button that is available in most forms which clears all the text boxes.

Submit

For the sake of testing, the browser that was implemented is only able to submit forms that are of type GET. Once the user utters this command, the browser would build up a string containing the information in the text boxes in the necessary format, e.g. for the example above the string generated could look something like the following:

name=Tarek+Haddad&address=haddad@uwindsor.ca

This string would be attached to the action that is specified in the HTML document with the ***"?"*** symbol separating the action part from the form input part. The browsers than executes the action along with the attached input string, as a response the server returns a new web document.

Exit form

This command simply exits the form the user is presently at and loads up the main grammar. No other changes occur and all the text the user typed in the text boxes is lost.

REFERENCES

[1] Frankie James. "Presenting HTML Structure in Audio: User Satisfaction with Audio Hypertext". ICAD '96 Proceedings. November 1996, p.97-103.

[2] Mike Paciello, "Making the Web Accessible for the Blind and Visually Impaired". http://www.webable.com/.

[3] Paul Fontaine, "Writing Accessible HTML Documents", http://www.webable.com/, June 1995.

[4] Markku Hakkinen and John De Witt, "pwWebSpeak: User Interface Design of an Accessible Web Browser". http://www.prodworks.com/

[5] Tarek Haddad, "Speech Recognition Technology And Its Applications", http://www.cs.uwindsor.ca/users/h/haddad/510/

[6] Daniel Tyman, "What You Say Is What You Get!", PCWorld January 1995.

[7] Michael Albers , "Auditory Cues for Browsing, Surfing, and Navigating", ICAD '96 Proceedings, http://www.santafe.edu/~icad/ICAD96/proc96/.

[8] Kai, Crispien, Klaus Fellbaum, Anthony Savidis, and Constantine Stephanidis, "A 3-D Auditory Environment for Hierarchical Navigation in Non-Visual Interaction", ICAD '96 Proceedings, http://www.santafe.edu/~icad/ICAD96/proc96/.

[9] "An Introduction To Speech Recognition", Philips Speech Processing 1997, http://muck2.piro.net:100/.

[10] Richard D. Peacocke and Daryl H. Graf, "An Introduction to Speech and Speaker Recognition ," IEEE August 1990, p26-50.

[11] Vanderheiden, G.C. "Design of HTML Pages to Promote Accessibility to Users with Disabilities: Strategies for Today and Tomorrow. Version 6.0", http://www.webable.com/

[12] G. Vanderheiden, W. Chisholm and N. Ewers, "Making Screen Readers Work More Effectively on the Web", http://www.webable.com/.

[13] Enrique Padilla, "Senior Technician on Voice Recognition Technology", http://www.voicerecognition.com/ , January 1997.

[14] Dufresne. A.. Martial. O., Ramstein. C. and Mabilleau. P. "Sound. Space. and Metaphor: Multimodal Access to Windows for Blind Users". ICAD '96 Proceedings. http://www.santafe.edu/~icad/ICAD96/proc96/.

VITA AUCTORIS

NAME:	Tarek El-Haddad
PLACE OF BIRTH	Cairo, Egypt
YEAR OF BIRTH	1971
EDUCATION	New English School, Kuwait 1976-1988
	University of Kuwait, Kuwait 1988-1990
	University of Windsor, Windsor, Ontario 1992-1994 B.CS.
	University of Windsor, Windsor, Ontario 1994-1997 M.Sc.

IMAGE EVALUATION
TEST TARGET (QA-3)