EFFICIENT NAVIGATION ON THE WORLD WIDE WEB FOR THE PHYSICALLY DISABLED

Leo Spalteholz, Kin Fun Li
Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada

Nigel Livingston
University of Victoria Assistive Technology Team, University of Victoria, Victoria, BC, Canada

Keywords: Web accessibility, single switch input, web navigation, physical disability.

Abstract: One of the major obstacles with current web access solutions for those with physical disabilities is the efficient selection of links and other web page elements. This is especially so for users of single switches. Current solutions simply build a list of the selectable web page elements and use a linear scan to iterate through them, selecting the highlighted link when the user activates an input device. We propose a new method based on incremental searching of the link text to select elements. This approach, paired with well-established switch typing solutions, allows a single switch user to select any element on a web page by typing very few (most often only one) letters.

1 INTRODUCTION AND MOTIVATION

For many people with limited dexterity or severe physical disabilities, interaction with a computer requires a specialized input device. The choice of which input device to use is highly dependent on the users' level of physical control. For example, for those at the advanced stage of Amyotrophic Lateral Sclerosis (ALS), eye muscles are often among the last to deteriorate, making the use of an eye tracking system suitable to control the computer mouse pointer. For other disabilities, such as muscular dystrophy, one or more hardware switches, activated by any body part capable of consistent, voluntary movement, may be used along with specialized software to control the computer. For the severely disabled, a single switch – although it is a low bandwidth input – is a widely used control device due to its simplicity and economy. A switch may take the form of a physical button, a relay activated by a puff of air through a tube, or a switch activated by the electrical signals produced by muscle cells as in electromyography (Prinz et al., 2006). The problem of how to produce text with a single input has been thoroughly researched, with the first typing devices for single switch users being proposed almost 30 years ago (Rowell et al., 1978).

While typing text is still a key component of computer access, efficient access to the information on the world wide web is becoming central to the use of computers. Especially for those with mobility impairments, the world wide web represents an exciting opportunity to move beyond the limitations they may face when interacting with people in the outside world. To that end, it is critical that the world wide web be accessible for those with physical impairments. Although standards for accessible web pages are an important component of the overall effort towards universal accessibility, for example the W3C Web Accessibility Initiative (w3.org, 2006), the reality is that the majority of web pages are not developed with accessibility in mind and this statistic is not improving (Hackett et al., 2004). The current state of accessibility support on the world wide web requires tools that can handle any web site, regardless of its structure or adherence to standards.

From the authors' experience with users of low bandwidth input devices as well as with their aides (University of Victoria Assistive Technology Team, UVATT.org), it is apparent that although there are many people that use their devices for computer control or text input, the web has remained largely inaccessible to them. Judging from requests received for assistance in accessing the web by users with a wide range of disabilities, we can conclude that an effective web access method would be useful to many. As there is no lack of either potential users or demand for a usable web access system, the fact that it is so rarely seen in practice may be partially due to the inadequate efficiency of currently available solutions. In our experience testing various accessible web browsing systems, the large number of links and the complex layout in modern web pages makes navigation unacceptably slow and cumbersome.

To address this issue, we propose a new technique for selecting web page elements for bandwidth limited input devices. This method will be a key component in a customized browser designed to be used with alternative, low bandwidth input devices. Section 2 discusses some of the previous work in this area, with particular emphasis on the linear scanning selection method (Section 2.1) for selecting elements as employed by many previous approaches. The major components of our proposed system are detailed in subsections of Section 3, including specifics of the implementation.

2 BACKGROUND SURVEY

One of the cornerstones of computer use has always been, and continues to be entering text into the computer. This has prompted a wealth of work on creating efficient typing interfaces that can be operated with a single input signal such as that generated by an alternative input device. A common approach to this typing interface has remained relatively constant from its inception, with letters being selected from a grid by automatic scanning and typed upon activation of the input device. The most efficient implementations use a divide and conquer approach by dividing the available characters into groups and narrowing down the selection on each switch activation. Commonly found in commercial switch scanning products (madentec.com, 2006) is the row-column scanning approach where letters are displayed in a two dimensional matrix and scanned through first by row, and then, upon switch activation, by column until the desired letter is selected (Steriadis and Constantinou, 2003).

Conversely, the primary method for navigating on the world wide web has traditionally been by “point and click”. This method is not easily adaptable to single input devices that are not capable of accurate two dimensional input. One approach to solving this problem is to use several on-screen buttons to move the mouse pointer on the screen. For single switch users, this is usually done by presenting a button for each direction of mouse movement (x and y, as well as diagonal) and then scanning through the buttons. While this method is a valuable fallback for applications that are not accessible via the keyboard, it is not an efficient interface for continuous use, and requires precise timing control in switch activations from the user. The inefficiency of these pointer control schemes has prompted research into other ways to navigate on the world wide web.

There are two notable attempts at creating a web browser specifically accessible to single switch users, the AVANTI browser (Stephanidis et al., 1997) and the MultiWeb browser (Owens and Keller, 2000). Both of these approaches use a linear scanning approach to link selection, with some differences in how the link scanning is presented to the user. In addition, (Mankoff et al., 2002) developed a web browser accessible to users capable of operating four separate switches using a similar linear scanning approach, with focus being advanced by switch activation instead of automatically on a timeout. The linear scanning technique is discussed more thoroughly in Section 2.1. Hanson et al. with their accessibilityWorks project, while focusing more on adapting the appearance of web pages, also consider the issue of control by devices with zero or one dimensional input (Hanson et al., 2005). Similar to other work in the area, they propose to extract the list of links and place them in an external list such that they can be selected by a user incapable of two dimensional cursor control. The Gnome On Screen Keyboard (gok.ca, 2006) implements a similar link extraction scheme for web browsers that can expose web page structure to an accessibility framework (Zhao et al., 2005).

In more mainstream web browsers, two alternate approaches to linearly iterating through links on a web page have emerged: incremental search and automatic access keys. Mozilla Firefox implements a feature called “find as you type”, which incrementally searches a web page and can highlight matching links as one types a search query (mozilla.org, 2006a). If only one link matches a search, it can be navigated to by pressing the “enter” button on the keyboard. This allows the user to select text and image links, but does not allow focusing of text entry boxes, clicking of form buttons, or selecting other form input fields. Another approach is automatic access keys, as implemented in the Konqueror (konqueror.org, 2006) web browser on the Linux platform. The automatic access keys approach assigns a unique alphanumeric character to each selectable element on a web page, and shows these characters with a tooltip over each element. Typing an assigned letter will select the corresponding element. Currently, the system only assigns the 36 alphanumeric characters, meaning that any web pages with more than 36 elements will not be accessible.

2.1 Linear Scanning Selection

The most common approach to web page element selection for switch specific browsers is to construct a list of selectable elements in the current web page, and then linearly scan through this list. There are two general approaches to scanning through selectable elements: in-place and external scanning. In-place scanning, as used by the MultiWeb browser, as well as all major mainstream browsers, scans through the list of selectable elements on the page by highlighting them within the original web page layout. The external scanning approach works by extracting all the selectable elements in the page and displaying them in a list beside the web page. This list is then scanned through top to bottom, and a link can be navigated to by activating the switch when the target is highlighted.

While the linear scanning approach is a logical extension of the scanning used in switch typing interfaces, it has some key disadvantages that limit its efficacy for navigating the web. In-place scanning has the advantage of lending context to links with non-descriptive names. Although it is considered bad practice in the field of web development, many hypertext links are given names such as “link” or “click here” which are only meaningful in the context of the surrounding text. If these types of links are extracted from the web page and scanned through in an external list, it becomes very difficult to determine which link is the desired target. Another disadvantage of the in-place scanning technique is that the scanning order can be unpredictable. Since the user must quickly activate the switch when his/her element is selected, it is crucial that the scanning order be predictable. Although scanning elements in a top to bottom order is logical for simple web pages, sites with a complicated layout can make element scanning very difficult to follow.

The primary disadvantage of both linear scanning approaches is their lack of scalability to web pages with many selectable elements. It is very common for web pages, especially those of news networks, search engine results pages, or site maps to contain many tens or even hundreds of links. Linearly scanning through large numbers of elements to find a desired target is a very tedious process. The external scanning approach gives more flexibility in grouping the list of links and selecting them via a divide and conquer approach to improve performance, but is still difficult to present to the user in an intuitive way for large numbers of links. The error cost of the linear scanning approach is also high, as a missed target would, in the best case, involve having to wait for the scan to loop around, and in the worst case result in a click on an incorrect link, necessitating multiple steps to return to the previous page.

3 PROPOSED SYSTEM OVERVIEW

Our proposed system is designed as an extension to the open source Mozilla Firefox web browser (Section 3.3), adding a more efficient text based navigation method for low bandwidth input. This system will facilitate the activation of selectable elements on a web page, such as hypertext links, text entry boxes, form buttons, or image maps. After a web page is loaded, the label builder (Section 3.2) constructs a textual label for each selectable element on the page. To select an element, the user employs a text entry interface (Section 4) to type the starting letters of his/her desired element’s label. After each letter, the selectable elements on the page are searched and highlighted according to the algorithm described in Section 3.1. When only a single element matches the entered query, the user is prompted to activate (navigate to) the selected element. Figure 1 depicts an overview of the major components of the system.

3.1 Incremental Search Selection

To address the usability issues found in the linear scanning approach, we propose an alternate approach to selecting elements on a web page based on text entry instead of direct selection. Using this approach, we can leverage the research done on efficient text entry using alternative input devices such as eye tracking systems or single switch devices.

The incremental search selection method is based upon selecting a web page element by typing a subsection of its user-visible label. For hypertext links or form buttons, the identifying label is simply the text of the link or button. For text input boxes and other selectable elements with no meaningful identifier, the system will generate a unique label to present to the user. Our approach to automatic label generation is discussed further in Section 3.2.

When a new page has completed loading, the system extracts all selectable elements from the page and builds a list of labels from them. This list comprises the search space for the highlighter module. When the

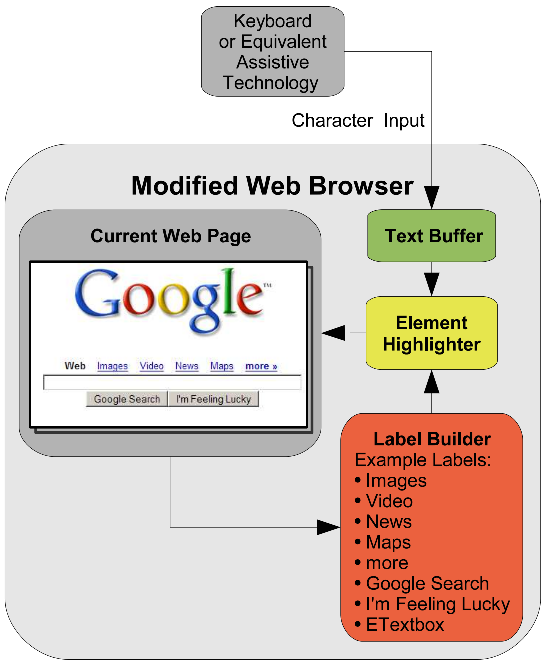


Figure 1: System Overview.

link function is selected, the system waits for text input from an external on-screen keyboard application designed for alternative input devices. To select an element on the page, the user starts typing the label of that element. When the first letter is typed, the system performs a prefix search of all the elements and visually highlights any matches. Every subsequent letter will narrow this search until only one element is highlighted. The system will then prompt the user if that selected element should be activated. In the case of a hypertext link, an activation entails navigating to the linked page, while a text box would be focused, and a button would be pressed. The goal of the highlighting algorithm is to be as intuitive as possible for the user while allowing fine grained specification of the desired link. Figure 2 details our algorithm used to highlight elements on the page.

In Figure 2, the step labeled "Widen Search Field" requires some additional explanation. When the user initially activates the link selection function, the search field is comprised of the visible selectable elements on the current screen. Thus, if a web page is longer vertically than the size of the screen, the algorithm will only match elements that are currently visible to the user. If no elements start with the letter(s) typed by the user, the search field expands to encompass all the selectable elements on the web page. If there are no matching elements at this level, the search criterion is relaxed to consider any substring

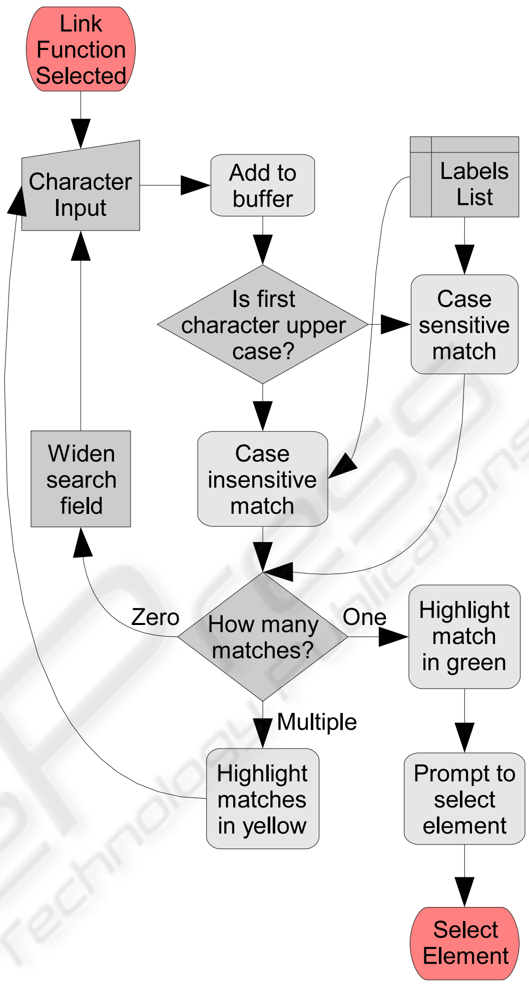


Figure 2: Search Selection Highlighting Algorithm.

of all the elements on the current page. In the vast majority of cases, the user will be attempting to select an element that is currently visible to them. This search fallback method maximizes the efficiency for the common case of a user wishing to select a visible element, while still allowing a user to select an off-screen target if he/she has existing knowledge of its label.

The selective case sensitivity is included to decrease the average number of necessary letters to uniquely identify any given element while not imposing additional constraints on the user. When using an on-screen keyboard, much like with a physical keyboard, producing a capital letter requires pressing an extra key. Thus, if a user types a capital letter, it is assumed that it is not accidental. Performing a case sensitive search when the user has typed a capital letter opens up the set of upper case letters to be populated with labels. For example, if a page contains multiple links starting with a lowercase ’b’, and the desired element label starts with a capital ’B’, then the desired element can be uniquely selected by typing a capital B, instead of possibly having to type multiple letters if the system was not case sensitive. Typing a lower case letter matches both upper and lower case labels since the system should not reduce efficiency in cases where there is only a single label starting with that letter (in upper case).

3.2 Element Labeling

There are several factors to consider for the element label generation algorithm. The primary goal is to minimize the number of characters necessary to select any given element on the web page. A secondary goal is to present link text intuitively and make as few modifications to the visual appearance and layout of the web page as possible. Initially, the system extracts the text of every link and every button on the page and constructs a list based on these labels. For elements without a user-visible label, the system constructs a list of the alphanumeric characters that are not already used as starting characters in the set of element labels. For each unlabeled element, a label is created consisting of an unused alphanumeric character prefixed to the type of the element. For example, a text edit box might receive the label “bTextbox” or “4Textbox” while an image link may receive the label of “tImage”. It should be noted that these kind of labels are only constructed if no meaningful information about the element can be determined which could be used instead. For example, some form input controls (check boxes, text entry boxes, radio buttons, etc.) are referenced by an external “label” HTML tag which provides a textual description of its purpose, and image links may have an alternate textual label specified.

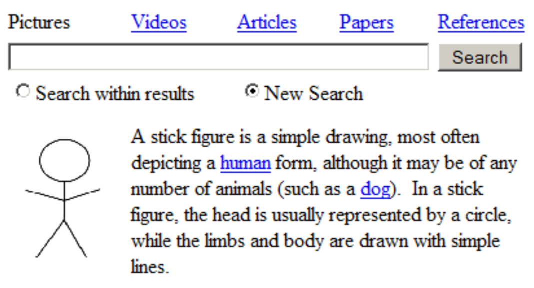
The tradeoff to consider when constructing labels for textual links is whether to provide as much coverage of the alphanumeric character set as possible versus leaving the labels as they appear on the page. Aggressive relabeling ensures maximum efficiency but necessitates more layout modifications and may reduce the intuitiveness of the labels for textual links. For example, if there are 30 selectable elements on a web page, it is possible to relabel elements that have the same starting letter(s) as other elements to maximize selection efficiency. This approach is similar to automatic access keys, and thus faces similar problems in that the relabeling of many elements contributes to visual clutter or causes layout changes. In our implementation, the aggressiveness of the label builder will most likely be tunable based on the input characteristics of the user. If a user has a very difficult time activating their input device, efficiency is of paramount importance, and elements should be relabeled for maximum coverage of the alphanumeric character set. In other cases, minimizing the changes to the existing labels may be desired to minimize the impact on the look of the web page.

3.3 Implementation

One of the reasons that previous attempts at custom browsers for low bandwidth input have been less useful than theorized is that implementations were often based around custom written HTML rendering engines. Although this approach gives absolute control over the presentation of web pages, it limits any implementation’s usefulness for today’s highly complex web pages. Fortunately for future developers in this area, two mature open source HTML rendering engines are now available for use. Both Webkit (webkit.org, 2006) (the core of Apple’s Safari Web Browser) and Gecko (mozilla.org, 2006b) (the foundation of the Mozilla Firefox browser) can be embedded into custom applications or extended in any fashion. We are in the process of implementing our navigation technique as an extension to the Mozilla Firefox browser.

One obstacle for users not able to move the mouse is determining the location of selectable elements on the page. For regular textual links, identification is not a problem as these links are usually underlined and colored blue. However, an image link or the links within an image map may be difficult to locate without having the visual feedback of the mouse cursor turning to a hand symbol. To overcome this problem, our implementation will darken any areas of the page that are not selectable when the link function is selected. Figure 4 shows an example page with non-selectable elements darkened. In the regular view (Figure 3) it is not possible to determine that the image of the stick figure is actually a link. This technique gives users a visual cue to discover selectable elements without introducing complexity to the user interface or modifying the page layout.

Additionally, in Figure 4 the label overlays are visible for the image link and the text entry box. These two elements normally do not show a user visible textual description, so our system assigns a label based on the information available about the element. In the case of the image, the system has extracted the alternate text specified in the HTML code and turned it into a label for the element. If no alternate text is specified, the system would generate a label consisting of an unused (as the first character in another label) al-

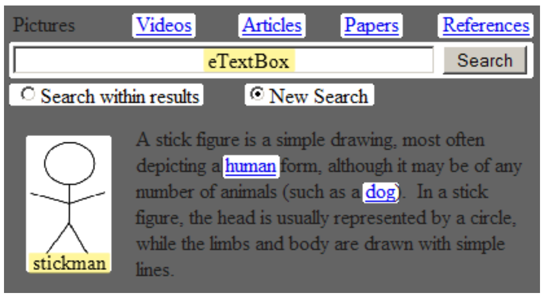

Pictures Videos Articles Papers References

Search

Search within results [✓ New Search]

A stick figure is a simple drawing, most often depicting a human form, although it may be of any number of animals (such as a dog). In a stick figure, the head is usually represented by a circle, while the limbs and body are drawn with simple lines.

Figure 3: Example Page Containing Different Selectable Elements.


Pictures Videos Articles Papers References

eTextBox Search

Search within results [✓ New Search]

A stick figure is a simple drawing, most often depicting a human form, although it may be of any number of animals (such as a dog). In a stick figure, the head is usually represented by a circle, while the limbs and body are drawn with simple lines.

stickman

Figure 4: Example Page Showing Selective Darkening and Label Overlays.

phanumeric character and the type of the element (image, check box, text box, etc.). The search query box, for example, has a label that was constructed in this manner.

4 TYPING INTERFACE

The appropriate typing interface is highly dependent on the specific input device and performance characteristics of the user. For single switch users, this may be a row-column scanning keyboard or Morse code, while for eye or head tracking users this will be an on-screen virtual keyboard. Some commercially available alternative input devices are paired with appropriate software for the user to type into any third party application. While future developments may require a more integrated typing interface (Section 6), initially we intend to leverage existing text entry solutions.

5 DISCUSSION AND CONCLUSIONS

This paper has presented a new approach to navigating on the web for people not able to easily use a standard computer mouse or keyboard. Using text input to search the labels of selectable elements on a given page and selecting matching elements, users are able to acquire and activate their desired element in a predictable and efficient way. Compared to the scanning selection technique, our approach improves on performance for web pages with many selectable elements, reduces the time cost of incorrect selections, and increases the system's predictability for the user. It is our hope that more efficient tools for web access will make it feasible for users with severe disabilities to take full advantage of the world wide web.

Although we believe that the incremental search selection process will be significantly more efficient than a linear scanning approach for most web pages, there are certain cases where it is not optimal. The most common case is when the web page contains only a small number of selectable elements. In this case, the system should fall back to a linear scanning approach. The exact number of selectable elements where a linear scanning approach is more efficient depends on the distribution of starting letters in the selectable elements and the characteristics of the virtual keyboard (which letters are fastest to type). This number may be calculated independently for each page based on these properties.

It remains to be seen whether users are more or less comfortable with the search based selection process versus the scanning based one. Although specifying one's desired target by typing part of its name seems intuitive, it is a departure from the direct selection modes the user may be familiar with such as "point and click" or automatic scanning. After the implementation phase is complete, this issue will be investigated further through user trials.

A beneficial side effect of the search selection method is that it gives the user more control over the process of navigation. With previous approaches, the system controlled the pace of the scanning and the user was merely reacting to events instead of controlling them. Although the scanning pace can be adjusted, these systems have long scanning processes that require the user's complete attention and cannot be interrupted. Not being in control of the system's operation can cause users to feel "pushed" and anxious about using the software.

6 FUTURE WORK

On many web pages, some elements are more frequently used or more important relative to other elements on the page. Important elements may be the main links in the navigation bar, the email body text box in a web based email page, or the submit button for a form. In addition, in most virtual keyboard interfaces some letters are easier and faster to type than others. For example, the characters closest to the top left corner are usually the fastest to type in a row-column scanning keyboard since the scanning starts in the topmost row and leftmost column. If typing using Morse code, symbols with shorter dot/dash combinations (e, t, a, i, m, n, etc.) are faster to type. Which characters are the most efficient depends on the specific typing interface being used.

In future work, we intend to examine matching the most important elements in a web page to the characters that are easiest to type in a given interface. This requires an algorithm to order the selectable elements based on their estimated importance. A measure for the importance of an element could be determined based on a combination of metrics such as text size of a link label, the actual text of a button, the name of an input field, or even a dynamic measure such as the PageRank (Page et al., 1998) of a link. In addition, this approach would require tighter integration between the typing interface and the web browser to determine the cost of typing each character in the interface.

REFERENCES

gok.ca (Nov 2006). Gnome on-screen keyboard. http://www.gok.ca.

Hackett, S., Parmanto, B., and Zeng, X. (2004). Accessibility of internet websites through time. In *Assets '04: Proceedings of the 6th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 32–39, New York, NY, USA. ACM Press.

Hanson, V. L., Brezin, J. P., Crayne, S., Keates, S., Kjeldsen, R., Richards, J. T., Swart, C., and Trewin, S. (2005). Improving web accessibility through an enhanced open-source browser. *IBM System Journal*, 44(3):573–588.

konqueror.org (Nov 2006). Konqueror - web browser. http://konqueror.org/features/browser.php.

madentec.com (Nov 2006). Discoverpro. http://www.madentec.com/products/discoverpro.php.

Mankoff, J., Dey, A., Batra, U., and Moore, M. (2002). Web accessibility for low bandwidth input. In *Assets '02: Proceedings of the Fifth International ACM Conference on Assistive Technologies*, pages 17–24, New York, NY, USA. ACM Press.

mozilla.org (Nov 2006a). Accessibility features in firefox. http://www.mozilla.org/access/features.

mozilla.org (Nov 2006b). Mozilla layout engine. http://www.mozilla.org/newlayout/.

Owens, J. and Keller, S. (2000). Multiweb: Australian contribution to web accessibility. In *Proceedings of the 9th Australasian Conference on Information Systems*.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998). The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project.

Prinz, R., Zeman, P. M., Neville, S., and Livingston, N. J. (2006). Emulating a single switch input device through wavelet de-noising of surface EMG signals. In *Proceedings of the 3rd Cambridge Workshop on Universal Access and Assistive Technology*.

Rowell, D., Dalrymple, G. F., and Olsen, J. (1978). UNICOM: A universal communication and control system for the non-verbal motor impaired. *SIGCAPH Computers and the Physically Handicapped*, (24):56–59.

Stephanidis, C., Paramythis, A., Karagiannidis, C., and Savidis, A. (1997). Supporting interface adaptation in the AVANTI web browser.

Steriadis, C. E. and Constantinou, P. (2003). Designing human-computer interfaces for quadriplegic people. *ACM Transactions on Computer-Human Interaction*, 10(2):87–118.

w3.org (Nov 2006). Web accessibility initiative. http://www.w3.org/WAI/.

webkit.org (Nov 2006). The webkit open source project. http://webkit.org.

Zhao, L., Yan, J., and Yuan, K. (2005). Mozilla accessibility on unix/linux. In *W4A '05: Proceedings of the 2005 International Cross-Disciplinary Workshop on Web Accessibility (W4A)*, pages 90–98, New York, NY, USA. ACM Press.