WEST: A Web Browser for Small Terminals

Staffan Björk,
Lars Erik Holmquist and
Johan Redström
Viktoria Institute
Box 620, SE-405 30
Göteborg, SWEDEN
{bjork,leh,johan}
@viktoria.informatics.
gu.se

Ivan Bretan
Telia Mobile AB
SE-131 86 Nacka Strand
SWEDEN
ivan.p.bretan@telia.se

Rolf Danielsson
Telia Research AB
SE-123 86 Farsta
SWEDEN
rolf.j.danielsson
@telia.se

Jussi Karlgren
and Kristofer Franzén
Swedish Institute of
Computer Science
Box 1263
SE-164 29 Kista
SWEDEN
{jussi,franzen}@sics.se

ABSTRACT

We describe **WEST,** a **WE**b browser for **S**mall **T**erminals, that aims to solve some of the problems associated with accessing web pages on hand-held devices. Through a novel combination of text reduction and focus+context visualization, users can access web pages from a very limited display environment, since the system will provide an overview of the contents of a web page even when it is too large to be displayed in its entirety. To make maximum use of the limited resources available on a typical hand-held terminal, much of the most demanding work is done by a proxy server, allowing the terminal to concentrate on the task of providing responsive user interaction. The system makes use of some interaction concepts reminiscent of those defined in the Wireless Application Protocol (WAP), making it possible to utilize the techniques described here for WAP-compliant devices and services that may become available in the near future.

Keywords

Hand-held devices, web browser, proxy systems, focus+context visualization, text reduction, flip zooming, WAP (wireless application protocol)

INTRODUCTION

The World Wide Web (WWW) currently consists of about half a billion pages, offering users a vast range of informational resources. However, these pages are almost exclusively designed for use with desktop computers, i.e. computers with large high resolution screens, powerful processors, and an abundance of primary and secondary storage.

Parallel to exponential growth of the web during the last few years, digital mobile telephony has evolved to become a basic commodity in the United States and many parts of Europe. Particularly in the Nordic countries, penetration can be as high as 60% (Finland). The world's largest manufacturers of mobile phones predict that there will be 1 billion mobile telephones in use in five years time. Currently, mobile communication is still mostly synonymous with voice telephony, but this is almost certain to change pending new mobile data communication technologies being deployed, increasing data speeds and improving usability. In particular, this development should be viewed in the context of network technologies such as GPRS (General Packet Radio Service), allowing for data speeds in the range of 115 kbps and service technologies such as WAP (Wireless Application Protocol) which sets an industry standard for web-like, interactive applications for use with mobile telephones.

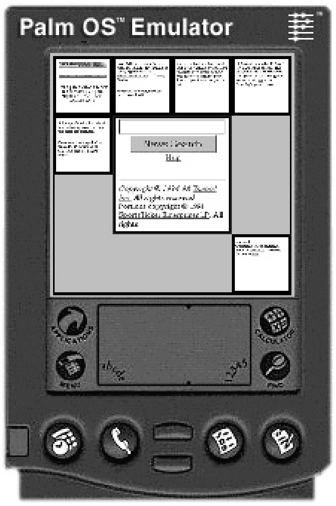


Figure 1: The WEST browser on a simulated Palm OS™ display

The work presented here focuses on this encounter between the WWW and mobile telephony, and more specifically on the need to provide gateways between mobile technologies and existing web resources. Although mobile terminals require specially designed formats for optimal usability due to the constraints of the user environment, it is not likely that all information available on the web will be translated into these format in advance. Thus, there is need for some kind of automatic on-the-fly transformation of existing web content to mobile formats, in order not to shut mobile users out from the bulk of web resources.

In dealing with this issue, the crucial problem is not as much a lack of bandwidth (which the new network technologies are dealing with) or the conversion from one mark-up language to another, but rather developing techniques for the adaptation of information to the usability requirements of mobile terminals. Innovative use of techniques for information filtering and information visualization seem to be a fruitful approach in dealing with this problem, as such techniques deal with issues that are part of the problem of providing information on small mobile devices.

New ways are needed to present web resources and to navigate among and within web pages, which is the target domain of the work described here. The constraints on information presentation posed by small terminals made it necessary to combine several different strategies in order to achieve a sufficiently compact presentation. In different fields of research, several techniques for creating compact representations have been developed. In WEST, techniques from computational linguistics and information visualization were combined. The original web-pages were compressed both in terms of their linguistic content by means of text reduction, and in terms of their visual presentation, and were then presented to the user by means of focus+context visualization.

The rest of the paper is organized as follows: First we give a brief overview of the WEST system and its components. We then give a background in related work, required for implementation of the system. A detailed example, where we see how a user may interact with the system follows. We then describe each of the components of the system in further detail, and give an account of an early user test of the system. Finally, conclusions and future work are discussed.

THE WEST BROWSER

WEST (**WE**b browser for **S**mall **T**erminals) is a web browser specifically designed for use on hand-held devices with limited resources (see **Figure 1**). Although most of the actual implementation was done in Java running on a standard PC, in order to achieve a realistic simulation of the conditions of mobile computing we based the design of WEST on the capabilities and limitations of popular PDAs, such as the 3Com Palm™ line of hand-held computers. Such a device would typically have a small touch-sensitive black-and-white or grey-scale screen with a resolution of about 160x160 pixels, a memory of about 0.5-4 MB, a processor running at about 10-20 MHz, and no provision for traditional keyboard input. These capabilities have proven to be quite adequate for the tasks which such devices are currently required to perform, but are of course far below the specifications of any current desktop computer. The challenge was to work within these limitations but still provide the user with a workable browsing experience, and in the process attempt to overcome the navigation problems that would typically occur on a small terminal.

Our solution consisted of two parts:

- A *proxy server* (running on the user's ISP server), that would take a standard HTML page and transform it in real-time into a format suitable for browsing on small screens
- A *client application* (running on the hand-held terminal), that would allow the user to view and interact with the web pages as provided by the proxy server

The reason for letting the bulk of the processing of the HTML pages be done on the proxy server rather than by the terminal was to relieve the comparatively under-equipped terminal of resource-intensive tasks, thus allowing it to concentrate its resources towards providing responsive user interaction. Furthermore, by stripping away unwanted information on the server rather than on the client, a saving on bandwidth might also be made.

The proxy processing was comprised of several stages:

- A *chunking stage,* where an HTML page was divided into a number of smaller pages, or *cards*, which were then collected into groupings, or *decks*
- A *text reduction stage,* where a set of keywords summarizing each card were extracted from the text
- A *link extraction stage,* where all the hyper-links on each card were extracted

The resulting cards, with supporting keywords and links, were then passed to the client. The client application would then provide the following display modes:

- *Thumbnail view:* Here, a focus+context visualization comprising miniature views of the cards (or top-most card of each deck) was provided
- *Keyword view:* Here, rather than presenting thumbnails, the keywords extracted from each card were presented
- *Link view:* Similar to the keyword view, but rather than displaying keywords, this view showed the links available on each card

(A *pure text view,* showing only the text with no images or formatting, was not included in this prototype but could be useful in some situations and might be added later.)

Each view allowed the user to zoom in completely on a card, providing a fully readable view of the content. The user interacted with the views using the *flip zooming* focus+context visualization technique [16], through which the system provided an overview of the material with simultaneous access to the individual cards.

RELATED WORK

In designing the system used in the WEST prototype, previous work from several different research areas were applied: Proxy systems to provide intermediate formats of the web pages; text reduction algorithms to find suitable keywords in the pages; and information visualization techniques to display information on the limited screen space available. Some properties of WAP, the Wireless Application Protocol, were also considered.

Wireless Application Protocol (WAP)

WAP [32] is a de facto standard for providing Internet-based content and services to wireless devices such as cellular telephones, and requires resources to be coded in a dedicated mark-up language called WML (Wireless Mark-up Language) adapted to the limitations of such devices. Although the WEST architecture is not specifically designed to work in conjunction with WAP, there is potential for interesting synergies when it comes to user interaction.

Firstly, the concept of deck (approximately corresponding to a page in WWW, i.e. a single resource transfer) and card (sub-unit of deck, i.e. a single display object) in WAP lends itself very well to visualization using flip zooming. The overview mode captures the collection of cards, i.e., the deck, whereas the zoomed view corresponds to viewing an individual card. Because of this nice correspondence, we have adopted the WAP terminology of cards and decks in the WEST system, although WAP protocols are not currently used in WEST.

Secondly, given a PDA type of device with WAP capabilities, a WEST browser could readily be converted into a WAP browser, i.e. processing WML instead of HTML. However, when it comes to mobile phones with small text-only displays, this is of course a completely different issue. In this setting, the simplest way of using a WEST browser would be to navigate in zoomed-only mode (i.e. without the context overview). A more advanced solution would involve creating overviews through keyword or link-extraction.

Proxy Systems

The notion of using a proxy server to mediate between the Internet and thin clients is well established [9, 23]. A proxy of this kind can have many functions: coding and conversion of protocols; filtering, compression and conversion of information, etc. The WEST proxy could be tailored to include protocol functions, but the work presented here focuses on the information handling aspects of a web proxy for mobile devices. By removing unwanted or unnecessary information, and by compressing and restructuring (chunking) the information, it can be made to better suit the usability demands of mobile terminals. When it comes to information compression, we can distinguish between *lossy* and *non-lossy* compression. This distinction is normally applied to images [9, 23], but in the context of WEST we will be dealing with lossy text reduction in the shape of text summarization techniques.

WEST has in common with the *Top Gun Wingman* browser for the 3Com Palm™ PDA [10] the basic principle of hiding complexity (such as HTML parsing and analysis) in the proxy server to off-load the handheld device. The idea of saving screen real estate by using text compression has been put to use in another proxy-based system known as the *Digestor* [2]. Proxy systems can also be used as support systems in “surgical” extraction of information from WWW and other sources, providing semi-automatic conversion of such pre-determined content into format suitable for thin clients, such as WML or Web Clippings in the Palm VII™ PDA. *Panama* from Oracle [26] is an example of such a system, which converts HTML and other formats into XML, from which selected WML fragments can be generated by means of stylesheets.

It should be noted that the work presented here does not take a stand as to whether pre-authored content (e.g., a WML source) or automatically converted and filtered content (e.g., HTML–>WML or HTML to simplified HTML) is the strategically correct way to produce services for the user of wireless handheld devices. We content ourselves with the observation that there will be a demand from mobile users for accessing arbitrary web-based resources, particularly in the initial deployment period where dedicated mobile services, WAP-based or not, will emerge slowly. Initially, the range of services and content for mobile use will be limited, since information providers will be reluctant to invest in parallel coding of content. Gradually, this will change (particularly if systems such as Panama are used, which allow for re-use of existing web resources), but there will always arise situations where users want to access material not pre-adapted to dedicated mobile formats. The WEST approach tries to address the needs of such users.

Text Reduction

For the keyword view, a text had to be summarized into a few words. We call this technique *text reduction*, to distinguish it from traditional text summarization. The major challenge for traditional text summarization techniques is two-fold: understanding which regions of a text bear the most pertinent information, and cobbling those bits of information together into a coherent summary. In the case of small screens, the space requirements are more demanding, which actually makes the task somewhat easier. Coherence will not be an issue, since the aim will be to extract a small number of information-bearing terms from the text, making the task closer to the field of index term selection than that of text summarization.

Index terms are typically selected based on term frequency as originally proposed by Luhn [24], selecting suitably frequent terms to represent a document. However, the most frequent words in a text are usually form words, which bear little or no topical information (“is”, “and” and the like). These words must be filtered out either through the application of a judiciously composed stop list or through the application of estimates of term specificity [30]. These are terms that occur in all documents in a document base and have no indexing power; terms that occur in few documents are more useful to that end. Typically, the two measures are combined, in a standard “tf.idf” formula (e.g. [29]). This was the basis for the keyword extraction algorithm used in our application.

Viewing Web Pages on Small Screens

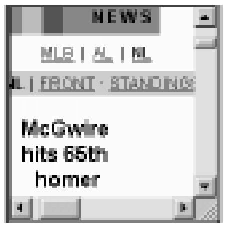
Although personal digital assistants and other hand-held devices have been available for a number of years, the problems associated with user interface design for small terminals have only recently started to attract attention from the human-computer interaction research community [18, 25]. While many general principles for human-computer interaction also apply to small terminals, they can not always be taken for granted, and to simply transfer interaction components from desktop computers will often lead to unexpected problems [15].

Earlier research in information visualization techniques have focused mainly on maximizing the use of screen space on ordinary computer screens. A number of *focus+context techniques* have been developed to give users access to simultaneous overview and detail. General focus+context visualizations techniques such as the *Generalized Fisheye View* [11] or techniques developed for text documents, such as *SeeSoft* [7] or the *Document Lens* [27], might be adapted to the WWW. General zooming or multi-scale interaction techniques which have also been used for visualizing web pages include *PAD++* [1], *Cone Trees* [28], *Hyperbolic Trees* [22] and *Elastic Windows* [19], and techniques developed specifically with the web in mind include the *WebBook* and the *Web Forager* [6], *Zippers* [5] and *CZ Web* [8].

Although most of the techniques above have been developed for use on traditionally-sized screens, some of them might feasibly be adapted for use on small screens. However, many of these techniques have advanced requirements in the form of computational resources for performing smooth graphical transformation and providing responsive interaction, and while they may often have proved useful on desktop machines, hand-held devices such as those on which the WEST system are intended to be used, are currently for the most part not capable of any advanced visual calculations. The focus+context technique *flip zooming* that was used in this project was also originally developed for ordinary screens, but because it is not very resource-intensive it has proven possible to transform it to smaller devices. For ordinary screens, it has previously been used for visualizing web sites [14], and has been generalized to handle hierarchical material such as hierarchically ordered image collections [17]. As part of the WEST project, we have evaluated flip zooming as an alternative to scroll bars on small screens [3].

INTERACTION IN WEST

To give a better idea of how the WEST browser works, we will now give a detailed account for how a user may interact with the system. This will take the form of a complete interaction scenario, with an illustration for each screen the user will see.



McGwire hits 65th homer

The example page viewed in a traditional browser on a 160x160 pixel display

As our example, we have used a page reporting baseball news at the Yahoo Sports site. The page was comprised mostly of text – 319 words, or about 1500 characters. There were 15 links to other pages, plus a banner advertisement and a search function. As the figure above will attest, viewing this page on a traditional browser on a 160x160 pixel screen presents serious problems. Only a very small part of the page would then be available at any time, giving almost no clues to the size or context of the material.

Flip Zooming in WEST

The interaction in WEST is based on *flip zooming,* a tile-based focus+context visualization technique. Flip zooming allows users to navigate a data set consisting of sequentially ordered discrete objects, e.g. images or pages of text. One object is in focus, the other objects provide the context. Users can move the focus backwards or forwards in the data set, or select any visible object as focus object by pointing at it. Users can also zoom in further on an object, allowing it to occupy the whole screen. Objects are ordered in a left-to-right, top-to-bottom fashion, so that any object that is after another object in sequence will be placed to the right and/or below the preceding object.

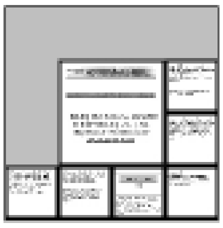
Earlier user studies of flip zooming applications [4] have indicated that users may become confused if thumbnails and focus objects are allowed to change their positions on the screen, or if the display is too packed with information. For this reason, we limited the maximum number of objects on the display at any one time to seven, which allowed us to keep the focus object at the center of the display with sufficient room to display the context objects at a reasonable size. It may seem that in some cases we are not using the available screen estate to the maximum, but this is a conscious trade-off to provide a clearer and more easily-understood display.

In WEST, some objects on the display are in fact representations of several objects, since they represent the top-most card of a deck. In this case, when zooming in on such a card, a user would be presented with a view of all the cards in the deck, which could then be navigated as usual. In this way, the user is in fact navigating a hierarchy comprised of decks and cards. In the example this hierarchy is only one deck deep, but there is no reason why it could not be more complex.

Interaction Example

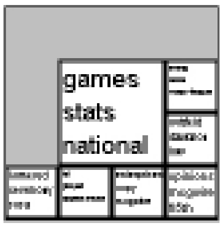
We will now follow a user interacting with the sample page using the WEST browser. The user wants primarily to read about her favorite Chicago Cubs player, Sammy Sosa, and possibly chat about his exploits with other supporters.[1]

1. The authors know very little about the game of baseball, and apologize in advance for any errors in this account that may be spotted by more knowledgeable readers!



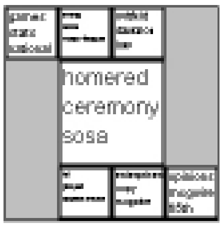
***1.** Thumbnail view, whole page, with first card in focus*

1. Initially in WEST, the viewer is presented with the *thumbnail view,* which gives an overview of the whole web page in flip zoom format. Here, each card is presented as a thumbnail image, not large enough to be readable, but still giving a sense of the overall nature of the page – e.g. image-heavy, text-heavy, many or few links, etc. The first card or deck is in focus, with the others presented as context. (Unfortunately, there is currently no clear visual indication of if a thumbnail represents a single card or a deck, something which might be addressed in future versions of the browser.)



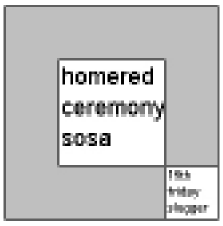
***2.** Keyword view, whole page, with first card in focus*

2. The user now chooses to switch to the *keyword view,* to see if she can locate some information about Sammy Sosa.



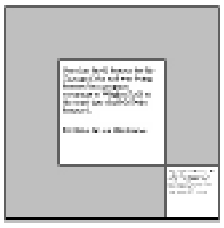
***3.** Keyword view, whole page, fourth card in focus*

3. The keywords on the fourth card in the sequence indicate that Sosa is mentioned. The user focuses on that card. This can be done either by explicitly pointing at the card, or by moving the focus sequentially until the desired card is reached. (Since what here looks like a single card may in fact be the top card in a deck, user will actually often be navigating among decks in this manner.)



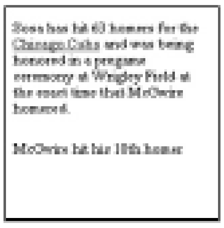
***4.** Keyword view, a deck open, first card in focus*

4. The card in question is in fact a deck, consisting of two sub-cards in total. By zooming in on the visible card, the first card in the deck, the deck is opened and displayed. The keywords indicate that some kind of ceremony has taken place, involving Sosa and home-runs.



***5.** Thumbnail view, a deck open, first card in focus*

5. The user now switches back to a thumbnail view of the deck, showing the original HTML formatting of the cards.

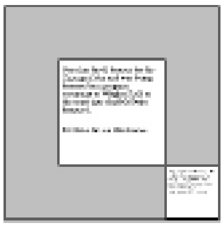


Sosa has hit 63 homers for the Chicago Cubs and was being honored in a pregame ceremony at Wrigley Field at the exact time that McGwire homered.

McGwire hit his 10th homer

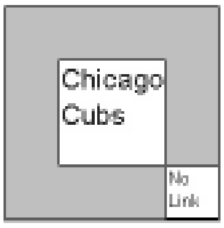
***6.** Thumbnail view, zoomed in completely on a card*

6. The user zooms in completely on the first card in the deck and reads the text on the card. It is indeed interesting news about Sammy Sosa. Staying in this view, the user can now advance to the next or previous card in the deck (e.g. by pressing a specified button on the PDA or tapping on a portion of the card with the pen), to read the full story. (If the card on view happens to be the last in a deck, when advancing, the first card in the following deck will be shown.)



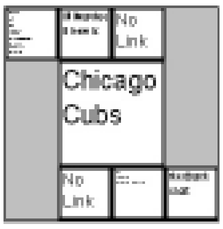
***7.** Thumbnail view, a deck open, first card in focus*

7. The user now wants to chat with other supporters about this development. She zooms out again, returning to the overview of the deck.



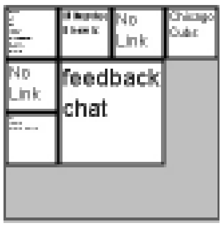
***8.** Link view, a deck open, first card in focus*

8. The user now switches to the *link view,* since she is looking for a link to the chat page.



***9.** Link view, whole page, with fourth card in focus*

9. Not finding the link she is looking for in this deck, she zooms out to reveal link view for the whole page.



***10.** Link view, whole page, with seventh (last) card in focus*

10. She sees a link to the chat room on the very last card in the page, and focuses on that card. By clicking on the link while the page is in focus she will be transported to the chat-room page, meaning that the current web page/deck will be removed from the screen and the chat-room page/deck will be displayed.

DESCRIPTION OF THE COMPONENTS

The components of the WEST system were designed as a number of modules that could be individually improved and expanded as the system was developed. In the following, we will describe each of these pieces separately.

Pre-processing, Including Card Chunking

Proxy servers for real-time pre-processing of web information to be accessed using a mobile terminal is a proven technique used for instance in current web services for palm-sized PDAs. In WEST, we made use of a proxy server to:

1. Filter and reduce the contents of web pages in order to adapt them to the capabilities of the mobile browser (this would mean among other things to get rid of JavaScript, image maps, frames etc.)
2. Convert the reduced web page into *n* sub-pages (cards), each of which can be readily presented on a mobile-sized display (e.g. 160x160 pixels). Cards are interlinked to form a deck by arranging them into a suitable reading-order
3. Produce alternative renderings of these cards corresponding to different levels of detail. Typically a card can be displayed in its full size, in reduced size and minimized. These alternative renderings are not necessarily derived from graphical reduction – in WEST, one alternative when reducing card size is to use automatic text summarization

Key element such as headings, paragraphs, pictures, tables etc. provide hints on how the original page is structured. These hints are used in the “chunking” of the page into cards, i.e. determining break-points for card creation. The maximum allowed size of a card is of course a limiting factor, which sometimes means that the information contained within a card’s minimal natural page-chunk cannot be presented without some modifications (for instance image or font size adjustments), or by splitting up the information into two cards. (For more information on the chunking algorithm see the appendix.)

The cards produced by the proxy are arranged into several decks linked together in the original reading order. Because of the limitations of the display, each deck was limited to seven cards, the maximum that could comfortably be displayed using the flip zooming variant we had chosen. If a page consisted of more than 49 cards (seven decks with seven cards each) some of the decks would in turn have to contain sub-decks of cards, creating a deeper hierarchy.

Extraction of Keywords

To extract the keywords that were to represent each card, the method chosen had to be suitably general to handle any kind of material. Since there is no way in advance to tell what type of web page a user will be browsing, the system should be equally at home at whatever topic it was subjected to, including general news, sports, entertainment, and so on. It would be feasible to allow the creator of a page to specify which key-words are most relevant, but this would require that pages were specially constructed for the purpose of this system, and as mentioned, the intention was to give users access to all pages of the web without any prerequisites.

Our text reduction algorithm relies on the fact that typically several texts or text chunks will be compressed and displayed simultaneously. The text chunks are all short, approximately thirty words. The word tokens in the text chunks are tabulated for frequency, after the application of a stop list filter of form words. Each chunk is represented by a list of words sorted by frequency. These raw frequency counts are then modified by inverse document frequency [30] – each word will have its raw frequency count divided by a factor depending on the number of texts it has been found in. Thus, if a text has two words with equal frequency, where one occurs in three texts and the other only in the text at hand, the latter term will be weighted higher.

The text reduction procedure thus disfavors words that are evenly spread out over the chunks at hand, and aims at representing each chunk by as unique words as possible, in that given context of chunks. Words that occur disproportionately often in a given chunk, compared to other visible chunks, are favored above the more generally frequent words. But words with high frequencies that occur in many texts are not discarded. They are set aside and used to generate a header for the group of text chunks under consideration, and can be used for hierarchical reduction of the entire group, although this is not done in the current version of WEST. A group of chunks can be reduced to words such as “baseball”, “scores”, “season”; individual chunks can be more finely reduced to “sosa”, “homered”, etc. As mentioned previously, taking advantage of these headers could be particularly fruitful when using small text-only displays where the contextual overview does not fit.

As it stands today, the algorithm does not make use of morphological analysis, thesauri or lexical categories, all of which would increase the reliability of the results. Adding surface level linguistic processing is a modular issue and can be done without a system redesign: there are several efficient general-purpose linguistic analysis components suitable for this purpose.

Link Extraction

To facilitate a view of which links were available on a given web page a simple link extraction procedure was created. This went through each of the cards constructed in the chunking processing and created a similar deck structure, but where the content of each card would only be the hyperlinks.

Web Page Rendering

For the graphical presentation of the different cards, each individual card had to be rendered as if it were a web page. However, we were unable to write a full-scale web rendering engine within the constraints of this project. Instead, we used the rendering engine provided by the HotJava Web Browser [31] to produce an image of each card as displayed on a screen of the required size (160x160 pixels). The same images where also graphically compressed to intermediate and thumbnail size. These pre-rendered images were then used by the system for the graphical presentation.

Presentation and Interaction

Based on the flip zooming technique, the WEST browser presents each web page as a number of discrete objects, representing individual cards or decks of cards. The user navigates between different objects by using directional buttons or by directly choosing the object to focus upon with a pen or other pointing device. For sequential reading of a whole page, a user would generally switch to a full-screen view and then advance through the cards by pressing a designated "forward" button.

Each view in WEST only presents one level of the hierarchical structure of decks and cards that represent the web page. To move between levels, the user zooms in on the object in focus (usually by clicking or tapping with the pen on it) and will thus go one level deeper into the structure. To go up one level, the user clicks or taps on the "white space" between the objects. This navigation might also be facilitated by the use of "up" and "down" buttons, for moving up and down in the hierarchy, analogous to zooming out and zooming in.

When the user goes down one level in the hierarchy, the focus object takes over the whole screen space to show its content. If the current focus represents a single card, this card will be allowed to fill the screen completely to facilitate reading. In the case of the focus representing a deck, however, a view of all the objects in the deck is presented.

The system provides three different modes in which the material can be viewed: thumbnail, summary and link view. When switching from one view mode to another, the position of the focus in the hierarchical structure is maintained, enabling the user to navigate in a suitable view mode to locate a card, and then change to another mode (typically the thumbnail view) to actually view the card. In the prototype, the user switched between the different views by accessing a pop-up menu.

USER EXPERIENCE

To gain some insight in how the WEST prototype performed with inexperienced users, we performed a qualitative evaluation in which the prototype was compared with the HotJava browser [31]. It is important to note that the test was in no way intended as a "fair" comparison between two browsers, since the HotJava browser was not developed with the intention of being used on very small screens. Rather, the intention was to gauge novice users initial reaction to the WEST browser, and the other browser was provided as a reference point only.

A test group consisting of ten subjects, all expert computer users but with no experience of browsing the web on a PDA, were set a number of tasks to perform both in the WEST system and the traditional browser. The tasks consisted of finding specific items in the material, and in some cases required returning to a part of a page which had previously been visited. The tests were performed on a traditional computer screen, but both browsers were given the same screen size as a typical PDA to operate in, i.e. a window of 160x160 pixels.

A questionnaire given to users after the test indicated that they thought that the prototype provided a better overview than the HotJava browser, ranking it on average 3.40 points higher in this respect (5.30 vs. 1.90, standard deviation being 0.68 and 0.99 respectively) on a scale of 1 to 7 with 7 being the best. It also showed that users thought searching was easier with WEST than with HotJava, ranking it on average 2.25 points higher on the same scale (5.55 vs. 3.30 with standard deviation 0.90 and 0.95 respectively). However, it was also noted that the flip zooming interaction technique took some time to get familiarized with, providing some initial difficulties.

Although we did not collect any quantitative measures during this preliminary experiment, the positive reactions of the users did provide us with an indication that the ideas behind the system should be worth pursuing further.

FUTURE WORK

At the moment, the system can be improved primarily in the following areas: improving the chunking of pages; improving the techniques used for text reduction; and improving the means of interaction with the system to make it useful in various realistic situations. We might also consider the division of tasks between the proxy and the client. At the moment, most of the work is performed on the proxy to offload the client machine as much as possible. With faster hand-held machines, there is no reason to believe that not more or maybe most of the information processing such as keyword- and link extraction could take place on the client rather than on the server.

The chunking process still leaves much room for improvement, since often the provided cards are not of the optimum size for the available screen space. Improving the chunking is difficult, however, since there will have to be a balance between producing chunks that are logically coherent to the user, and chunks that are of maximum size. To achieve maximum chunk size it is sometimes necessary to break the pages at inconvenient places, even breaking text in mid-sentence, but this should be avoided for the sake of the user. A more thorough analysis both of page structure and user behavior will be needed to improve this process. Also, integrating the chunker more closely with the actual rendering of the HTML pages would make the judging of available space much easier.

The text reduction algorithm as it now stands is very simple. It is based on well established and understood techniques from text indexing, which guarantees a predictable, stable, and somewhat mediocre result. There are two well known bottlenecks in this type of information access techniques: 1) we have too little knowledge of texts as texts to be able to answer the question of what a certain text is about, and 2) we have too little knowledge of what the text will be used for and why the user wants it. The second problem is somewhat less pressing for this specific application: we know that the text needs to be compressed, and we know what the context is, namely what else is being displayed at the same time. This knowledge we already utilize to some extent, since we are able to generate a header for the texts in view at any given time. The first problem is harder. Knowledge of texts is limited if we view texts as simple bags of words. In future work, we plan to utilize stylistic information [20] to reduce different types of text differently: a legal text might be reduced to a paragraph header, while a long-winded error message might be reduced to a generic icon. We intend to experiment with using text structure to tailor the chunking algorithm so that it will feed homogenous bits of text to the reduction algorithm (e.g. [13]). We might use language technology such as surface syntactic analysis [21] and text extraction techniques [12] to extract topical terms and other topical items such as names, links, or dates from the text segments. We are currently running a pilot project for multi-document summarization, to be able to impose a middle level of analysis: the idea is to collapse several texts into one short summary, whereupon that summary in turn can be reduced.

Finally, it might be possible to improve the interaction with the WEST system in certain usage situations. Using a pen to interact with a hand-held device is sometimes undesirable, since it requires the user to hold the device with one hand (or place it on a flat surface) and use the pen with the other. Essentially, flip zooming only requires four navigational actions to navigate a hierarchical data set (move the focus back, move the focus forward, zoom in and zoom out), and while the WEST browser requires additional input for switching among the different views, it is in many cases possible to use perform the majority of the navigation using only four buttons without relying on a pen. This might allow users to navigate with more precision and efficiency in some situations, and ideally it might even be possible to construct the system so that all navigational buttons were accessible using just one hand, thus freeing up the other hand for other tasks. This would make the human-computer interaction far more flexible, as there might be many situations when having one hand free would be beneficial: while talking in a phone, taking notes, etc.

CONCLUSIONS

Truly mobile web access will evolve along several paths. One path is the development of the “stripped-down” web, reminiscent of browsing with text-only browsers such as Lynx. The other extreme will result from miniaturizing standard computers into hand-held devices capable of handling the same resources as stationary machines. These paths will of course cross, and we will see combinations of dedicated mobile resources and advanced hand-held computers. No matter what, the restrictions of mobile terminals will always hold with respect to the usage environment. We believe work like WEST is important because it focuses on ways to enable advanced interaction on small devices, ways that are largely independent of the capabilities of both the network and the terminal.

By constructing the WEST system, we have shown how material on the World Wide Web can be made available for mobile users and others who are restricted to accessing the web from small terminals. By placing the major work-load on the proxy server, and by providing a novel combination of visualization and text summarization, existing web pages can already be made much more suitable for such devices. In the future, with the continued acceptance of hand-held devices and high-speed wireless network, browsing the web from a PDA or a mobile phone will be a common occurrence. In these cases, systems such as WEST may aid in making this a much more pleasurable and productive experience.

ACKNOWLEDGMENTS

Thanks to the other members of the WEST project, Roberto Busso and Peter Nilsson, and to the anonymous UIST reviewers whose valuable comments helped improve this paper. This work was part of the project *Effective Display Strategies for Small Screens* within the *Mobile Informatics* research program, sponsored by SITI, the Swedish Institute for Information Technology.

REFERENCES

1 Bederson, B., Hollan, J., Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. In *Proc. UIST ’94,* ACM Press, 1994.

2. Bickmore, T.W. and Schilit, B.N. Digestor: Device-Independent Access to the World Wide Web. In *Proc. Sixth International World Wide Web Conference,* pp. 655-663, 1997.

3. Björk, S. and Redström, J. An Alternative to Scrollbars on Small Screens. In *Extended Abstracts of CHI '99,* ACM Press, 1999.

4. Björk, S. and Holmquist, L.E. Formative Evaluation of a Focus + Context Visualization Technique. In *Proc. HCI ’98* (poster presentation), The British HCI Society, 1998.

5. Brown. M.H., Weihl, W.E., Zippers: A Focus+Context

Display of Web Pages, in *CD-Rom Proc. WebNet '96,* Association for Advancement of Computing in Education (AACE), 1996.

6. Card, S.K., Robertson, G.G. and York, W. The WebBook and the Web Forager: An Information Workspace for the World Wide Web. In *Proc. CHI '96,* pp. 111-117, ACM Press, 1996.
7. Eick, S.G., Steffen, J.L. and Sumner, E.E. SeeSoft - A Tool for Visualizing Line Oriented Statistics Software. *IEEE Transactions on Software Engineering,* 18(11), 1992.
8. Fisher, B., Agelidis, M., Dill, J., Tan, P., Collaud, G., Jones, C., CZWeb: Fish-eye Views for Visualizing the World Wide Web. In *Proc. HCI International '97,* pp. 719-722, Elsevier, Amsterdam, 1997.
9. Fox, A., Gribble, S.D., Chawathe, Y. and Brewer, E.A. 29. Adapting to Network and Client Variation Using Active Proxies: Lessons and Perspectives. *IEEE Personal Communications* (invited submission), Sept. 1998.
10. Fox, A., Goldberg, I., Gribble, S.D., Lee, D.C., Polito, A. and Brewer, E.A. Experience With Top Gun Wingman: A Proxy-Based Graphical Web Browser for the USR PalmPilot. In *Proc. IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98),* Lake District, UK, 1998.
11. Furnas, G.W. Generalized Fisheye Views. In *Proc. CHI '86,* pp. 16-23, ACM Press, 1986.
12. Grishman, R. Information Extraction: Techniques and Challenges. *Materials for Information Extraction (International Summer School SCIE-97),* ed. Maria Teresa, Pazienza, Springer-Verlag, 1997
13. Hearst, M. and Plaunt, P. Subtopic Structuring for Full-length Document Access. In *Proc. ACM SIGIR '93,* ACM Press, 1993.
14. Holmquist, L.E. Focus+Context Visualization with Flip Zooming and the Zoom Browser. In *Extended Abstracts of CHI '97,* ACM Press, 1997.
15. Holmquist, L.E. When Will Baby Faces Grow Up? In *Proc. HCI International '99,* 1999. (to appear)
16. Holmquist, L.E. and Ahlberg, C. Flip Zooming: A Practical Focus+Context Approach to Visualizing Large Information Sets. In *Proc. HCI International '97,* pp. 763-766, Elsevier, Amsterdam, 1997.
17. Holmquist, L.E. and Björk, S. A Hierarchical Focus + Context Method for Image Browsing. In *SIGGRAPH '98 Sketches and Applications,* ACM Press, 1998.
18. Johnson, C. (ed.). *Proc. First Workshop on Human Computer Interaction with Mobile Devices.* URL: http://www.dcs.gla.ac.uk/~johnson/papers/mobile/HCIMD1.html, 1998.
19. Kandogan, E., and Shneiderman, B. Elastic Windows: A Hierarchical Multi-Window World-Wide Web Browser. In *Proc. UIST '97,* pp. 169-177, ACM Press, 1997.
20. Karlgren, J. and Cutting, D. Recognizing Text Genres with Simple Metrics Using Discriminant Analysis. In *Proc. COLING 94,* Kyoto, 1994. (In the Computation and Language E-Print Archive: cmp-lg/9410008).
21. Karlsson, F., Voutilainen, A., Heikkila, J. and Anttila A. (eds.) *Constraint Grammar,* Berlin: Mouton de Gruyter, 1995.
22. Lamping, J., Rao, R. and Pirolli, P. A Focus+Context Technique Based On Hyperbolic Geometry for Viewing Large Hierarchies. In *Proc. CHI '95,* ACM Press, 1995.
23. Liljeberg, M., Helin, H., Kojo, M., and Raatikainen, K. MOWGLI WWW Software: Improved Usability of WWW in Mobile WAN Environments, in *Proc. IEEE Global Internet 1996 Conference,* London, England, November 20-21, 1996.
24. Luhn, H. P. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development,* 1 (4) 309-317, 1957. Reprinted in *Luhn, H.P.: Pioneer of Information Science, selected works.* Claire K. Schultz (ed.). New York: Sparta, 1968.
25. Marcus, A., Ferrante, J.V., Kinnunen, T., Kuutti, K. and Sparre, E. Baby Faces: User-Interface Design for Small Displays. In *CHI '98 Summary,* pp. 96-97, ACM Press, 1998.
26. *Oracle Project Panama, Connecting Oceans of Information.* Oracle White Paper, March 1999. URL: http://www.oracle.com/mobile/panama/panamawp.htm
27. Robertson, G.G. and Mackinlay, J.D. The Document Lens. In *Proc. UIST '93,* pp. 101-108, ACM Press, 1993.
28. Robertson, G.G., Mackinlay, J.D. and Card, S.K. Cone Trees: Animated 3D Visualization of Hierarchical Information. In *Proc. CHI '91,* ACM Press, 1991.
29. Robertson, S.E. and Sparck Jones, K. *Simple, proven approaches to text-retrieval.* Technical report 356, Computer Laboratory, University of Cambridge, 1996.
30. Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation,* 28:1, pp. 11-20, 1972.
31. Sun Microsystems. *HotJava HTML Component.* URL: http://java.sun.com/products/OV_hotjavaProduct.html
32. WAP Forum, *Wireless Application Environment Overview,* February 3, 1999. URL: http://www.wapforum.org

APPENDIX: THE PAGE CHUNKER

To divide a page of HTML code into a number of pieces or *chunks*, each suitable for displaying as a single full-screen card on a small display, a page chunking program was developed. It was based on an existing HTML parser (or more accurately, an SGML parser with a description of HTML's elements) written in Java by Richard M. Tobin and available at the following address:

`http://www.cogsci.ed.ac.uk/~richard/ftp-area/html-parser`

First, the page chunker establishes a number of constants, such as the size of a card (e.g. 160x160 pixels), the typical width of a character, the height of a line, maximum number of lines that can fit on a card, and so on. It then reads a piece of HTML code from an URL and performs a number of operations depending on the HTML elements encountered. Operations include:

- Setting flags for elements that can not be split and/or that are suitable as break-points (e.g. H1-H6, HR, A, IMG)
- Reducing the value for the total remaining space on the card (e.g. IMG)
- Adapting the width of the HTML element to the maximum available (e.g. PRE, HR, TABLE)
- Adapting the total size of the HTML element (width and height) to the maximum available (e.g. IMG, APPLET, OBJECT)

Additionally, some tags are replaced with tag combinations that will be handled in a more predictable way during page rendering; for instance, the paragraph tag <P> was replaced with <BR>&NBSP;&NBSP;&NBSP;

The chunker also makes sure that no tags are left "open" on a card, e.g. an opening <H1> with no corresponding closing </H1>. HTML elements are then added to the card until it is full, or as close to full as the algorithm can manage, at which time a new card is started.

After creating the cards, a number of decks are created. The current design of the flip zooming display in WEST limits the number of cards simultaneously visible at any time to 7. For simplicity's sake the deck creation algorithm simply tries to create a maximum of 7 decks with as equal a number of cards as possible. The resulting HTML files are saved in a file structure corresponding to the decks (i.e. one directory for every deck) which can then be read by the WEST browsing component.

Pseudocode for the page chunker is as follows:

```
chunkPage (parameter: URL for HTML page)
    parse HTML page
    save away header (<head> … </head>)
    chunkBody; (divides page into cards)
    collect cards into decks and create corre-
      sponding file structure

chunkBody (parameter: HTML element)
   if not (tag=skiptag)
   then
      modify tag if needed, and add the (starting)
        tag (e.g. <IMG>), including attributes
        (e.g. an image) to body
   else
      if (tag=<p>) then reduce available space on
        this card with no. of characters on a line
   for (all sub-elements)
      if (element=text)
        then
          addText; (add this text to the new card)
        else (i.e. element=tag)
          addTag; (add this tag to the new
            card)
     if not (tag skipped) then add finishing tag
       (e.g. </IMG>)

addText
   (divide until it fits)
   while (number of characters added so far +
     length of new string >= maxlength)
     if (tag can not be split)
       then
         add string to current body
         add finishing tags to all open start tags
           and finish this body
         add corresponding start tags to new body
       else
         check where it is suitable to break (at
           end of paragraph/sentence etc.)
         add what we can fit in to the current body
         add finishing tags to all open start tags
           and finish this body
         add corresponding start tags to new body
     if (text left)
       then add remaining text to current body

addTag
   if (break condition) (i.e. is this a tag that
     can cause the creation of a new card?)
   then
     if (available space on current card is less
       than 10% of maximum size)
       then
         add finishing tags to all open starting
           tags and finish this body
         add corresponding start tags to new body
   chunkBody; (continue chunking body until done)
```