Collapse-to-Zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content

Patrick Baudisch[1], Xing Xie[2], Chong Wang[3], and Wei-Ying Ma[2]

[1]Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
baudisch@microsoft.com

[2]Microsoft Research Asia
Zhichun Road,
Beijing 100080, China
{xingx, wyma}@microsoft.com

[3]Tsinghua University
Department of Automation
Beijing 100084, China
wangchong99@mails.tsinghua.edu.cn

ABSTRACT

Overview visualizations for small-screen web browsers were designed to provide users with visual context and to allow them to rapidly zoom in on tiles of relevant content. Given that content in the overview is reduced, however, users are often unable to tell which tiles hold the relevant material, which can force them to adopt a time-consuming hunt-and-peck strategy. *Collapse-to-zoom* addresses this issue by offering an alternative exploration strategy. In addition to allowing users to zoom into relevant areas, collapse-to-zoom allows users to collapse areas deemed *irrelevant*, such as columns containing menus, archive material, or advertising. Collapsing content causes all remaining content to expand in size causing it to reveal more detail, which increases the user's chance of identifying relevant content. Collapse-to-zoom navigation is based on a hybrid between a marquee selection tool and a marking menu, called *marquee menu*. It offers four commands for collapsing content areas at different granularities and to switch to a full-size reading view of what is left of the page.

Keywords

Web browser, collapse-to-zoom, marquee menu, small screen device, PDA, pen, overview, fisheye, zooming, marking menu, dragging, gestures, user interface.

INTRODUCTION

Web pages are often designed with a desktop screen in mind, i.e., they use multi-column layouts and/or are preformatted to a certain page width. On a small screen device, such as a PDA, such pages can be hard to read. If rendered in a desktop-like two-dimensional layout, such pages can require users to scroll horizontally while reading text (Figure 1). To avoid the need for horizontal scrolling, some of the recent browsers narrow down cells (e.g. MS Pocket Internet Explorer™) or even rearrange the table structure into a single column to make the page fit the screen width (e.g. Opera™ opera.com).

While single-column views can facilitate the reading process, they result in a correspondingly larger amount of vertical scrolling. In Figure 1, for example, accessing material in the center column now requires users to scroll past the entire content of the left column, as that column will appear at the top of the single-column page. For a typical page, this accounts for several PDA screens worth of material.

Figure 1: The problem: viewing a page designed for the desktop on a small screen device requires users to scroll not only vertically, but also horizontally.

To reduce the need for vertical scrolling and to provide users with visual context, researchers have proposed providing users with a miniature version of the page. Such overviews are either thumbnails, i.e., zoomed-out copies of the page [15, 16] or summarized versions of the page [6, 17]. Overviews allow users to zoom in on content in order to view it in a close-up view. However, given that content in the overview is reduced users are often unable to tell which tiles hold the relevant material. In the example shown in Figure 2a even the headlines have become unreadable, which can force users to adopt a time-consuming hunt-and-peck strategy.

In this paper, we present collapse-to-zoom, a technique that addresses this issue by offering an alternative exploration strategy. In addition to the known strategy of zooming into relevant areas, collapse-to-zoom allows users to collapse areas deemed *irrelevant*, such as columns containing menus, archive material, or advertising (Figure 2b). Collapsing content causes all of the remaining contents to be redrawn in more detail (Figure 2d), which increases the user's chances of identifying relevant content. When finally switching to the full-size view (Figure 2g), the page has been reduced significantly, which allows users to scroll through the remaining content in an efficient way.

We begin by giving a brief summary of the related work. Then we give a detailed walkthrough of collapse-to-zoom and describe its gesture-based navigation mechanism. We then briefly describe our implementation before we conclude with a summary and an outlook on future work.

RELATED WORK

There are four general approaches to displaying web pages on small screen devices [4]: device-specific authoring; multiple-device authoring; automatic re-authoring, and client-side navigation. The first two allow obtaining high-quality results by taking the specifics of the device into account during page authoring, but in exchange they require the cooperation of the individual page authors. This prevents these techniques from being applied to already existing pages, which limits the practical applicability of these approaches. Automatic re-authoring, such as web page summarization [10, 6] does not require the collaboration of page authors; neither does it require any user effort. However, techniques in this category are limited to what can be extracted from the page, such as its structure or text content. Oblivious of the user's task at hand, a summarizer cannot know, for example, whether the user plans to make use of a page's link menu or whether it is safe to remove the menu.

Collapse-to-zoom falls into the remaining category of client-side navigation which encompasses a wide range of techniques. *Outlining* transforms pages into sets of tiles. Tiles may then be presented to the user one at a time (*power browser* [7]) or as a layout of cards (*flip zooming* [5], WebThumb [16]). As mentioned above, other browsers combine tiling with zooming, so that users can use an overview to access the individual tiles (e.g., [9, 15, 17]).

The approach of collapsing page content makes collapse-to-zoom a member of the family of generalized fisheye views [11]. Existing research prototypes allow users to collapse text sections (*active outlining* [13]) or pan a fisheye lens across the page (*fishnet* [2]). Collapse-to-zoom is different from these in that it allows users to collapse tiles in a 2D layout, such as cells in a table. In particular, collapse-to-zoom uses space gained from collapsing tiles to *immediately magnify* the remaining page content.

Collapse-to-zoom uses gestures for collapsing and expanding content (a so-called *marquee menu*). For a more general taxonomy of selection gestures see *paintable interfaces* [1]. Unlike gestures in existing systems, such as power browser [7], marquee menus offer command gestures that simultaneously define a target *area* and the command to be executed on that area.

COLLAPSE-TO-ZOOM: A WALKTHROUGH

Figure 2 gives a walkthrough of collapse-to-zoom using the example of the news page from Figure 1. The user's task is to find news articles about the Bertelsmann Company.

(a) As the page loads, the browser detects that the page is significantly bigger than the screen and thus shows a thumbnail view instead. (b) The headlines are scaled below recognizability, so the user cannot decide if any of the

Figure 2: Collapse-to-zoom walkthrough

shown headlines involve the topic of interest. However, layout conventions allow the user to recognize the main structure of a news page, i.e., menu, articles, and advertising columns. To see the news headlines in more detail, the user performs a pen gesture to collapse the advertising column and then (c) the menu column (we will present our gesture language in detail in the next section). (d) As a result, these columns are now replaced with thin gray placeholders and the freed screen space has been used to render the remaining column with the news headlines at a larger size. While the abstracts are still fairly small, headlines have now become readable. Note how the placeholders provide the user with visual context. In addition, they allow the user to restore the respective tile. Tapping the left placeholder, for example, would restore the view to the state shown in Figure 2c. In the expectation to return to this news site, the user bookmarks it.

(e) The user scrolls through the page and finds a headline containing the name Bertelsmann. Without leaving overview mode, the user follows the link by tapping it. (f) The article page is loaded and since it is too long to fit into the browser window, it too is displayed as an overview. To be able to read the story the user performs an *expand-and-zoom* gesture on the story area. (g) This removes all content outside the story area and switches to single-column view. The user can now advance through the story efficiently using the device's hardware scrolling buttons.

(h) The next day, the user invokes the bookmark created earlier and the browser loads the new edition of the page. However, the browser also restores the collapse-state the page was in when the bookmark was created. All headlines are therefore readable right away; no further user interaction is required. In addition, the page loads faster, as image material in the collapsed areas is not loaded.

The collapse-to-zoom mechanism described in this walkthrough minimizes the need for zooming and trial-and-error exploration. Instead of forcing users to iterate between overview and detail view, collapse-to-zoom allows users to make continuous progress towards the page content that is relevant to the user's task at hand. By collapsing elements deemed irrelevant users narrow down the page, thereby reducing the amount of content that has to be examined when finally switching to detail view. The need to scroll over irrelevant content is eliminated. In addition, collapsing content causes the remaining material to be re-rendered with increased detail, which often enables users to continue zeroing in on the relevant material.

With collapse-to-zoom, all page navigation is done up front (often using only a single *expand-and-zoom* action). Once users switch to single-column view, subsequent navigation reduces itself to scrolling, which most devices support conveniently with a set of hardware scroll buttons.

NAVIGATION USING A *MARQUEE MENU*

Navigation in collapse-to-zoom is based on a novel menuing technique what we call a *marquee menu*. Marquee menus are based on a *marquee selection.* Marquee selection is commonly used in image editing programs, spreadsheets (Figure 3b), etc. Dragging the pen on the screen creates a rectangular selection enclosing the start and end point of the drag gesture. Marquee menus also share properties known from *marking menus* [14], namely the use of a directional gesture for command selection.

The main strength of marquee menus is that they combine area selection and command selection into a single gesture. The user performs a marquee selection (visual feedback shown in Figure 3a) and when the user lifts the pen a command is executed. Which commands is executed is determined by the direction of the drag movement. Since there are four directions for selecting a given rectangular area (starting at any of four corners), marquee menus allow users to choose from up to four commands (Figure 3a).

The four commands of collapse-to-zoom's marquee menu are based on the following mnemonic scheme (Figure 3c). **Vertical:** *Up* means *expand*, *down* means *collapse*. Expand stand for "collapse everything outside this area", which effectively grows the selected area. **Diagonal:** The two indispensable functions are placed on the top-right/bottom-left diagonal, where right-handed users can access them conveniently by pivoting around the wrist. The main functions are *collapse column*, which is guaranteed to cause remaining screen content to grow, and *expand-and-zoom*, which expands the selected area and switches to full-size single column reading mode. The opposite diagonal offers cell-wise commands for manual fine tuning. Left-handed users can mirror the menu (as in Figure 2).

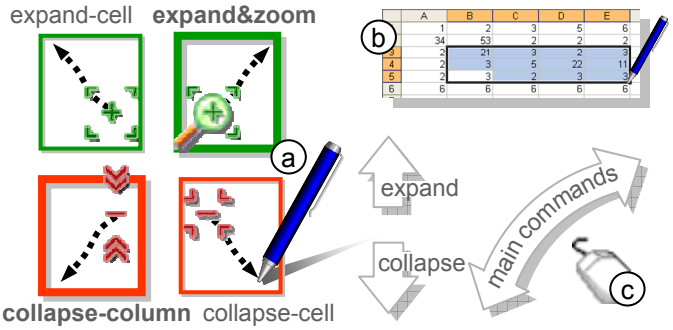

Figure 3: (a) The *marquee menu* and the commands it offers in collapse-to-zoom. Dragging the pen creates a selection. On pen lift the command associated with the drag direction is executed. During dragging, the icon of the upcoming command is shown; the lasso assumes the color red for collapse and green for expand commands. (b) Example of a marquee selection in a spreadsheet. (c) Mnemonics behind the arrangement of the four commands.

The fact that all the expand/collapse navigation is based on drag gestures allows collapse-to-zoom to use pen tapping for following links. Unlike most existing systems that require users to switch to detail mode first [9], this allows users to follow links directly from the overview.

Marquee menus allow users to abort a command by dragging the pen back to the starting point (marked by the

command icon) and lifting it. Besides that, our browser offers an *undo* button that undoes the last navigation command, and buttons that allow users to switch between overview, single column view, and 2D spatial view of the page (see the four right-most buttons in Figure 2; the other five buttons offer traditional browser navigation). Hitting the overview mode button when already in overview mode restores a page, i.e., restores all placeholders in it.

IMPLEMENTATION

We implemented collapse-to-zoom and its marquee menu as a simulation of a Compaq iPAQ that we run on a TabletPC. Compared to a PocketPC-based implementation, this provides us with a more powerful development platform and allows us to anticipate the graphics processing power that the GPU-supported small screen devices of the next generation will offer. The tablet also allows us to explore pen-based interactions.

Our implementation is based on the Microsoft Web-Browser Control that parses, renders, and scales the loaded pages (CSS zoom feature supported in Microsoft IE since version 5.5, see http://msdn.microsoft.com). This allows the prototype to load and render live web pages. Marquee gestures are implemented by intercepting mouse events and bypassing the function that normally highlights text while dragging. In its current version, our prototype is limited in that marquee gestures cannot be invoked on top of links.

Our prototype splits web pages into a set of tiles using the algorithm described in [9]. In a first pass, an iterative separator detection algorithm breaks the page down into tiles and transforms the page into a slicing tree based on the position information of each tile [17]. In a second pass, the slicing tree is transformed back into a web page. This is done by creating a nested table reflecting the structure of the slicing tree and pasting the tile content generated during the first pass into the individual table cells. The resulting page is displayed using the web browser control. The page now contains individual cells for each collapsible unit, so that collapse actions can be executed by replacing cells with placeholder contents. The lasso around selected cells is implemented by changing the cell border color and re-rendering the page.

CONCLUSIONS

Collapse-to-zoom is a technique that helps users view web pages on small screen devices. Our two main contributions are (1) the concept of collapsing irrelevant cells to cause the remaining content to be rendered at higher detail and (2) a single-stroke-based gesture language called *marquee menu*. Collapse-to-zoom supports the user's two main tasks: Marquee menu commands provide efficient access to relevant content located *inside* the page; link tapping directly in the overview allows users to instantly move on the content located *elsewhere*. Our experience so far is that the relevant content in most pages is arranged in the shape of a rectangle, which allows users to handle these pages with a single *expand-and-zoom* gesture. Pages that do not fall into this category can be handled by combining *collapse cell* and restore placeholder commands.

As future work we plan to run an experimental comparison between collapse-to-zoom and other web browsers for small screen devices. Furthermore, we plan to explore in how far the concepts described in this paper can be applied to the desktop, e.g., to remove advertising or to prepare pages for printing.

Acknowledgements

Thanks to Aaron Filner, Autumn Stroupe, and Henry Chen for their comments on our earlier design prototypes. Thanks also to Mary Czerwinski, George Robertson, and Ken Hinckley for their comments on a draft of this paper.

REFERENCES

1. Baudisch, P. Don't Click, Paint! Using Toggle Maps to Manipulate Sets of Toggle Switches. In *Proc. UIST'98*, pp.65–66.
2. Baudisch, P., Lee, B., and Hanna, L. Fishnet, a fisheye web browser with search term popouts: a comparative evaluation with overview and linear view. In *Proc. AVI'04*, pp. 133-140.
3. Bederson, B., Hollan, J., Perlin, K., Meyer, J., Bacon, D., and Furnas, G. Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics. *J. of Visual Languages and Computing*, 7, 3–31, 1996
4. Bickmore, T., and Schilit, B, Digestor: Device-Independent Access to the World Wide Web. In *Proc. Seventh Intl. WWW Conference 1997,* pp. 655–663.
5. Björk, S., Holmquist, L.E., Redström, J., Bretan, I., Danielsson, R., Karlgren, J., and Franzén, K. WEST: a web browser for small terminals. In *Proc. UIST'99*, pp. 187–196.
6. Buyukkokten, O., Garcia-Molina, H., Paepcke, A. Text Summarization for Web Browsing on Handheld Devices. *Transactions on Inf. Sys. 20*(1):82-115.
7. Buyukkoten, O., Garcia-Molina, H., Paepcke, A., and Winograd, T. Power browser: efficient web browsing for PDAs. In *Proc. CHI'00*, pp. 430–437.
8. Cai, D., Yu, S., Wen, J.R., and Ma, W.Y. VIPS: a vision-based page segmentation algorithm. *Microsoft Technical Report, MSR-TR-2003-79,* 2003.
9. Chen, Y., Ma, W.Y., and Zhang, H.J. Detecting Webpage Structure for Adaptive Viewing on Small Form Factor Devices. In *Proc. WWW'03*, pp 225–233.
10. Gupta, S., Kaiser, G., Neistadt, D., ad Grimm, P. DOM-based Content Extraction of HTML Documents. In *Proc. WWW 2003,* pp. 207–214.
11. Furnas, G. Generalized Fisheye Views. In *Proc. CHI'86*, pp. 16–23.
12. Gessler S., and Kotulla, A. PDAs as Mobile WWW Browsers. *Computer Networks and ISDN Systems 28*(1-2):53-59, 1995.
13. Hsu, J., Johnston, W., and McCarthy, J. Active Outlining for HTML Documents: An X-Mosaic Implementation. In *Proc. Second Intl. WWW Conference 1994*.
14. Kurtenbach, G. and Buxton, W. User Learning and Performance with Marking Menus. In *Proc. CHI'94*, pp. 258–264.
15. Milic-Frayling, N. and Sommerer, R. SmartView: Enhanced Document Viewer for Mobile Devices. *Microsoft Research Technical Report MSR-TR-2002-114*.
16. Wobbrock, J., Forlizzi, J., Hudson, S., Myers, B. WebThumb: interaction techniques for small-screen browsers. In *Proc. UIST '02,* pp. 205–208.
17. Chen, L.Q., Xie, X., Ma, W.Y., Zhang, H.J., Zhou H.Q., Feng H.Q., DRESS: A Slicing Tree Based Web Representation for Various Display Sizes. *Microsoft Research Technical Report MSR-TR-2002-126*.