Interacting with Big Interfaces on Small Screens: a Comparison of Fisheye, Zoom, and Panning Techniques

Carl Gutwin and Chris Fedak

Department of Computer Science, University of Saskatchewan
57 Campus Drive, Saskatoon, SK, S7N 5A9
{carl.gutwin, chris.fedak}@usask.ca

Abstract

Mobile devices with small screens are becoming more common, and will soon be powerful enough to run desktop software. However, the large interfaces of desktop applications do not fit on the small screens. Although there are ways to redesign a UI to fit a smaller area, there are many cases where the only solution is to navigate the large UI with the small screen. The best way to do this, however, is not known. We compared three techniques for using large interfaces on small screens: a panning system similar to what is in current use, a two-level zoom system, and a fisheye view. We tested the techniques with three realistic tasks. We found that people were able to carry out a web navigation task significantly faster with the fisheye view, that the two-level zoom was significantly better for a monitoring task, and that people were slowest with the panning system.

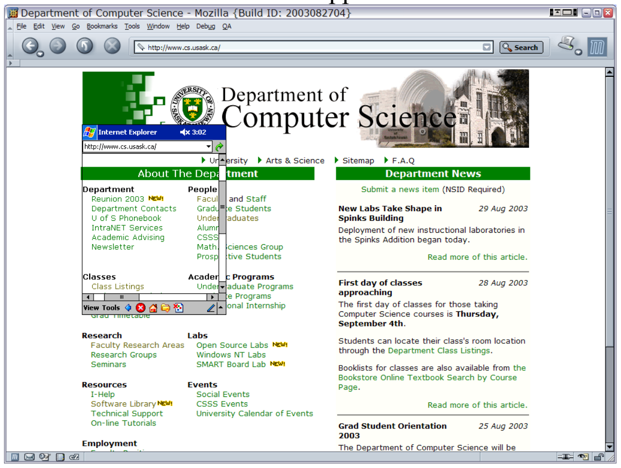
Key words: Large interfaces, small screens, mobile devices, screen space, zoom and pan, fisheye views.

1 Introduction

Many interactive applications have complex interfaces that consume considerable screen space. Toolbars and tool palettes, help windows, outline views, and status bars all take up space, even without considering the workspace and data. However, there are times when applications cannot be run on a screen that comfortably contains the interface. This problem becomes acute when using mobile and handheld computers such as personal digital assistants (PDAs) or sub-notebooks. These kinds of systems are becoming more and more common, and will soon have (or already do have) the power to run complex interactive applications [6].

The form factors used for these mobile devices constrain their screens to be only a fraction of what is available on the desktop (see Figure 1). For example, a 320-pixel by 240-pixel display of a PocketPC system has less than 1/16 the screen area of a 1280x1024 screen, and only 1/25 of a 1600x1200 screen. Although the resolution of PDA screens is increasing, limitations on the physical size of the screen will prevent these devices from ever reaching parity with the desktop.

This means that large-interface applications cannot be adequately shown on small screens. There are two ways to solve this problem. First, applications can be redesigned for the smaller screen. Although there are examples of this approach (e.g., Pocket Word or Internet Explorer for PocketPC devices), it requires that multiple versions of the application be produced, and requires that users become familiar with a second GUI. There will also be cases where no redesigned version of an application is available—so another approach is needed.


Department of Computer Science - Mozilla {Build ID: 2003082704}
File Edit View Go Bookmarks Tools Window Help Debug QA
http://www.cs.usask.ca/
Search
Internet Explorer
3:02
http://www.cs.usask.ca/
View Tools
Department of Computer Science
University
Arts & Science
Sitemap
F.A.Q

About The Department

Department
Reunion 2003 NEW!
Department Contacts
U of S Phonebook
IntraNET Services
Academic Advising
Newsletter

People
Faculty and Staff
Graduate Students
Undergraduates
Alumni
CSSS
Math. Sciences Group
Prospective Students

Classes
Class Listings
Grad Timetable

Academic Programs
Undergraduate Programs
Graduate Programs
Professional Internship

Research
Faculty Research Areas
Research Groups
Seminars

Labs
Open Source Labs NEW!
Windows NT Labs
SMART Board Lab NEW!

Resources
I-Help
Software Library NEW!
Technical Support
On-line Tutorials

Events
Social Events
CSSS Events
University Calendar of Events

Employment

Department News

Submit a news item (NSID Required)

New Labs Take Shape in Spinks Building *29 Aug 2003*
Deployment of new instructional laboratories in the Spinks Addition began today.
Read more of this article.

First day of classes approaching *28 Aug 2003*
The first day of classes for those taking Computer Science courses is **Thursday, September 4th**.
Students can locate their class's room location through the Department Class Listings.
Booklists for classes are also available from the Bookstore Online Textbook Search by Course Page.
Read more of this article.

Grad Student Orientation 2003 *25 Aug 2003*
The Department of Computer Science will be

Figure 1. A 240x320 PDA browser overlaid on a 1024x768 desktop browser.

The second solution, and the one we concentrate on here, is to use the small screen as a window onto a larger virtual space that is the size of a desktop display. This allows applications to maintain their interfaces unchanged, but requires that users navigate the UI through a small window. This second solution can also be seen in existing systems: for example, many laptops allow the display to be set to a resolution higher than what the screen can show; the user can then make use of the larger space by panning the screen with the mouse.

Although there has been considerable research into mechanisms for navigating documents or 2D data spaces, GUIs present a new type of problem in that the navigation mechanism must also support all the interaction techniques that are used in the interface. For example, any small-screen system must support targeting, steering, scrolling, selection, dragging, and drawing, in addition to moving the view around.

In this paper, we report on an exploratory study that compared three navigation methods: a panning system similar to that used on laptops; a two-level zoom showing either an overview or a fully-zoomed-in view; and a fisheye view that provides both an overview and a detail region. We had participants carry out three realistic tasks that included a range of interaction styles: editing a presentation, navigating a set of web pages, and monitoring a control panel for particular events.

We found that for all three tasks, people performed poorly with the panning view. It was clear that the tasks were difficult without the global context provided by the other techniques. Performance with the two-level zoom and the fisheye view was generally significantly better, although these effects depended on the task.

Below we review issues related to large data spaces on small screens, then report on the methods and results of the experiment, and lastly discuss how our findings can be used in the design of small-screen systems.

2 Large spaces and small screens

2.1 2D navigation mechanisms

The general problem of how to deal with data that does not fit on a single screen is well-known in HCI research. The main focus has been on navigating large documents and 2D data spaces; work has been done to develop different representations and navigation mechanisms, and also to compare different techniques against one another. Four approaches that have been identified are panning, zooming, multiple views, and focus+context.

Panning. The earliest systems that were able to navigate a large space did so by scrolling, panning, or paging, all of which either move a fixed viewport over the document, or move the document under the viewport. Studies have compared different mechanisms in a variety of task situations (e.g., [10,12]). One result shows that for touch screens, panning by dragging the background was superior to either pushing the viewport or touching the window border [10].

Zooming. When documents are particularly large, scrolling alone becomes tedious. A number of systems have implemented the idea of zooming, both to speed navigation and also to provide multiple perspectives on the data. For example, Pad++ is a toolkit that allows infinite zoom and pan in a 2D space [3]. Zoom can either be dynamically controlled by the user, or can use a set of levels that have some meaning to the task at hand. Systems where zooming is used to provide an overview of the space have been shown to perform better than basic scrolling systems (e.g. [12]).

Multiple views. Although zooming allows people to look at the data from any distance, it allows only one type of view at a time. It is often useful to have more than one of these perspectives available concurrently, and this is provided in multiple view approaches. The most common of these provides an overview that shows the entire data space in miniature, and a detail view that shows a portion of the data at normal size. The overview often shows a viewfinder to indicate the current location of the detail view, and often allows navigation by dragging this viewfinder [19]. The overview and detail approach has been shown to outperform both panning and fisheye representations for some tasks, and is easy for users to understand [9].

Focus+context views. A variation on the multiple-view approach is to provide both overview and detail in the same window. Focus+context techniques include fisheye views [17], bifocal displays [16], and table lenses [14]. Fisheye systems have been studied in a number of contexts, and have been shown to perform well for navigating networks [18], for using menus [2], and for large tracing and steering tasks [7]. However, distortion-oriented techniques can also cause problems for some interactions such as targeting [8].

2.2 Small-screen design strategies

Some of the above techniques have been applied to small-screen displays (e.g. [1,2]), but design for mobile devices has more often looked at changing the representation of the interface or the data. Past work on the interface side has explored transparent interface objects [11], the use of enhanced interaction techniques such as tilting to make up for reduced screen space [15], or the use of sound to reduce the dependence on visual data [4]. Changes to the data often involve creating summaries of documents that fit more easily on the small screen (e.g. [5]), or using context and location awareness to show a subset of the data space (e.g. [13]).

These techniques for redesigning the interface or the presentation of the data to fit a small screen are valuable, but only in situations where redesign is possible, where the original interface is not too large, and where users are willing to learn the new UI and the new interaction techniques. For scenarios in which it is useful to maintain the layout and interactivity of the original system, however, there is a scarcity of research about how different navigation approaches like zoom and pan or focus+contex might affect performance and satisfaction.

To investigate this problem in more detail, we carried out a exploratory experiment with three different mechanisms for navigating a large interface on a small screen.

3 Study methodology

3.1 Participants

Fifteen people (fourteen male, one female) were recruited from a local university and were given course credit for participating in the study. All participants were frequent users of mouse-and-windows based systems (> eight hrs/wk). All had seen both panning and zooming systems, but none had prior experience with fisheyes.

3.2 Apparatus and Study Conditions

The experiment was conducted on a P4 Windows XP PC, running a custom-built C++ application. The system used two monitors, one large (41cm x 30cm, 1600x1200 resolution) and one small (26cm x 19cm, 1024x768 resolution). The large display was used for the source window of an application's interface, and the small display was used to present the small-screen version. To simulate different small display sizes, a window was displayed on the small screen. Therefore, no actual handheld devices were used – all study conditions were simulated on this apparatus. For the small-screen conditions, participants could only see the small monitor (Figure 2).

Three small-screen techniques were tested in the study: a panning system, a two-level zoom, and a fisheye view. We did not test any multiple-view techniques because of the extra screen space they require. All the techniques are shown in Figure 3 below.

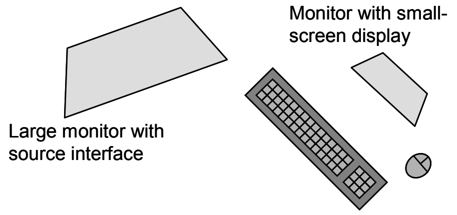

Monitor with small-screen display
Large monitor with source interface

Figure 2. Experiment setup.

Panning. This technique implements the 'sliding window' used by some laptop computers. The system shows only a portion of the source screen, but at full size. This viewport moves whenever the user pushes their mouse pointer on the boundary of the window; the amount of view movement is equal to the amount of mouse movement. This allows much faster panning than other techniques (such as autoscroll or dragging the background), because the motion of the mouse is not constrained to the size of the screen.

Two-level zoom. The two-level zoom provides users with two different magnifications: an overview, which shows the entire screen in a reduced form; and a zoomed view, which behaved in the same way as the panning system. Users can switch between the two magnifications using a key combination (Control-F1). When zoomed out, the system shows a viewfinder rectangle on the screen to indicate what will be visible after zooming in.

Fisheye view. The fisheye presents an overview of the entire large screen, like the two-level zoom, but also includes a detail region inset into the same view. This region uses nonlinear magnification to provide a smooth transition from the de-magnified surrounding context to the magnified focus area. The system used a flat-topped pyramid lens built with the Idelix PDT library (www.idelix.com). The lens was always centred around the mouse cursor, and the magnification at the focus of the lens was set to show the source screen at normal size.

All three techniques actually did operate the large-screen system; when participants carried out interface actions with the small screen, they worked normally as they would with the large interface. The small-screen techniques were all implemented using the following general algorithm: first, an image of the large screen was captured to memory; second, transformations were applied according to the type of technique (e.g. cropping, scaling, or adding the fisheye effect); third, the image was copied to the window of the simulated small-screen device. The steps are repeated as often as possible; with our study setup, the system showed about 10 frames per second. This meant both panning and the motion of the fisheye lens were noticeably slow, but still usable.

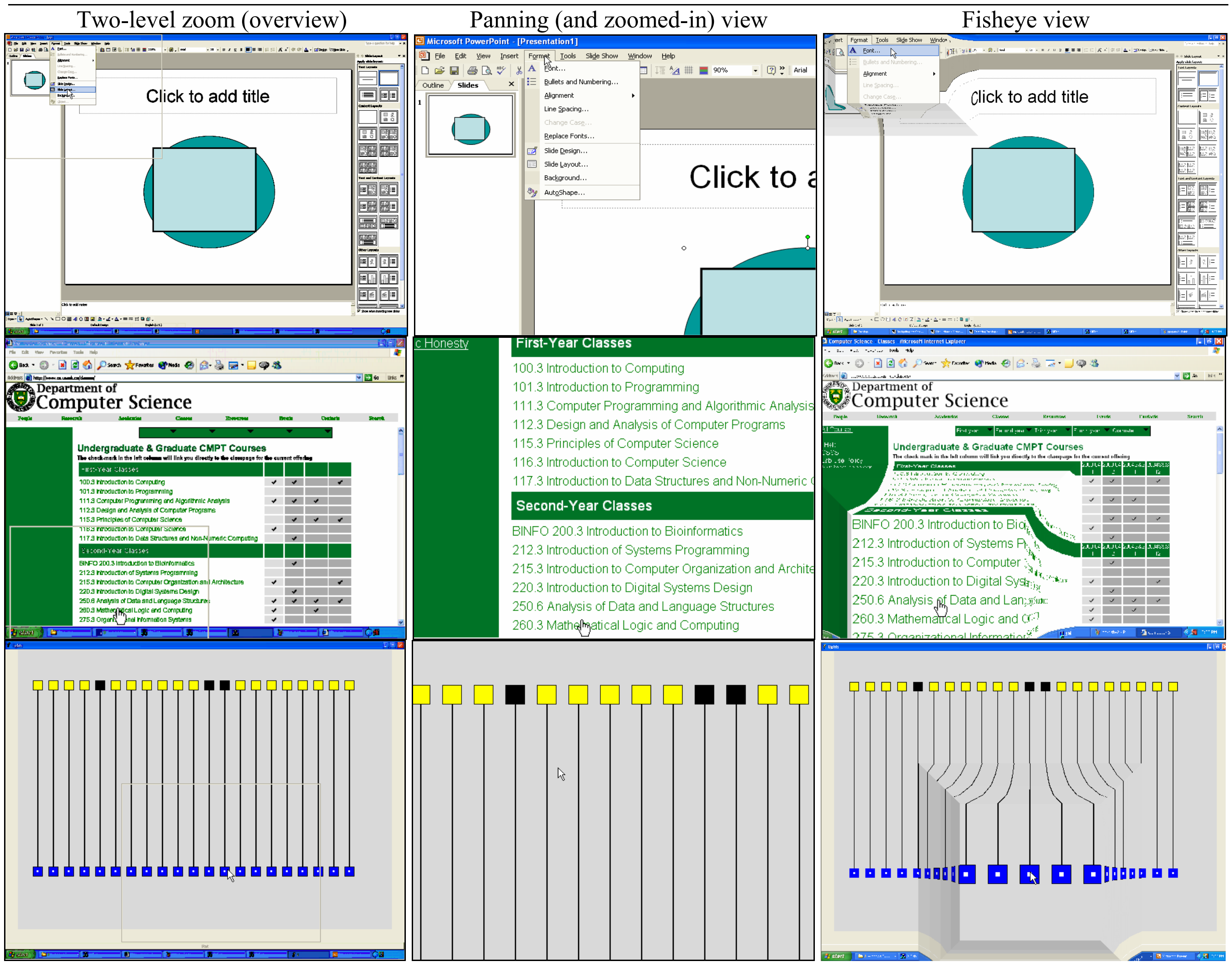
3.3 Tasks and measures

We chose tasks and task situations that were realistic enough to reflect real-world performance and usability issues. We focused on tasks where the user is already familiar with the interface, as in the case of a user who frequently uses a desktop version of an application, but who on occasion must work from a mobile device. We chose three tasks that covered a range of pointer-based interaction techniques: editing a document, navigating the web, and monitoring a control panel. Screens from each tasks are illustrated in Figure 3. For each task, we chose screen sizes for both large and small screens that were appropriate for the task scenario (see Figure 4). Our experiment only looked at selected situations in the design space rather than a full crossing of independent variables, in order to get an initial broad range of experience with the different possible size combinations.

Editing task

This task involved using a standard office application (Microsoft PowerPoint) to create a presentation document and add objects to a slide. The usage scenario is that of someone needing to edit a presentation while out of the office. Participants followed a set of instructions that had them carry out a set of actions using the menus and toolbars of the application: e.g., create a file, change the layout of a slide, add a rectangle and a circle of particular sizes to the slide, select and change the colour of certain objects, and save the updated presentation.

The interaction techniques used in this task included: finding and selecting icons and menus, drawing shapes (by dragging the mouse), selecting data objects on the slide, and aligning slide elements. The experimenter watched the session and ensured that all steps were completed correctly. No shortcuts or additional steps were allowed. The primary measure recorded for this task was the time to complete the steps.



First-Year Classes

100.3 Introduction to Computing
101.3 Introduction to Programming
111.3 Computer Programming and Algorithmic Analysis
112.3 Design and Analysis of Computer Programs
115.3 Principles of Computer Science
116.3 Introduction to Computer Science
117.3 Introduction to Data Structures and Non-Numeric

Second-Year Classes

BINFO 200.3 Introduction to Bioinformatics
212.3 Introduction of Systems Programming
215.3 Introduction to Computer Organization and Archite
220.3 Introduction to Digital Systems Design
250.6 Analysis of Data and Language Structures
260.3 Mathematical Logic and Computing

Figure 3. Small-screen display types and tasks used in the study. From top: editing task, navigation task, monitoring task.

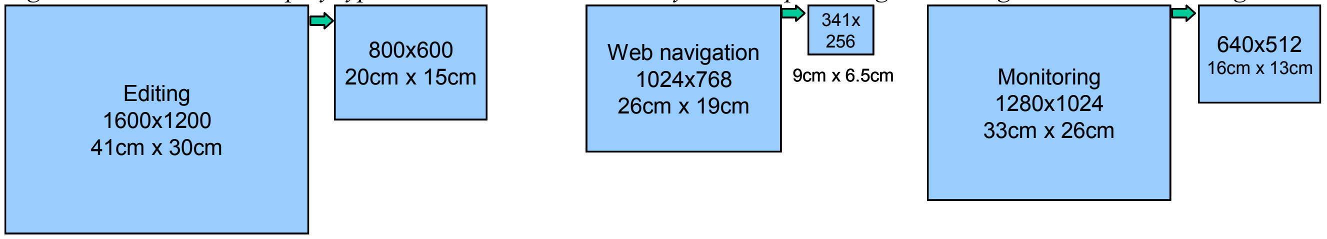


Figure 4. Relative screen sizes for source interfaces and small-screen versions. All small interfaces appeared as windows on the small (26cm x 19cm) monitor.

Navigation task

This task used a standard web browser (Internet Explorer) and asked participants to visit a given sequence of links as quickly as possible. The scenario for this task is that of a student who wants to check for updates to a set of pages: announcements on class pages for the courses she has that day, new job postings, and the seminar schedule for that afternoon. This is a task that would be done regularly, and in a situation where the user had only a mobile device (e.g. riding the bus). In the task, participants were given verbal instructions about which web link or browser function to select next. In total, there were 18 different actions to be performed. Pages were cached locally for consistent retrieval times.

Interaction techniques involved in this task are: finding and selecting text links, scrolling, constrained visual search to find particular information, and finding and clicking buttons in the browser's toolbar. When an incorrect link was selected, the participant had to repair the error and continue. The primary measure was the time to complete the tour through the links.

Monitoring task

This task used a custom Tcl/Tk application to display a control panel for a simulated real-time device array. The user's job was to monitor the panel for failures, and then restart failed devices. Unlike the other two tasks, this used an artificial simulation. Although there are many real-world situations where users monitor dynamic displays (e.g., instant messaging, live web content, interactive games, command-and-control systems), we chose to regulate the progress of the monitoring activity and the number of events with an artificial system. Our work scenario for this task is that of a control-room operator who must periodically visit different locations on the work site, but must continue monitoring events using a handheld computer.

In the task, participants monitored a display showing 21 devices and their associated restart switches. Participants worked for two minutes, watching for any device alarms (shown by a change from a yellow status icon to black). When a device failed, the user had to operate the corresponding switch by clicking an on-screen button with the mouse. Failures occurred randomly with a 1% probability (per device), calculated every second. On average, there were about 24 failures over the 120 seconds of the task. This task involved visual monitoring, determination of which switch belonged to which device, and finding and targeting particular switches. The primary measure was the amount of time needed to attend to failures.

3.4 Procedure

Participants were randomly assigned to one of three order groups that determined which technique they would use first in each task (tasks were done in the same order). Participants were then introduced to the study, the tasks, and each of the small-screen techniques. For each task and each technique, the participant first practiced until all task elements could be carried out successfully, and then did the task on the large screen to get a baseline completion time. The task was then carried out again with each of the three study systems, with rest provided between tasks.

After each task, participants completed a short questionnaire asking them which system they preferred for that task, considering both speed and accuracy.

3.5 Study design

The study used a 3x4 within-participants factorial design. The factors were task type (editing, navigating, or monitoring) and display type (normal screen, panning, two-level zoom, or fisheye). Order was balanced so that each task and each small-screen system was seen in each position equally. With 15 participants carrying out three tasks with four display techniques (including the normal screen), 180 data points were collected.

4 Results

We first compare performance on the small-screen interfaces to that on the normal screen, and then look more closely at small-screen performance for each task. We carried out separate analyses by task, since each task had a different expected completion time.

4.1 Small screen vs. large screen

A preliminary analysis was carried out to compare performance on the small-screen interfaces and the normal-sized screen. An overall analysis of variance (ANOVA) was carried out with the data from the normal view included. As expected, there was a large main effect of display type ($F_{3,42}$=88.55,p<0.001). Follow-up analyses showed significant differences between the normal view and each of the small-screen displays, in every task (all p<0.005). The charts below include the normal view data for comparison, although analysis from this point on considers only the three small-screen views.

4.2 Comparison of small-screen displays

Editing task

ANOVA showed a main effect of display type on completion time ($F_{2,28}$=11.04,p<0.001). Subsequent one-way analyses between each pair of conditions showed that the fisheye and two-level zoom were significantly faster than panning (for the fisheye, $F_{1,14}$=9.26,p<0.01; for the two-level zoom, $F_{1,14}$=20.86, p<0.001). There was no difference between the fisheye and two-level zoom ($F_{1,14}$=1.66,p=0.218). Mean completion times for all display types are shown in Figure 5.

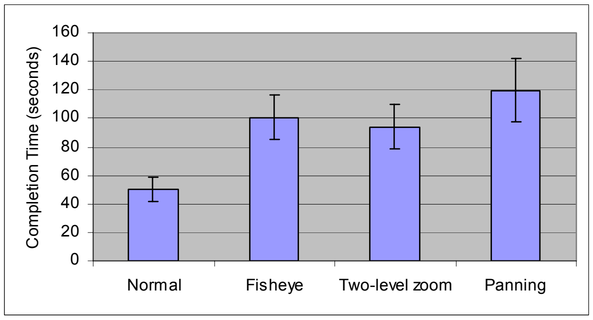


Figure 5. Mean completion times for the presentation editing task. Error bars show standard deviation.

Navigation task

A main effect of display type was also found for the web navigation task ($F_{2,28}$=6.66,p<0.005). Follow-up analyses showed that the fisheye technique was significantly faster than either the two-level zoom ($F_{1,14}$=17.57, p<0.005) or the panning system ($F_{1,14}$=6.77,p<0.05). There was no difference between panning and the two-level zoom ($F_{1,14}$=0.26,p=0.62). Completion times are shown in Figure 6.

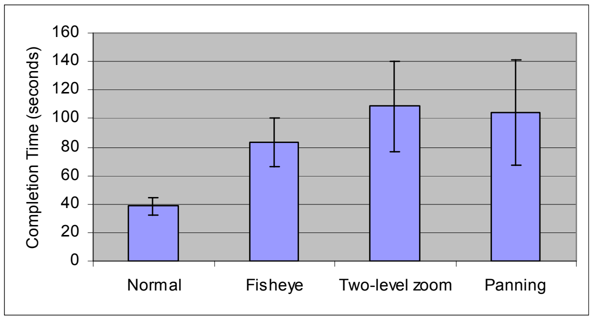


Figure 6. Mean completion times for the web navigation task. Error bars show standard deviation.

Monitoring task

A main effect of display type was also found for the monitoring task ($F_{2,28}$=26.65,p<0.001). In this task, the two-level zoom was shown to be better than either the fisheye ($F_{1,14}$=7.53,p<0.05) or the panning system ($F_{1,14}$=30.68,p<0.001). In addition, the fisheye was faster than panning ($F_{1,14}$=25.53, p<0.001). Again, completion times are shown in Figure 7. With the two-level zoom, people were able to restart failures about three seconds faster than with the fisheye, and almost 14 seconds faster than with the panning technique.

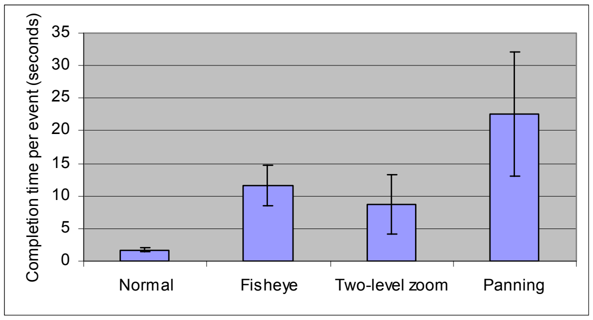


Figure 7. Mean completion times for the monitoring task. Error bars show standard deviation.

4.3 Preferences

At the end of the session, we asked participants which small-screen display they preferred. As shown in Figure 8, most people chose the two-level zoom – even for the task where it had the slowest completion time.

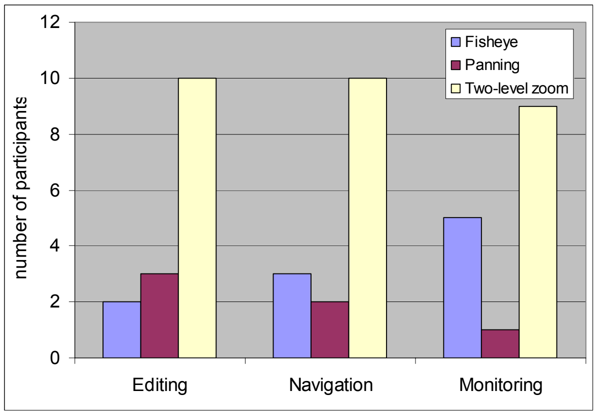


Figure 8. Participant preferences.

5 Discussion

The study showed that there can be substantial performance differences depending on the interface navigation technique used on the small-screen system. In particular, the fisheye and the two-level zoom were significantly faster than the panning method in all cases except for web navigation, where two-level zoom was also slow. In this task, the fisheye view was better than the two-level zoom, and in the monitoring task, the two-level zoom was better than the fisheye. In the following paragraphs we consider explanations for these results, and discuss how our findings generalize.

5.1 Explaining the differences

Speed of the panning system

There were four reasons that participants took longer with the panning system: the time required to move the viewport, the costs of finding the correct location, the problem of missing events that happen outside the viewport, and the problem of scrolling the document.

- Although the viewport could be moved quickly with a large mouse move, the total amount of movement was necessarily larger with this system than the overview-based methods (since these allow movement at the reduced scale of the overview).
- Since the target of a move was often outside the viewport, participants made errors in the direction they moved. Although they were easily able to correct these errors, the extra movement took time.
- In the monitoring task, it was clear that not having an overview hindered participants as they tried to watch the control panel for failures. People had to resort to moving back and forth across the large interface in order to notice changes in device icons.
- In several cases, participants got confused between navigating the screen and the actual document (i.e.

remembering to use the scroll bars). In the web task (where the data occasionally required scrolling), people would sometimes pan to the bottom of the interface, but then wonder why they still couldn't see the links that they were looking for (which were in the document but 'below the fold').

Differences in the navigation task

The performance difference between the fisheye and the two-level zoom in web navigation can be attributed to the two-level zoom, since the fisheye view performed similarly in relation to the panning view as it did on other tasks. Our observations suggest that people had to do more zooming in this task, which may have caused problems that slowed them down.

Switching back and forth between the overview and the zoomed-in view incurs costs that are not present in the fisheye. The more switching that is required, the more time will be needed by the two-level zoom. In the other two tasks, people were often able to carry out most of the task without zooming in at all; but the smaller size (and greater reduction) of the web navigation task made the web links too small to read in the overview. We saw three types of switching costs: adjusting to the new context, remembering to switch, and performing the switch itself.

- Changing zoom levels requires that users adjust to the new view – for example, they must find the cursor and reorient themselves to where they are in the interface.
- Switching between modes requires that the user remember to do so, and we noticed that some users did not zoom optimally. When people were concentrating on the task, they would sometimes stay at the zoomed-in view longer than was strictly necessary – that is, longer than they needed to read and select the text link.
- The switch was performed with a key combination, a mechanism that required effort. People would often remove their fingers from the keyboard, and when switching had to be performed, it was clear that looking away from the screen and finding the correct keys took extra time.

Differences in the monitoring task

The differences between the fisheye and the two-level zoom in the monitoring task appeared to be related primarily to targeting. For each device failure, the user had to target the associated switch – accurately positioning the cursor over the on-screen button, and ensuring that it was the correct one among several that looked similar.

The fisheye view caused difficulties in both aspects of the task. First, targeting with a fisheye lens can be difficult because the magnification makes it appear that objects under the lens move as the lens itself comes closer. This can cause people to overshoot the target, since objects move just as the cursor arrives, and we observed some participants having this difficulty.

Second, the 'wires' connecting devices to switches showed the distorting effects of the fisheye, particularly on the shoulder of the lens (see Figure 3). As the participant moved towards the switches, this distortion and the magnification effect just described could make it more difficult for people to keep track of which switch lined up with which device icon.

5.2 Generalizing the results

The generality of our results can be considered in terms of the tasks, the implementation of the techniques, and the physical characteristics of real small-screen devices. The realism of our study tasks provides good evidence that the results can be generalized to real-life situations where people are familiar with the large interface. The editing and navigating tasks in particular already involve real applications, real data, and a large subset of the interactions that will occur in most applications. There may be additional differences, however, in tasks that involve other interaction styles (e.g. reading of text, visual search).

Although we believe that the differences between the individual techniques will also be apparent in other situations, it is possible that these differences could be reduced by changing the way that the techniques were implemented. For example, targeting in the fisheye can be improved by coupling the lens's magnification to the speed of the cursor [8]; similarly, a simpler and more ready-to-hand mechanism for switching levels in the zoom system could improve performance in situations where switching is frequent.

Finally, there are various limitations in the physical devices themselves that may affect the navigation scheme. For example, in some stylus-based systems, there is no notion of moving the pointer without dragging, because the screen cannot detect the stylus unless it is pressed onto the screen (others detect the stylus a few centimeters away). This makes moving a fisheye lens or viewfinder more difficult, and these techniques may require modification to work properly with this type of input system. Similarly, panning with stylus systems often entails either dragging the background or using a second control, since the stylus is restricted to the visible screen area. This would likely cause further problems for the panning technique. However, there are also many devices where the input systems are similar to our simulation (e.g. sub-notebooks with ordinary mice), and where our results should hold.

6 Conclusions and future work

We carried out an experiment to compare different methods of navigating a large interface through a small screen. There are five main findings from the study that can be used in designing such systems:

- even at its best, navigating on the small screen is considerably slower than on the normal screen;
- overviews of the entire interface are valuable, both because the global context allows faster navigation and because many interactions can be carried out at the overview level;
- fisheye views and two-level zoom systems are both effective techniques for navigating interfaces, particularly if targeting (in the fisheye) and view-switching (in the two-level zoom) can be improved;
- for tasks where movements are large or events happen elsewhere on the screen, panning strategies perform poorly and are disliked by users;
- users like the two-level zoom, even when its performance is less than optimal.

In future, we will both expand on the results reported here, and investigate the techniques in a wider variety of usage situations. First, the performance of the overview techniques we tested can be improved – perhaps dramatically, if we move the processing to a pixel shader program on the graphics card. They can also be better tuned to provide more appropriate magnification levels – for example, our observations that people do not need full size in order to carry out tasks could allow a fisheye view to get by with considerably less magnification (and thus fewer problems from distortion). Second, we want to look more closely at issues and problems raised by having to interact with an interface component (e.g. a menu) and navigate the view at the same time. Third, we also plan to consider these techniques in situations where the reduction in size from the original interface is not so extreme – for example, we plan to determine whether it possible to have an effective and usable 1600x1200 desktop on a 1280x1024 monitor.

References

[1] Bederson, B., Czerwinski, M., and Robertson, G., A Fisheye Calendar Interface for PDAs: Providing Overviews for Small Displays, *University of Maryland HCIL Tech Report #HCIL-2002-09*, 2002.

[2] Bederson, B., Fisheye Menu Selection, *Proc. ACM UIST 2000*, 217-225.

[3] Bederson, B., and Hollan, J., Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics, *Proc. ACM UIST 1994*, 17-26.

[4] Brewster, S., Leplatre, G., and Crease, M., Using Non-Speech Sounds in Mobile Computing Devices, *Proc. 1st Workshop on Mobile HCI*, Glasgow, 5-14.

[5] Buyukkokten, O., Garcia-Molina, H., Paepcke, A., and Winograd, T., Power Browser: Efficient Web Browsing for PDAs, *Proc. ACM CHI 2000*, 430-437.

[6] Fitzmaurice, G., Zhai, S., Chignell, M., Virtual Reality for Palmtop Computers, ACM ToIS, 11,3, 1993, 197-218.

[7] Gutwin, C., and Skopik, A., Fisheye Views are Good for Large Steering Tasks, *Proc. ACM CHI 2003*,

[8] Gutwin, C., Improving Focus Targeting in Interactive Fisheye Views, *Proc. ACM CHI 2002*, 267-274.

[9] Hornbaek, K., and Frokjaer, E., Reading of Electronic Documents: The Usability of Linear, Fisheye, and Overview+Detail Interfaces, *Proc. ACM CHI 2001*, 293-300.

[10] Johnson, J., A Comparison of User Interfaces for Panning on a Touch-Controlled Display, *Proc. ACM CHI 1995*, 218-225.

[11] Kamba, T., Elson, S., Harpold, T., Stamper, T., Sukaviriya, P., Using Small Screen Space More Efficiently, *Proc. ACM CHI 1996*, 383-390.

[12] Kaptelinin, V., A Comparison of Four Navigation Techniques in a 2D Browsing Task, *Proc. ACM CHI 1995*, 282-283.

[13] Pascoe, J., Ryan, N., and Morse, D., Using while Moving: HCI Issues in Fieldwork Environments, ACM ToCHI, 7,3, 2000, 417-437.

[14] Rao, R., and Card, S., The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information, *Proc. ACM CHI 1994*, 318-322.

[15] Rekimoto, J., Tilting Operations for Small Screen Interfaces, *Proc. ACM UIST 1996*, 167-168.

[16] Robertson, G., and Mackinlay, J., The Document Lens, *Proc. ACM UIST 1993*, 101-108.

[17] Sarkar, M., and Brown, M., Graphical Fisheye Views of Graphs, *Proc. ACM CHI 1992*, 83-91.

[18] Schaffer, D., Zuo, Z., Greenberg, S., Bartram, L., Dill, J., Dubs, S., and Roseman, M., Navigating Clustered Networks through Fisheye and Full-Zoom Methods, *ACM ToCHI*, 3,2, 1996, 162-188.

[19] Smith, R., Hixon, R., and Horan, B., Supporting Flexible Roles in a Shared Space, *Proc. ACM CSCW 1998*, 197-206.