

Wideband Visual Interfaces: Sensemaking on Multiple Monitors

Jock D. Mackinlay, Jeffrey Heer, and Christiaan Royer

Palo Alto Research Center

3333 Coyote Hill Road, Palo Alto, CA 94304

{mackinlay, jheer, royer}@parc.com

ABSTRACT

Although vendors have made multiple-monitor systems for many years, our interfaces have been stuck in a 30-year old windows paradigm focused on displays much smaller than the desktops we use when working with paper. Advances in flat panel displays and graphics cards now enable affordable personal computers with 6-8 monitors and may someday eliminate seams. This paper argues that vendors should be developing wideband visual interfaces that are designed for displays that fill the human visual field. We describe a longitudinal field study of window activity that found that windows almost always filled a typical single monitor display and that subjects occasionally struggled with window thrashing when they needed to work with two or more windows at the same time. Vendors need not wait for affordable seamless wideband displays before addressing these findings. We have implemented several novel user interface techniques for creating seam-aware applications that target wideband displays based on multiple monitors.

Author Keywords

Seam-aware visual interfaces, wideband displays, window activity, multiple monitors

ACM Classification Keywords

H5.2. Information interfaces and presentation (e.g., HCI): User Interfaces *Graphical user interfaces (GUI), Screen design (e.g. text, graphics, color), Windowing systems.*

INTRODUCTION

Human limitations, such as fixed working memory and sequential information assimilation, cause cognition to scale poorly as the amount and complexity of information increases in our lives. We often cope with large-scale cognitive tasks through “external cognition,” using desks, maps, documents, and computers to hold and provide information as needed [8].

Sellen and O’Hara describe field and lab studies of knowledge workers engaged in reading and writing (a sensemaking activity) that found that the spatial layout of documents was used to gain a sense of the overall structure of documents, to cross-reference documents, and to interleave reading and writing [7,9]. Paper documents, in particular, were laid out in space to visualize a great deal of

information, placed in juxtaposition for cross-referencing, and organized into independent reading and writing spaces that could be accessed concurrently and manipulated independently [7].

These findings suggest that workspace size can impact the effectiveness of external cognition. Although people can occasionally use airplane tray tables to work successfully with paper documents, they typically use desks and sometimes need a dining table or something larger. However, typical computer workspaces are no larger than airplane tray tables. The Sellen and O’Hara lab study found that subjects working with online documents experienced lost resolution through zooming documents, overlapping windows, and difficulties integrating reading and writing [7]. They concluded that multiple monitors might improve the reading and writing of online documents.

However, few users have multiple-monitor computers, even though such products have existed for many years. This is surprising—a field study by Grudin found that multiple monitor use confers many benefits, including peripheral awareness and improved access to resources [4]. Furthermore, Czerwinski et al review many studies that also indicate benefits for increased display size [2]. Their recent study found that a large research display had significant benefits over a standard LCD monitor for complex multi-application computer tasks. Grudin, in particular, suggested that one reason for the lack of interest in multiple monitors is that they do not connect seamlessly [4]. “A window opened across two monitors is interrupted.” He also found that applications do not make good use of multiple monitors. Dialog boxes are placed randomly, sometimes across seams. Grudin concluded that there are opportunities for user interface designs that take advantage of multiple monitor scenarios.

This paper argues for the development of wideband visual interfaces that fill the human visual field. Clearly, given the disparity in size of paper and online workspaces and the paucity of multiple monitor computers, interactive user interfaces have done much to reduce the impact of small displays, at least for routine tasks. However, we believe that users still have problems with small displays. In particular, large-scale cognitive tasks unbalance the amount of information relative to the workspace size—increasing the overhead associated with moving windows, scrolling

them, resizing them, or bringing them to the top—resulting in a rapid, non-linear drop in the usability of a window system [5]. The impact is similar to the thrashing that occurs in virtual memory operating systems when an application needs more data than fits simultaneously in main memory [10].

We describe a longitudinal field study of window activity that found what appears to be window thrashing on a typical single-monitor computer used for sensemaking tasks. Furthermore, the sensemaking windows almost always filled the display, suggesting the need for larger displays. Although researchers are experimenting with seamless wideband displays [2], advances in flat panel displays and graphics cards now enable affordable wideband displays that use multiple monitors to fill the human visual field. However, the resulting seams can disrupt windows. To address the findings of the study with current display hardware, we argue that windows are typically container spaces or metric spaces. Seams disrupt each type of space differently. We have implemented novel user interface techniques that mitigate seam disruption for these two cases, supporting the creation of seam-aware applications for multi-monitor wideband displays.

LONGITUDINAL FIELD STUDY OF WINDOW ACTIVITY

Background

Novel Intelligence from Massive Data (NIMD) is a government research program focusing on information analysts and how to improve their work. A notable feature of this program is the Glass Box Analysis environment, custom software that instruments Microsoft Windows to capture a large amount of data including mouse events, keyboard events, window events, documents, queries, notes, and the topics the analysts are researching. A goal of the NIMD program is to use the Glass Box to capture the activity of information analysts both before and after the introduction of new technologies.

Method

The longitudinal field study of window activity described in this paper is based on the baseline activities of two highly-skilled information analysts paid to do their typical work, which is using open source material obtained from the Internet to write reports. The subjects are called Analyst 3 & 4 (more analysts are planned in the NIMD program). The Glass Box environment was running on a standard PC in a typical office setting with Internet access and one 1280x1024 monitor. Human subject guidelines were followed in the development and deployment of the Glass Box. In particular, subjects can use the Glass Box application to protect their privacy by turning the capture off and on at any time, which groups the data into sessions.

The data used in the paper was captured from April 16 to July 31, 2003 and stored in an SQL database (see Table 1). The activities of Analyst 3 were captured on 33 days (more than 106 hours) and the activities of Analyst 4 were

captured on 52 days (more than 269 hours). Given the volume of data captured by the Glass Box, we focus here on window events, which describe when windows are created or destroyed and when they change their size, position, or title. Window scrolling is not noted by window events but may someday be inferred from the mouse events also captured by the Glass Box.

Given the quantity of data collected, we developed custom software to filter and visualize the data. Interpreting Microsoft window events is not entirely straightforward because applications also use windows as part of their user interfaces widgets. A single user interaction can thus produce many confusing window events associated with nested window widgets. We removed many of these confusing events by filtering an event when the associated window had a parent window or did not have a title. For the remaining events, we determined the top level parent window and used that window for the analysis reported in this paper.

Analysis

The visualization we developed to analyze window manipulation activity is described in the caption for Figure 8. The visualization was generated by sorting the window events into temporal order and simulating the windowing activity by interpreting events such as creation, resize, and destruction. Although sessions start with unknown window configurations the Glass Box reports on the size and position of all windows at the start of each session, which is used to update the window simulation. The rectangles in the bars of the visualization represent windows. Their height represents the percentage of the monitor covered by that window. Their length represents the time to the next window event.

The primary advantage of this visualization is that it compactly shows both the frequency of window events and monitor usage. Although the data is full of simultaneous events for a single user action and unexpected system-generated events, the visualization emphasizes events that are generated with a frequency consistent with human activity. Given additional evidence from samples of the mouse and keyboard events, we believe the pattern of activity in Figure 8 represents reading and writing. Since the window title changes when new pages are fetched into IE, we believe the short duration events at the start of Figure 9 represent browsing activity. The full activity of Analyst 3 prints on a 40 by 40 inch plot. Analyst 4 prints on a 40 by 1440 inch plot. When we look at these plots, most of the time is spent in activities that have reading, writing, and browsing patterns similar to Figure 8 and Figure 9.

The visualization and the associated statistics (see Table 1) clearly show that the analysts spend most of their time reading and writing with windows that cover most of the 1280x1024 display. On average, the focused windows of Analyst 3 cover 51% of the monitor and those of Analyst 4

cover 72%. The finding of monitor usage is even larger when we focus on sensemaking applications. Word windows covered on average more than 90% of the monitor for both analysts, and IE windows covered almost 100% of the monitor. Reflecting their dual use, Glass Box windows covered 13% for Analyst 3 and 61% for Analyst 4. Analyst 4 clearly used this application to author meta-notes and routinely made the window large. Analyst 3, on the hand, only made it large on two days, and there were also some long events when it was on top and small that suggest that Analyst 3 had left for the day.

Analyst	3	4
Days	33	52
Duration	106:37:32	269:33:29
Avg. Coverage	51%	72%
Word	7:08:48 (97%)	59:13:34 (92%)
IE	48:02:14 (100%)	66:15:34 (99%)
GB	51:23:54 (13%)	31:27:20 (61%)
Abobe	0:0:28 (34%)	0:05:51 (85%)
Mail	0:0:6 (6%)	10:27:52 (85%)
Other	0:2:04 (5%)	2:03:40 (27%)

Table 1. Activity statistics for Analyst 3 & 4. The percentages indicate the amount of coverage of the monitor by a given type of application.

Evidence of Window Thrashing

The visualization also provides evidence that window thrashing occurs periodically. For example, in both Figure 10 and Figure 11 the analyst appears to be using Word and IE together, presumably writing about information found while browsing and reading. Figure 10 illustrates a “space-multiplexing” strategy where the analyst frequently changed the size of the Word window, presumably to reveal information in the underlying IE window. The visualization does not show repositioning of windows, which would be another indicator of space-multiplexing. Figure 11 illustrates the alternate “time-multiplexing” strategy, where the analyst switches rapidly between applications that cover 100% of the monitor. Time-multiplexing is more common in the data probably because space-multiplexing requires window resizing, which is an expensive subtask. The extremely short switches shown in Figure 11 represent cut-and-paste activity. The short glances into IE with longer durations into Word after the cut-and-paste probably represent writing time, where it would have been useful to have separate reading and writing spaces on the display.

Implications

Although the frequency of window events indicate that the analysts spent most of their time reading, writing, and browsing, they used windows that filled most of the

1280x1024 monitor, and they occasionally experienced what appeared to be window thrashing. Given the limited time spent in window thrashing, users engaged in sensemaking tasks may not feel the need to invest in wideband displays. However, Table 2 shows that Analyst 4 experienced at least one episode of window thrashing on almost half of the days of the field study and both analysts experienced one day with four episodes of window thrashing. The thrashing episodes are generally similar to Figure 10 and Figure 11 but shorter in duration. Since Analyst 3 used IE much more than Word, there was less opportunity for window thrashing between those two applications.

Analyst	3	4
Total Days	33	52
Thrashing Days	9	22
Percent	27%	42%
Max episodes	4	4

Table 2. Thrashing statistics for Analyst 3 & 4.

The visualizations shows that time-multiplexing is more common than space-multiplexing, which makes sense given the work required to resize and position windows when the space-multiplexing is used. Clearly, a second monitor would reduce this window thrashing by giving the analysts a space for reading and a space for writing. The study data does not indicate whether more than two monitors would be desirable.

Although our current field study data is limited to two subjects, the longitudinal nature of the data collected makes it possible to see window thrashing even though it occurred infrequently. Although the impact of the thrashing is not directly measured by the field study, we expect that the thrashing might have had considerable impact on the sensemaking activity as a whole because it occurred at critical moments in the sensemaking activity when the analysts were trying to write about the information they found when reading and browsing.

The information analysts used in this field study are atypical users of computer systems, engaging in complex sensemaking tasks that require weeks to complete and involving many documents. Since these subjects were quite skilled at using small displays, the thrashing found in this study probably represents a lower bound for thrashing activity. Less skilled sensemakers, such as students writing research papers, are likely to experience more frequent window thrashing when working on small displays.

THE IMPACT OF SEAMS

Although multiple monitors are a cheap and easy way to create a wideband display, the resulting seams in the workspace create gaps in words and divide diagonal lines into nonaligned segments, as shown in Figure 1. Words

and graphics with gaps are hard to read and view. Also, monitors can be vertically misaligned, which makes the reading of text and the viewing of graphics even more difficult.

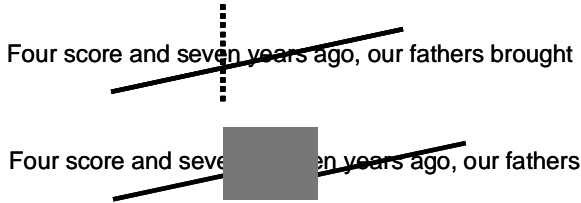


Figure 1: The top half shows text and graphics that cross a seam between two monitors at the location indicated with the dotted line. The bottom half shows how it looks with the grey rectangle representing the seam between two monitors. The gap in the text makes reading more difficult, and the line appears to be broken into two segments.

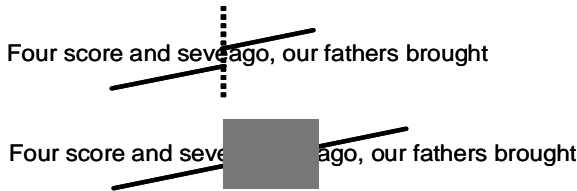


Figure 2: The top half shows how space can be removed so that a seam does not make a line appear broken. The bottom half shows what the user sees. However, text is occluded.

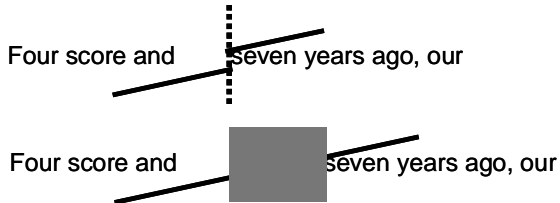


Figure 3: The top half shows how text can be moved off the seam when a display spans two monitors. The bottom half shows what the user sees.

Even though multiple monitors create seams, they have proven to be effective in niche applications such as CAD and graphic design, particularly when the seams help to organize and align the work [2]. Multiple monitors are also effective when the information for a task fits in windows that are small enough to be placed on individual monitors and those windows do not end up being placed across seams. For example, a stock broker might assign various windows showing different types of information to different monitors in the morning and not have to worry about the seams for the rest of the day.

However, tasks that involve the frequent creation of windows may require additional support to avoid the overhead of moving windows off seams. For example, nVidia, a major supplier of graphics cards that support multiple monitors, provides a driver for their card that automatically moves windows off seams onto the closest monitor. However, this automatic movement can obscure

windows needed for the task or move windows that are not impacted by seams. Furthermore, some tasks require windows that do not fit in a single monitor, in which case simply moving the window will not be effective.

MITIGATING SEAMS

The basic insight about mitigating seams is to acknowledge that seams have a perceptual impact. The user sees the lower half of Figure 1 rather than the top half. Given this insight, we can develop techniques to create seam-aware applications. For example, Figure 2 shows that the line will appear linear if it is drawn as if there were a display behind the seam. However, text becomes occluded by the seam. Figure 3 shows how text can be moved off a seam, avoiding both the broken word in Figure 1 and the occlusion in Figure 2.

Two Types of Seam Disruption

A formal analysis of graphical presentations indicates that a seam can disrupt window space in two ways: when the application uses space as a *container* to hold graphical objects or when it uses it as a *metric field* to position objects meaningfully with respect to quantitative axes [6]. In fact, an application can use space simultaneously as a container and a metric field. For example, charts can have a quantitative axis in one direction and an ordinal or nominal axis in the other direction [6]. Container spaces and metric spaces require different techniques for mitigating seams.

Container spaces and seam-awareness

When an application uses space as a container, it has the freedom to draw *seam-aware* graphics that compensate for the perceptual impact of the seams. Figure 4 is a screen shot of a node/link graph that is not seam aware. Links appear disjoint, and nodes are split across monitors. In contrast, Figure 5 shows the same graph drawn with seam-awareness turned on. Links appear to be drawn through the seams and nodes are moved off the seams. The implementation section describes how this can be done interactively, as shown in the related video figure.

Metric spaces and City Lights indicators

Metric spaces, on the other hand, must be drawn through a seam to maintain the metric from one monitor to the next, which may cause important information to be occluded. For example, the scatterplot shown in Figure 12 clearly shows the linear trend of the data. The distance between points is meaningful even across the seam. Maps, which often need to be large, are another example of a metric space that should be maintained across seams.

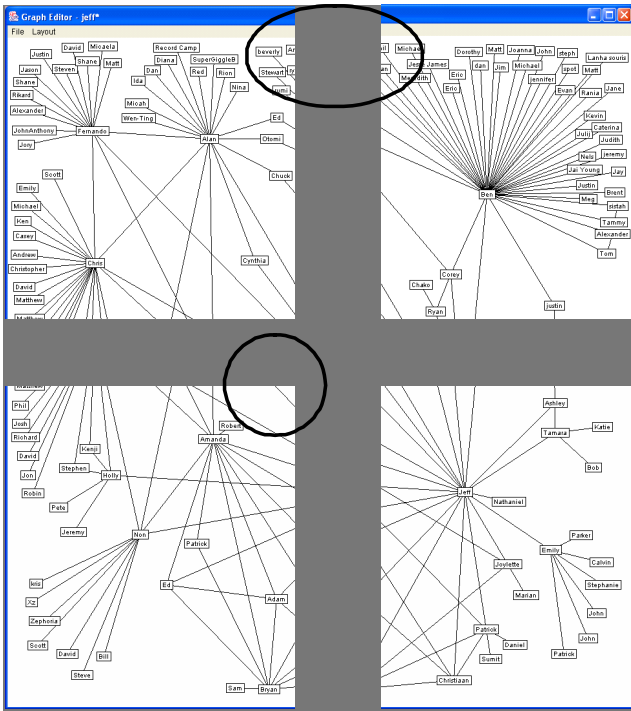


Figure 4: Seam-ignored FreeformGraphLayout. The circle marks a line segment that does not appear to be part of a link between nodes. The oval marks nodes that are split across the seam.

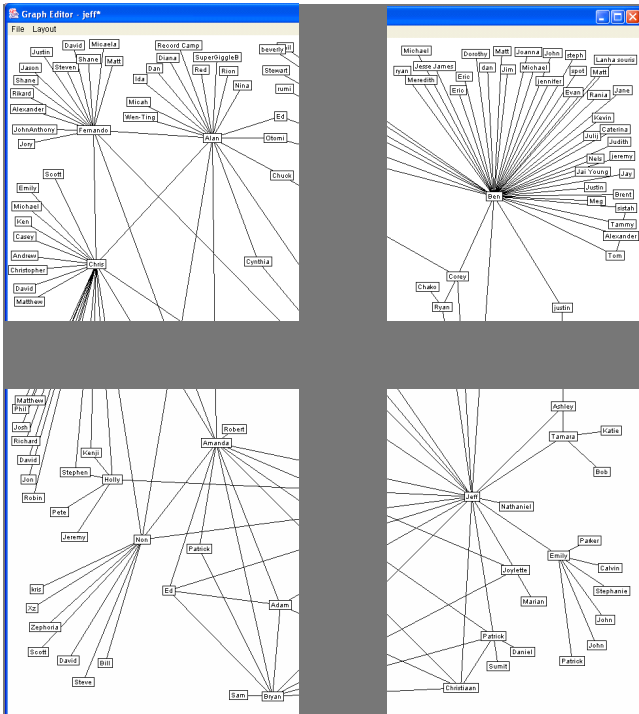


Figure 5: Seam-aware FreeformGraphLayout. Links are drawn under seams and nodes are moved off seams.

However, drawing metric spaces through seams can force graphics objects constrained by the metric to be obscured by the seam. For example, the seam obscures many points in the scatterplot shown in Figure 12. When this occurs, City Lights indicators can be used to help the user see that graphical objects are obscured [11]. The implementation section also describes how seam-awareness was added to the third party application that produced Figure 12.

IMPLEMENTATION

Our approach to mitigating seams is to regard the entire display configuration as one large display surface and treat each individual monitor as a viewport into this larger space (as in Figure 6). Physical separations such as seams are explicitly modeled as off-screen pixels in a virtual, seam-aware coordinate space. Naively, user interface components can draw themselves onto a virtual canvas, and then paint only the visible regions of this canvas to the screen. However, this can result in items being drawn “behind the seams”. To make better use of display resources we can make interface components *seam-aware*, such that they can structure their content optimally in the face of possible occlusion by seams. Our technical solution consists of two parts: infrastructural support for computing the seam-aware coordinate spaces of interface components, and software methods for assisting application-specific aspects of seam-awareness. We have implemented our solution in the Java programming language as a general library supporting seam-aware user interfaces built in the Java AWT and Swing user interface toolkits.

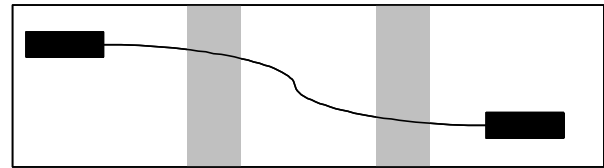


Figure 6: The application is given a graphics object that includes space for the seam. The visible regions are then drawn to the monitors.

Infrastructural Support

At the heart of our infrastructural support is a globally accessible class called the ScreenGraphFactory. Upon loading of the Java Virtual Machine, the ScreenGraphFactory queries the operating system for all the available displays and creates two models of the display set-up. The first model is merely the system of *screen coordinates* provided by the OS. This representation has no knowledge of the actual physical location of monitors, and assumes that adjacent screens form a continuous display. The second model is a *seam-aware coordinate* system that models physical separations (such as seams) as distances measured in pixels. This model allows us to represent the perceived separation between displays.

Successful creation of this seam-aware coordinate model depends on knowing the actual physical distances between display surfaces. This requires knowing the measurement of the seam widths or physical separations in terms of display pixels. In initial studies, we measured these distances physically and included them as an input parameter. A more general solution is to acquire the seam widths dynamically with a simple calibration application, depicted in Figure 7. For each display boundary, the user performs a simple line fitting task. The system can then use basic trigonometry to infer the correct distance in pixels between the displays. Such calibrations need only be performed once for any given display configuration, as the results can be stored and reused.



Figure 7: Example of seam width calibration. After the user fits the line on the right side, the seam width x_s can be computed as $x_s = (h / \tan \theta) - (x_L + x_R)$

The `ScreenGraphFactory` assists user interface components in achieving seam-awareness by providing custom display information on a per component basis. Using the bounding screen coordinates of the supplied user interface component and the global seam-aware coordinate space, the `ScreenGraphFactory` computes a custom seam-aware coordinate space local to the component. This information is encapsulated in a `ScreenGraph`, an object that keeps track of both the screen and seam-aware coordinates of the component and methods for mapping between the two. In particular, the `ScreenGraph` abstraction provides methods for:

- Retrieving the size of the virtual, seam-aware coordinate space
- Determining if a point or bounding box is contained within or intersects a seam region
- Providing the nearest visible display region in any given direction

Seam-aware interface components must explicitly query the `ScreenGraphFactory` to receive `ScreenGraph` instances, and should listen for move and resize events from their parent windows to request updated `ScreenGraph` objects as appropriate.

Application-specific seam management

The `ScreenGraph` allows user interface components to be aware of seams and separations in the display. How the seams are dealt with, however, is left to application-specific code. For example, custom paint routines can draw the component into an offscreen buffer using the seam-aware coordinate space, and then paint only the visible regions to the actual display. Layout algorithms within the application can determine the spatial position of interface items to

avoid seam crossings or occlusions. In the next sections we discuss the implementation of multiple classes of seam-aware applications.

Seam-aware Graphics for Container Spaces

As an example of container spaces, we have incorporated seam-awareness into the `prefuse` toolkit, which was designed using an extended model-view-controller architecture – entities in the abstract graph model are first mapped into visual analogues. These items are run through layout and presentation algorithms to determine screen position and appearance. They are then placed into rendering queues, which a double-buffered view component uses to paint them to the screen.

To make the toolkit naively seam-aware, we simply modified the view component. The view component was changed such that it requests `ScreenGraph` objects from the `ScreenGraphFactory`, and adjusts its offscreen paint buffer to match the size of the virtual coordinate space. The paint routine draws all the graph items (nodes, edges, and aggregates) into this buffer, but then only draws to the screen those portions of the buffer that correspond to visible display regions. While this causes all lines and shapes to be rendered correctly, it can cause items to be drawn “behind the seams.”

To remedy this problem, we added custom layout procedures to our applications. The `prefuse` architecture employs a modular pipeline architecture, allowing custom processing components to be placed in the pipeline at will. This allowed us to completely avoid rewriting intricate graph layout algorithms, instead adding an additional layout module further down the pipeline that perturbs graph nodes and aggregates so that they do not intersect any seam boundaries. Of course, custom layout algorithms for more specialized seam-aware layouts may also be desirable.

City Lights Indicators for Metric Spaces

The scatterplot in Figure 12 shows that our architecture can be added to third-party Java applications that require multiple-monitor metric spaces. The scatterplot was drawn by `JFreeChart`, an open source software package that required minimal modification to generate [3]. The top level paint method was modified to render into an offscreen buffer, which was then mapped to the monitors using `ScreenGraphFactory`.

Adding City Lights indicators to an application requires information about the existence and location of the components in the interface, which can be difficult to determine in third party code. Luckily, `JFreeChart` generates an `EntityCollection` during rendering that contains all relevant components and their locations. The City Lights indicators were added to Figure 12 using `ScreenGraphFactory` to identify which components were obscured and the closest visible point for the indicator.

CONCLUSION

Progress in information visualization and in our understanding of human-information interactions provides an opportunity to develop wideband visual interfaces that take advantage of displays that fill our field of view. These could radically improve productivity in many knowledge management tasks, analogous to the improved productivity of a craftsman who has the right tools and a big enough workbench. We described a longitudinal field study of two information analysts using single monitor computers. The study, which found that windows typically filled the display and the analysts were occasionally interrupted by what appeared to be window thrashing, suggests the need for larger displays.

Although researchers are currently developing seamless wideband displays [2], they are expensive. However, their cost will ultimately become affordable, driven down by computer gaming, entertainment, and teleconferencing. In the meantime, we can already explore the potential of wideband displays by building affordable multiple-monitor wideband solutions with current commercial hardware as long as we mitigate seam disruption. Given the differing requirements of container and metric spaces, we have implemented several novel user interface techniques for creating seam-aware applications that target wideband displays based on multiple monitors. Given a practical method for implementing wideband visual interfaces, the next step is to improve our applications by exploiting the power of the human visual system to work effectively in computational workspaces that are at least as large and high-resolution as the desks where we work with paper.

ACKNOWLEDGMENTS

This research has been funded in part by contract # MDA904-03-C-0404 awarded to Stuart K. Card and Peter Pirolli from the Advanced Research and Development Activity, Novel Intelligence from Massive Data program. We thank Stuart K. Card for many discussions about window paradigms and Polle T. Zellweger for her skilled suggestions about multiple drafts.

REFERENCES

1. Card, S., and Nation, D. Degree-of-Interest Trees: A component of an attention-reactive user interface. *Advanced Visual Interfaces '02*, Trento, Italy, 2002.
2. Czerwinski, M., Smith, G., Regan, T., Meyers, B., Robertson, G., Starkweather, G. Toward characterizing the productivity benefits of very large displays. *Proc. Interact 2003*.
3. Gilbert, D. and Morgner T. JFreeChart. <http://www.jfree.org/jfreechart/index.html>
4. Grudin, J. Partitioning digital worlds: Focal and peripheral awareness in multiple monitor use. *Proc. CHI 2001*, CHI Letters 3 (1) 458-465.
5. Henderson, D. A., & Card, S. K. Rooms: The use of multiple virtual workspaces to reduce space contention in a window-based graphical user interface. *ACM Transactions on Graphics*, 5, (1986) 211-243.
6. Mackinlay, J. Automating the design of graphical presentation of relational information, *ACM TOG 5 (2)* ACM Press (1986) 110-141. Also in *Readings in Information Visualization: Using Vision to Think*. Card, S., Mackinlay, J., and Shneiderman, B. (1999) Morgan Kaufmann.
7. O'Hara K. and Sellen A. A comparison of reading paper and on-line documents. *Proc. CHI 1997*, ACM Press (1997), 335-342.
8. Scaife, M. and Rogers, Y. External cognition: How do graphical representations work? *International Journal of Human-Computer Studies*, 45, (1996) 185-213.
9. Sellen A. and Harper R. Paper as an analytic resource for the design of new technologies. *Proc. CHI 1997*, ACM Press (1997), 319-326.
10. Tanenbaum, T. *Modern Operating Systems* (2001) Prentice Hall.
11. Zellweger, P., Mackinlay, J. Good, L. Stefik, M. and Baudisch, P. City Lights: Contextual views in minimal space. *Ext. Abstracts CHI2003*, ACM Press (2003) 838-839.

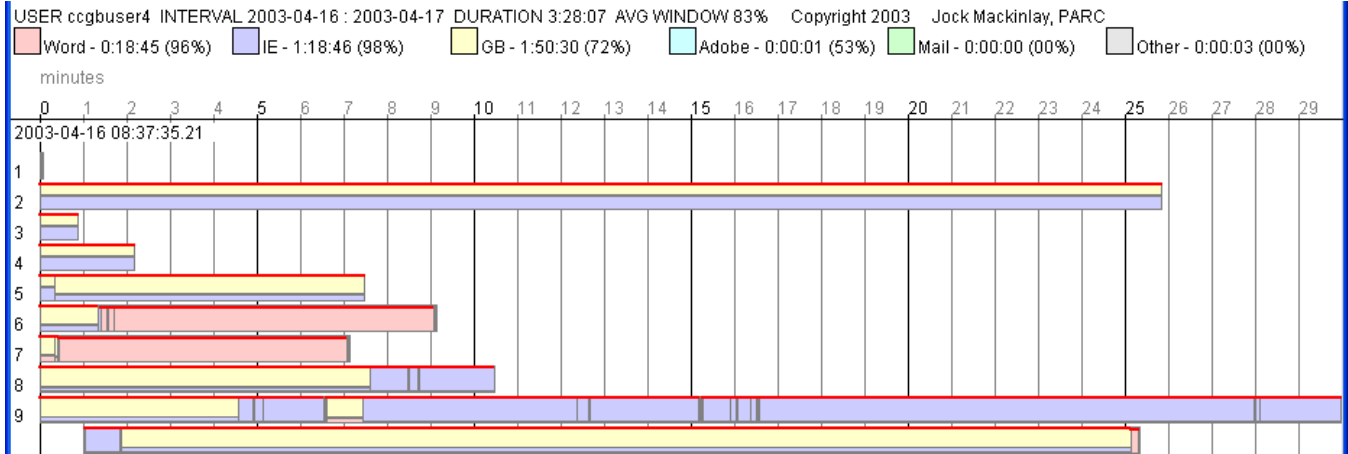


Figure 8: Analyst 4 on 4/16. These 9 sessions show typical window event behavior. The horizontal axis is minutes. The vertical axis is sessions. Color indicates type of window. Bar length is segmented into window events, showing their duration. Bar height is segmented into windows, showing their display area. Horizontal red lines indicate the focused window, if known. Sessions 1-5 show notes written in the focused Glass Box application (GB), which covers around half of the display. MS Internet Explorer (IE) fills the rest of the display. Sessions 6 and 7 start in the GB and switch to MS Word, which covers the whole display. Session 8 and 9 start in GB and switch to IE. Session 9, which is wrapped to a second line because of its length, concludes in the GB application.

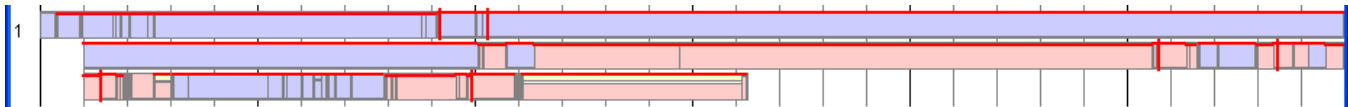


Figure 9: Analyst 3 on 4/17. This session, which is also typical, started with IE window events of short duration, which is probably browsing. Next there are longer events in IE (reading) and Word (writing). Vertical red lines indicate a new window for the same type of focused application. The rapid switches between applications toward the end of this session may be window thrashing.

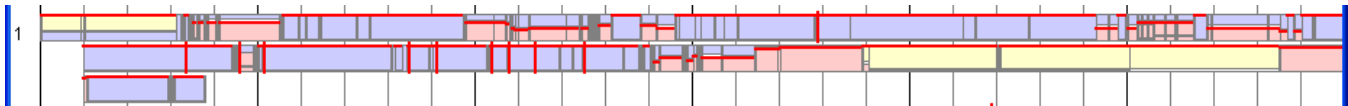


Figure 10: Analyst 4 on 6/9. Window thrashing between Word and IE. Word, in particular, covers only part of the monitor even though it is generally used full screen and it is resized, presumably to see information in the overlapped IE window. This visualization does not show window movements, which would be another indicator of window thrashing.

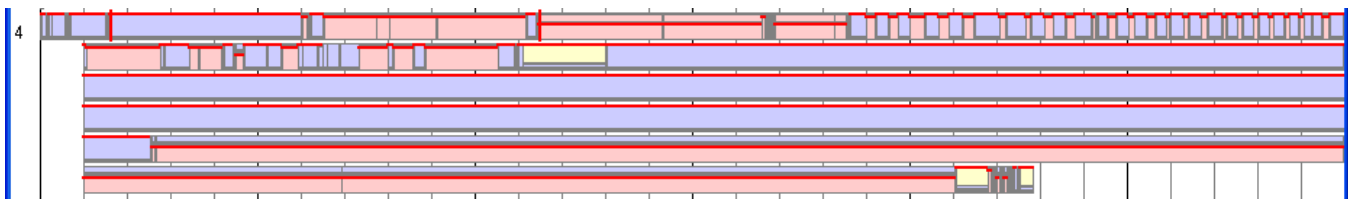


Figure 11: Analyst 4 on 7/1. Window thrashing between Word and IE. The analyst generally used this time strategy.

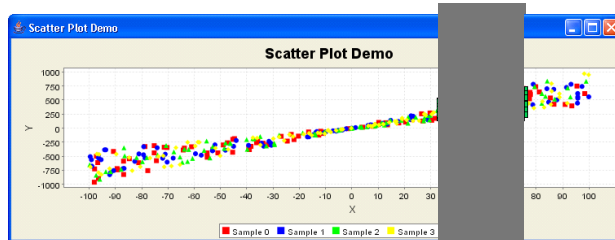


Figure 12: Seam-aware third party scatterplot, which is a metric space. Green City Lights indicators [11] mark obscured points.