Content-Aware Layout

Edward W. Ishak
Columbia University
Department of Computer Science
New York, NY 10027 USA
ishak@cs.columbia.edu

Steven K. Feiner
Columbia University
Department of Computer Science
New York, NY 10027 USA
feiner@cs.columbia.edu

Abstract
We describe *content-aware layout* (CAL), a technique that automatically arranges windows on a user's desktop. Unlike conventional window managers that automatically cascade or tile each window without regard to its content, CAL uses information about the contents of windows to help decide if and where they should be placed. We present the approach to designing CAL, as well as its implementation. We then conclude with a discussion about future work and CAL's potential use in large display environments.

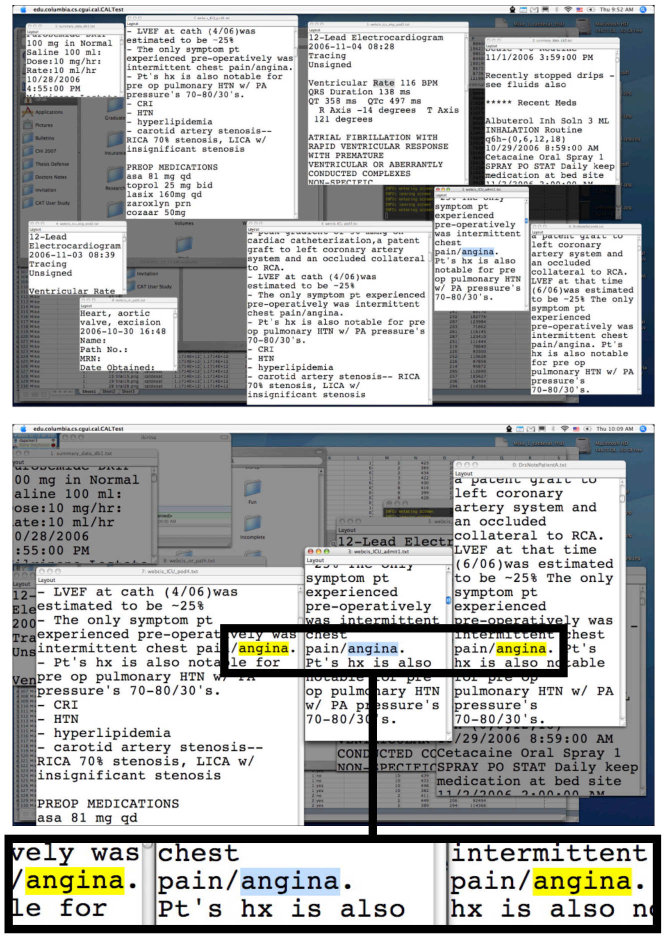
Keywords
Layout, content-aware, window manager, transparency, scrolling, fisheye

ACM Classification
H.5.2. Information interfaces and presentation: User Interfaces—Graphical user interfaces (GUI), Interaction styles, Screen design, Windowing systems

Introduction
Computer users often need to interact with more than one document at a time. Consequently, they often rearrange their windows by resizing and/or repositioning them. Most window managers provide ways to bring obstructed content to the front through the use of "alt-tab" (or, in the case of Mac OS X, "command-tab"), which invokes a menu containing a list of open windows

(a)

(b)



```
vely was  chest          intermittent
/angina.  pain/angina.   pain/angina.
le for    Pt's hx is also hx is also n
```

Figure 1: Comparison between Exposé and CAL prototype for a desktop with overlapping windows in which a user wants to locate other windows that contain the selected text, "angina". (a) Using Exposé, a user can view scaled versions of all windows of that application and visually search for the text. (b) Using CAL, the system presents only those windows that contain the word "angina," where search results are aligned horizontally with the selected text.

(in Mac OS X, a list of open applications) from which the user can choose.

Some commercial window managers automatically rearrange windows, such that each one is unobstructed. For example, Microsoft Windows can tile or cascade each window ordered by recency of use. In contrast, Apple's Exposé scales, tiles, and neatly rearranges all open windows (or all open windows of a particular application) at the press of a key. However, none of these approaches take into account window content (as opposed to window type).

In this paper, we discuss the design and preliminary implementation of *content-aware layout* (CAL), an approach that considers the content of a window to help determine if and where it should be placed on the screen. For example, in Figure 1(b), after a user has selected a text string in a window, CAL presents only those windows that contain the selected text, arranging the windows so that the search results are aligned horizontally. We will show that taking into account the characteristics of the content, along with the user's task, potentially makes for more effective layouts.

Related Work

We summarize relevant work in bringing obscured content into focus and automatically arranging content in user interfaces.

Bringing Obscured Content to Focus

Both alt-tab and Exposé are familiar techniques used to bring focus to an obstructed object. Applications, such as 3ds Max, Adobe Photoshop, and our mouse-over pie-menu [16] allow users to cycle through a list of overlapping objects beneath the current mouse cursor position. However, in all of these techniques, a user must know the approximate bounds of an obscured window to invoke the selective list containing it.

Automatic Layout

There has been extensive research in automatic layout using design grids [10, 14, 17, 23], machine learning [4, 25, 30, 34], evaluation techniques [8, 9, 18, 28, 29, 31], and constraint solvers [1, 4, 6, 7, 11, 12, 13, 19, 24, 32, 33], including those that adhere to aesthetic design criteria [21, 27]. (See Lok and Feiner [20] for a survey of this work.)

CAL Approach

Content-Aware Interaction

CAL is built on the concept of making traditional techniques *content-aware* by taking into account various characteristics of a document's content to determine how to view or interact with it, an approach we have previously applied to transparency [16] and scrolling [15]. In applying it to layout, we hypothesized that by modifying an existing layout approach and making it content-aware, user interaction would be improved. Part of this involves applying layout criteria to the window's contents, rather than to just the window bounds. Although this could be achieved by manually imposing constraints in other constraint-based layout systems [1, 6], CAL does this automatically.

Visual Scope

We designed CAL with the intent to increase what Norman and colleagues refer to as *visual scope*: "the degree to which the user is able to integrate information across a display of multiple windows or screens and to grasp the whole of whatever is being displayed" [26]. Maintaining visual scope is important because many tasks require visual access to multiple screen objects. Since it is not necessarily true that presenting more information to the user will make her more productive, CAL attempts to display no more than what is needed at a given time. For example, consider copying text from one window to another. If the system presents the user with more visual information than just the source and destination windows, the task may not be completed as quickly, or even correctly at all. In fact, according to Norman and colleagues, if there is no relationship spanning the various screen objects, this may decrease the visual scope. Our goal is to match the user's visual expectations of what is to be displayed on the screen with what is actually displayed.

CAL Design

Anchoring the Working Set

With large displays, if content is not placed strategically, users may spend several seconds trying to locate it. During normal interaction, users often view or interact with only a part of the display (or what has been referred to as the *working set* [5]). This can vary both physically and semantically from a paragraph of text to some meaningful subset of windows. CAL regards this area as important for two reasons: (1) it is the region where the user is currently working, and therefore, contains information that is important to the user at that time, and (2) it is the physical part of the screen containing content around which the user may wish to see additional contextual information. For these two reasons, if this region is known by the system during a layout operation, CAL uses it as an *anchor area*, a portion of the screen around which a layout occurs.

CAL allows the user to directly specify an anchor area. Although one could use an eye tracker to determine where the user is gazing, or record the user's mouse activity to determine what parts of the screen she is interacting with more than others, these methods may not accurately reflect the parts of the screen containing content in which the user is *currently* interested. For example, a user's eyes may be fixated on one part of the screen, while her mouse is on another, making it ambiguous to the system as to which area is more important. However, if the user directly indicates her area of interest, this gives CAL a well-defined screen region containing content in which the user is interested.

Layout Scheme

Although alternative layout schemes may be more appropriate for other types of content, we have chosen a *linear horizontal array* layout when dealing with text because it resembles side-by-side book pages, facilitating an easier visual context switch between the working set area and peripheral textual content. However, we made two distinct content-aware modifications. First, only those windows containing relevant content are laid out. Second, for each of these windows, the layout is performed with regard to the content within the window, not just the window bounds, as with most other layout managers. For example, in Figure 1(b) CAL horizontally aligns relevant content.

To avoid obstructing content, windows do not overlap in the current implementation. We are currently incorporating content-aware transparency [16] (CAT) to take advantage of screen space occupied by unimportant regions of neighboring windows. CAT varies the transparency level of different window regions based on their importance. If unimportant regions are rendered transparent, they can overlap otherwise hidden important regions of a neighboring window and thus expose them to the user.

Application

We have developed a prototype in which users can peruse text documents, called the *CAL manager*. The CAL manager is cross-platform and allows users to interact with custom text editor windows that behave like any other window on their desktop.

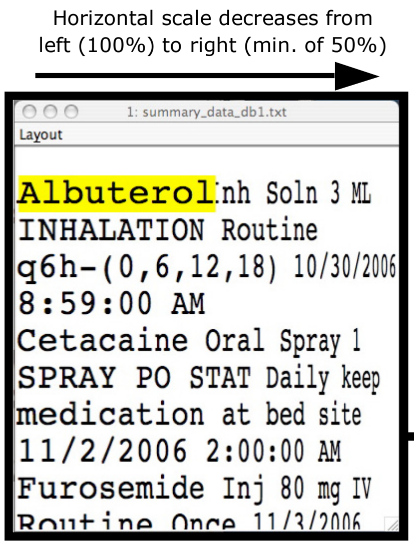
Implementation

We implemented the CAL manager using the Java 1.4 SDK with a custom Swing library. We use our lab's dynamic space manager [3] to control free space.

Search Open Documents

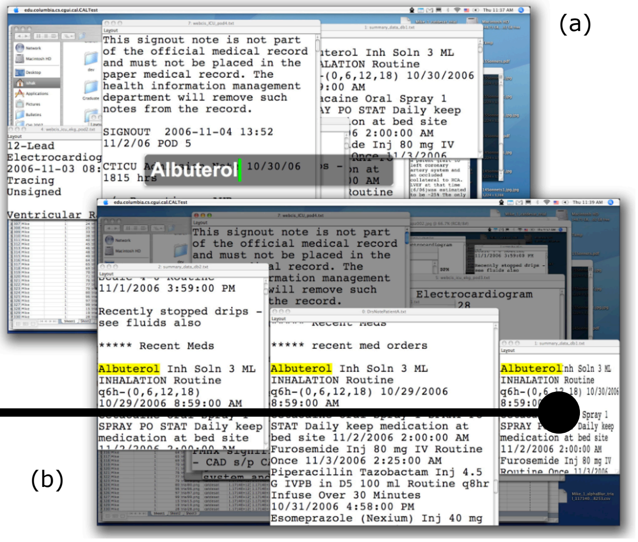
When a user peruses a text document created with the CAL manager, she can focus on windows containing similar content. By first selecting text in a window using any conventional selection mechanism (e.g., double clicking on a word), the user can then invoke a *layout-peripheral* operation across all other open windows (through either a menu option or a ctrl-L shortcut key). Once invoked, the CAL manager takes the selected text as input, sets its current screen location as the anchor area, and for each document containing that search string, highlights the first occurrence, and then rearranges it using a linear horizontal array layout scheme, such that the first occurrence within each document is lined up with respect to the anchor area, as shown in Figures 1 and 2. This allows the user to visually scan the search results in a single spatial dimension, ordered left-to-right in decreasing order of result hit count. The rearrangement is performed using a slow-in-slow-out animation over 500 ms, similar to that of Apple's Exposé and earlier work on automated layout [3]. The location of each window is restored using the same shortcut key, or by pressing the "escape" key.

A user can alternatively perform a *search-and-layout* operation across all open windows without first selecting any text. Pressing ctrl-space causes an input dialog to appear, which allows the user to enter a text string. Upon pressing return, windows containing that string are centered on the screen and arranged using the linear array layout scheme with its search results highlighted, as shown in Figure 2.


Horizontal scale decreases from left (100%) to right (min. of 50%)
1: summary_data_db1.txt
Layout

Albuterol Inh Soln 3 ML
INHALATION Routine
q6h-(0,6,12,18) 10/30/2006
8:59:00 AM
Cetacaine Oral Spray 1
SPRAY PO STAT Daily keep
medication at bed site
11/2/2006 2:00:00 AM
Furosemide Inj 80 mg IV
Routine Once 11/3/2006

The contents of windows that would otherwise overlap the screen edge are non-linearly scaled in width, such that the entirety of the window contents remains visible. No part of the window is scaled at less than 50% of its original size to maintain content legibility. Highlighted text always remains at 100% scale.



This signout note is not part of the official medical record and must not be placed in the paper medical record. The health information management department will remove such notes from the record.

SIGNOUT 2006-11-04 13:52
11/2/06 POD 5

12-Lead Electrocardiogram
2006-11-03 08:
Tracing Unsigned

Albuterol Inh Soln 3 ML
INHALATION Routine
q6h-(0,6,12,18) 10/30/2006
8:59:00 AM

11/1/2006 3:59:00 PM
Recently stopped drips - see fluids also

***** Recent Meds

Albuterol Inh Soln 3 ML
INHALATION Routine
q6h-(0,6,12,18)
10/29/2006 8:59:00 AM
SPRAY PO STAT Daily keep
medication at bed site

***** recent med orders

Albuterol Inh Soln 3 ML
INHALATION Routine
q6h-(0,6,12,18) 10/29/2006
8:59:00 AM
STAT Daily keep medication at bed site 11/2/2006 2:00:00 AM
Furosemide Inj 80 mg IV Routine Once 11/3/2006 2:25:00 AM
Piperacillin Tazobactam Inj 4.5 G IVPB in D5 100 ml Routine q8hr Infuse Over 30 Minutes
10/31/2006 4:58:00 PM
Esomeprazole (Nexium) Inj 40 mg

Electrocardiogram

Figure 2: A user performs a search for "Albuterol" across all open windows. (a) By pressing ctrl-space, a search dialog appears. The user enters text and pressing return. (b) CAL then presents those windows that contain the text, where search results are horizontally aligned.

For cases in which the screen cannot accommodate all relevant windows in one view, we have incorporated seam-awareness [22] by implementing an orthogonal non-linear magnification view of window content that would otherwise overlap a monitor seam, as shown in Figure 2. This is an effect similar to that used in the Fishnet web browser [2], except that, in CAL, distortion is performed non-uniformly, where content nearer to the screen edge is scaled by a larger factor. We do not allow any part of the window to be scaled at less than 50% of its original size to maintain legibility. Before the layout is reverted, a user can click on the scaled window to view it temporarily in its original size. In all cases, the scaled window's original size is restored when the layout is reverted.

If the screen is still insufficiently wide, we display an initial set of windows, and then allow the user to iterate through the next set by pressing ctrl-right or ctrl-left to advance forwards or backwards, respectively. We are investigating supporting content-aware scrolling [15], to support scrolling through each search result within a window before the next set of windows is presented.

Conclusions and Future Work

We have presented an early prototype of content-aware layout, a technique that considers the content of windows to determine if and where they should be placed on the screen. Users can perform a search operation that places the relevant content peripherally in a linear array layout scheme for easier visual access to the important content.

We are incorporating CAL into a multi-monitor environment in the Cardiothoracic Intensive Care Unit at New York Presbyterian Hospital, where attending and resident physicians need to create and peruse patient status notes. The system they use currently is very error-prone, and requires referencing contextual content in other windows. With the help of our colleagues, we are incorporating a high-level classification algorithm to help retrieve documents that are “relevant” to completing a daily patient status note without having to match a textual search. We believe that CAL has the potential to improve productivity in this environment, as well as other large display systems.

Acknowledgements

This work is funded in part by IBM Research and is part of a project being conducted under the Open Collaboration Research program.

References

[1] G. J. Badros, J. Nichols and A. Borning, "Scwm: An Extensible Constraint-Enabled Window Manager," *Proc. FreeNIX 2001,* June 2001.
[2] P. Baudisch, B. Lee and L. Hanna, "Fishnet, a fisheye web browser with search term popouts: a comparative evaluation with overview and linear view," *Proc. AVI 2004,* Gallipoli, Italy, May 25–28, 2004, 133–140.
[3] B. Bell and S. Feiner, "Dynamic Space Management for User Interfaces," *Proc. UIST 2000,* San Diego, CA, November 5–8, 239–248.
[4] A. Borning and R. Duisberg, "Constraint-based tools for building user interfaces," *ACM Trans. on Graphics,* Vol. 5 (4), Oct. 1986, 345–374.
[5] S. K. Card, G. G. Robertson and J. D. Mackinlay, "The information visualizer, an information workspace," *Proc. CHI 1991,* 181–186.
[6] E. S. Cohen, A. M. Berman, M. R. Biggers, J. C. Camaratta and K. M. Kelly, "Automatic Strategies in the Siemens RTL tied window manager," *Proc. Computer Workstations 1988,* Mar. 7–10, 1988, 111–119.
[7] E. S. Cohen, E. T. Smith and L. A. Iverson, "Constraint-Based Tiled Windows," *IEEE Comp. Graphics and App.,* Vol. 6 (5), 35–45.
[8] T. Comber and J. Maltby, "Evaluating usability of screen design with layout complexity," *Proc. OZCHI 1995*, 1995, pp. 175–178.
[9] T. Comber and J. Maltby, "Investigating layout complexity," *Proc. Graphics Interface 1988*, June 1988, pp. 192–197.
[10] S. K. Feiner, "A grid-based approach to automating display layout," *Proc. Graphics Interface 1988*, June 1988, pp. 192–197.
[11] W. H. Graf., "Constraint-based graphical layout of multimodal presentations," *World Sci. Series in Comp. Sci.,* Vol. 36, 1992, 365–385.
[12] S. E. Hudson and S. P. Mohamed, "Interactive specification of flexible user interface displays," *ACM Transactions on Information Systems,* Vol. 8 (3), July 1990, pp. 269–288.
[13] S. E. Hudson and I. Smith, "Ultra-lightweight constraints," *Proc. UIST 1996*, 1996, pp. 147–155.
[14] A. Hurlburt, *The Grid*, Van Nostrand Reinhold Company, Melbourne, Australia, 1978.
[15] E. W. Ishak and S. K. Feiner, "Content-Aware Scrolling," *Proc. UIST 2006,* Montreux, Switzerland, October 15–18, 2006, pp. 155–158.
[16] E. W. Ishak and S. K. Feiner, "Interacting with Hidden Content Using Content-Aware Free-Space Transparency," *Proc. UIST 2004,* Sante Fe, NM, October 24–27, 2004, pp. 189–192.
[17] C. Jacobs, W. Li, E. Schrier, D. Bargeron and D. Salesin, "Adaptive grid-based document layout," *ACM Trans. on Graphics*, 2003, 838–847.
[18] R. Jeffries, J. R. Miller, C. Wharton and K. M. Uyeda, "User interface evaluation in the real world: A comparison of four techniques," *Proc. CHI 1991*, 1991, pp. 119–124.
[19] S. Kochhar, J. Marks and M. Friedell, "Interaction paradigms for human-computer cooperation in graphical-object modeling," *Proc. Graphics Interface 1991*, June 1991, pp. 180–191.
[20] S. Lok and S. Feiner, "A survey of automated layout techniques for information presentations," *Proc. SmartGraphics Symposium 2001*, March 2001, pp. 61–68.
[21] S. Lok, S. Feiner and G. Ngai, "Visual Balance for Automated Layout," *Proc. IUI 2004*, January 2004, pp. 101–108.
[22] J. D. Mackinlay and J. Heer, "Wideband displays: mitigating multiple monitor seams," *Proc. CHI 2004,* Vienna, Austria, pp. 1521–1524.
[23] J. Muller-Brockmann, *Grid Systems in Graphics Design*, Arthur Niggli Publishers, Niedersteufen, Switzerland, 1981.
[24] B. A. Myers, "The garnet user interface development environment," *Proc. CHI 1994 Conference Companion,* 1994, 25–26.
[25] B. A. Myers, R. G. McDaniel and D. S. Kosbie, "Marquise: Creating complete user interfaces by demonstration," *Proc. INTERCHI 1993,* Amsterdam, The Netherlands, pp. 293–300.
[26] K. L. Norman, L. J. Weldon and B. Shneiderman, "Cognitive layouts of windows and multiple screens for user interfaces," *International journal of man-machine studies,* Vol. 25, pp. 229–248.
[27] L. Purvis, "A genetic algorithm approach to automated custom document assembly," *Proc. Workshop on Intelligent Sytems Design and Application,* Atlanta Georgia, 2002, pp. 131–136.
[28] A. Sears, "Layout appropriateness: A metric for evaluating user interface widget layout," *IEEE Transactions on Software Engineering,* Vol. 19 (7), July 1993, pp. 707–719.
[29] A. Sears and A. M. Lund, "Creating effective user interfaces," *IEEE Software,* Vol. 14 (4), July/August 1997, pp. 21–24.
[30] S. Stille, S. Minocha and R. Ernst, "A^{2}DL-an Adaptive Automatic Display Layout system," *Proc. HICS 1996*, pp. 243–250.
[31] T. S. Tullis, "A computer-based tool for evaluating alphanumeric displays," *Proc. IFIP INTERACT 1984*, 1984, pp. 719–723.
[32] B. Vander Zanden and B. A. Myers, "Automatic, look-and-feel independent dialog creation for graphical user interfaces," *Proc. CHI 1990*, pp. 27–34.
[33] L. Weitzman and K. Wittenburg, "Automatic presentation of multimedia documents using relational grammars," *Proc. Multimedia 1994,* New York, October 1994, pp. 443–452.
[34] M. X. Zhou and S. Ma, "Toward applying machine learning to design rule acquisition for automated graphics generation," *Proc. 2000 AAAI Smart Graphics,* Stanford, CA, March 20–22, 1999, pp. 16–23.