

GraphStract: Graphical Abstracts for Explaining Interaction

Bo Lu and Michael B. Twidale

Graduate School of Library and Information Science

University of Illinois at Urbana-Champaign

Champaign, IL 61820

{bolu, twidale}@uiuc.edu

ABSTRACT

We explore the use of abstracted screenshots as part of a system to support help giving. GraphStract extends the ideas of textually oriented Minimal Manuals to the use of abbreviated screenshots, enabling multiple small graphical elements to be shown in a small space. This enables the user to get an overview of a complex sequential task as a whole while allowing users to “zoom in” on a single step of the interaction. Additionally, GraphStract aims to facilitate multiple levels of user help. This idea has been developed by iterative prototyping, and pilot tested on a small scale.

Keywords

Help, Minimal Manuals, Visualization.

INTRODUCTION

Current implementations of automated user help on end-user applications such as Microsoft Word [8] rely substantially on textual explanations to guide the user to perform the correct operations. Various user studies have found that users are unlikely to read carefully, if they read the text at all [10,18]. Users are often looking for a quick fix to make a small change to their work, and may be unwilling to make the investment of time and concentration to read a lengthy explanation. Even when users are highly motivated by the importance of solving a particular problem, conventional online help is mostly textual (Figure 1) which can be difficult to map to the 2D GUI environment of the application.

We are working on ideas for creating abbreviated forms of graphical help using layered elements of the application window, inspired both by the metaphor of abstracts of text, and Carroll’s Minimal Manuals [3]. This is intended to support both individuals and semi-formal peer user to peer user help giving. GraphStract is currently in the prototype investigation and evaluation phase.

CURRENT APPROACHES

Much work has been done in modeling the process of users receiving help [5,6,9,15]. Here we describe three general classes of current approaches.

Text-centric Help

Current paper and online manuals attempt to combine text and graphics but often relegate graphics to a secondary role. Thumbnail-sized pictures of a specific control or a relevant

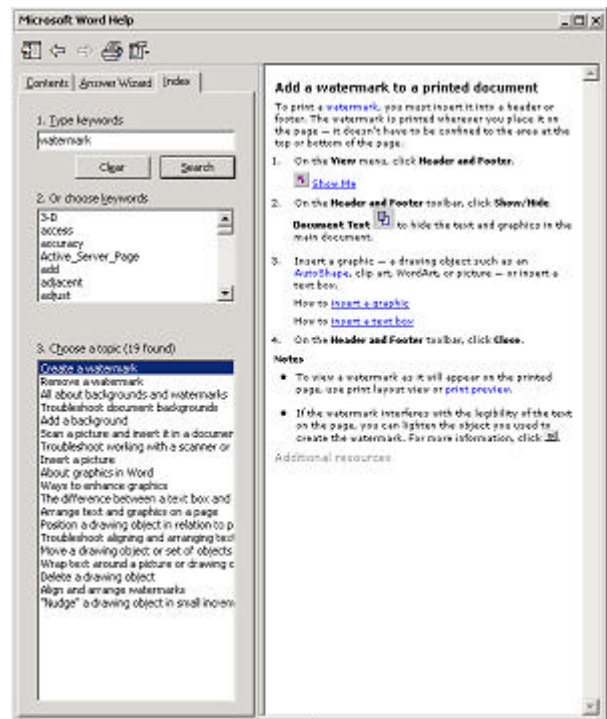


Figure 1. A typical text-centric help page, this one from Microsoft Word on applying watermarks.

dialogue box are used only to supplement textual description. In addition, the images in text-centric manuals are presented as individual tokens of information, instead of a part of a cohesive group of interactions to achieve a task.

Naive Screenshots

Between colleagues, peer-to-peer help giving often utilizes the interface seen on screen as a focus of pointing and discussion [18]. When creating help texts, screenshots are frequently used. They have many advantages in clarifying for the end user what to do, and what they should be seeing next in a complex sequence of actions. The crucial disadvantage is that this method often involves numerous, large and complex screenshots which yield far more information than is necessary for most users. This adds to perceptions of complexity and being overwhelmed by technology, discouraging the user receiving help. In addition,

the size and number of the screenshots can pose a problem in understanding a task as a whole. It may be hard to spread out a series of screenshots, either in the user's mind or even on a desk, to form a coherent and holistic sense of the entire task. This is particularly true when these help-giving screenshots must be viewed on a computer, where users maybe can only fit a single snapshot of the interaction on the screen at once, and even then must switch between the screenshot and the application repeatedly in order to apply the help they are getting.

Is Animation the Solution?

Animation can be used to provide effective, detailed and step-by-step help to complete a task [1,2,4,8,11-14,16,17]. Animation may be excellent as a means to introduce unfamiliar ways of working, but it may not always lead to optimal learning of interface use [11]. Animated help does not normally support the multiple different ways of reading (in-depth, skimming, selection, and re-reading) that are possible with conventional text and graphics. An animation makes it difficult for users to move at their own pace, confining each user to follow a single level of software proficiency – the level the animation was written for. It is difficult for the novice user to pause an animation frequently to find the right controls on the application interface and catch up. It is also frustrating for the power user who may only be missing one key step of the interaction to have to sit through an exhaustive description of what he or she already knows. In addition, while most animated help allows skipping forward and backwards, animated help by its very nature enforces a temporal view of a task. Animation fails to give a representation of the task as a whole. The steps must be watched in sequence and subsequently remembered if they are to be perceived as a set. This hinders a user's ability to get the bigger picture of the task as it fits together into a whole. That is not a problem for someone who already understands the overall task, but can be unnecessarily confusing for a less well-informed user.

GRAPHSTRACT

Combining the idea of minimal manuals [3], the ease-of-recognition of graphical tokens, and animation's ability to convey the element of time in an interaction, we arrived at GraphStract. GraphStract creates a single page of image tokens for a multi-step task, so users can retain their sense of the entire task. Images are "focused screen grabs" which center on the actual control a user will have to manipulate. This reduces information clutter while still mapping easily to the 2D GUI environment. In addition, GraphStract allows for self-paced user help where a power user missing one step of the interaction can find that step at a glance, but the same GraphStract help page can also walk a novice user through a complex interaction step-by-step. One design option we are exploring is to have no textual description at all. This naturally saves space and of course in a very extreme way addresses the problem of conventionally verbose and indeed unread

descriptions. But will users be able to understand what to do solely from abstracted screenshots? Our pilot studies imply that they can. If larger testing proves otherwise, we can intersperse small amounts of text between the images as needed.

Design

The basic GraphStract idea is to show just a small part of the interaction screens such as the main top window frame, the menu bar and the button to be clicked (Figure 2). Fortunately, these elements are often located close together on many interfaces. We take the traditional one-dimensional 'lower means later' convention for sequence information and add to it indentation, which serves to signal the opening of a new sub-window (often a dialog box). This allows better perception of the larger idea, especially for times when the desired result requires many levels of choices, as can happen especially in actions involving the setting of preferences. In Figure 2 this is illustrated by a rather complex task (creating a watermark using the printer's built-in watermarking feature accessed through Microsoft Word) involving three nested dialog boxes, with completion of each indicated by clicking

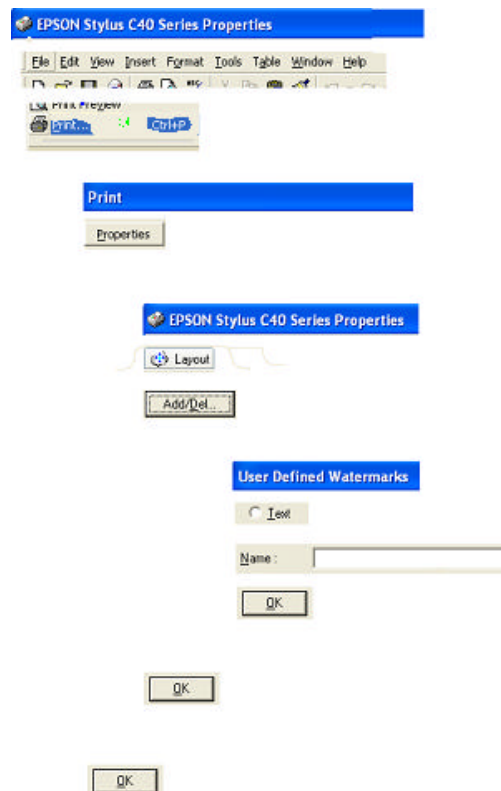


Figure 2. A GraphStract explanation of how to insert a watermark in Microsoft Word

on an OK button and the alignment of that button showing which dialog box it is associated with. Note that in actual use, the correspondence of the screen fragment with the users interaction with actual interface serves to disambiguate the meaning of the graphical steps.

We have done small-scale informal user studies and iterative design modifications. The idea of image-only explanations initially caused confusion with first time users due to its unfamiliarity. However, this perceptual gap only seems to occur once – repeat users would quickly jump into a task with only a glance at the GraphStract help sheet. To address this initial confusion, we are currently exploring adding an animated title bar to the system, wherein a copy of the target application’s title bar detaches from the application window and moves to the top of the help window – helping to form the link in the user’s mind between the application at hand and what the GraphStract help system represents (See Future Work). Given that it seems that users soon learn the meaning of the representation, we envision the animated title bar to be a feature that is needed only for the first few runs of the GraphStract help system.

Levels of Help

We are also exploring the idea of multiple levels of abstraction, allowing the system to help users with varying levels of experience. Level one, designed for the power user, shows just indented title bars with only the necessary-for-this-task buttons listed. In level two, both

miniature windows provide additional context, and exploit the power of thumbnail image to support recognition and quick location of a control.

The miniature screenshots do not need to be full size. They are intended to be used alongside the actual application. Thus they serve as a small-scale map for supporting recognition of the appropriate part of the real interface. The ideal size is still under investigation. The trade-off is between the level of legibility needed and screen real-estate.

PILOT STUDIES

Throughout the development of the GraphStract concept, small-scale informal pilot studies were conducted to get quick feedback on new elements and small tweaks we were making. Prototypes were initially developed on paper, using pencil sketches of the UI. Later prototypes were developed in Microsoft Paint, using elements of the UI cut and pasted from screenshots.

Subjects were computer-literate undergraduate students, most of whom owned a computer and used it daily for tasks like email, word processing, web browsing, and instant messaging. GraphStract is designed to be an end-user system, and as such we actively sought out users who were not majoring in a computer-related discipline.

For purposes of brevity and rapid prototyping, we constrained our pilot testing to tasks within Microsoft Word. We began each test by giving the user a task and asking them to perform it without any help. On most occasions, users quickly began to get agitated when they couldn’t figure it out themselves quickly or when the regular help (i.e. Mr. Paperclip) proved uninformative. The user was then given the GraphStract help sheet for the task (printed in color on a single page of 8.5x11 paper, much like figures 2 and 3) and asked to “see if this helps.” The GraphStract printout was explained as a help-sheet that a co-worker or friend put together. Investigators did not answer any subsequent questions, maintaining the situation of single-user interaction with GraphStract. At the completion of each pilot test, we asked users what confused them the most and encouraged them to help us brainstorm ideas that would improve GraphStract.

Most users found getting started with GraphStract to be the most difficult part. One user expressed sentiments very representative of the entire group of test users when his first words after seeing the GraphStract help page were “What *is* this?” Most users were able to overcome their initial “What is this?” reactions by noticing that the top title bar in GraphStract corresponded to the application title bar at the top of the computer screen, though the time it took users to notice this ranged from nearly-instantaneous to several seconds, prompting the user to look to the investigator for help (we of course did not respond). These reactions acknowledge that the GraphStract system is still a work in progress, and part of our planned future research will center on the transition into

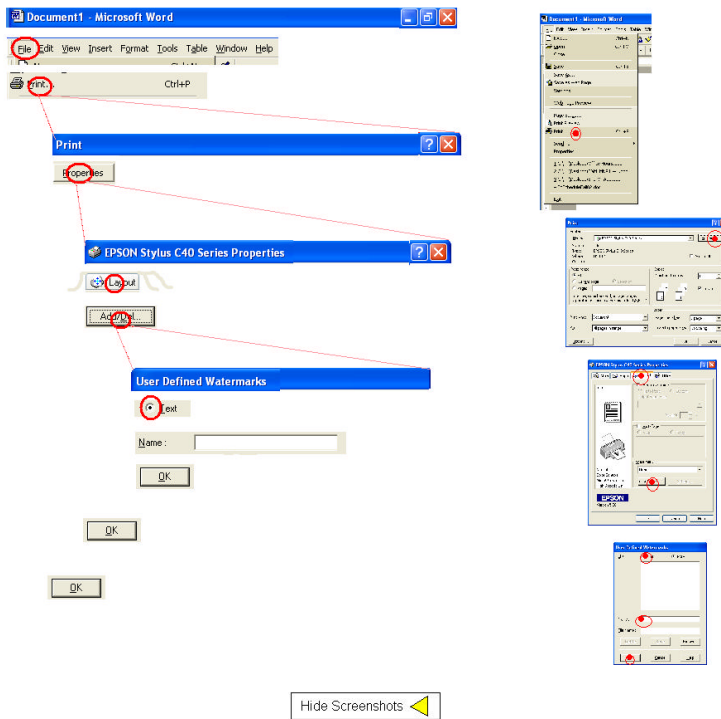


Figure 3. Providing more detailed graphical explanation on demand

the abstract help and miniature modular-window screenshots of each step of the interaction are shown, using two side-by-side panels (Figure 3). Red dot annotations on the miniature screenshots guide the user to the location of a control, and serve as a double-check in cases where two similarly named controls exist. The

the GraphStrat system for first-time users, possibly using animation to copy the application title bar into the top of the GraphStrat window in order to facilitate first-time understanding, an ideal currently being implemented in Visual Basic .NET.

From watching the eye and head movements of test users it was clear that users were using the GraphStrat information in small packets. Usually, test users would read aloud to themselves one indentation-worth of GraphStrat information, look up to the computer, and then repeat these words to themselves as they worked with the application interface. While this seems to show empirically that the divide-and-conquer indentations of GraphStrat seem to be working, at the same time this points to GraphStrat's limits in handling tasks in which users must make many interacts with the same application window.

Future Work

Planned future work with GraphStrat includes continuing iterative prototyping and testing, as well as larger-scale testing. We are continuing to develop the GraphStrat concept, including the addition of minimal animation to help familiarize first-time users of the system. In addition, we are exploring methods of adapting the system to address more complex explanation scenarios. These include: tasks involving many interactions with different parts of the screen at one level (i.e. without launching new dialogue boxes), tasks of significant length, and cross-application tasks where the indentation metaphor no longer holds as strongly. Development proceeds using a combination of pencil sketching, Microsoft Paint cut & paste collages from screenshots, demo development in Macromedia Flash, and rapid development in Visual Basic .NET. We envision this concept to be eventually implemented as a high-speed alternative to current text-based help.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under Grant No. 0081112. The authors would like to thank the anonymous reviewers of an earlier version of this paper, as well the help of as test users.

REFERENCES

1. Bharat, Krishna and Piyawadee "Noi" Sukaviriya. Animating user interfaces using animation servers. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology, Visualizing Information*, pages 69--79, 1993.
2. Byrne, Michael D.; Wood, Scott D.; Sukaviriya, Piyawadee Noi; Foley, James D.; Kieras, David: Automating Interface Evaluation. *Proceedings of the Conference on Human Factors in Computing Systems (CHI94)*. Boston, MA, USA, 1994. 232-237.
3. Carroll, J.M. (Ed.), *Minimalism beyond the Nurnberg funnel*. The MIT Press, Cambridge, MA, 1998.
4. de Rosi, F. and de Carolis B. and Pizzulito S. Software documentation with animated agents. In

- Proceedings 5th ERCIM Workshop on User Interfaces For All*, 1999.
5. Encarnacao, M. and S. Stoev, "An Application Independent Intelligent User Support System Exploiting Action-Sequence Based User Modelling", in *Proceedings of the Seventh International Conference on User Modeling*, 1999, pp. 245-254.
6. Lonczewski, F.: "Providing User Support for Interactive Applications with FUSE", in *Proceedings of the 1997 International Conference on Intelligent User Interfaces IUI'97* (Orlando, 6-9 January 1997), J. Moore, E. Edmonds, A. Puerta (Eds.), ACM Press, New York, USA, 1997, pp. 253-256.
7. Macromedia Corporation. <http://www.macromedia.com/>
8. Microsoft Corporation. Microsoft Word 2002.
9. Mynatt, Elizabeth D., Terry, Michael. Side Views: Persisten, On-Demand Previews for Open-Ended Tasks. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology* (October 27-30, Paris, France), ACM, NY, 2002, pp. 71-80
10. Nielsen, J. *Usability Engineering*. Academic Press, London, 1993.
11. Palmiter, Susan and Jay Elkerton: An evaluation of animated demonstrations of learning computer-based tasks, *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, p.257-263, April 27-May 02, 1991, New Orleans, Louisiana, USA.
12. Qarbon Corporation. <http://www.qarbon.com/>
13. Sukaviriya, Piyawadee Noi and Jeyakumar Muthukumarasamy and Anton Spaans and Hans J. J. de Graaff: Automatic Generation of Textual, Audio, and Animated Help in UIDE: The User Interface Design. *Advanced Visual Interfaces 1994*: 44-52.
14. Sukaviriya, Piyawadee Noi: Dynamic Construction of Animated Help from Application Context. *ACM Symposium on User Interface Software and Technology 1988*: 190-20
15. Sukaviriya, Piyawadee Noi; Foley, James D.: Supporting Adaptive Interfaces in a Knowledge-Based User Interface Environment. *Proceedings of the International Workshop on Intelligent User Interfaces (IUI93)*. Orlando, FL, USA, 1993. 107-114.
16. Sukaviriya, Piyawadee Noi; Isaacs, Ellen; Bharat, Krishna: Multimedia Help: A Prototype and an Experiment. *Proceedings of the Conference on Human Factors in Computing Systems (CHI92)*. Monterey, CA, USA, 1992. 433-434.
17. Sukaviriya, Piyawadee Noi; Foley, James D.; Griffith, Todd: A Second Generation User Interface Design Environment: The Model and The Runtime Architecture. *Proceedings of the Conference on Human Factors in computing systems (INTERCHI93)*. Amsterdam, The Netherlands, 1993. 375-382.
18. Twidale, M.B. Over the shoulder learning: supporting brief informal learning embedded in the work context. *Technical Report ISRN UIUCLIS--1999/2+CSCW*.