Graphical Abstract Help

Jeff Huang, Bo Lu, Michael B. Twidale
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
Champaign, IL 61820
huang6@uiuc.edu, bolu@illinoisalumni.org, twidale@uiuc.edu

ABSTRACT
We explore the use of abstracted versions of screenshots as part of an interface to support help giving. Graphstract, the software implementation of this graphical help system, extends the ideas of textually oriented Minimal Manuals to the use of 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. Graphical hints, such as jagged edges, red dots, and icons are also explored. The idea has been developed by iterative prototyping. In cases where the minimalist help is insufficient, ways of providing more detailed information on demand are investigated.

ACM Classification Keywords
H.5 Information Interfaces And Presentation: User Interfaces

Author Keywords
Help, Minimal Manuals, Visualization.

INTRODUCTION
Current implementations of automated user help on end-user applications such as Microsoft Word 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 (e.g. [8, 11]). Users are often looking for a quick way to make a slight change to their work, and may be unwilling to make the investment of time and effort 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 apply to the 2D GUI environment of the application.

We have developed methods for creating abbreviated forms of graphical help using layered elements of the application window, inspired both by the metaphor of abstracts of text,

CHINZ '05 July 6-8, 2005 Auckland, NZ

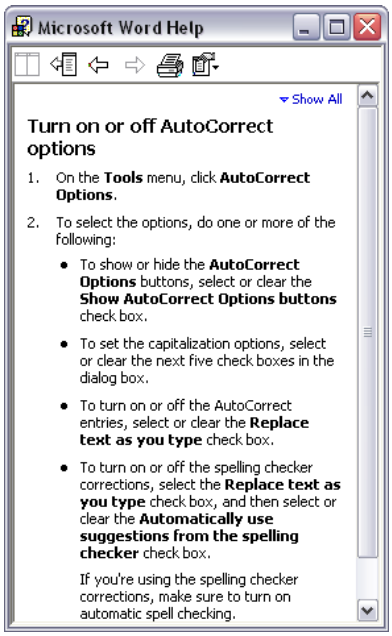

Figure 1. Traditional textual help in Microsoft Word.

and Carroll's Minimal Manuals [2]. This is intended to support both individuals and semi-formal peer user to peer user help giving. The new concept, Graphstract, has been tested in user studies and shown to work in a structured, test environment. We explore Graphstract through iterative prototyping and user studies, through which we gain insight on which features work and which don't. The process and evaluation of the studies are described later in the paper.

CURRENT APPROACHES

Text-centric Help
Current paper and online help attempt to combine text and graphics but often relegate graphics to a secondary role. Thumbnail-sized pictures of a specific control or a relevant dialog 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 as part of a cohesive group of interactions to achieve a task. Our assumption is that users don't want to spend the time and effort to read through the text, and there should be a better way to present the information.

Conventional Screenshots

Between colleagues, peer-to-peer help given often utilizes the interface seen on screen as a focus of pointing and discussion [11]. When creating help texts, screenshots are frequently used, both in online help and product manuals. 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 [4,5]. 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, since several pages are required to demonstrate a single 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. Lastly, this makes it harder for the user to skim the help if they are looking for just one piece of information.

Animation

Animation can be used to provide effective, detailed and step-by-step help to complete a task [1,5,7,9,10]. 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 [9]. Animated help does not normally support the multiple different ways of in-depth reading, 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 an explicit 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.

THE GRAPHSTRACT CONCEPT

Combining the idea of minimal manuals [2], the ease-of-recognition of graphical tokens, and the need to convey time in an interaction, we developed Graphstract (short for Graphical Abstracts). Graphstract aims to create a single page of image tokens for a multi-step task, so users can retain their sense of the entire task throughout the process. Images are "focused screen grabs" which center on the actual control where action is required. This reduces information clutter while still mapping easily to the 2D GUI environment. This naturally saves space and of course in a very extreme way addresses the problem of conventionally verbose and indeed unread descriptions. The objective is to give the user the information to know what actions to perform in a GUI environment using the combination of clues that Graphstract presents on screen.

Graphstract employs a metaphor of ripping out parts of screenshots of the steps required to perform a task, and pasting them into a scrapbook. Each snippet of help is a piece of the puzzle for the user. This metaphor is further expanded upon using features such as jagged edges and different levels of help.

The Graphstract Prototype

Our current prototype of Graphstract is an extreme implementation of graphical help containing no text help at all. Although we do not dispute that some accompanying text is often useful, we wanted to explore graphical help in its purest form, and understand how users react to image-only help. The current iteration of Graphstract was developed with user studies in mind, and incorporates many possible graphical help features: different levels of help, red guide dots, jagged edges, and hints using various icons. These features are described in detail in the following section.

DESIGN AND LAYOUT

The basic Graphstract idea is to show just small snippets of the interaction screen, such as the controls one must use: a menu, a tab, or a button (Figure 2). Luckily, these elements are often located close together. 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 the user a better overview of the entire task while they are several levels deep in the interface, which can happen with tasks such as setting preferences. Figure 2 illustrates Graphstract demonstrating toggling the auto capitalization feature, a task that involves a degree of complexity because it includes interactions with multiple nested dialog boxes. Graphstract is not intended to be used on its own, but always in conjunction with the application it is supporting, as illustrated in Figure 3. As such, it can exploit the advantages of minimalism by serving as pointers to elements of the interface of the main application without having to replicate them in their entirety. It aims to be the equivalent of a map to a city.

Layout of Controls

Controls are placed relative to their actual location on the windows of the application. Snippets of menus are positioned directly below title bars to replicate their placement in the actual interaction space. In general, we found that sticking with systematic token placement (ie. fixed distance between each token regardless of token location on the screen) was not as successful as roughly replicating the interaction space. Thus, controls are placed relative to their location in

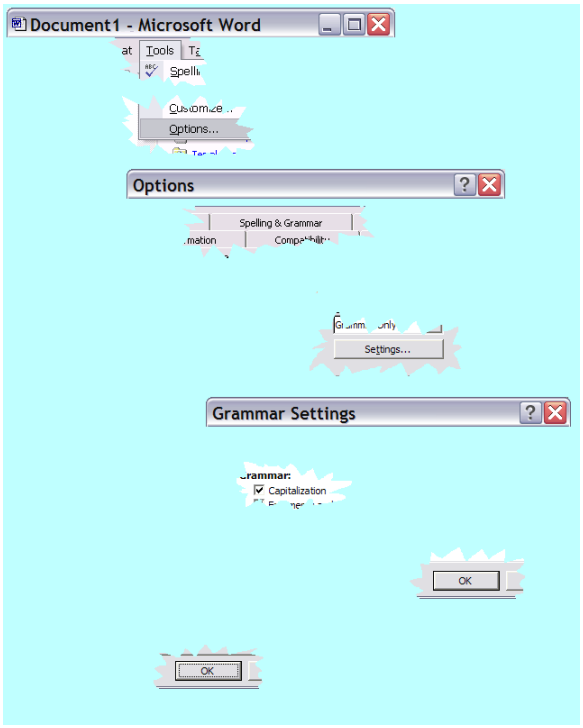

Figure 2. The Graphstract *snippet view* prototype, demonstrating how to toggle the auto-capitalization feature in Microsoft Word. Jagged edges emphasize the metaphor of ripping out interface elements and pasting them in a scrapbook. The layout of the image snippets show their relative placement in their parent control, ie. the menu or dialog.

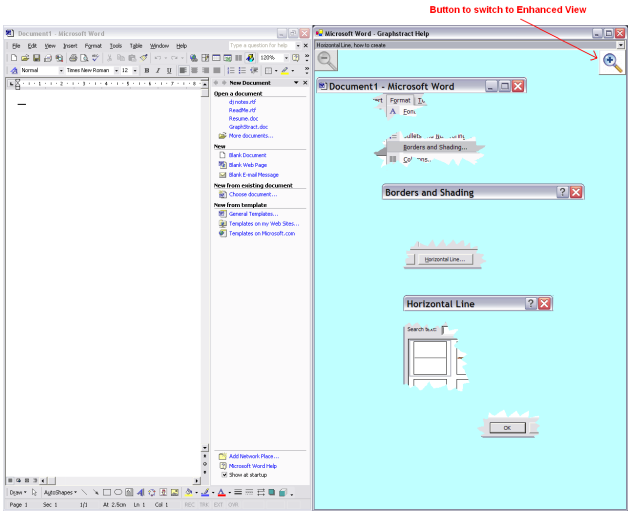

Figure 3. The Graphstract help system in *snippet view* running alongside Microsoft Word, demonstrating how to insert a horizontal line into a document in Microsoft Word.

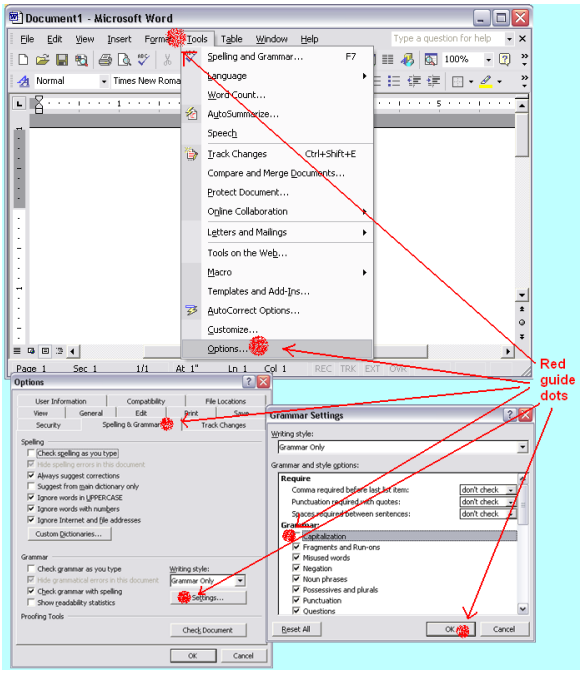

Figure 4. A Graphstract *enhanced view* showing the three step process of toggling the auto-capitalization feature in Microsoft Word. This is all displayed on half a screen and thus does not take up much screen estate. The red guide dots indicate to the user where to click.

the menu or dialog. For example, the OK button is placed closer to the right edge, which is where it can be found on the dialog.

Levels of Help

Graphstract explores the idea of multiple levels of abstraction, allowing the system to help users with varying levels of experience. Our implementation of Graphstract defaults to *snippet view*, which shows only the necessary actions required to perform the task, and offers an *enhanced view*, which consists of screenshots of the complete application to guide the user all the way. *Snippet view* (Figure 3) was designed to both allow power users to skim through and allow near-novice users to step themselves through an interaction. Alternately, *enhanced view* (Figure 4) allows very novice users to see the entire interaction screen step-by-step, using thumbnails to confirm and reassure users. Users navigate between the two views with the + and - magnification icons near the top of the window (visible in Figure 3).

We had originally implemented a version of Graphstract with 3 levels of help available, where the additional level of help included was even more concise than the current *snippet view*. It presented only the title bar of the dialogs. However, a few pilot tests using the 3 levels of help indicated that this concise level wasn't very useful. It lacked sufficient information to be of any help. Consequently it was removed, and *snipped view* and *enhanced view* are the only two levels of help implemented in the final Graphstract system.

Red Dots

Red dot annotations on Graphstract's *enhanced view* guide the user to the location of a control by showing the entire interaction space and highlighting the location of pertinent controls. The red dots are blotchy, rather than solid, to differentiate them from the standard UI an application might have. In addition, the red dots serve as a reassurance in cases where two similarly named controls exist.

Jagged Edges

Each graphical token in *snippet view* has jagged edges to illustrate that Graphstract is a representation of the interaction rather than a part of the interaction space. This gives the user the impression that the tokens are pieces of UI that have been ripped off the screen and thus helps to complete the metaphor. The only straight edges are in cases where a token rests near the border of the interaction space; an example is the OK button on many dialog boxes. Through pilot tests, we found that this system of preserving straight edges where they exist in the UI serves the dual purpose of meshing more closely with what the user sees on the screen (thus reassuring) and hinting as to the location of the control (especially in cases where the snippet is of an OK button, where the snippet often includes two straight edges that form a corner).

Hints with Icons

Icons were also found to be very helpful in providing the user with clues on performing the task when the snippets themselves were unclear. An example of this is the linked text task (Figure 5), which instructed the user to use the right-click button of the mouse, rather than the left, to access certain functions, and to drag and drop with the mouse. Icons were placed near the location of the action; in this case, it meant the mouse icon was useful when placed next to the location the user should be right-clicking.

FORMAL USER STUDIES

The objectives of the studies were to see whether users could understand the Graphstract concept and how users performed a task given Graphstract help versus Microsoft Word Help. We wanted to explore users' interactions with the help, rather than the act of locating the right help page. After roughly a dozen rounds of iterative informal pilot tests of Graphstract conducted in small batches, we conducted a formal study with 20 users in various disciplines, recruited from the University of Illinois.

Each user completed four tasks with Microsoft Word Help, and four tasks with our Graphstract prototype. The eight tasks used in the study were broken down into four pairs of similar difficulty. Half of the user group was asked to use Graphstract on the first task and Word Help on the second task for each pair of tasks, and vice versa for the other half of users. Microsoft Word was displayed on the left of the screen taking up about 2/3 of the screen, and the help system was displayed on its right. The users performed the task with Microsoft Word and were able to simultaneously use the help system.

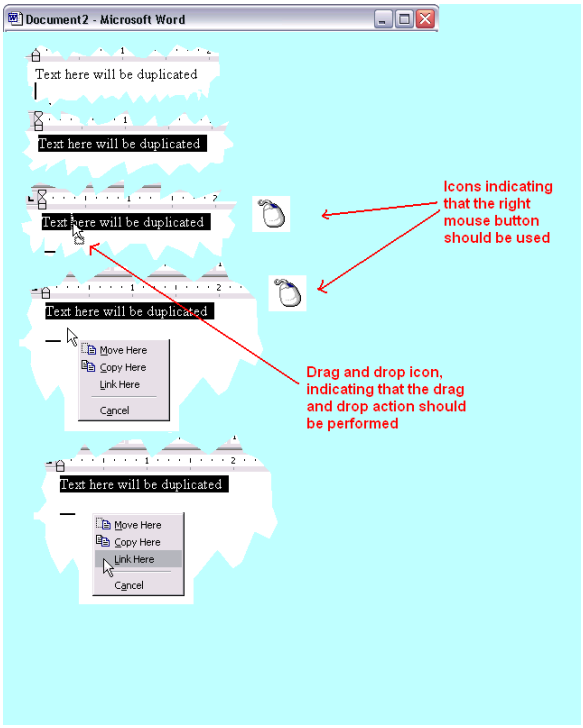

Figure 5. Icons hints that suggests using the right mouse button and drag and drop.

Task	Difficulty	Description
1	1	Highlight some text in yellow
2	1	Change the document to Outline View
3	2	Insert a horizontal line
4	2	Change some text to be upper-case
5	3	Set a new keyboard shortcut for FileClose
6	3	Create text that is linked from some other text
7	4	Disable Grammar Check's capitalization checking
8	4	Overlap two layers of text

Table 1. Summary of tasks presented to each user. A difficulty rating of 1 indicates an easy task while 4 is a hard task to complete.

Tasks

The eight tasks used in the formal studies are shown in Table 1, arranged in pairs in order of increasing difficulty (based on how likely it would be for an average user to have experience with a particular task). The users performed the tasks from the easiest to the most difficult task. Tasks were chosen to represent the most variety of actions a user could be asked to perform in Microsoft Word.

Procedure

Our approach for determining the effectiveness of Graphstract was similar to those used in other instructional help user studies [5]. Each user was told that Graphstract was a graphical help system and they could use either Microsoft Word Help or Graphstract (whichever one was presented to them) to help them complete various tasks. We did not explain how to use Graphstract, since we wanted to know whether they could learn how to use Graphstract for them-

User	**Task Number** (refer to Tasks section for task descriptions)							
	1	2	3	4	5	6	7	8
1	2:00	0:18	0:45	0:20	3:38	F	1:28	2:36
2	I	I	0:23	0:27	2:48	F	0:52	1:31
3	I	I	1:30	I	1:27	4:40	1:03	1:57
4	0:38	0:11	0:20	0:20	4:58	I	0:45	1:59
5	0:21	1:02	0:21	0:50	2:18	0:18	0:17	5:51
6	0:20	I	0:42	0:13	4:13	I	2:34	4:41
7	I	I	0:55	0:15	1:34	I	1:40	2:55
8	0:27	I	0:50	I	4:55	1:20	0:56	2:00
9	0:20	2:48	2:12	0:25	4:55	4:44	2:43	5:10
10	1:00	I	0:25	0:19	6:56	F	0:40	F
11	I	I	0:20	I	1:53	F	0:14	2:50
12	2:27	1:10	0:53	2:27	4:09	2:49	0:55	3:14
13	I	I	0:30	0:22	0:58	0:31	1:38	1:00
14	0:50	0:38	0:22	I	2:50	3:48	0:50	1:34
15	I	0:17	0:27	0:17	1:34	1:19	0:29	F
16	1:32	0:48	0:49	I	4:21	2:00	3:35	3:33
17	F	F	1:29	F	2:41	0:23	0:42	2:30
18	1:26	I	0:22	0:20	3:17	1:21	1:50	1:22
19	2:32	1:13	F	1:04	7:29	F	2:02	4:01
20	0:33	4:10	0:50	0:57	4:38	F	0:38	2:44

Table 2. Time in minutes it took each user to complete the task. Times in grey cells were trials done with the help of the Graphstract prototype. The remaining trials were done using Microsoft Word Help. An *I* indicates that the user performed the task instantaneously, ie. in 10 seconds or less. An *F* indicates that the user was unable to complete the task.

selves in order to emulate a real-world experience. For each task, we briefed the user on the details of the task, and opened up the corresponding Graphstract or Word Help application. Qualitative experiences were gathered by encouraging users to talk out loud during the interactions, debrief questions at the end of each task, and an overall questionnaire at the end of the study. Each task was timed to see how long a user took to complete it.

Quantitative Results

The complete set of results is shown in Table 2. A task is marked I for Instantaneous if the user completed the task immediately and without any hesitation as if they knew exactly what to do. We didn't record the exact time for these because any variation in time between instantaneous tasks would have been the result of arbitrary circumstances, such as the cursor being further away from a button to start with, rather than a measure of the help system. A task is marked F, ie. fail, if the user could not complete it. Fortunately, we found similar numbers of failed tasks in both help systems. If we ignore the few failed tasks for now, a summary of the results can be seen in Table 3. Note the considerable variability in completion time both within a task and between tasks. We explore this below.

We had hoped that learning to use Graphstract with no prior instruction would not impose too much of a performance burden, and that using graphical abstracts would be faster and less confusing for the average user to learn how to complete a task. This was partially confirmed; on average over all tasks, users were able to solve their problems 15% faster when using Graphstract than with the textual

Task	1	2	3	4	5	6	7	8
Difficulty	1	1	2	2	3	3	4	4
Graphstract Avg. Time	0:23	0:40	0:44	0:34	2:58	1:56	0:45	2:47
Word Help Avg. Time	1:08	0:51	0:47	0:24	4:04	1:31	1:50	2:57
Graphstract over Word Help Time	0.34	0.85	0.94	1.42	0.73	1.27	0.41	0.94

Table 3. Summary of results from user studies. Highlighted results indicate the faster of the two help systems. To calculate average speeds, the results in Table 2 were averaged. We counted tasks that were completed instantaneously (10 seconds or less) as 10 seconds; failures were not included in the average. A difficulty rating of 1 indicates an easy task while 4 is a hard task to complete.

Microsoft Word Help. However, Graphstract was not the fastest in every task.

Graphstract was substantially worse in tasks 4 and 6, the two methods were similar in effectiveness in tasks 2, 3 and 8, and Graphstract was substantially better in tasks 1, 5 and 7.

The two tasks in which Graphstract was less effective involved changing the text to upper-case and linking text respectively. In task 4, the textual help was very concise and clear about how to change the case of the text. In task 6, users had trouble figuring out the right click drag and drop because Graphstract was unable to show this easily. Another explanation is because these tasks were mainly text focused, and Graphstract lacked the ability to demonstrate features that were text-based because of its graphical nature. The results also indicated that using Graphstract versus textual help did not change the success rate, ie. the number of times the user gave up.

Qualitative Results

From the user studies, it was gathered that overall, users using Graphstract were able to solve their problem in less time than users using the built-in Microsoft Word textual help. On average, Graphstract users were able to speed up the time it took to complete a task in 6 of the 8 tasks. In a post study survey, all users stated that they enjoyed using Graphstract more and thought using Graphstract helped them complete a task quicker; they also stated that they would prefer it to textual help. Most users took some time to learn how the Graphstract prototype worked, but the time lost was easily compensated for once the users realized what Graphstract was.

Our observations of the users and their interaction with the different levels of help were interesting. Approximately two-thirds of the users stayed within the *snippet view* and did not use the *enhanced view* because they were unaware of the method that switched to that view (ie. clicking on the magnification button). See Figure 3. Of the users that did switch, approximately half of them switched back and forth between the two views, and the other half stayed with *enhanced view*. From these observations, it seems that

enhanced view is beneficial to some users but detrimental for others, and that its usage is based on preference. We believe that it would be a good idea to offer both modes as options to the user.

When users used Graphstract for the first time, about 30% of them attempted to click on the images in the Graphstract window. This is a problem with Graphstract because in a 2D environment on the computer, images (or controls) of controls and actual controls look exactly the same. A map of Canada would never be confused with Canada itself, because they are "made" of different materials. However, images of controls and controls are both just pixels. In order to prevent this type of confusion, we plan to explore the possibility of "fading" the Graphstract images into grey in the future.

When we implemented Graphstract, we hoped that several key features would enhance the user experience. We found that these were a hit-or-miss with each user. In *snippet view*, most users understood that the jagged edges meant that each pictorial piece was a snippet; however, there was a minority that did not understand their purpose and it only confused them. Likewise, of the few users that explored the *enhanced view*, most of them thought the red dots were useful guides, while the rest stated that the red dots got in the way.

The hint icon was used for the right-click drag and drop in the linked text task, and although it was helpful to the users, we believe that it still is weak in indicating that a right-click drag and drop action should be performed. Many users did not conclude from the icon that a right-click drag and drop was needed. Those that got it took a long time to figure it out. We believe that a possible solution is minimal animation, showing a mouse in the drag and drop action, combined with a pressed right mouse button.

The post study surveys also showed that the biggest problem users had with Graphstract was that it lacked any textual direction. Most commented that some text to explain Graphstract or guide the user in using Graphstract would be beneficial. We agree with this and believe that help systems should include some guiding text when necessary, based on the complexity of the task. The Graphstract prototype used in the user studies was purely graphical to examine the results of such a system.

Graphstract demonstrated that it was especially strong in one particular area - in tasks where the user had to navigate dialogs and interact with certain controls. From our studies, tasks 3, 5, and 7, which were *inserting a horizontal line*, *setting a keyboard shortcut*, and *disabling capitalization checking* were examples of this. For each of those tasks, users using Graphstract were able to complete them much faster, and our observations showed that they performed the tasks more smoothly. On average for the 3 tasks, Graphstract users were about 40% faster, much better than the average improvement for all the tasks. We believe this is because users are able to recognize the dialogs and controls to discern the next step to complete a task. It's less obvious what to do when presented with an unfamiliar image or one you cannot find.

Lastly, Graphstract was found to have a weakness at displaying interactions with combo boxes. Because of the way combo boxes work, both states of the combo box (opened and closed) can not be shown at the same time. We chose to display the open combo box in order to be able to show which item in the combo box to select, but some users were confused when seeing the dialog in the application without an open combo box, since the default state of a combo box is closed.

RESPONSES TO COMMON CONCERNS

The authors, in developing Graphstract, have found that several concerns commonly come to mind to those who see the system for the first time. To encourage discussion and help direct future work, we highlight some of these comments here and include our current thoughts:

- Crafting a Graphstract help system for large software packages such as Microsoft Office will admittedly take significant human effort and cannot be easily automated. However, the authors feel that this effort would not be significantly more than writing textual help files, which also must be done completely manually.
- With its heavy reliance on graphical representation, the Graphstract metaphor is very vulnerable to changes in the user environment. For example, Word's Personalized Menus feature hides rarely used menu options, and thus presents users with a different menu than what they see in a Graphstract representation. The authors acknowledge that this has been a stumbling block in user studies, and this comprises part of the future work.
- Graphstract lacks the ability to express keyboard commands. However, Graphstract's target platform is one which heavily involves a graphical user interface, where almost all commands can be done through interactions with menus, controls, dialogs, and toolbars.

FUTURE WORK

We are continuing to develop the Graphstract 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 (ie. without launching new dialog boxes), tasks of significant length, and cross-application tasks where the indentation metaphor no longer holds as strongly. For some of our studies, we noticed that users attempted to interact with the Graphstract images as if they were controls themselves. We plan to explore the possibility of "fading" the Graphstract images into grey to reduce the confusion some users may have experienced. We will also try to implement the suggestion of replace the red dots with red arrows instead in order to draw the attention of the user better. Lastly, we hope to explore using Graphstract to represent less-common interactions in the Windows user interface such as right-click drag

and drop as well as explore its adaptability to pen-based interactions.

Given that creating help takes a lot of time and effort, we are looking at ways in which a demonstration created for the benefit of helping a colleague can be recorded and reused in the future. One interesting side effect of the extreme minimalist version of Graphstract that we developed for the purpose of this study is that its lack of any accompanying text means that it has the potential to support the creation of multilingual help. That is, a sequence of steps in one language version of the software would yield a Graphstract sequence that in theory could be mapped directly to its equivalent sequence in another language. We hope to eventually bring this concept to a stage where it can be adapted to large-scale home and office software as an alternative to current textual help.

CONCLUSION

After looking at results from an experimental evaluation of the Graphstract prototype, the authors would like to propose the Graphstract concept as a relatively new concept for help systems. More space-friendly than their screenshot counterparts, and a better way to present information than text-centric help, we believe Graphstract is a strong alternative. Although Graphstract has some problems representing things such as right click drag and drop and keyboard shortcuts, its strength is in aiding a user to perform a task that involves dialogs and controls. The advantage of Graphstract in this situation is that users will be able to skim the graphics and match them with the situation they are seeing, so they are able to quickly decide the next course of action.

The Graphstract prototype presented in this paper was also an extreme case of pure graphical help. We acknowledge that and believe the correct balance of graphical help and guiding text to be the ideal implementation for a suitable help system. The Graphstract iteration used in the user studies was a focus on a pure graphical solution. Ideally, a user will be able to open up Help and search for help files as they do now with text help. Then, they are presented with a Graphstract solution to the problem they are facing on the side of their viewing screen. Following those steps, they are able to simultaneously work through the solution, skimming Graphstract when needed, and solving their problem quickly. This is what we believe Graphstract can do.

ACKNOWLEDGEMENTS

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. In addition, the test users provided invaluable feedback as well as great ideas, and without them this would not have been possible.

REFERENCES

1. Bharat, K. and Sukaviriya, P.N. “Animating user interfaces using animation servers,” in *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, ACM Press (1993), 69-79.
2. Carroll, J.M. (Ed.), *Minimalism beyond the Nurnberg funnel.* The MIT Press, Cambridge, MA, 1998.
3. 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.
4. Gellevij, M. and Van der Meij, H. “Empirical Proof for Presenting Screen Captures” in *Software Documentation Technical Communication 51*, 2 (2004) 224-238.
5. Harrison S., “A comparison of still, animated or nonillustrated on-line help with written or spoken instructions in a graphical user interface,” in *Proceedings of CHI95 (International conference on human factors in computing systems)*, Denver, May 1995, ACM Press, New York, pp 82-89.
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. Mynatt, E.D., Terry, M. “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.
8. Nielsen, J. “Usability Engineering,” *Academic Press*, London, 1993.
9. Palmiter, S. and Elkerton, J. “An evaluation of animated demonstrations of learning computer-based tasks”, *Proceedings of CHI 1991*, ACM Press (1991), 257-263.
10. Sukaviriya, P.N. “Dynamic Construction of Animated Help from Application Context,” *ACM Symposium on User Interface Software and Technology 1988*. ACM Press (1988), 190-20.
11. Twidale, M.B. “Over the shoulder learning: supporting brief informal learning embedded in the work context,” Technical Report ISRN UIUCLIS–1999/2+CSCW.