Some Design Refinements and Principles on the Appearance and Behavior of Marking Menus

Mark A. Tapia
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada, M5S 1A1
Tel.: 1 416 925 3953
E-mail: markt@dgp.utoronto.ca

Gordon Kurtenbach
Alias Research Ltd.
110 Richmond St. E
Toronto, Ontario, Canada, M5C 1 P 1
Tel.: 1416 362 8558
E-mail: gkurtenbach@aw.sgi.com

ABSTRACT

This paper describes some design refinements on marking menus and shows how these refinements embody interesting and relevant design principles for HCI. These refinements are based on the design principles of: (1) maintaining visual context, (2) hiding unnecessary information, and (3) supporting skill development by graphical feedback. The result is a new graphical representation and a more effective form of visual feedback and behavior for marking menus.

KEYWORDS: Menu layout, user interface design, pie menus, gestures, marking menus.

INTRODUCTION

While marking menus are functionally equivalent to standard linear pop-up or pull-down menus, they dramatically accelerate selection time for expert users and simplify the transition from novice to expert. Essentially, marking menus are a refinement of radial (or pie) menus [4] [10] integrating zigzag marks and hierarchical radial menus.

Marking menus support two selection methods. The first, "press and hold", is intended for a novice user not familiar with the particular menu layout. The method allows the novice to pop-up the menu by pressing down the mouse button and holding the mouse still for a fraction (1/3) of second, causing the menu to be displayed and allowing menu item selection by moving in the direction of the desired one. Like traditional menu systems, the menu item is executed as soon as the mouse button is released. The second selection method, "making a mark", is intended for an expert familiar with the layout of a particular menu. Instead of waiting for the menu display, the expert makes a selection simply by moving the mouse immediately after pressing the mouse button, causing an "ink trail" (the "mark") to be made while moving the cursor rather than displaying the menu. When the user releases the mouse button, the system examines the angle of the mark to determine the menu item to execute. In practice we have found that "press and hold" is not annoying for novices and easily avoided by experts.

Laboratory research has shown that marking menus are used as designed [6]: novices begin by pressing and holding to display the menu, and graduate to using marks as they become experts a technique that can be up to 10 times faster than using the menu display.

The major design principles and empirical testing of marking menus have been presented elsewhere [7] [8].

In this paper, we present seven design refinements on marking menus based on three design principles (see Table 1). We organize refinements by principle, providing examples to illustrate each and indicating how to apply them to other situations.

Principle	Refinement
Maintain visual context	- Display only the labels - Violate pie wedges - Make labels symmetric
Hide unnecessary information	- Hide parent menus
Support skill development using graphical feedback	- Use eight item menus - Use compass star with menu center - Show idealized marks

Table 1: The refinements to marking menus described in this paper and the underlying design principles.

Recently, marking menus have been introduced into StudioPaint V3, a paint program by Alias Research (see Figure 1). The performance we have observed under laboratory settings also occurs during "in field" use of the program. Many of the design refinements, presented here, are the result of incorporating marking menus into StudioPaint.

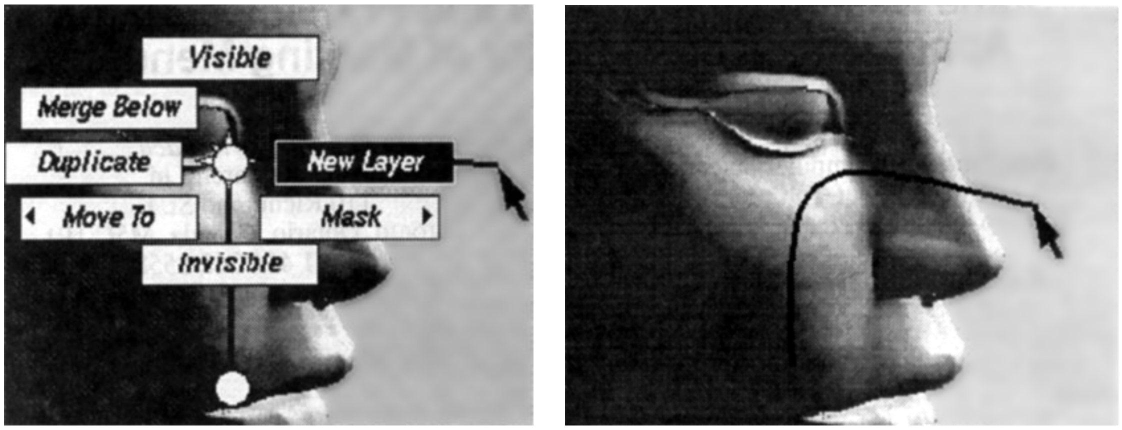

Figure 1: The new graphical representation for marking menus in Alias StudioPaint V3. On the left, the user has selected from the root menu (which only displays its center) and is now selecting a command (New Layer) from the second level menu; on the right, the user makes a mark to perform the same selection.

PRINCIPLE: MAINTAIN VISUAL CONTEXT

The graphical representation of marking menus shown in Figure 1 is a result of observing several problems with the initial implementation of pie chart menus, the more traditional graphical representation for radial menus (Figure 2).

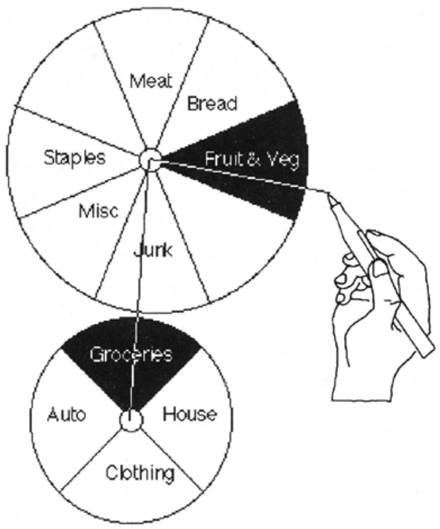

Figure 2: The "pie chart" style of graphical representation originally used for marking menus.

Refinement: "labels only" display

Figure 2 shows our previous graphical design for marking menus. Similar to other pie menu graphical designs [4], menus are displayed like pie charts with command names appearing within the wedges. The major problem with this graphic design is that menus are much larger than their linear menu equivalents - a consequence of the interaction of vertical pie wedges and horizontal text. In turn, the size of the menus made them visually disruptive when they were displayed and cleared. Furthermore, their large size made them awkward to use in constrained locations on the screen. For example, when menus appear near the edge of the screen, they can either be left clipped off by the edge of the screen it or translated towards the center of the screen to be completely visible. Using a graphical representation that consumed less screen real estate seemed desirable. In addition, the circular design was not aesthetically similar to rectangular graphics found in modern GUIs.

We realized that the problem was not that the menu popped up and "blinded" the user like a camera flash, but that the menu obliterated the user's visual focus. This effect was poignantly revealed when, in a real application, we compared popping up the menu with using a mark (see Figures 3a and 3b).

The application, called ConEd, allows a user to edit and view timelines of speech events, with each event detailing who is speaking and when they spoke [9]. The data appears in a "piano roll" representation with black rectangles representing speech events (Figure 3a shows a typical window in ConEd). The user applies a command to a particular event either by pointing to the event with the mouse and either making a mark over it (see Figure 3a) or using "press and hold" (pointing to it and pressing down the mouse button to display the menu, and then selecting from it, Figure 3b).

We observed that with "press and hold'" users would sometimes hesitate after the menu disappeared. They reported that sometimes they were not sure that they had really performed the action on the intended event (e.g., "had the event been deleted?"). However, this was not the case when using the mark, since the event was always in view.

The new "labels only" graphical representation (Figure 4) reveals more of the underlying context than the previous one (Figure 3b), since it only obscures the context underneath the textual labels, at the menu center, and along the mark. We believe that this helps the, user to maintain a visual awareness of the context when selecting from the menu

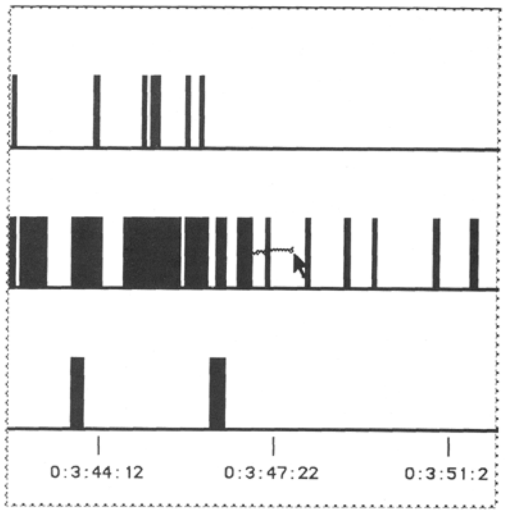

Figure 3a: Using a marking menu mark to delete an event in the "event time line" application. The mark occludes very little of the underlying data.

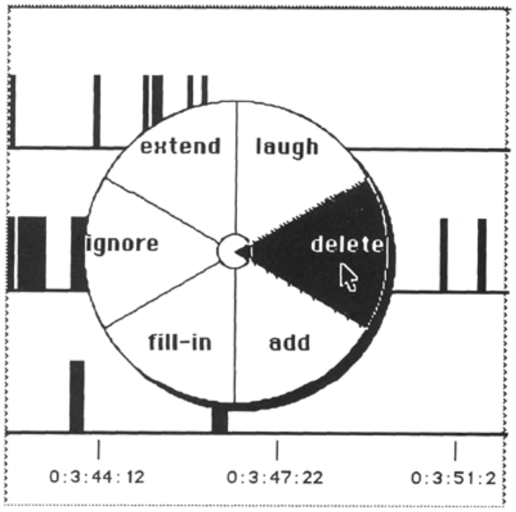

Figure 3b: The equivalent operation using the menu. The menu display obscures a large portion of the underlying data.

While the background behind the text of each label is not needed when the text itself can be distinguished from the data, this cannot be guaranteed in most applications. Instead, we use a technique similar to that used to display movie subtitles: the text appears in an opaque rectangle with a contrasting frame around it to distinguish it from data. A selected menu item is highlighted by reversing its color. Consequently, selected and unselected items have different visual representations and can be easily distinguished from the underlying data.

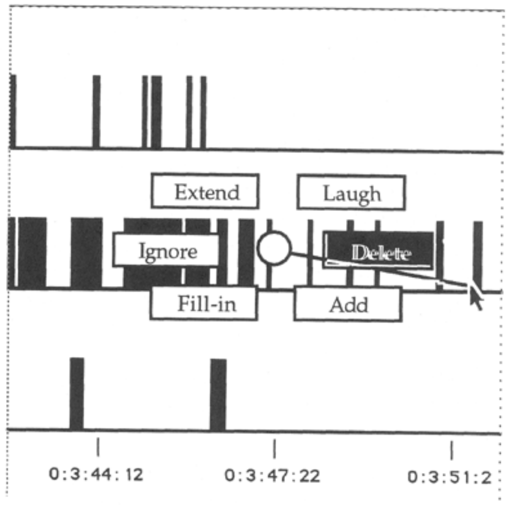

Figure 4: The new "labels only" graphical design for marking menus. The menu allows more of the underlying data to be visible even when displaying the menu.

Refinement: violate pie wedges

The "labels only" graphical representation has the additional advantage of compressing the menus, reducing both the screen space covered by the menu (its about the same size as traditional linear menus) and the size of the movements required for selection. Figure 5 shows how we allow text labels to "violate" pie wedges. To select an item, release the cursor in the associated wedge, as before, or inside the menu label when it is displayed.

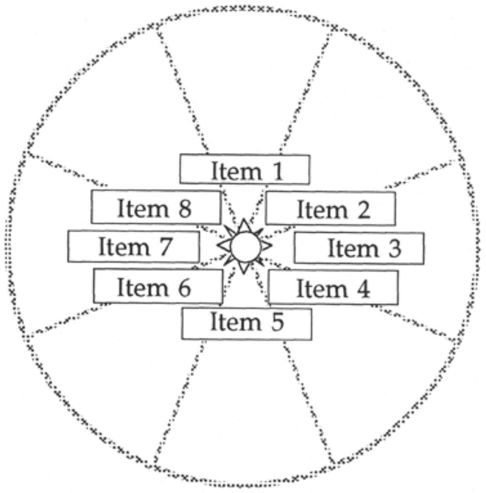

Figure 5: The "labels only" representation "violates" the wedges of a pie menu, however, the behavior remains mainly unchanged.

Another advantage of the new graphical representation is that it allows longer text items to be used. For example, menu labels on the left (or right) side of the menu can extend arbitrary distances to the left (or right). Menu items at the top or bottom can extend both to the left and right. This scheme allows marking menus to handle the same length of menu items considered reasonable in traditional linear menus.

Refinement: graphical symmetry of labels

We have also observed that it is important to use graphical symmetry in a marking menu to make it visually attractive. We found that a menu with each menu label a different size appears busy and disorganized. Our current scheme sets all the menu label boxes to the size of the largest menu item. This ensures symmetry but has the disadvantage that one overly large menu item can cause all menu items to be overly large. A refinement of this scheme treats single large menu items as exceptions, maintaining horizontal symmetry by forcing horizontal pairs of menu items to be of equal size.

We explored other ways of reducing menu size before settling on the design described above. Since menu size depends on text size, smaller menus can be generated by using smaller font sizes. Unfortunately, as the text becomes smaller, it becomes more difficult to read, effectively limiting the solution.

Another possibility is to change the orientation of the text, displaying it at an angle. Unfortunately this makes text difficult to read (e.g., reading a title on the spine of a book placed vertically). Implementing this method would have required us to devise our own low level routines to draw text at varying angles. For these two reasons, angling text was not an attractive method.

PRINCIPLE: HIDE UNNECESSARY INFORMATION

Refinement: Hide parent menus

In traditional linear menu systems, when a user descends a menu hierarchy, child submenus and their ancestors remain on the screen, allowing a user to move back to parent menu items and select from different submenus. Our original design of marking menus used such a scheme (see Figure 2): Journeying through a hierarchy of menus left a trail of parent menus. This made it easy for the user to back-up in the menu hierarchy using a technique similar to linear menus first, pointing to a parent menu item closed all other submenus and displayed items for that particular parent menu item; and second, pointing to the center of a parent menu caused all child submenus to be closed, displaying items in the parent menu.

While this scheme allows users to back-up, reselect, and browse menu hierarchies, it creates a clutter of parent menus on the screen that occludes the data. Adopting our new "labels only" graphical representation made this problem disappear while creating a new problem: When parent and child menus overlapped, it was hard to determine whether a particular menu label was part of a parent or child menu.

We solved this problem by closing the parent menu and leaving only the "center hole" graphic of the parent menu, as soon as the user selects a submenu. This not only solved the problem that parent menus cluttered the screen but also eliminated the risk of accidentally pointing to a parent menu item. While it is still possible to back-up in the menu hierarchy, this is now restricted to parent menus, not to items within them.

There are other advantages to hiding parent menu items: first, reducing the clutter allows the user to concentrate on the currently available menu items; and second, it emphasizes the path to a particular menu selection (i.e., the centers of the parent menu items are connected by lines; see Figure 1), with each path corresponding to the shape of the zigzag mark needed to select the particular menu item. We believe that this may help users to learn both the menu items available at each level and the correspondence between zigzag shapes and menu items.

PRINCIPLE: SUPPORT SKILL DEVELOPMENT BY GRAPHICAL FEEDBACK

Refinements: Use eight item menus, compass star center

In practice, we have found that when text labels violate wedge boundaries, there is no affect on selection performance. We believe this is due to our design decision to constrain all our marking menus to eight items, based on the eight directions of a compass. In our new graphical representation (Figure 5) the center of the menu is a compass star In this way, even though a menu item may spread beyond its wedge, it is still clear that each menu item corresponds to one of the compass directions.

Refinement: Show Idealized Marks

After introducing marking menus into Alias StudioPaint, users complained that when they used a mark it was hard to tell whether they had drawn the mark correctly and invoked the intended command. We asked ourselves "why didn't the users of ConEd have this problem?". It might have been because the menu in ConEd was simpler and therefore a user remembered the menu item associated with a mark. In contrast, StudioPaint's menu was more complicated making it easier to forget and consequently users were unsure of whether they had made the correct mark. StudioPaint users reported uncertainty about a mark even when they where sure they drew it correctly; they just did not know whether the system had correctly recognized it.

Upon closer examination, we discovered the problem was not with marking menus but with StudioPaint, since it did not provide feedback indicating successful invocation of the command. However, many of the commands in the marking menu could also be found in the traditional linear pull-down menus from the menubar. With this method, users were not confused about whether a command had been successfully invoked.

Thus, it was clear that unless the system responded quickly and with sufficient feedback, users were not sure that the system really recognized the mark. As a result, since they frequently thought the mark was not recognized correctly, users re-issued the mark and forced the system to re-execute the command. In cases where the command required a long time to complete (e.g., when creating a new image layer for painting), the delay was very annoying.

Initially, we tried to persuade the designers of StudioPaint to provide better feedback on command execution and completion. However, they pointed out that traditional pull-down menubar items did not require the extra feedback. Thus, it was clear that using a mark to issue a command did not provide feedback present in traditional menubar menus.

Therefore, we modified marking menus to provide feedback to the user on how the system interpreted the mark. In some respects, this mechanism corresponds to the way Macintosh menu items flash when selected. In contrast, after our system has recognized the mark and removed it from the screen, it continues to display an idealized mark (see Figure 6) with the associated label of the selected menu item until the system completes the command. The size and position of the idealized mark are based on the user's mark.

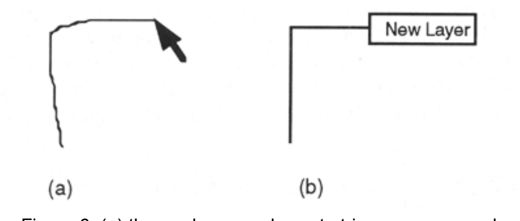

Figure 6: (a) the mark a user draws to trigger a command; (b) the feedback given to the user to indicate how the system interpreted the mark and which command it invoked.

DISCUSSION

In many cases, features of a good interaction technique are artifacts of good design principles. Since the fundamental design principles behind marking menus have been covered elsewhere [8], we now discuss the general design and cognitive principles that we believe underlie our design refinements.

Maintaining Visual Context

Reducing the amount of occlusion created by a pop-up menu is based on the hypothesis that the graphic occludes a user's visual focus and causes the user to lose the context, forcing the user to spend time re-acquiring the context after the menu disappears.

This issue is related to the more general notion of visual attention in interface design. In many applications, user must divide their visual attention between the data (or the "context") being operated upon and widgets that trigger operations. This division can occur both in the spatial domain, displaying menus and dialog boxes in a different space on screen, and in the temporal domain, by temporarily layering menus or dialogs over top the context.

This division creates a dilemma for the user interface designer. The problem is that, since the context and operations on the context are conceptually intertwined, it may be desirable to see the object being operated upon while operating on it. Unfortunately, both spatial and temporal divisions preclude this.

Effectively, our "labels only" design for marking menus circumvents the spatial and temporal division constraint by supporting dual attention, since both the menu (UI widget) and the context appear at the same time and are close together in the visual field. Other researchers [1] [2] [3] have explored this general notion by creating see through UI widgets. The "labels only" technique in marking menus is only one of several that support dual attention. For example, [2] investigates the effect of varying the transparency (opacity) of UI widgets.

As for tangible benefits, the effect of supporting dual attention is that a user is not forced to divide visual focus between a spatially displaced context and a UI widget and to re-acquire the visual image of the context after it has been obliterated by a pop-up graphic. Both these benefits translate into faster task performance.

Users reported another more intangible benefit when using marks: Using marks, instead of menus, has a more direct feel, analogous to applying an operation directly on an object. Because this observation is vague, we can only speculate on the source. First, the speed of the mark may influence the perception since there is evidence that the more responsive a system is, the more a user feels that the system is being directly manipulated (Hutchins, Hollan, & Norman 1986). Second, the proximity of the menus or the marks within the users focus may be another influence (e.g., the user does not have to make trips to and from the menu bar). Investigating these observations may help to define more clearly what is meant by "direct manipulation" and to quantify what makes some interaction techniques "feel right".

Information Hiding

One example of information hiding is to conceal parent menu items when descending a marking menu hierarchy. In many cases, leaving the parent menu items displayed produces clutter that confuses the user. Hiding the menu has a cost - while a user cannot back-up to a cousin menu in a single step, they can select the parent menu by pausing on the central hole and choosing another item. We feel that the cost is worth the benefit over time, since this situation only occurs when a user browses an unfamiliar menu structure. One fundamental design principle for marking menus is optimizing speed of selection for the expert user, not for the novices who prefer guided exploration to speed.

We claim that, frequently, UI designs that work for novices are clumsy for experts. In contrast, with marking menus novices report that while browsing is slower than with traditional linear menus, when they become experts, they do not notice the problem.

Support skill development by graphical feedback

Showing the user an idealized version of a mark may encourage expert behavior (i.e., selection using small, fast marks). The success of correctly recognizing a mark depends on the user's accuracy in drawing it. Showing a user an idealized version of the mark not only helps to determine the menu item selected, but also provides clues on how to make a more accurate mark. The intention is that this, in turn, will help users to improve the accuracy of their marks and the recognition rate.

Idealized marks appear whenever the user makes a selection from the menu. The intention is to reinforce to the user the image of the mark needed to invoke the menu item. A similar scheme could be applied to other recognition based systems.

One key to performing extremely fast selections with marking menus is by drawing very small marks. The recognition of a mark depends only on its shape not its size. Thus, a particular selection from a menu four levels deep may be made by a four-inch long mark, or more quickly by an one-inch mark of the same shape. Since the idealized version of the mark is drawn at the same size as the mark the user entered (e.g., a one inch mark creates a one inch idealized mark), we hope that as the user draws smaller marks, the smaller idealized mark will reinforce the visual image of smaller and faster marks.

CONCLUSIONS AND FUTURE RESEARCH

This paper described some design refinements on marking menus and how these refinements embodied interesting and relevant design principles for HCI. The design principles arose iteratively from analyzing design artifacts, rather than from first principles. We were then able to reapply design principles to refine the design.

While we have not conducted formal tests of the refinements in this paper, our refinements were based on user preferences. During the iterative process, we listened to our own design preferences and those from a pool of approximately 15 users of marking menus in StudioPaint and ConEd. As a result, we feel our refinements are valid. After implementing the "labels only" representation, we have never used or been asked to use the pie style representation. Similarly, the "idealized mark" feedback seems to have addressed user insecurities about mark recognition. The notion of "graphical symmetry of labels" arose from a graphical designer who complained that asymmetric menu items in StudioPaint looked graphically messy. While the benefits of "hiding parent menus" are immediately apparent when the parent menus are displayed, none of our users has requested the display. Finally, no one also complained about the labels "violating pie wedges".

We could run formal experiments to perform rigorous tests of our design refinements to answer a number of questions. Does a user maintain more visual context with the "labels only" display than with the "pie style"? Do idealized marks help users learn marks more quickly? Currently we are using our design refinements because users clearly prefer them.

While three design principles can be extended to other situations, the designer must exercise caution. For example, consider invoking a submenu that does not apply to the current application (e.g., an accessory menu that selects electronic mail). Should the underlying application data be hidden to eliminate unnecessary information? Should the data remain to maintain visual context? Future research will define our design principles (especially "maintain visual context") in more detail so that they can be applied to different situations.

We are continuing to make refinements to marking menus, especially while introducing them into other commercial products. The "labels only" representation is a large step forward in making marking menus "industrial strength" and graphically compatible with modern GUIs. The focus of our current research is on using marking menus in conjunction with other GUI interaction techniques and ToolGlass technologies.

Is this level of attention to detail warranted for something as trivial as menu selection? Our feeling is that menu selection is a fundamental, high frequency operation in modern GUIs and that, consequently, small improvements can have major benefits. The positive response from users supports this claim.

We hope this paper will help future implementers of marking menus and that other HCI designers can use the design principles presented in this paper to generate or refine designs.

ACKNOWLEDGMENT

We thank George Fitzmaurice and Bill Buxton for their comments and assistance in writing this paper. We gratefully acknowledge the support of our laboratory from the following organizations: the Natural Sciences and Engineering Research Council (NSERC), Apple Computer, the Information Technology Research Center (ITRC), Xerox PARC, Alias Research Inc., and the Ontario Telepresence Project.

REFERENCES

1. Bier E., Stone M., Fishkin K., Buxton W.A., and Baudel T. A Taxonomy of See-Through Tools. In *Proceedings of CHI `94,*. pp. 358-364.

2. Harrison B.L., Ishii H., Vicente K.J., Buxton W.A.S) Transparent Layered User Interfaces: An Evaluation of a Display Design to Enhance Focused and Divided Attention. In *Proceedings of CHI `95*, (May 07-1), Denver, p. 317 -324.

3. Harrison B.L., Kurtenbach G., and Vicente K.J. An Experimental Evaluation of Transparent User Interface Tools and Information Content. To appear in *Proceedings of UIST '95*., 1995

4. Hopkins D. The design and implementation of pie menus. In *Dr. Dobb's Journal 1*, 1991, 6:12 pp. 16-26.

5. Hutchins E., Hollan J., and Norman D. Direct Manipulation Interfaces. In Norman D. and Draper S. (Eds.), *User Centered System Design*, Lawrence Erlbaum Associates, 1986, pp. 118 -123.

6. Kurtenbach G. The Design and Evaluation of Marking Menus, *Ph.D. Thesis, University of Toronto*, Toronto, Canada., 1993.

7. Kurtenbach G. and Buxton W.A.S User Learning and Performance with Marking Menus, *Proceedings of CHI '94*, April 24-28, pp. 258 -264.

8. Kurtenbach G., Moran T. and Buxton W.A.S Contextual Animation of Gestural Commands. In *Computer Graphics Forum*, V13 (5), 1994, pp. 305-314.

9. Sellen A.J. "Speech patterns in video-mediated conversation. In proceedings of the CHI `92 *Conference on Human Factors in Computing Systems*; New York: ACM.. May 3-7, pp 49 –59.

10. Wiseman N.E., Lemke H.U., and Hiles J.O. "PIXIE: A New Approach to Graphical Man-machine Communication". In *Proceedings of 1969 CAD Conference Southhampton* 463 IEEE Conference Publication 51, 1969.