Pies: Implementation and Evaluation of Circular Menus

Don Hopkins Jack Callahan Mark Weiser

Heterogeneous Systems Lab Computer Science Department University of Maryland, College Park

ABSTRACT

Menus on computer displays are usually a linear row or column of choices. We propose an alternative to these rectilinear menus, called the Pie menu. The choices of a Pie Menu are positioned in a circle around the cursor. The direction in which the cursor is moved makes the menu selection, and the length of motion is available as a second input. We discuss the implementation, evaluation, and application of pie menus. We have implemented them in the MIT X window system, where they are completely compatible with the standard menu package. This is a robust implementation in everyday use. We have also evaluated pie menus in a controlled experiment comparing rectilinear menus with pies on a series of tasks chosen to show both rectilinear and pies to good advantage. Our computer-naive subjects, regardless of task, made selections faster and made fewer mistakes using pie menus. Finally, we offer a bit of speculation on the theoretical advantages of pie menus in new applications (or re-organized old applications). Three such advantages are: (1) the return of two kinds of information from every pie selection (angle and radius), and (2) the kinesthetics of nested menu selections and large angular arm motions, (3) the existence of natural opposites (left/right, up/down) in a pie menu.

1. Introduction

We are proposing a new menu format, pie menus. We believe pie menus offer significant advantages over other kinds of menus, yet because they are so different they deserve require from many viewpoints. This paper offers three viewpoints: the implementation of pies in an existing window system (X Windows), the evaluation of pies vs. traditional menus in a controlled experiment, and the potential for pies over traditional menus for future applications. We believe the three viewpoints converge to an implementable, efficacious, and exciting idea. Before starting into the viewpoints, the introduction below reviews traditional menus, and introduces pies.

1.1. Traditional Menus

When it comes to presenting a list of choices to the user, most computer systems have been limited, largely by the available hardware and software, to a rectilinear format (Figure 1). The items are listed from top to bottom, sometimes with an index number corresponding to the item of choice. Occasionally, the lists are multicolumned, have multiple items per line, or are even hierarchical (i.e. indented sub-choices), but for the most part lie in a strictly one dimensional structure. Many of these menus are static on the display screen or activated from mouse actions in two formats: pull-down (menu appears at a fixed label on screen when mouse directed) or pop-up (menu appears anywhere within a fixed area, occasionally the whole screen)[Shneiderman 86].

Even so, many systems have used the two dimensional nature of the computer display to the advantage of certain menu applications. Many flight simulation program, for example, lay out directional headings in a typical compass format (Figure 2).

Item placement in menus has been an important research topic for many years. Menu organization is typically divided into three types[Dray et al 81]: alphabetical/numerical, categorical (functional), and random ordering. It is generally agreed that the performance of subjects (i.e. time to seek a target) with different placement styles converges with practice[Card 82, Perlman 84]. Further studies [McDonald et al 83, Perlman 84] revealed that a functional placement of items is superior when the task domain is unambiguous to the user whereas an alphabetic organization can be useful in uncertain task descriptions. All of these studies have mainly concentrated on the rectilinear display format.

Most commercial systems available today make a general package of linear menu presentation available to the programmer. Libraries of software of menu display routines are widely used as a default by programmers of many window systems and applications[Xerox 85, Sun 86]. These libraries often have default display mechanisms which could easily accomodate other presentation methods without changing a single line of application code. Pie menus are such an alternative presentation method, one with many advantages and few disadvantages (see final section).

1.2. Pie Menus

Pie menus are a concept which can have many implementations, one of which is described in a later section. As a concept, pie menus have the following properties:

(1) The selection made from a pie menu is based upon the angle between the position of the mouse *down* event (the menu center) and that of a later mouse *up* event. Sometimes, in some implementations, there may be intervening mouse *up*/*downs* before the final selecting *up* event.

(2) The center of a pie menu contains a dead zone a few pixels wide. When the cursor is in the dead zone no selection is made. Normally when a pie menu first appears the cursor is warped into the dead zone.

(3) Pie menus may permit selection even when the cursor is outside menu borders, or they may de-select when the cursor leaves the window borders.

(4) Pie menus optionally return the distance from the menu center, as well as the item selected.

Conceptually pie menus can parallel rectilinear menus in functionality. One can have nested pie menus, either ‘stack-of-cards’ or ‘walking’ (Figure 3). Pies can be thought of as simply an alternative user interface into a common underlying selection process.

Notice that pies have two important differences from normal menus on simple selection tasks: all items are equidistant from the initial mouse position, and greater resolution of selection is easily available via greater motion. The last point would have significance in a situation made noisy by any of the following factors: human (hand jitter), electro-mecahnical (mouse jitter), environmental (room jitter). By moving the mouse a larger distance from the pie menu center a more accurate selection is assured. This feature of pie menus also facilitates ‘mouse-ahead’. Our experience has been that even an experienced and confident menu user will not mouse ahead to a rectilinear menu item more than two or three from the top, while a pie menu user easily mouses ahead to any item on menus with as many as eight items.

2. Pie Implementation

2.1. Introduction and problems

Currently, pie menus are do not exist in any commercial systems. As far as we know, we have the only implementation. Imaginative menu formats, however, are an inevitable future with the latest advances in window management systems[Gosling 85]. Window imaging systems using technology from laser printing protocol standards such as Interpress[Xerox 84] and PostScript[Systems 85] will make it possible to display conical section shaped windows effectively on a bitmapped display. Circular, elliptical, or even triangular windows and menus will be possible to implement easily. There are some obvious advantages to this organization for particular applications: compass directions, time, angular degrees, and diametrically opposed or orthogonal function names are some groupings of items that seem to fit well into the mold of the pie menu design.

Before discussing our implementation itself, consider some problems that any implementation will confront: layout within windows, and menu warping near screen edges.

There are two kinds of layout problems. The first is simply that pies tend to take up more space than rectilinear menus because of the non-uniform space in a pie selection ‘slice’. Although all our menus have had strictly horizontal or vertical letter placement, even slanted letters would not fit much better (Figure 4). A triangle is hard to fill completely with most text or graphics found in menu items. One could argue that the unoccupied space in pie slices is not wasted, since it participates in selection, but if screen space is critical it is still annoying.

The second layout problem is simply how to arrange text within a pie menu. Which way should letters travel, or if images are used, where should they be within the circle? Should the radius of the pie enlarge to make all text and/or pictures fit naturally? These are unresolved questions which experience and experiment can help answer.

The menu warping problem is common to all popup menu systems, but is particularily acute for pie menus because of the requirement that the cursor start in the *center* of the menu. Normally a menu will popup (or pulldown) starting at the position of the cursor when the menu is first activated. However, if the cursor is close to the edge of the screen the menu may not completely fit, and must be displayed in a slightly different screen position. Because of mouse-ahead and other user expectations, if the menu is displayed askew then the cursor must also be moved so that it is in the same position relative to the menu that it would have been had the menu not need to be moved. This automatic moving of cursor and menu near the edge of the screen is called ‘mouse warping’.

Rectilinear menus are normally smaller than pie menus, and the cursor normally starts near the edge of a rectilinear menu, often at the top edge or the top item. The large size of pie menus, and the need to move the mouse to the middle of this larger size, make good warping important.

The initial warp algorithm looked at the offset between the mouse down event and the current mouse location, and warped the mouse to the center of the menu plus that offset. However, this complicates things if there are more input events pending. The positions associated with pending events are still relative to the unwarped mouse position, and in particular the angle towards the pending mouse up event that specifies the current selection is different after the mouse has been warped. Also, if the last pending event is a mouse down, the angle between that event and the current mouse position changes.

The solution we currently use is to only put up a menu when there are no pending mouse button events. When there are events pending, they are acted upon without putting up a menu.

When there are not, and more are needed to complete a selection, a menu is actually put up. If the menu would not fit completely on the screen, it is moved by the appropriate offset back onto the screen, and the mouse is warped to its current location plus that same offset. Normal tracking then begins.

For instance, if at the time the mouse down event is received, the mouse up event that completes the selection is already pending, then there is already enough information to make the selection. It is not appropriate to look at the current mouse position, because the selection has already been completely specified. At this point, putting the menu up serves no purpose but feedback, and slows down interaction. The selection can be made and acted upon without even showing a menu, if feedback is not required, or if acting on a selection provides the feedback.

2.2. Implementation Details

Our implementation of pie menus in MIT's X windows (V10r2)[Gettys 86] uses no new window management techniques, and in fact places the circularly arranged selection items within a rectangular popup window (Figure 5). Adding pie menus to X Windows required writing ZZZZZZZZZZZZZZ new lines of code, and changes to ZZZZZZZZZZZZZ lines of existing code in ZZZZZZZZZZZZZ different modules. The task took one of us (Don Hopkins) about a month of part-time work.

At the time pie menus were implemented (June 1986) there was not a standard menu package in X windows. Different applications used their own libraries. Therefore, our implementation was put into one of those applications, 'uwm'. Uwm is the most flexible of the X window managers, and is extensively customized for individual users by a special startup file ('.uwmrc'). Customization can included the specification of new popup menus, and the action to be taken when menu items are selected.

We augmented the language of the .uwmrc file to include new menu specification keywords. There is a global keyword, 'piemenu', which when set causes all menus to be pie menus. There are also new keywords for individual menu specifications: 'piemenu' and 'pulldownmenus'. 'Pulldownmenus' forces the usual uwm menu, 'piemenu' the new style, regardless of the global keyword. For pie menus, the user can also specify the radius of the menu, and the initial offset in degrees of the first item. By default the first item is put at the '12 o'clock' position.

The pie code was integrated into the existing uwm menu code so that no part of uwm *except* the menu implemenation (and the .uwmrc parser) knew anything about pies. The result was a very smooth integration, upward compatible with existing use but by setting one variable upgradable to make all menus pies.

There is now (January 1987) a second implementation of pie menus, mostly compatible with the normal Sun Microsystems menu library, and living in SunView. This required changing no lines of existing Sun source code, but is itself about 500 new lines of code. Existing modules can use pie menus by compiling against the new library. Not all features of the Sun menu library have been implemented, just all those that are used by the background menu process, 'suntools'. This implementation took one of us (Mark Weiser) a weekend.

3. Evaluation

3.1. Hypotheses

We performed a controlled experiment[1] to test two hypothesis: that pie menus decrease the seek time for menu items and that pie menus are especially useful in menu applications suited for a circular format or diametrically opposed item sets.

We expected pie menus to do better primarily because of the decreased distance to the target in pies when compared to rectilinear menus. The distance to an item in any menu style can be defined as the minimum distance needed to highlight the item as selected. In both menu styles, this is defined by a region rather than a point. This region is typically of greater area than the actual target (Figure 6). Once the cursor has entered the region, the item is highlighted as feedback to the user. Note that the regions extend beyond the menu window itself. Even with linear menus where the cursor may be placed initially in the middle of the list, the outer item choices suffer from their distance from the initial cursor location. Pie menus, in fact, enjoy a two fold advantage because of their unique design: all items are equidistant, and that distance is in fact quite small when measured as the cursor motion needed to select an item.

The use of the mouse in the experiment as the pointing device emphasizes the distance advantage as predicted by Fitts Law[Card et al 78, Card et al 83]. **[review fitt's law]** Since IM is nearly constant and S must vary depending on the application, the only variable we are able to deal with is D. By fixing D, pie menus should exhibit an improved performance over linear menus.

3.2. Experimental Design

The experiment is fairly straightforward and is constructed in a 2x3 randomized block design [Kirk68] (Table 1). Each cell is an element of the cross product of menu type and task. A typical pie task would be a compass because it seems best suited functionally for pie menus. List of elements, like OPEN/CLOSE and UP/DOWN, whose meanings are antonyms are also classified as pie tasks. Lists, like numbers, letters and ordinals, are best suited for linear menus and are thus classified as linear tasks. Groups of menu items that have no relation to each other fall in the unclassified category. Figure 7 shows an example of the groupings.

There are a total of 15 menus, a group of 5 for each task type (column). Subjects perform the experiment for all cells in the experiment matrix in random order in accordance with a randomized block design [Kirk68]. The subjects see the same menu four times, twice in each menu format. Each subject therefore sees a total of 60 menus.

The target item in each menu varies for the menu type and are balanced in their placement. This is, of course, particularly crucial for the linear menus whose mean seek time may be skewed if too many targets are found near the bottom of the list. Targets fall into 3 categories: top, middle, and bottom, splitting the menu into thirds. This terminology is derived from their placement in linear menus but is also used to describe placement in pie menus where the first menu item is at 90 (north) on the unit circle and items proceed clockwise. In a run of ten menus per cell, a typical sampling would be 5,6,3,8,5,2,7,4,1,4.

3.3. Subjects

Subjects were taken from the University of Maryland Psychology Department Subject Pool. 26 subjects performed the experiment in addition to 16 pilot subjects. All subjects were undergraduate students with little or no mouse experience. They were rewarded with 1 extra credit point for participating.

1. The experiment was primarily the work of Jack Callahan.

3.4. Materials

The experiment was run Sun Microsystems Workstation Model 3/160, running MITs X windows system as modifed for pie menus. (This modification is described earlier in this paper). The screen is a 19 bitmapped high resolution black-and-white display, approximately 80 spots/in. Cursor location is controlled by a three button optical mouse on a moveable mousepad made of a specially formatted reflective material. Lighting was by overhead flourescent tubes. Screen contrast, visibility or glare was not mentioned as a problem by any subject.

To invoke a menu on the Sun, the user positions the cursor with a pointing device, in this case a mouse, and presses a mouse button. The menu will appear on the screen at the cursor location as long as the button is held down. The whole process of selecting items from a pop-up menu, regardless of format style, can be characterized in three stages: invocation, browsing, and confirmation. To make a selection, the user invokes the menu by pressing a mouse button (invocation), continues to hold the mouse button down and moves to an item which is then highlighted (browsing) and releases the mouse button confirming the selection (confirmation).

A typical sequence of events for the user is shown in Figure 8. The computer posts the target name at the top of the screen, the user invokes the current menu, moves to the target item, and confirms the selection by releasing the mouse button. This sequence, called a task, was repeated 60 times by each subject. Each subject saw 6 sequences of 10 menus each. In each ten menu sequence, the menu type was the same, either pie or linear, and since there are only 5 menus per task type, each menu appears twice in the sequence. The 10 menus sequences correspond to the cells in the experiment table design. Each subject performed a sequence for all 6 cells and in random order. 60 data points are collected per subject. A total of 26 subjects performed the experiment for a total of 1560 data points.

For each task, the time from the first mouse button down to the correct target selection is the seek time for the item. If the user selected the wrong item, this time is included in this interval (Figure 10).The number of errors made as well as the sub-interval times when errors are made is recorded during the experiment by the system.

There were few difficulties with the execution of the experiment. The simplicity of the task and brevity of the experiment was intention so that any subject would have little problem. All subjects performed the test adequately and no person failed to finish the assignment.

3.5. Pilot Study

Sixteen subjects Results of the pilot study of 16 subjects showed that users were approximately 15% faster with the pie menus and that errors were less frequent with pie menus. Statistical significance was shown for a difference in item seek time. There did not, however, appear to be any significance attached to the task type. Subjects were split on their subjective preference of pie and linear menus. Some commented that they were able to visually isolate an item easier with linear menus and that it was hard to control the selection in pie menus because of the sensitivity of the pie menu selection mechanism. These subjects tended to be the most mouse naive of all whereas those who had heard of or seen a mouse/cursor controlled system but had not used one extensively tended to prefer pie menus. The most mouse naive users, while finding linear menus easier, tended to be better at pie menus and commented that with practice, they would probably be superior and in fact prefer the pie menus because of their speed and minimiz

Experience in the pilot caused only slight changes in the design: a better distribution of menu targets and doubled number of menu trials, though the number of menus total remained constant.

3.6. Results

A standard analysis of variance (ANOVA) was performed on the data. Table 2 shows the means per cell, per row, and per column. Table 3 displays the ANOVA results of statistical significance A Tukey analysis revealed that there is a statistical significance to the difference between overall menu type performance, task type performance, and the effect of target placement on performance. Pie tasks and linear tasks did not significantly differ from each other, but both organizations are an improvement over the unclassified menu tasks. Items placed in the third group, items 6,7,8, differed significantly from those in the other two groups. No other interaction was observed to be significant.

Disregarding pie menus for a moment, the shape of our results confirm earlier experiments on menus. The task type difference affirms the results of Card[Card 82] and McDonald[McDonald et al 83] that some organization is helpful. The significance of the menu targets in group 3 confirms Perlman's result that linear menus suffer performance problems with increased menu length at the extremities[Perlman 84]. But the exciting news for us was how well pies did.

Figure 11 displays the target location by item, not group, plotted against the mean seektime. The mean seektime across target location for pie menus is fairly constant. As expected for linear menus, the mean see time increases proportional to the distance of the target from the initial cursor location. Figure 12 plots the number of menus seen against the mean seek time for all menus that appeared at a particular point in the sequence of 60 menus. The early spike to the value of 6 seconds was probably caused by the misunderstanding of one subject in particular when an error occurred. Nevertheless, the region between pie and linear menu seek times seems fairly constant. It may be reasonable to assume that for this configuration of 8 items on a pie menu, linear menu performance will always be slightly slower. This can be attributed to the constant item distance advantage of pie menus. Figure 12 shows a slight learning effect for both types of menus, without obvious convergence in this number of trials.

Figure 13 displays the target location plotted against the number of errors. Pie and linear menus seem to suffer from a similar phenomenon - errors are made more on items in the central region of the menu display. These are the items with the most interaction with neighboring items[Card 82].

Some of the subjective results obtained in the pilot study repeated themselves in the actual experiment. Subjects were split on preferring one menu type over another but those who preferred linear menus had no string conviction in this direction and most admitted that with further practice they might prefer the pie menu structure. Those who preferred pie menus generally felt fairly confident in their assessment and this is reflected in the questionnaires.

3.7. Discussion

Should we go and program pie menus into our bitmapped window systems tomorrow and expect a 15% increase in productivity based on the fact that user can find the items slightly faster with pie menus. Well, maybe. There are several issues to examine first and several more experiments to run before one can issue a global recommendation of pie menus based on experimental evaluation.

First, this experiment only addresses fixed length menus, in particular, menus consisting of 8 items - no more, no less. Secondly, there remains the problem of increased screen real estate usage. In one trial a subject complained because the pie menu obscured his view of the target prompt message. Finally, the questionnaire showed that the subjects were almost evenly divided between pie and linear menus in subjective satisfaction. Many found it difficult to home in on a particular item because of the region of activation sensitivity of the pie menu design.

On the other hand, many users complained that the cursor migrated down towards the bottom of the screen and they were called on the reposition the mouse (i.e. pick it up and place it further forward on the mouse pad to allow more free movement space) quite often with the linear menus. Many preferred the limited movement and static placement nature of the pie menus. We only looked at computer-naive subjects, but it would be useful to examine the use of pie menus by experienced mouse and linear pop-up and pull-down menu users.

One continuing issue with pie menus is the limit on the number of items that can be placed in a circular format before the clutter or size of the menu window is impractical for implementation of an application as a pie menu. Perhaps, like the limiting factors active in linear menus concerning their lengths, pie menus reach a similar breaking point beyond which other menu styles would be more useful. Additional experiments could determine where, if anywhere, this point is.

4. Applications

Pie menus are not just another way of showing a menu. They offer an additional degree of information, the radius of selection, and open the door to some rather different interaction styles. In this section we speculate on some future uses of pie menus.

4.1. New Principles

Menus can be displayed as labeled icons around the menu name label in the center, which may have arrows, spears, wedges, or other embelishments around it. For entries that invoke a submenu, the icon for the entry could be an scaled down image of the menu that it invokes (possibly with less detail) so that you can know by looking at the first menu what the next stroke should be. Alternativly, just the name of the submenu and maybe an icon could be displayed, but when tracking the mouse, selections that are menus would be highlighted by drawing the mini-menu. This might look less cluttered. When a submenu is selected, its mini-menu would zoom to normal size, and the parent menu could either vanish or stay underneath.

For instance, the following menu that would work well in miniature form as a submenu:

```
              ______
              |    |
              |    |
              |    |
 ______       |    |
 |    |       |    |
 |    |       | 60 |
 |    |       |    |       ______
 |    |       |    |       |__6_|
 | 48 |       |    |
 |    |       |____|
 |    |
```

```
|____|       Window Height
 ______
|    |                      ______
|    |          ______     |    |
|    |         |    |      |_12_|
| 36 |         |    |
|    |         | 24 |
|____|         |____|
```

The user could choose from a set of 6 commonly used window sizes, with each being in a unique direction, or you could choose any intermediate value in between, by not constraining the choices to 6 sizes.

One thing that should be explored is the philosophy of convenient directions. What does it mean for a direction to be convenient? Easy to remember is convenient, but are there directions that are easier to specify with a mouse, touchpad, joystick, trackball, lightpen, footmouse, or whatever? Under what circumstances does a direction's convenience differ?

A selection's position relative to other selections is an important factor. A selections could be placed opposite of one that complements it, and 90 (or whatever) degrees away from ones that are orthogonal to it. What other ways are there to make selection directions easy to remember are there?

I think the following menu demonstrates this philosophy:

```
                    Any Height
       Any Size                      Next Size

 Any Width          Window Size         Common Width

     Previous Size                    Common Size
                   Common Height
```

A graphical representation could show a small version of each submenu, as well as the text label. The Common menus might have pointers showing the direction of each icon, while the Any menus would have the icons but not the arrows.

There could be another menu to choose a window size from a window's size ring (like an emacs kill ring: a circular array with one "current item" (displayed at the top of course)), and a way add and delete sizes. i.e. a size ring menu, where you choose a verb (select, replace, add, delete, next, previous, etc), and then (if appropriate) a noun (by selecting one of the sizes depicted in a menu). With the current item at the top, operations on the current item would always be the same sequence of strokes. This could be applied to rings of whatever you want.

Another similarly arranged set of menus could let you scale windows without changing the number of lines or columns. (i.e. change the font size.)

Visual feedback is not required in making choices, as with pull down menus. When familiar with choice directions, the user can traverse menus in quick, easily remembered strokes, without even looking. There is no need to slow the mouse down and "park" in in a small rectangle as with conventional pull down menus. If the user is familiar with the menu, then no visual attention at all is necessary. Thus you can be looking at something in one window while traversing menus in another. Rectilinear menus require that you move in the same direction every time you choose them (thus discarding theta), and depend on how far you move the mouse (depending on the radius instead). With rectilinear menus, there is no advantage to having fewer menu items, because you need just as precise control to choose each item.

4.2. New Applications

Choosing between n items, each item having a sector of 360/n degrees.

Setting the time of an analog clock. Use one button to set the minute hand, and the other to set the hour hand. Move in the direction you want it pointing.

Choosing from a color pallete. The menu is a circle of colors. Move in the direction of the color you want.

Answering yes or no questions. Up for yes, down for no.

Scrolling the screen. The radius could supply an argument for how far to scroll. Compare to scroll bars: You do not have to stop looking at what you're scrolling to aim the mouse at a scroll bar.

Choosing percentages. Right for 25%, down for 50%, etc.

Specifying a direction (and distance) to move in a game or

Incorporate audio feedback for the blind.

5. Conclusions

Pies are great. (etc.)

Acknowledgements

Ben Shneiderman provided advice on experimental design.

Xerox 84.
*Interpress Electronic Printing Standard*1984.

Sun 86.
SunView Programmer's Guide,.ie !"Revision A of 19 September 1986"" Revision A of 19 September 1986 1986.

Card et al 78.
S.K. Card, W.K. English, and B.J. Burr, ''Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a CRT,'' *Ergonomics* **21**(8) pp. 601-613. (1978).

Card 82.
Stuart K. Card, ''User perceptual mechanisms in the search of computer command menus,'' *Proc. Human Factors in Computer Systems,* pp. 190-196. (March 1982).

Card et al 83.
Stuart K. Card, Thomas P. Moran, and Allen Newell, *The Psychology of Human-Computer Interaction,* Lawrence Erlbaum, London (1983).

Dray et al 81.
S. M. Dray, W. G. Ogden, and R. E. Vestewig, ''Measuring performance with a menu-selection human computer interface,'' *Proc. Human Factors Society, 25th Annual Meeting,* pp. 746-748 (1981).

Gettys 86.
James Gettys, ''Problems Implementing Window Systems in UNIX,'' *Proceedings of the 1986 Winter Usenix Technical Conference,* pp. 89-103 (January, 1986).

Gosling 85.
James Gosling, ''SunDew – A Distributed and Extensible Window System,'' pp. 47-57 in *Methodology of Window Management*, ed. F.R.A. Hopgood,Springer-Verlag (1985). (SunDew has now gone commercial under the name NeWS.)

McDonald et al 83.
James E. McDonald, Jim D. Stone, and Linda S. Liebelt, ''Searching for items in menus: The effects of organization and type of target,'' *Proc. Human Factors Society, 27th Annual Meeting,* pp. 834-837. (1983).

Perlman 84.
Gary Perlman, ''Making the right choices with menus,'' *INTERACT '84, First IFIP International Conference on Human Computer Interaction*, pp. 291-295 North-Holland, (September 1984).

Shneiderman 86.
Ben Shneiderman, *Designing the User Interface,* Addison-Wesley Publishing Company (1986).

Systems 85.
Adobe Systems, *Postscript Language Reference Manual,* Addison-Wesley (1985).

Xerox 85.
*Mesa Programmers Manual and ViewPoint Programmers Manual*1985.

Figures (need double checking)

1. Rectilinear menu
2. flight simulation compas menu
3. nested pies
4. show various kinds of potential pies, with wasted space shaded.
5. show a pie menu
6. Jack’s figure 5.
7. Jack’s figure 6.
8. Jack’s figure 7
9. Jack’s figure 8
10. Jack’s figure 10.
11. Jack’s chart 1.

12 Jacks’ chart 2

13. Jack’s chart 3.

Tables

1. jack’s table 1.
2. jack’s table 2.
3. jacks’ table 3.