

Don't Click – Paint!

Using Toggle Maps to Manipulate Sets of Toggle Switches

Patrick Baudisch

Integrated Publication and Information Systems Institute (IPSI)
German National Research Center for Information Technology (GMD)
64293 Darmstadt, Germany
+49-6151-869-854
baudisch@gmd.de

ABSTRACT

A toggle map is a set of toggle switches that allows the manipulation of several switches with a single mouse drag interaction. Because toggle switches are functionally equivalent to black and white pixels interaction techniques from paint programs can be adopted for this task. A controlled experiment shows that toggle maps can speed up interfaces containing many toggle switches such as the interactive definition of user profiles. To maximize time savings toggle maps have to be laid-out according to co-occurrences between toggles. Efficiency gains resulting from the paint method open up new application areas such as segmented interval sliders. As an example an efficient timer dialog is presented.

KEYWORDS

toggle map, toggle switch, user interface, efficiency, image map, multiple select, painting

INTRODUCTION

Some applications, such as the interactive definition of user profiles, require a large number of Boolean variables to be set. To this purpose, toggle switches are often used (Figure 1). However, setting a large number of toggle switches can be time consuming. So how can toggle switches be handled in an efficient way?

Spreadsheet programs, desktop GUIs and paint programs provide means to select a number of items (cells, icons or pixels respectively) with a single mouse drag operation. These temporary selections define the range of the subsequent action, e.g. deletion.

Toggle switches can be handled the same way. First, a number of switches is selected with a drag operation, then an action such as *set all* can be performed. But toggle switches are so simple that it makes sense to combine selection and manipulation into a single interaction—let's *paint* toggle switches!

RELATED RESEARCH

Mary Valk, as well as Plaisant et al., have done interesting work on the visual design of toggle switches [20, 21, 23]. Concepts and interfaces about entering times and dates found in [3, 10, 13, 17] relate to the timer interface presented in this article. Research about mouse dragging can be found in [9], the involved basic research in the original work by Fitts [8]. Since sets of toggle switches have much in common with menus, research done on menu layout can be transferred to toggle maps. For an excellent overview over menu layout see [18, p. 261-280]. See Chin [5,6] on bottom-up and top-down clustering approaches to menu layout. Layout of two-dimensional menus according to item similarity or according to frequency of co-occurrence is discussed by McDonald et al. in [14, 16]. In [15] color-coding of menu items is analyzed.

PAINTING

Since a toggle switch is functionally equivalent to a black and white pixel, setting a whole dialog of toggle switches is similar to painting a black and white image. Therefore black and white paint tools found in painting programs such as *pencil*, *filled rectangle*, *polygon* or *line* are directly applicable to toggle maps. Since selecting and painting is syntactically equivalent, selection tools can be used as well. They can be turned into painting tools by automatically filling the selection. Figure 2 shows a variety of tools from a commercial painting program.

For most toggle map applications a single tool is sufficient, although different applications favor different tools. The pencil tool for example is most useful with a smaller number of toggle switches laid-out in an irregular pattern. The rectangle tool is especially effective if there is a large number of toggle switches.

The *painting mode* defines how to manipulate painted-over toggles. Extending toggle behavior into a paint mode leads to simply inverting all painted-over toggles. More useful is this slightly different mode: Only the first toggle switch is inverted, subsequent toggles are set to the new state of the first toggle switch. In this mode the rectangle tool paints rectangles of set toggles if painting starts on an unset toggle, otherwise it paints rectangles of unset



Figure 1: Dialog allowing users to input their personal TV channel profile

toggles. This allows users to over-paint fragmented regions with a single drag interaction, e.g. to set or reset a whole map¹. Since still at least the first toggle is inverted, users always get visual feedback, which simplifies trial-and-error learning. All interactions are triggered by a single mouse button, allowing toggle maps using this painting mode to be run on a single button mouse system or on touchscreen-based systems like palm-top computers.

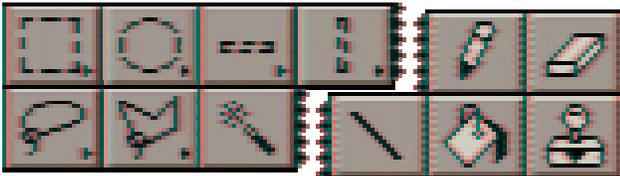


Figure 2: Tools for differently shaped selections (left) and tools for black and white painting (right). Painting tools are pen, rubber, line, paint bucket and copy stamp (All icon screenshots taken from Adobe Photoshop [1]. Reproduced with the kind permission of Adobe Systems Incorporated.)

REQUIREMENTS FOR APPLYING TOGGLE MAPS

Providing an additional painting function for sets of toggle switches does not do “any harm”, but for them to become real power tools the following requirements have to be met.

- 1.) Individual items should bear no or only short descriptions or names and should not require much

¹ Actually, it can take up to two interactions. If within the bounding box of all set switches at least one corner toggle is set then the map can be reset at once by starting painting at this corner toggle. Otherwise an additional click is required to set a corner toggle first.

time for decision making. Otherwise users prefer to release the mouse button and to click switches individually.

- 2.) It must be possible to manipulate several toggle switches per mouse drag. Otherwise there is no speed-up. This requires two things: First, a significant number of switches must be manipulated during individual sessions. Toggle maps are therefore not useful as menus, where usually only a single item per usage is picked. Second, a significant frequency of co-occurrence between toggles has to exist and to be reflected by the layout (clustering) [15]. Setting dialogs, e.g. for customizing printing options of a word processor, usually lack such co-occurrence relations and are therefore not a good application area for toggle maps.

LAYOUT

Toggle map layout has much in common with menu layout. Similar to spatial menus, toggle maps profit from two-dimensionality: Users can make use of visual recognition and spatial memory. The goal of toggle map layout is to maximize usage speed. In contrast to menus, toggle maps not only help in reducing the cognitive effort for finding items but also in reducing the manual effort for picking items.

There are two basic layout concepts: Layout by subjective similarity of items and layout by frequency of co-occurrence between items. These two approaches were compared by McDonald et al. [14] who conducted experiments on the layout of a simulated fast-food cash register. Layout generation techniques applied in this experiment can be used to layout toggle maps in a similar

way. In the first part of the experiment (layout generation) subjects rated pairs of food items on the basis of similarity (“How similar are these foods?”) and on the basis of co-occurrence (“How well do these two foods go together?”). The ratings were analyzed using non-metric multidimensional scaling and disjoint cluster analysis. Menu items were placed at the resulting coordinates with boxes placed around them to serve as selectable areas on a touch screen. In the second part of the experiment subjects had to enter orders of one to four items each into this simulated fast-food cash register. Usage speeds of the similarity layout and the frequency of co-occurrence layout were compared. McDonald et al. concluded “When tasks involve multiple-item selections and minimum task-execution time is important frequency of co-occurrence offers greater efficiency”.

The results by McDonald et al. refer to menus—items still had to be picked individually. Therefore these findings should be even more clearly reproduced in the context of toggle maps, where the grouping of items not only allows them to be found more easily but also selected more easily. Layout based on frequency of co-occurrence therefore seems to be the method of choice to reduce manual effort in using toggle maps. Kent L. Norman summarizes: “It is likely that in real-world applications menu layouts based on frequency of co-occurrence are, in general, superior to layouts based on similarity. ... Rating of item relatedness by the users may result in structures that are in some sense meaningful, but not appropriate to the task at hand.” [18, p.272]. For good results toggle map layouts can be constructed using bottom-up clustering [12]. Finding *optimum* layouts for toggle maps using the rectangle tool is, on the other hand a difficult undertaking. Here, frequently selected subsets should not only be grouped but arranged in rectangles. This requirement is not reflected by traditional clustering algorithms so that new algorithms still have to be found.

In some applications cognitive user effort can be significantly higher than the manual effort, e.g. if items bear complex descriptions or if the dialog is to be used mostly by first-time users. In such a situation layout should sacrifice part of its potential to the reduction of the cognitive user effort for finding and deciding. In such a situation layout based on subjective similarity can be useful. Similarity-based layouts are more common and appear more natural to users [15].

Another strategy to reduce cognitive effort is to emphasize grouping of items graphically. Graphical highlighting can help users to recognize related items as a group and thus to choose whole groups of items at once. In Figure 1 grouping is done by using columns, and blocks within columns. In the example shown in Figure 3 TV channels are grouped according to their geographical locations (for details on semantic space layout see [18, p. 269]). McDonald et al. conducted experiments on using color-coding to highlight groups of menu items [15]. They were

not able to show a positive effect of color-coding on recognition speed, but there are indications that these results were caused by a side-effect of their color-coding mechanism, i.e. the poorer readability of item names on the colored menu buttons. See the results section at the end of this article for more unexpected results about graphical highlighting of grouping.

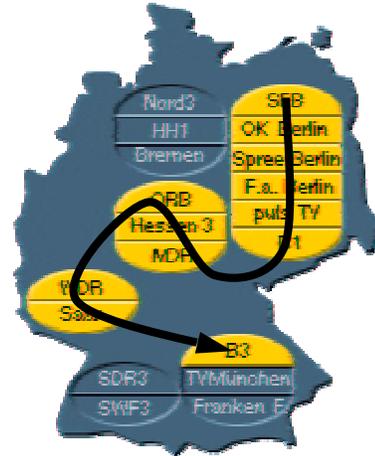


Figure 3: Toggle map offering German TV channels grouped according to their geographical locations. The black line shows one possible path to activate the highlighted switches with a single mouse interaction using a pen tool.

DON'T MOVE BOUNDARIES, PAINT AREAS!

Since large numbers of toggles are rendered manageable using the toggle maps concept, this opens another promising application area: When a continuous variable like time is segmented it can be represented as a set of toggle switches. The toggle switches in turn can be manipulated as a toggle map.



Figure 4: A toggle map timer interface. It allows users to input intervals for a whole week. Intervals of set switches are labeled as single intervals to reduce cluttering.

Figure 4 shows a toggle map timer interface. Programming the shown state (e.g. controlling house lighting during absence) is possible with only three rectangle paint operations. At the shown moment the time intervals for the weekend are enlarged by adding the hours starting at 9 o'clock. When the mouse button is released old and

new intervals unite automatically. Figure 5 shows how this works.

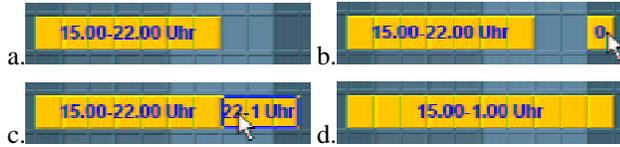


Figure 5: Scaling a toggle map interval only requires painting the addition. Touching or overlapping intervals unite automatically.

The main conceptual difference between toggle map interval sliders and handle-based interval sliders (Figure 6) is the same as between painting and drawing. Toggle map sliders work on segments while traditional interval sliders work on boundaries. Painting deals with surfaces while drawing deals with contours.

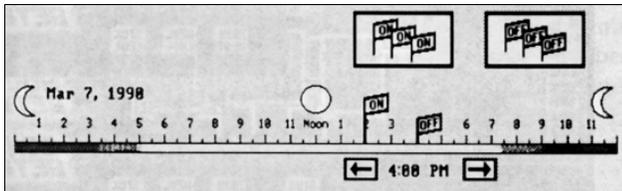


Figure 6: Slider element of the touchscreen VCR timer dialog by Pleasant et al. [19, 22, p.214]. On- and Off-Flags can be taken from the containers at the top left and be dragged onto the time scale.

Preliminary user tests showed that toggle map interval sliders are highly efficient, especially when several intervals can be manipulated at once. Sweeping across several days allows users to directly input quantified task descriptions like “For all days of the week... but on Fridays and Saturdays...”.

Unlike classical interval sliders toggle maps can do without any handles. Enlarging an interval only requires painting the addition. In a similar way intervals can be shortened or even divided. Furthermore toggle map interval sliders are especially easy to read, because a large share of the screen surface is used for visual feedback. And finally, like all toggle maps, they generate feedback on every possible user interaction, which simplifies trial and error learning.

The limited granularity of toggle maps may not be acceptable for some applications. To overcome this problem, again techniques from paint programs can be used: Since time intervals and bitmap images are both digitized continuums of limited resolution, zooming and scrolling can be transferred from pixel painting to toggle maps. Zooming-in magnifies pixels/toggles and thereby splits them into several finer segments.

FUZZY MAPS

Some applications require entering more information than can be expressed using toggle switches: User profiles can contain several degrees of liking and disliking, allergy tests result in skin reaction of different intensities, multi-

user calendars may work on probabilities. To reflect these requirements in a user interface, pixel painting again delivers the metaphors. As black and white pixels are similar to Boolean values, gray-scale pixels are similar to fuzzy values. Replacing toggle switches with elements that can represent multiple distinct values turns toggle maps into “fuzzy maps”. To manipulate fuzzy maps the described black and white painting tools can be complemented with the gray-scale painting tools shown in Figure 7. The airbrush, for example, works like a pencil but increases the value of fuzzy elements the longer they are painted over. One possible painting mode for airbrush painting is to incrementally paint with the left mouse button and to incrementally “erase” with the right mouse button. The advantage of tools like the air-brush is that they allow users to work on profiles as a whole instead of adjusting individual elements.



Figure 7: Painting tools for grayscale images (airbrush, gradient tool, smoothing, brush).

IMPLEMENTATION

Since today’s operating systems do not define any drag method on toggle switches, the described interaction techniques can be integrated without any conflicts with existing applications.

The prototypes described in this article were implemented as Java applets. Applet syntax was derived from the calling format of *html image maps* [7]. An Image map is a menu that is implemented as an image providing a set of *html links*. A link is followed when its associated region within the image is clicked. Since toggle maps deal with switches, not buttons, they have to display two different states for each item. Therefore toggle maps require two bitmap parameters instead of one. The first contains the appearance of the map where all switches are unset, the second contains the state where all switches are set. Similar to image maps, dimensions of the individual toggle switch regions are passed as parameters. The approach to render switches on fore- and background images guarantees full freedom of the graphical design. At the same time it allows to reuse all Java classes without modification—only images and toggle switch regions have to be changed.

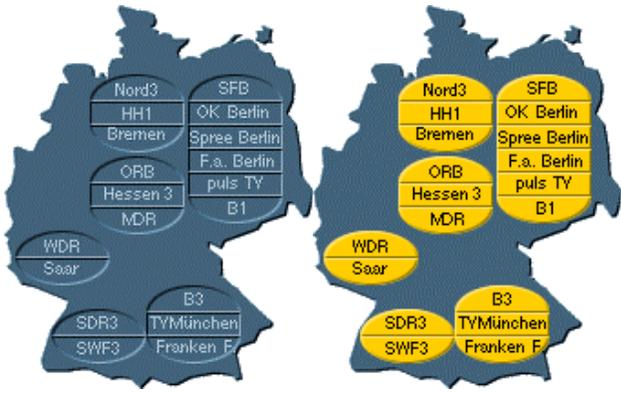


Figure 8: Implementation of a toggle map requires a background and a foreground image.

EXPERIMENT

To verify the validity of our concepts we conducted a controlled experiment on different interfaces allowing the selection of subsets of channels from a TV channel user profile dialog.

Subjects

Subjects were 74 persons from computer rooms at the Darmstadt University of Technology who volunteered in the experiment. Subject ages ranged from 15 to 55, 32% were female. All subjects had at least some previous computer experience. There was no significant influence of age, sex, education and computer experience on performance during the experiment.

To acquire the theoretical optimum performance we trained four expert users from our lab on performing all individual tasks on the different interface versions.

Apparatus and material

Experiments were run on Toshiba Tecra 740 CDT notebook computers with a 13.3 inch (33.8 cm) TFT color display and an external two-button mouse. The operating system was Microsoft Windows 95. Screen resolution was adjusted to 1024 x 768 pixels. Interfaces were programmed in Java and were run on a Microsoft Internet Explorer 4. Interfaces had a screen size of 24.5 cm x 12.5 cm. Individual toggle switches where 30 mm x 7 mm large.

Interfaces

Four different versions of the TV channel user profile dialogs were included in the experiment. Interfaces were similar in several aspects. They offered 61 channels receivable in Germany that were grouped in a table-like interface.

Interfaces differed in the following two aspects. The first two interfaces used switches in Windows-style as shown in Figure 1, while the last two interfaces used button-style switches as shown in Figure 9. Both types of switches were functionally equivalent and had the same sensitive regions. Interfaces one and three allowed manipulation of several switches at once using a rectangle paint tool,

while interfaces two and four only permitted clicking in individual switches.



Figure 9: TV channel user profile dialog with button-style toggle switches.

Procedures

Subjects were randomly assigned to one of eight groups. Groups were defined by the three variables paint/click, graphical toggle style and layout as shown in Table 1. The twelve items to select in the “good layout” condition were grouped in three larger blocks (= high frequency of co-occurrence between items). In the “poor layout” condition they were more distributed (three single items, three groups of two and one group of three).

	“Good layout”		“Poor layout”	
	Paint	Click	Paint	Click
Button-Style				
Windows-Style				

Table 1: The eight subject groups in the experiment

All subjects were given the same general instructions. To test true applicability with first-time users, subjects were not provided with *any* training and were not allowed to see the interfaces before the experiment. The four groups using paint interfaces were given the additional instruction “This dialog allows you to set or reset several switches at once by dragging the mouse with depressed button.”

Subjects had to select three sets of twelve TV channels each from their interface. Performance was measured as time to complete the task. Each selection process was recorded individually. To exclude times for reading task lists during the experiment, subjects had to learn channel lists by heart *before* using the interface. The three selection tasks consisted of different sub-tasks. The first selection task required a) learning how to operate the interface, b) finding the right toggle switches and c) setting the switches. In the second task, subjects were given a different set of channels². Since users already knew how

² Half of the subjects got the two involved sets in inverse order to avoid effects based on differences between sets.

to operate the interface the second time, this task consisted only of b) finding the right toggle switches and c) setting the switches. For the third selection task subjects were asked to select the same channels from the second task again. This time users already knew where to find the switches, thus the task was only c) setting the switches. Calculating the differences between these times allowed the times for the three subtasks a, b and c to be separated. After each selection users had to reset all toggle switches. Times for resetting were recorded as well.

After the experiment subjects had to fill in a questionnaire about their subjective satisfaction. Then they had an opportunity to try out the other three interface types (different toggle switch style and/or different possibility to paint) and selected which of the four interfaces they preferred. The overall session lasted about 20 minutes.

Expert users did not participate in the three phase conception. They had to complete all sixteen tasks (the eight groups x two different sets) in random order. They were given two trials on each interface to reduce the effect of outliers.

Hypothesis

1. Users provided with a painting method should perform better than those clicking switches individually. This should hold for first-time users as well as for experts.
2. Because more switches can be manipulated at once, differences in performance between paint group and click group should be higher in the “good layout” condition.
3. The button-style toggle switches should give better optical feedback and allow for a better use of spatial memory. Especially when entering the same set of items a second time, button-style groups should therefore perform better.
4. Rectangle painting allows users to reset the whole map so efficiently that a “reset all” button becomes dispensable.

RESULTS

Analysis of the timed tasks was done using t-tests. First-time users: During the first two selection tasks subjects in the paint groups performed slightly better, but none of the differences was significant due to high variations in times for finding items. Learning times, i.e. differences between the completion times for the first two tasks, did not show any significant differences.

The third selection task (selecting the same set of items again) contained only the manual effort for setting switches. Here the speed-up of painting showed clearly. Figure 10 shows the average task completion times. Differences in the “good layout” condition are significant at $p < 0.001$. In the poor layout condition differences are not significant due to two outliers in the paint group that re-

quired 12 seconds more than the next faster user in their group.

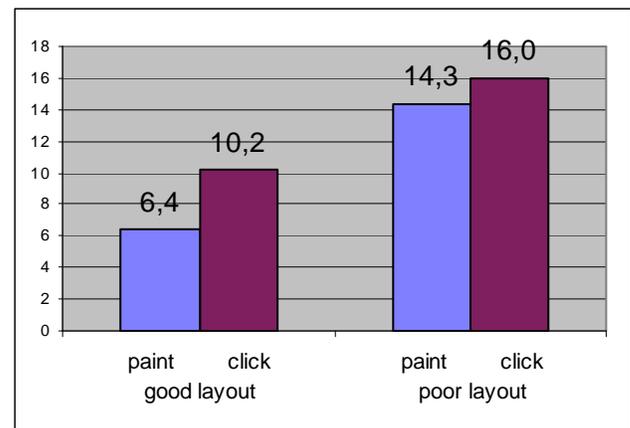


Figure 10: Average task completion times in seconds for first-time users

The results in the experts group showed the same trends as the first-timers. In the “good layout” condition painting users required an average of 1.97 seconds for task completion, which is more than twice as fast as the 4.1 seconds of the click-only group (significant at $p < 0.001$). In the “poor layout” condition paint users were, at 4.86 seconds, only slightly faster than the click-only group at 5.5 seconds (significant at $p < 0.01$).

In all first-time user and expert user groups, layout had a significant ($p < 0.01$) interaction with task completion time. The performance gain was always highest in the “good layout” groups, i.e. if more switches could be manipulated per mouse interaction.

When asked about their preference for any of the four interfaces styles, 88% of all subjects chose an interface providing a painting method. This ratio was independent of the interface type used during the experiment. The preference for interfaces with painting method was especially high in the “good layout” groups.

Subjects using the button-style toggles did not perform better than subjects using the Windows-style toggles. The only advantage of the button-style toggles was that 78% of all subjects subjectively preferred it. This style was described as being easier to read and as providing a better overview.

LESSONS LEARNED

First-time users needed an average of 6.6 seconds to reset maps, which was much longer than expected. This contradicts the fourth hypothesis with respect to first-time users. Expert users never needed more than a second to reset the whole map. This enormous difference between first-timers and expert users was caused by the fact that only two out of the 37 of first-timers provided with a painting method figured out, how to reset whole sets with a single paint interaction. Most first-time users reset the map column per column, others reset the map exactly in

the same way they had set it. When we investigated this phenomenon we determined:

1. Users avoided painting over unset toggle switches, because they expected them to become set when painting over them (“invert” paint mode). When discussing this effect after the experiment subjects rated the actually implemented paint mode as more useful, but stated that “invert” paint mode would be more common. Expectation of “invert” paint mode was especially common for computer experts.
2. The column layout of the channel selection applet kept users from painting across columns. While the columns helped grouping switches they kept users from understanding the two-dimensional nature of the interface.

The second finding was of even larger scope: Subjects seemed to derive a mental model of possible interactions from the first interaction they performed. Users who could apply painting for their first interaction (that was possible in the “good layout” conditions) were much more likely to make use of the painting function during the rest of the experiment. Users who started by clicking were more likely to keep on clicking even when painting could be usefully applied later. Some subjects even kept on clicking to reset maps.

Can we keep first-time users from expecting a different paint mode? Maybe it just takes some extra time for them to examine their expectations and to discover the additional functionality hidden in the actually implemented paint mode. Only an experiment containing a longer list of tasks can clarify that. On the other hand we surely have to reconsider the misleading layout, i.e. the columns in the presented example. To check the restricting influence of the column layout we added another reset-task at the end of the experiment. Before this task we gave subjects the hint “It is possible to paint across columns” which caused 27 out of 37 paint method users to discover the optimum reset strategy and let the average task completion time drop to one third (2.12 seconds). Obviously graphical highlighting of grouping should be used with care.

CONCLUSION AND FUTURE WORK

Toggle maps profit from defining a drag method on toggle switches—an interaction still undefined in today’s operating systems. Experimental results suggest that defining such a drag method as toggle switch “painting” leads to performance improvements and increased subjective satisfaction. Efficiency gains resulting from the paint method open up new application areas such as segmented interval sliders.

To apply toggle maps successfully, layout requires additional attention. Relations within the set, first of all frequency of co-occurrence, have to be determined and translated into layout. Layout enhancements such as graphical grouping have to be considered carefully. They

can help users in reducing cognitive costs but can as well mislead users to restrict their interactions to the highlighted structures.

Future work will include automated toggle map layout, applications on palm top computers and controlled experiments on toggle map interval sliders. More parameters like painting methods, layout types and graphical grouping will be formally evaluated. The experiment presented in this article can therefore only be a first step in understanding the involved variables.

All toggle map dialogs in this article were developed as part of the TV-program recommender project at GMD-IPSI [2]. They can be freely downloaded from <http://www.darmstadt.gmd.de/~baudisch/Publications/ToggleMaps>

ACKNOWLEDGMENTS

I thank Dieter Böcker, Ulrich Thiel and Barbara Lutes for their support. Special thanks to my students Robert Werner and Matthias Eilers for the implementation, Henning Meyer for conducting the user tests and Tom Stöltzing, the art director of TV TODAY, for his contribution to the graphical design of the applets shown in this article.

REFERENCES

1. Adobe Photoshop 4.0 User Manual, Adobe Systems Incorporated, San Jose, CA, 1996
2. Baudisch, P., Recommending TV programs on the Web: How far can we get at zero user effort, to appear in: *Proceedings of Recommender Systems 98, Workshop within AAAI 98*, Madison, Wisconsin, 1998
3. Beard, D., Humm, A., A visual calendar for scheduling small group meetings. technical report, Department of Computer Sciences, Technical report, Department of Computer Sciences, Univ. of North Carolina at Chapel Hill, 1989
4. Card, S., Moran, T. and Newell, A. (1983), *The Psychology of Human-Computer Interaction*, Hillsdale, NJ: Lawrence Erlbaum Associates
5. Chin, J.P., *Mental Models: Hierarchical Organization of computer menu functions, derived from top-down/bottom-up processing*. Master thesis, University of Maryland, College Park, MD, 1986
6. Chin, J.P., Top-down and bottom-up menu design, *Proceedings of the Second international Conference on Human/Computer Interaction*, p.144, Honolulu, HI, 1987
7. Durdning, B.M., Becker, C.A., and Gould, J.D., Data organization. *Human Factors*, 19, 1-14, 1977
8. Fitts, P.M., “The information capacity of the human motor system is controlled by the amplitude of movement”, *Journal of Experimental Psychology*, 7 0, 93-242, 1954

9. Gillian, D.J., Holden, K., Adam, S., Rudisill, M., Magee, L., How does fitts' law fit pointing and dragging?, *CHI'90 Conference Proceedings: Human Factors in Computer Systems* (p.227-234), New York, ACM, 1990
10. Gould, J., Boies, S., Meluson, A., Rasamny, M., and Vosburgh, A., Entry and selection methods for specifying dates, *Human Factors*, 1989, 31(2), 199-214
11. Image Maps: Online documentation at <http://theory.lcs.mit.edu/~boyko/imagemap.html>
12. Lindsey, P.H., and Norman, D.A., *Human information processing*, New York: Academic Press, 1977
13. Mac Carthy J.R., *A Matter of Time - the Story of the Watch*, Harper & Brothers, 1947
14. McDonald, J.E., Dayton, T., McDonald, D.R., *Adapting menu layout to tasks* (MCCS-86-78). Las Cruces, NM: Memoranda in Computer and Cognitive Science, Computer Research Laboratory, New Mexico, State University, 1986
15. McDonald, J.E., Molander, M.E., and Noel, R.W. Color coding categories in menus. *CHI'88 Conference Proceedings: Human Factors in Computer Systems* (p.101-106), New York, ACM, 1988
16. McDonald, J.E., Stone, J.D., Liebolt, L.S., (1982) Evaluating a method for structuring the user-system interface. *Proceedings of the Human Factors Society—26th Meeting* (p. 551-555). Santa Monica, CA: Human Factors Society.
17. Norman, D.A., *The Psychology of Everyday Things*, Basic Books, New York, 1988
18. Norman, K. L., *The Psychology of Menu Selection: Designing cognitive control at the human/computer interface*, Norwood, New Jersey, Ablex, 1991
19. Plaisant, C., Shneiderman, B., Battaglia, J., Scheduling home control devices: A case study of the transition from the research project to a product, Technical report 90-10, University of Maryland, 1995, available online at <http://www.cs.umd.edu/hcil>
20. Plaisant, C., Wallace, D., Touchscreen Toggle Design, *CHI'92 Conference Proceedings: Human Factors in Computer Systems* (p.667-668), New York, ACM, 1992
21. Plaisant, C., Wallace, D., Touchscreen Toggle Switches: Push or slide? Design issues and usability study. University of Maryland technical report CAR TR-521, CS-TR-2557, 1990
22. Shneiderman, B., *Designing the User Interface: Strategies for effective human-computer interaction*, Third edition, Reading MA: Addison-Wesley, 1998
23. Valk, A.M., An experiment to study touchscreen "button" design, *Proceedings of the Human Factors Society*, 29, 127-131, 1985