21

Selection

One of the fundamental behaviors in graphical user interfaces is the selection of objects or actions for interaction. This selection can be organized in many ways including pull-down menus, pop-up menus, palettes, toolbars, icons and semantic objects of various kinds. In this chapter we will look at the basic theory behind selection and the variety of techniques that have been developed to improve selection performance.

Most of this discussion will concentrate on menus because they are the most common form of selection. We will start with a discussion of the theory of selection and how it applies to the more common menu mechanisms. We will then discuss a variety of improvements that have been proposed for organizing menus. We will then examine marking menus which are a pen-based alternative to the standard menu mechanism. This will be followed by stroke-based selection where the standard click-and-drag model of selection is replaced by pen strokes to combine several selections. Lastly we will look at distortions of the mouse/cursor relationship to improve selection times and error rates.

Selection theory

Fitt's Law

The foundation of selection theory is Fitts' law first described by Paul Fitts in 1954[1]. The law can be simply stated as:

$$T = a + b \bullet \log\left(\frac{A}{W} + 1\right)$$

T is the time required to perform a selection. *A* is the distance from where the user starts to the target to be selected. *W* is the width of the target. The coefficients *a* and *b* are empirically determined for some particular combination of input device and display. Fitts' law effectively partitions selection theory into two parts that can be considered separately. For a given selection task the terms *A* and *W* are fixed. We can then try a variety of input devices and derive the coefficients *a* and *b*. By comparing these coefficients among various input devices we can make a choice for a given set of tasks. This scenario, however, is not common.

In most cases the input devices are given (such as screen and mouse) and we must create techniques and designs that will be the most efficient. For this case we ignore *a* and *b*, which characterize the devices, and focus on *log(A/W+1)*, which characterizes the selection problem. This is referred to as the *index of difficulty*. The greater this number, the more time the selection will require. Looking at this formula we see that we can reduce selection time either by bringing the target closer to where the user starts (*A*) or by increasing the size of the target (*W*). This is the basis for most selection theory.

Fitts' law makes several assumptions about the selection task. The first is that any path from the start position to the target is acceptable and the user is free to optimize that path. The second assumption is that the time to visually locate the target is zero. There are many situations where neither of these assumptions is valid.

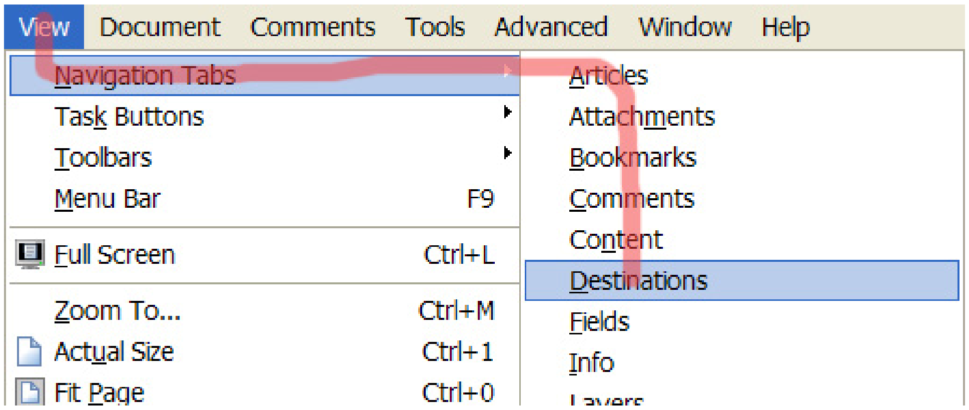


Figure 21.1 – Selection path for a cascaded menu

Steering Law

Accot and Zhai[2] created the *steering law* to address situations where the path that the user must take in reaching the target is important. Figure 21.1 shows a normal cascaded menu. Note that to select "Destinations" the user must select "View" and navigate straight down to "Navigation Tabs". If the user tries to cut this corner the "Document" menu will be selected and the user must start over. After reaching "Navigation Tabs" the user must turn right and stay within the menu item while navigating to the right. Any deviation will cause a different menu to be selected. Having navigated to the right, the user must turn down and reach “Destinations”. Accot and Zhai observed that staying within the “tunnel” slows users down. Their analogy is that when driving on a narrow road, one must slow down so as to not inadvertently steer off the road. With wider roads one can drive faster because there is a greater margin of error.

Their abstraction of the problem was a “tunnel” that is *A* long and *W* wide which the user must navigate without bumping into the edges of the tunnel. This is clear from the movement along the “Navigation Tabs” menu but also occurs when navigating down from “View” without cutting the corner. The paper describes the steering law as an integral along the path. For simple tunnels like that shown in figure 21.1, the index of difficulty becomes *A/W* rather than *log(A/W)*. The essence of the result is that narrow paths will increase selection times. The tunnel abstraction was further extended by Pastel’s[3] analysis of corners in the path. He showed that corners slowed down selection and increasing the path width at the corners would improve selection times.

Perception

The previous theories address the problem of controlling an input device from a starting location to an accurate target selection. They ignore the visual search problem where the user must locate the desired target before navigating to it. Card[4] addressed this question through a series of experiments on menu organization. In his experiments the structure of the menus were fixed, thus neutralizing the Fitts’ law and steering law effects. What changed was the order in which items appeared in the menu. The three conditions were random order, alphabetical order and grouping by function. For first-time users of a particular menu, alphabetical was fastest (0.81 sec), followed by functional order (1.28 sec) with random order being the slowest (3.23 sec). The hypothesis is that with a known structure, fewer saccades (eye movements) are required to locate the desired item. The structure optimizes the visual search. However, the same experiments showed that after 800 menu selections there was no difference in performance times among the three conditions. Once users learned the structure (even if random) they no longer needed visual search and those differences disappeared.

Standard menus

Pull-down menus have been a staple of graphical user interfaces since the Macintosh was introduced. There are two basic strategies. The Macintosh placed application menus at the top of the screen. No matter where the application window was placed on the screen the menu was always across the top. Windows places its application menus at the top of each window. Application of Fitts' law would indicate that selection time would be lower for the Windows strategy because the menu would generally be nearer to the start point where the user is working. Thus Windows would have a much smaller average *A* than the Macintosh.

Early experiments showed that this was not so. Users were faster on the Macintosh. The reason is that menus at the top of the screen are backed up by a border beyond which the mouse will not go. Users could rapidly shove the mouse to the top of the screen without having to slow down to hit the menu. In essence the top-of-screen menu strategy has a very large *W* in mouse control space. This is an important conceptual point. The Macintosh menus are of comparable size to the Windows menus. In terms of screen space their *W* is virtually identical. However, Fitts' law and the steering law are about the control system in the hand, not about screen space. Hitting a top-of-screen menu offers a much larger range of mouse positions (control space) than the menu-per-window strategy. These comparison experiments were done on relatively small screens. Modern workstations with multiple high resolution screens may yield different results due to much larger differences in *A*.

Another application of Fitts' law is the use of popup menus. In Windows these are activated using the right mouse button. Popup menus offer two advantages. The first is that the menu is adjacent to the current mouse position and therefore *A* is much smaller. The second is that the menu can be tailored to the type object immediately under the mouse location. This greatly simplifies the user's task of finding an appropriate action from the menu. One of the difficulties with popup menus is that they obscure the objects underneath. There has been some use of semi-transparent menus that allow the objects beneath to show through.

Improving menus

A variety of designs have been proposed for improving selection times on menus. Most of these are driven by Fitts' law and the steering law. An early innovation was the *pie menu*, where the items for selection are arranged in a pie shape around the mouse start position[5]. Figure 21.2 shows a standard linear popup menu on the left with a corresponding pie menu on the right. The advantage of the pie menu over the linear menu is that the average distance to each selection is lower for pie menus than for linear menus. The experiments showed that selection time for pie menus is 15% less than for linear menus with half the error rate. Most pie menus are arranged closer to the start point than shown in figure 21.2's early prototype.

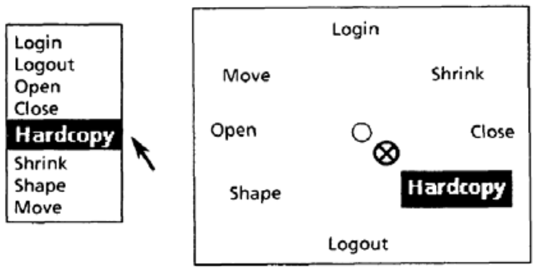


Figure 21.2 – Linear and pie menus[5]

There are several drawbacks for pie menus. The first is that they generally consume more screen space than linear menus. They are also limited to about 16 items where linear menus are not. Pie menus can be cascaded, but subsequent levels of the menu must appear over the top of previous levels creating screen clutter. There is also the problem of the screen edges. When the start point is very close to the bottom of the screen a linear menu can appear above the start point. A pie menu, however, derives its advantage from appearing around the start point, which is impossible at the very edges or corners of the screen. Some pie menus adapt by using a larger semi-circle to arrange items that otherwise would be invisible.

Walker, Smelcer and Nilsen[6] proposed an improvement to menus by increasing the size of menu items that are farther away. The idea is to give menu items that are farther away a size that will compensate for the distance. Figure 21.3 shows how such a menu might be laid out. The problem with this strategy is that the size of the menu is an exponential function of the number of menu items. As each menu item gets larger to compensate for the distance, subsequent items are pushed out even farther from the start position.

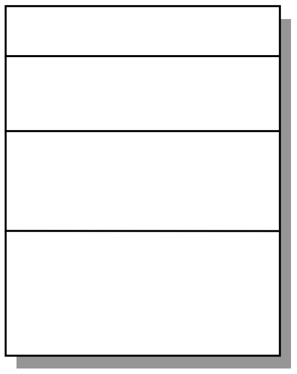


Figure 21.3 – Variable sized menu items

Subsequent improvements to menu selection focused on the steering law problems of cascading menus shown in figure 21.1. Kobayashi and Igarashi[7] looked at the problem of users cutting corners as they attempt to navigate cascaded menus. Figure 21.4 shows an example of the steering law problem that they were addressing.

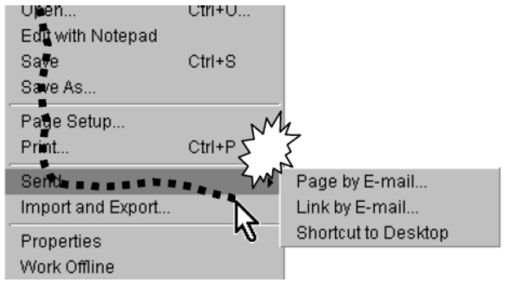


Figure 21.4 – Corner-cutting in cascaded menus[7]

Their solution was to consider the mouse movement direction rather than just the position when activating menus. Moving the mouse up or down would move the menu highlight to match the current mouse position. If the mouse is on sub-menu node then moving it to the right would immediately open the sub-menu near the current mouse position as shown in figure 21.5. This solves two problems. The extended steering law problem in going all the way to the right is eliminated and the sub-menu is very near the mouse reducing the Fitts' law distance to the next select. If a sub-menu is opened inadvertently, moving the mouse to the left will close it. Their experiments showed that selection time was reduced by 12% and the total path length traversed by the mouse was reduced by 31%. However, some users were confused by the non-standard behavior of the menu and by the fact that sub-menus would obscure their parent menus.

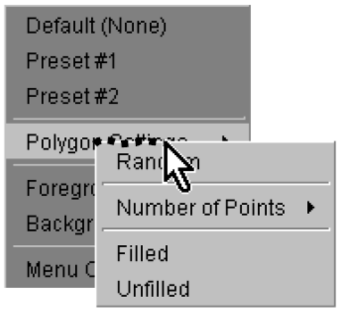


Figure 21.5 – Early opening of sub-menus[7]

Ahlström[8] proposed that the steering problem be alleviated using a "force field strategy to guide the mouse towards appropriate selection, rather than interpret gestures. This strategy subtly modifies the menu's control space size without modifying its visual presentation. The force fields are shown in figure 21.6. When the mouse is inside of a parent menu item, the force field guides the mouse toward the point that will open the submenu. When the mouse is inside of a non-parent item the mouse is guided towards the center of the item.

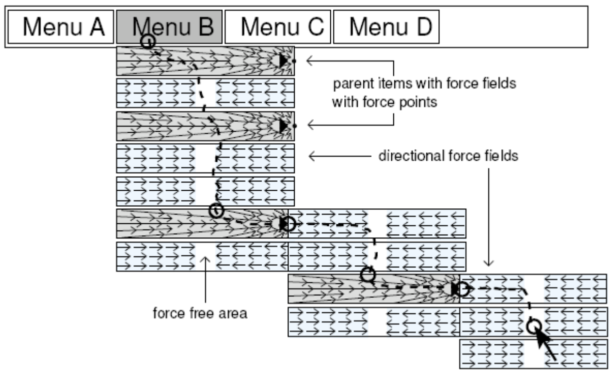


Figure 21.6 – Force-field menus[8]

This force field guidance is possible because the connection between mouse movement and cursor position is indirect and can be modified. For each type of menu item there is a force point **f** towards which the field is guiding the mouse.

For parent item it is the center of the right edge and for non-parent items it is the center of the item. Many systems allow the software to “warp” the mouse by setting the cursor to an arbitrary position on the screen. Ahlström’s force field formula is:

$$\mathbf{n} = \mathbf{a} + s \cdot \|\mathbf{a} - \mathbf{p}\| \cdot \frac{\mathbf{f} - \mathbf{a}}{\|\mathbf{f} - \mathbf{a}\|}$$

n = new position to where the cursor should be warped.

a = active position of the mouse from the current mouse-move event

p = previous mouse position before the current mouse-move event

f = the desired force point

s = a strength parameter for the force field that can be adjusted

This technique was tested on several input devices and with novice and expert users. Selection times improved from 11% for infrequent track point users up to 30% for novice touch pad users. The average of all mouse users was a 17% improvement.

These improvements assume that the user is dragging continuously to select a menu. Many users click on parent menus rather than hold down the mouse while navigating to their choices. Ahlström et. al.[9] modified this click behavior. When a user clicks on a parent item, the sub-menu is opened and the cursor is “jumped” to the top of the submenu eliminating the steering task entirely. If after pressing the button the user moves the mouse more than 5 pixels the jump is cancelled and the submenu is closed. This allows the user to cancel the jump in a natural way. This technique showed an increase in selection errors over standard menus. In selection times the results were mixed between force menus and jumping menus with no clear advantage.

Marking Menus

Popup menus tend to obscure the data beneath them. Gestures (chapter 20) resolve this problem, but tend to be difficult for people to learn and remember. With gesture interaction there is a gulf of execution problem because there is no visible menu to help guide the user to the correct gesture. To resolve these issues Kurtenbach and Buxton[10] invented *marking menus*. Figure 21.7 shows the two different modes of operation for marking menus. If the user holds down the mouse button or pen tip switch, for 1/3 of a second, a pie menu appears. By moving the stylus into the desired region the selection is made. The alternative mode is just to make the same mark without waiting for the menu to appear. Mode (b) behaves like a gesture though it is actually a menu selection. Mode (a) trains the user in the appropriate stroke for each menu selection.

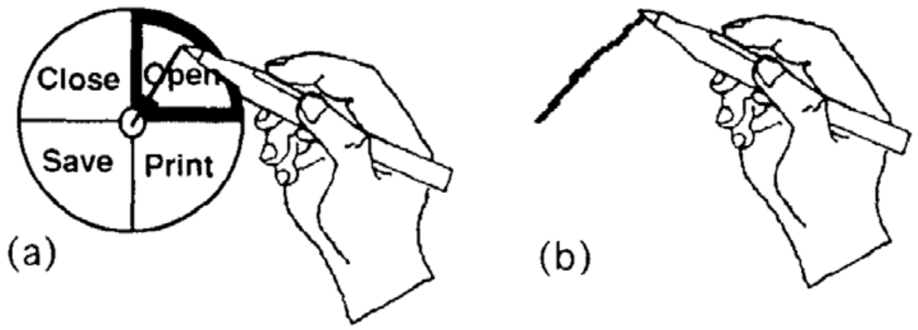


Figure 21.7 – Marking Menus[11]

The gesture recognition algorithm for simple marking menus uses only the stroke angle or the differences between the end point and the start point for features. Recognition is trivial.

Initial experiments led to several design principles for using marking menus. The first is that there should be an even number of choices up to a maximum of 12. The second is that the same command should appear in the same position among various marking menus so that the user learns a specific stroke for that command. Experiments also showed that marking menus should be used where the commands are frequent so that users learn to take advantage of the marks. When users were away from an application for a while, they tend to forget the marks. This effect disappears with experience.

An early advance was the introduction of hierarchic marking menus[10]. In this variant a menu selection could be itself the parent of another menu. The result was that selection becomes a continuous stroke with corners separating the segments for each menu. This was shown to be 3.5 times faster than normal menus and performed better with a pen than a mouse. If each menu has 8 choices, then a depth of greater than 2 became inaccurate with too many errors (64 total choices). If each menu had 4 choices then users could go to 4 levels without excessive errors (256 choices). These experiments also showed that gesture strokes on the primary X and Y axes reduced errors.

Tapia and Kurtenbach published several design refinements to make marking menus more effective[12]. These are based on their experience using marking menus in the Alias StudioPaint system. Their first proposal was to discard the pie and use only the menu text. This text was placed in similar positions but without regard for the location of the pie as shown in figure 21.8. Removing the pie shows more of the underlying task and frees the text messages to be longer. Boxes behind the text (all of the same size) provide the visual contrast necessary to see the text.

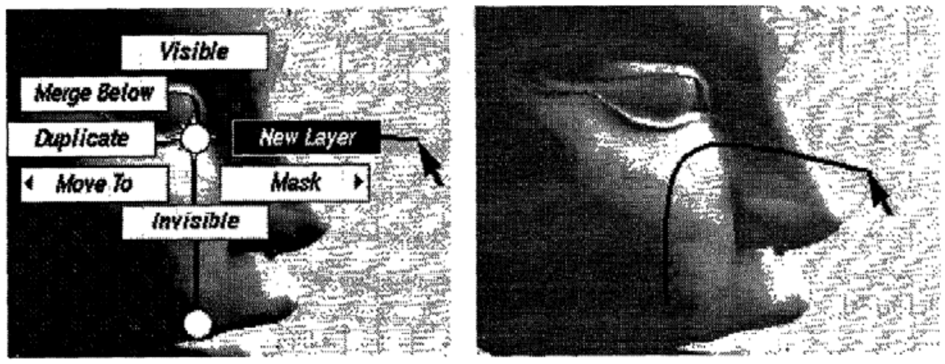


Figure 21.8 – Removing the pie from marking menus[12]

Their experience indicates that 8 menu items is best and any parent items should not be shown in hierarchic menus. Instead the parent center dot is shown along with a line indicating the stroke that produced the current menu. Moving the stylus back to a parent dot will bring up the parent menu again in case of inaccurate selection.

One of the key causes for problems in deep hierarchies is the steering law effect of drawing the correct stroke without cutting corners. Zhao and Balakrishnan addressed this issue by breaking up the strokes[13]. Rather than carefully turning the corner with the pen down, the user picks up the pen and makes a new stroke in a new direction. This also resolves the ambiguity of two successive menu selection in the same direction. By repeatedly making the marks in the same location the total screen space requirement is reduced. Selection accuracy (percent correct selections) was significantly higher for separate marks than for long compound strokes. Unlike single stroke techniques the separate strokes showed very little decline in accuracy as the menu hierarchy got deeper. In terms of total selection time, the separate strokes performed slightly better. This is consistent with Pastel's work on steering through corners[3].

Expanding on the individual stroke concept, Zhao, Agrawala, and Hinckley[14] introduced *zone* and *polygon* menus. The innovation is to allow the starting position of the stroke as well as its direction to serve in selecting the menu choice. Figure 21.9 shows a zone menu. The user first taps the pen and waits for the menu to appear. The tap establishes the menu center. Four menus in each of four zones appears. Each menu has 4 choices for a total of 16. The start position of the stroke (relative to the center point) selects the zone and the direction makes a choice within that zone. On the left is the learned gesture for the same selection. This technique expands the breadth of the menu to 16 making it faster.

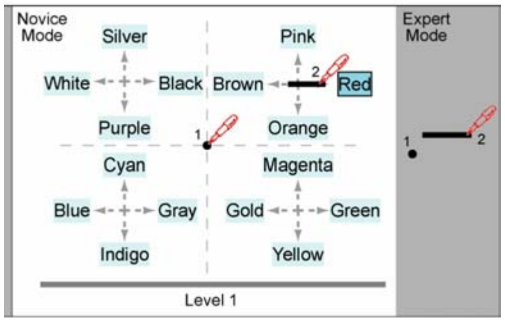


Figure 21.9 – Zone menus[14]

The polygon menu, shown in figure 21.10, also uses position and direction. The center tap defines an 8-sided polygon. The user is expected to stroke near one of the edges of the polygon. For each stroke direction (out of 8) there are two possible menu selections. By looking at where the stroke is located relative to the center tap the desired selection can be identified. Selecting the most similar polygon edge plus stroke direction yields 16 choices.

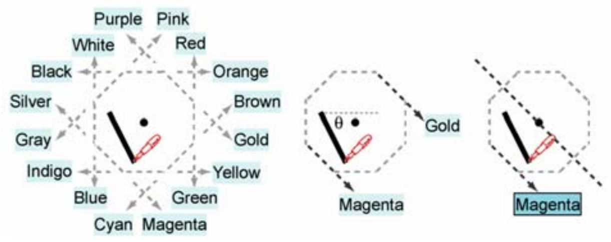


Figure 21.10 – Polygon menu[14]

Experiments showed that zone and polygon menus produced similar accuracy and similar speed to previous techniques, but they have twice to 4 times the breadth of choices at a given level of menu. This results in a sharp decrease in selection time for large numbers of choices.

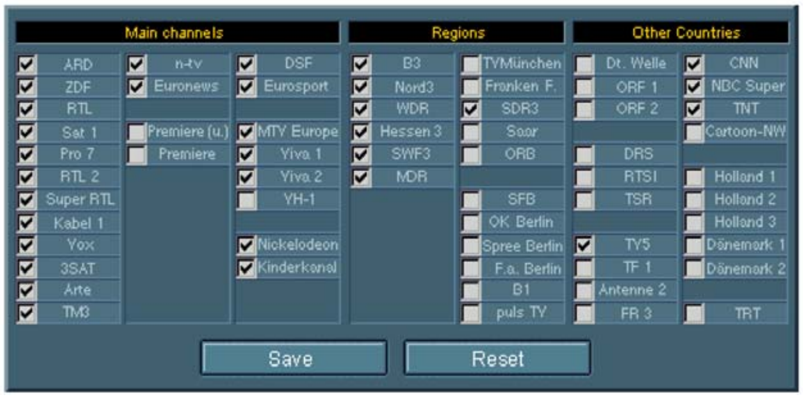
Stroke Selection

As pen devices have become more common most software has treated the pen as if it were a mouse. This helps the software architecture, but is not necessarily the best interactive solution. The advantage of this strategy is that with a pen, all existing software still works in a familiar way.

There are several difficulties with a pen. The double-click, without changing position, is hard to do correctly with a pen. The pen tip switch functions well as the primary button but the switch on the side of the pen is an awkward substitute for the secondary or menu button. Unlike a mouse, the hand holding the pen can obscure important data and in the case of pop-down menus, it obscures the menu itself.

In addressing the unique characteristics of pens it is observed that pens are more fluid stroking devices rather than clicking devices. There have been several proposals for adapting selection tasks to use strokes rather than clicks and drags. The core idea is to perform a selection by crossing a region rather than by clicking on a spot.

One of the first proposals was toggle maps[15]. This is useful when there are many binary selections possible and one wants to select large groups of them rapidly. There are many problems that can be cast as a presentation of a large number of binary choices. Figure 21.11 shows a selection of television channels from a German television service. The user's task is the select those channels to be shown and those to be blocked. The channels are grouped by region and by subject. When the primary button is first pressed, the checkbox under the button is toggled. As the stroke proceeds, all selections under the stroke are changed to the original toggle value, or left alone if they already have that value. If, for example, the user speaks Dutch the three channels from Holland can be selected in a single stroke rather than by three separate clicks.



Main channels

✓ ARD
✓ ZDF
✓ RTL
✓ Sat 1
✓ Pro 7
✓ RTL 2
✓ Super RTL
✓ Kabel 1
✓ Vox
✓ 3SAT
✓ Arte
✓ TM3
✓ n-tv
✓ Euronews
☐ Premiere (u.)
☐ Premiere
✓ DSF
✓ Eurosport
✓ MTV Europe
✓ Viva 1
✓ Viva 2
☐ VH-1
✓ Nickelodeon
✓ Kinderkanal

Regions

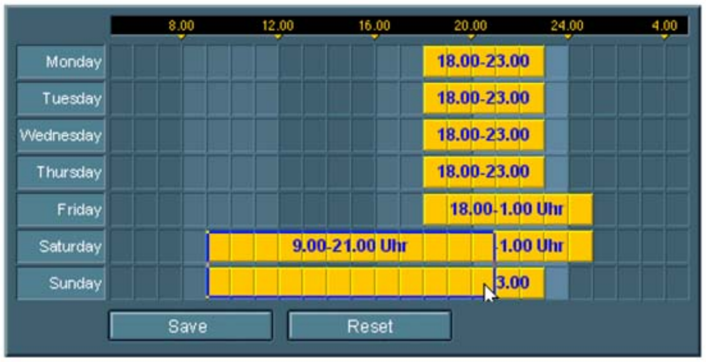
✓ B3
✓ Nord3
✓ WDR
✓ Hessen 3
✓ SWF3
✓ MDR
☐ TVMünchen
☐ Franken F.
✓ SDR3
☐ Saar
☐ ORB
☐ SFB
☐ OK Berlin
☐ Spree Berlin
☐ F.a. Berlin
☐ B1
☐ puls TV

Other Countries

☐ Dt. Welle
☐ ORF 1
☐ ORF 2
DRS
☐ RTSI
☐ TSR
✓ TV5
☐ TF 1
☐ Antenne 2
☐ FR 3
✓ CNN
✓ NBC Super
✓ TNT
☐ Cartoon-NW
☐ Holland 1
☐ Holland 2
☐ Holland 3
☐ Dänemark 1
☐ Dänemark 2
☐ TRT

Figure 21.11 – Toggle Maps with check boxes[15]

Figure 21.12 shows a table of times laid out with days of the week. The task is to select times when lights should be turned on or automatic door locks released. Each day/time period is a boolean value. Selecting "every day at 16:00" is a single stroke down the column. Turning off everything on Sunday is also a single stroke. Performing these tasks by clicking would be painful even with a mouse.



Monday 18.00-23.00
Tuesday 18.00-23.00
Wednesday 18.00-23.00
Thursday 18.00-23.00
Friday 18.00-1.00 Uhr
Saturday 9.00-21.00 Uhr 1.00 Uhr
Sunday 3.00

Figure 21.12 – Toggle times[15]

A second stroke selection system is CrossY[16]. Some of the key insights of CrossY are that in many cases the user wants to select several properties at once for a particular task and stroking through a region is easier for a pen than clicking on a spot. Figure 21.13 shows an example of such selections. The pen, its thickness and its color are all selected in a single stroke rather than by three separate clicks. Note that above and below the centers of each selection there are small tick marks. Selection is defined as a stroke that passes between these marks. Only the regions between the marks are selection active. This eliminates problems from the steering law because the user need not steer down the whole box before making a turn. Only the cross regions must be navigated.

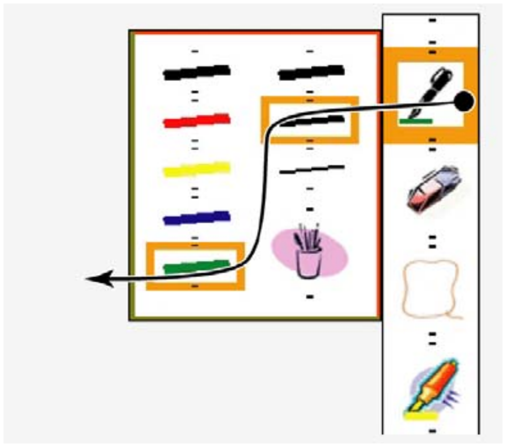


Figure 21.13 – CrossY selection[16]

Figure 21.14 shows a different crossing style. The brush properties have several selections that are not exclusive. As with toggle maps, selection will toggle the value. Note that instead of the tick marks, a diagonal line is presented. This line selection allows crossing with either vertical or horizontal strokes for more user flexibility.

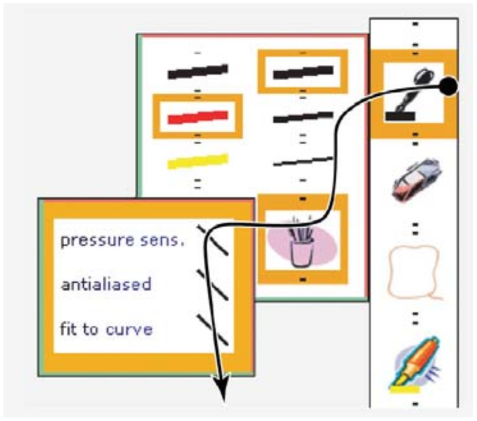


Figure 21.14 – CrossY toggle[16]

Figure 21.15 shows two adjacent sliders, one for hue and one for value. In a single stroke the user can select a desired color.

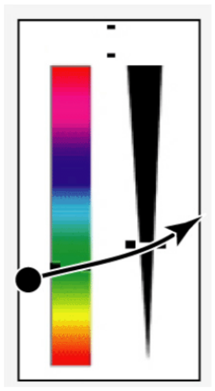


Figure 21.15 – CrossY color selection[16]

Area/Cursor modification for icon selection

Our last topic in selection has to do with distorting the mouse/display relationship. The first distortion is to use a cursor that is an area or region rather than a point and the second is to manipulate the relationship between hand movement and cursor movements.

One of the first problems is the rapid selection of very small targets. For example, trying to hit a point on the screen with a small cursor is very difficult. Kabbash and Buxton introduced *area cursors*. Their idea was to increase the size of the cursor rather than increase the size of the target. Their experiments showed that Fitts' law holds when the width of the cursor rather than the width of the target is used for such tasks[17]. The problem with area cursors is when there are many targets. The area cursor will then have a problem disambiguating among the targets. The only way to disambiguate is to keep the targets farther apart than the size of the area cursor. Usually we have small targets because we want to pack many of them on the screen. This defeats the advantage offered by area cursors.

Keyson[18] introduced the idea of changing the mouse gain. A mouse operates by sending incremental movement signals to the cursor controller whenever a hand movement is detected. By adding up these mouse movements the controller derives a cursor position. To give users better control over their cursor most systems use a *mouse gain* control. A common value is 3 to 1 or three mouse move ticks for each pixel movement. If you increase the gain, the mouse must be moved farther to achieve the same amount of cursor movement. This slowing down of the cursor tends to increase control by increasing the size of physical hand movement space. Whenever the cursor approaches a target, the mouse gain is increased. The result is that the size of the target remained the same size on the screen, but became much larger in hand movement space. Since Fitts' law is a property of hand movement rather than visual search, this has the effect of increasing *W* without increasing screen space.

Worden, et. al.[19] noted that when there are many targets on the screen, the cursor will always be slowing down and thus tend to increase the hand distance in reaching a particular target if there are many intervening targets. Thus the *A* value in Fitts' law is also increased, negating the advantage. Their insight was that when a user begins to move towards a target they move very quickly. As they approach the target they slow down to hit it accurately.

Worden, et. al. used this insight to modify Keyson's increased gain. Initially their mouse gain is set at normal. As the cursor is moving, they measure its velocity. When the velocity drops below 30% of the peak velocity, they raise the gain on the targets. This increase in gain creates *sticky icons* that are only sticky when the user slows down to land on a target. They reported that sticky icons show a 50% improvement in speed for elderly users with poor motor control and a 40% improvement for young adults.

These ideas were generalized by Blanch, Guiard and Beaudouin-Lafon in their concept of *semantic pointing*[20]. They propose the notion of a measure of *semantic importance* associated with objects on the screen. Increased semantic importance increases the mouse gain. This can be exploited in the visual redesign of some widgets. Figure 21.16-a shows a traditional scroll bar. Figure 21.16-b shows a redesigned scroll bar that takes less screen space, but is easier to see. By increasing the semantic importance of the slider and the end buttons, the scroll bar feels like figure 21.16-c in motor space.

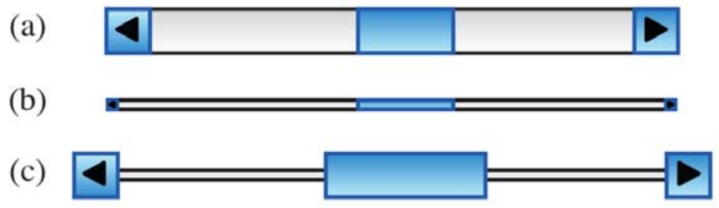


Figure 21.16 – Semantic pointing[20]

Grossman and Balakrishnan's Bubble Cursor[21] addresses the selection problem differently. They stayed with the notion of area cursors, but made them round and made cursor size adaptable to the current selection problem. The cursor changes size so that only one target is inside of the cursor. The idea is that the user need only navigate as close to the target as needed irrespective of the size of the target or its distance. If the target is not completely inside of the bubble, a "bump" on the bubble will enclosed the selected target so that it is obvious to the user what the current selection will be. Experiments showed a 30% decrease in average selection time over normal cursor selection.

Summary

The task of selection is fundamental to graphical user interfaces. Fitts' law shows that increasing the target size or decreasing the distance can reduce selection times. The steering law showed that small paths that must be navigated as well as careful corners can increase selection time. Various mechanisms have been proposed for overcoming these challenges including repositioning of selection items, force fields to "nudge" the mouse, larger or adaptable cursors and distortion of the size of objects in motor space. The special needs of pen users have been addressed by marking menus and stroke selections.

---

[1] Fitts, P. M., "The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement," *Journal of Experimental Psychology*, 41, (1954), pp 381-391.

[2] Accot, J. and Zhai, S., "Beyond Fitt's Law: Models for Trajectory-Based HCI Tasks", *Human Factors in Computing Systems (CHI '97)*, ACM, (1997), pp 295-302.

[3] Pastel, R. L., "Measuring the Difficulty of Steering Through Corners," *Human Factors in Computing Systems (CHI '06)*, ACM, (2006), pp 1087-1096.

[4] Card, S. K. "User Perceptual Mechanisms in the Search of Computer Command Menus," *Human Factors in Computing Systems*, ACM, (1982), pp 190-196.

[5] Callahan, J., Hopkins, D., Weiser, M. and Shneiderman,B., "An Empirical Comparison of Pie vs. Linear Menus", *Human Factors in Computing Systems (CHI '88)*, ACM, (1988), pp 95-100.

[6] Walker, N., Smelcer, J. B., and Nilsen, E., “Optimizing Speed and Accuracy of Menu Selection: a Comparison of Selection Times from Walking and Pull-Down Menus,” *International Journal of Man-Machine Studies,* 35, (1991), pp 871-890.

[7] Kobayashi, M., and Igarashi, T., “Considering the Direction of Cursor Movement for Efficient Traversal of Cascading Menus,” *User Interface Software and Technology (UIST ’03)*, ACM, (2003), pp 91-94.

[8] Ahlström, D., “Modeling and Improving Selection in Cascading Pull-Down Menus Using Fitts’ Law, the Steering Law and Force Fields,” *Human Factors in Computing Systems (CHI ’05)*, ACM, (2005), pp 61-70.

[9] Ahlström, D., Alexandrowicz, R., and Mitz, M., “Improving Menu Interaction: a Comparison of Standard, Force Enhanced and Jumping Menus,” *Human Factors in Computing Systems (CHI ’06)*, ACM, (2006), pp 1067-1076.

[10] Kurtenbach, G. and Buxton, W., “The Limits of Expert Performance Using Hierarchical Marking Menus,” *Human Factors in Computing Systems (CHI ’93)*, ACM, (1993).

[11] Kurtenbach, G. and Buxton, W., “User Learning and Performance with Marking Menus,” *Human Factors in Computing Systems (CHI ’94)*, ACM, (1994), pp 258-264.

[12] Tapia, M. A. and Kurtenbach, G., “Some Design Refinements and Principles on the Appearance and Behavior of Marking Menus,” *User Interface Software and Technology (UIST 95)*, ACM (1995), pp 189-195.

[13] Zhao, S. and Balakrishnan, R., “Simple vs. Compound Mark Hierarchical Marking Menus”, *User Interface Software and Technology (UIST ’04)*, ACM (2004), pp 33-42.

[14] Zhao, S., Agrawala, M. and Hinckley, K., “Zone and Polygon Menus: Using Relative Position to Increase the Breadth of Multi-Stroke Marking Menus,” *Human Factors in Computing Systems (CHI ’06)*, ACM (2006), pp 1077-1086.

[15] Baudisch, P. “Don’t Click, Paint! Using Toggle Maps to Manipulate Sets of Toggle Switches”, *User Interface Software and Technology (UIST ’98)*, ACM (1998), pp 65-66.

[16] Apitz, G. and Guimbretiere, F., “CrossY: A Crossing-Based Drawing Application”, *User Interface Software and Technology (UIST ’04)*, ACM (2004), pp 3-12.

[17] Kabbash, P. and Buxton, W., “The “Prince” Technique: Fitts’ Law and Selection Using Area Cursors”, *Human Factors in Computing Systems (CHI ’95)*, ACM (1995), pp 273-279.

[18] Keyson, D. V., “Dynamic Cursor Gain and Tactile Feedback in the Capture of Cursor Movements”, *Ergonimics*, 12, pp 1287-1298.

[19] Worden, A., Walker, N., Bharat, K. and Hudson, S., “Making Computers Easier for Older Adults to Use: Area Cursors and Sticky Icons,” *Human Factors in Computing Systems (CHI ’97)*, ACM, pp 266-271.

[20] Blanch, R., Guiard, Y. and Beaudouin-Lafon, M., “Semantic Pointing: Improving Target Acquisition with Control-Display Ratio Adaptation,” *Human Factors in Computing Systems (CHI ’04)*, ACM, (2004), pp 519-526.

[21] Grossman, T. and Balakrishnan, R., “The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor’s Activation Area,” *Human Factors in Computing Systems (CHI ’05)*, ACM (2005), pp 281-290.