AN INFORMAL STUDY OF SELECTION POSITIONING TASKS

William Buxton

Computer Systems Research Group
University of Toronto
Toronto, Ontario
Canada
M5S 1A1

ABSTRACT

Several techniques for performing selectioning-positioning tasks are compared. The comparison takes the form of a case study where the task is to select from among three geometric shapes and position them in two-space. The study emphasizes how much syntactic, lexical and pragmatic variables can influence the relative ease with which a particular task can be performed. In addition, it is shown how each approach has properties which make it optimal in some contexts. The overall impact of the study is to demonstrate the importance of actually implementing toy systems to test ideas and to point out the shortcomings of pencil and paper exercises.

1. INTRODUCTION

We are interested in achieving a better understanding of input structures. When confronted with a design decision we would like to have a good methodology for answering the following three questions:

- What are the alternatives?
- What are the relevant characteristics of each alternative?
- How can the best match be found between our needs and the means available?

In addition, we would also like to know:

- What would constitute a set of tools sufficient to render any viable alternative equally accessible? That is, how can we eliminate the bias of the path of least resistance.

Demanding an answer to these questions is a tall order. Even finding where to start is a problem. This largely explains why we have characterized user interface design an art rather than a science (Baecker, Buxton & Reeves, 1979).

The study reported herein considers these questions. It is an attempt to crystallize certain ideas that have formed over the past years in the course of our work at designing user interfaces. We have sought to isolate some of the key concepts that have developed and capture them in a case study which would permit their investigation. The study is informal in that no controlled tests were run. It is a preliminary probe rather than an experiment.

The work is rooted in the observation that systems which we have built or observed were constructed out of a set of reoccurring classes of transactions, regardless of application. For example, whereas a musician might select a note and place it in a particular position in pitch and time, so might a circuit designer place a particular gate in its proper place in a circuit. While what is accomplished in each case is quite different, both constitute the same generic transaction: a selection-positioning task.

If we accept the premise that the number of such generic transactions is finite, then attempting to enumerate the constituents of the canonical set:

- constitutes a first step in answering the first question posed above, "What are the alternatives?", and
- serves as the basis for designing the tools to support the implied interactions.

Such an approach is not new and forms the basis for the "logical devices" of the Core standard (GSPC, 1979; ACMCS, 1978). The problem is that the taxonomy that results is not of a fine enough grain to serve our purposes. The reason is that such generic categorization does not penetrate much below the syntactic level. The logical devices of the Core standard let us describe our musician's and circuit designer's actions in terms of "picking devices" and "locators", but does not contribute much in the way of characterizing the properties, for example, of the differing picking devices used.

Thus, in order to approach the questions posed at the start of this paper, we must penetrate a bit deeper into the structure of human-computer dialogues. Foley and Van Dam (1982) describe the user interface as consisting of four layers:

- conceptual
- semantic
- syntactic
- lexical

To this list we would add:

- pragmatic

Our point is that designing with logical devices is useful, but incomplete. This is due to an inability to adequately deal with the lexical and pragmatic levels. To illustrate what we mean by this is one of the main reasons that the current study was undertaken.

2. CASE STUDY OVERVIEW

If one step of systems analysis is to categorize a particular transaction as belonging to a particular type, then the next step is to understand the alternatives within that type. Alternatives at this level are distinguished by syntactic, lexical and pragmatic properties. It is these properties which we wanted to investigate. To do so, we implemented an environment which supported several different ways of performing one simple task: selecting a shape from among a square, circle and triangle, and positioning it in two-space. To provide a common basis for comparison, the techniques implemented were all constrained to function using the same hardware: a graphics tablet, a vector-drawing display and an ASCII keyboard. Within these constraints, the questions which we wanted to investigate were:

- What different techniques could we implement?
- What issues did each bring to light?
- To what extent could we characterize the properties of each?
- To what extent did our input support tools suffice in handling each of these techniques?

We will deal all but this last question in the remainder of this report.[1]

Five basic techniques for performing the designated task were implemented:

- Dragging
- Moving-Menu/Stationary-Pointer
- Character Recognition
- Typing
- Function Keys

Before discussing the techniques individually, there are a few general comments worth making. First, none of the techniques (except perhaps the second) is original. Hence, the value of the exercise does not lay in enumerating the possibilities. Rather, it rests in actually implementing the various techniques *in a common environment* so that they can be compared side-by-side through actual use. Second, our choice of a compound task, selection and positioning, was not accidental. The task is simple but non-trivial. Furthermore, it introduces the issue of syntax into the problem, as shall be seen.

1. Details on the menu system which supported the implementation described can be found in Buxton, Reeves, Patel and O'Dell (1979). A study of the consequences of this investigation on the input tools can be found in Ray and Kroll (1981).

3. DRAGGING

3.1 Simple Case

The first technique, "dragging", is illustrated in Figure 1. The user selects the desired shape by positioning the tracking cross over the appropriate menu item (on the right side of the display), and depressing the tablet cursor's selector ('Z') button.

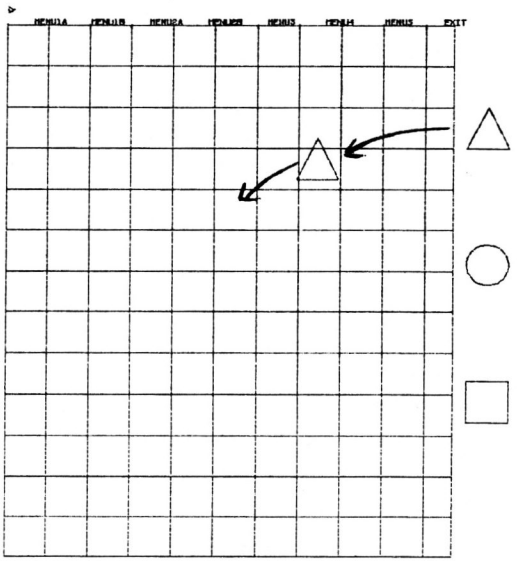

Figure 1. Dragging

A copy of the shape is thereby "picked-up", and follows the cursor's motion (for as long as the Z-button remains depressed). The shape can be positioned, therefore, by cursor movement and "anchored" by releasing the Z-button when in place.

The technique is simple and well known. Nevertheless, it brings to light a few interesting questions. For example, how does the location of the menu region affect the interaction? When, if ever, is it an advantage to have menus positioned along the top or bottom of the screen?

A less obvious issue is seen in the interplay between the syntactic and lexical components of the example. Two separate tasks are being performed: one selection and one positioning. For such multi-step dialogues not to be prone to error, it is desirable to design the interaction in such a way that it channels the user's actions along the "right path". Such a series of low-level operations is often (usually?) viewed as a single conceptual "chunk" by the user. Therefore, it could be that it is most appropriate for the gesture to

perform that task also to form a single "chunk". The Z-down/move/Z-up gesture, for example, binds the constituent operations into a single "compound word". This is in marked contrast with the common alternative of a Z-down/Z-up move Z-down/Z-up command sequence. In this latter case, the first Z-up is an act of *closure*, which disrupts the binding of the sub-tasks, and therefore the coherence of the overall task performance. The point to make here is that syntactic tokens can be bound together to the advantage of the user interface and that this is often made possible by recognizing the differing degrees of closure inherent in various lexical elements. Just as hyphens can bind words together, so can appropriate spellings of the tokens in the user dialogue. The question that begs to be asked, therefore, is how can we develop an understanding of these features and learn to use them to best advantage?

3.2 Redundant Case

In the previous version of the dragging technique, the user had to go back to the menu to select an item before each placement. If not, depressing the Z-button resulted in the tracking symbol becoming a question mark icon. This is clearly inefficient in terms of hand movement in cases where several instances of the same shape are to be laid down. A variation on the technique, therefore, is to use a "paint-pot" analogy and have the last item selected "remain on the brush". Once the first instance of a shape has been selected and positioned, subsequent instances of that shape can be input by a simple Z-down/Z-up gesture at the appropriate position.

While this modification will often result in a more efficient system (as measured by the keystroke model of Card, Moran & Newell, 1980), there is one observation worth making. The potential savings will not always be taken advantage of by the user. For example, a designer will not generally lay out all the AND gates and then all the OR gates of a circuit. Rather, the circuit will be built up in logical order. The semantics of the task will dictate the order rather than syntactic/lexical efficiency. Recognition of such facts will help the interface designer to prioritize where effort should be invested in attempting to improve the quality of the user dialogue.

4. MOVING-MENU/STATIONARY-CURSOR

4.1 Simple Case

With the dragging technique, much hand motion was expended in moving between the menu and the work areas. This is illustrated in Figure 2, where we have drawn vectors to connect all points on the screen where an interaction occurred during the course of performing a simple layout task.[2] One way to save

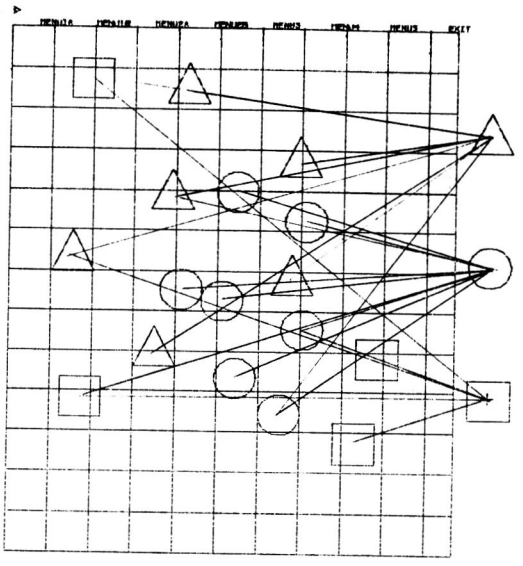

Figure 2. Hand Motion in Dragging

much of this hand motion (and the time that it consumes) is to have the menu come to us rather than us going to it. We do this by placing the tracking cross over the position where we want a shape located and depressing the Z-button. This causes two things to happen (for as long as the Z-button remains depressed):

- the tracking cross becomes anchored at its current location.
- the menu seen in Figure 3, becomes the tracking symbol *i.e.*, it follows the motion of the tablet's cursor.

Conceptually, what we now have is a moving menu and a stationary pointer. The shape desired is selected by placing it over the (stationary) tracking cross and releasing the Z-button.

The technique is effective in many contexts. However, it has some interesting properties and begs some important questions. When compared to dragging, for example, it breaks down as the number of items on the menu increases.

Also, the technique is not self-obvious. That is, there are no explicit cues to prompt the user as to the nature of the interaction. Dragging, on the other hand, is self documenting for anyone who has previously seen it. On the other hand, the technique does not permanently consume display real-estate to hold the selection menu, thereby providing a larger

2. Diagrams such as this can be made from the "dribble file" in which a time-stamped record of all interactions can be placed. The data that results can be used in the evaluation of the user interface through the technique of protocol analysis. Such a file can be generated on request from any program which uses our menu-support tools. The example hints at the potential value of such tools, but also forces us to realize the inadequacy of our current knowledge concerning techniques of protocol analysis, *viz.*, how to interpret the data.

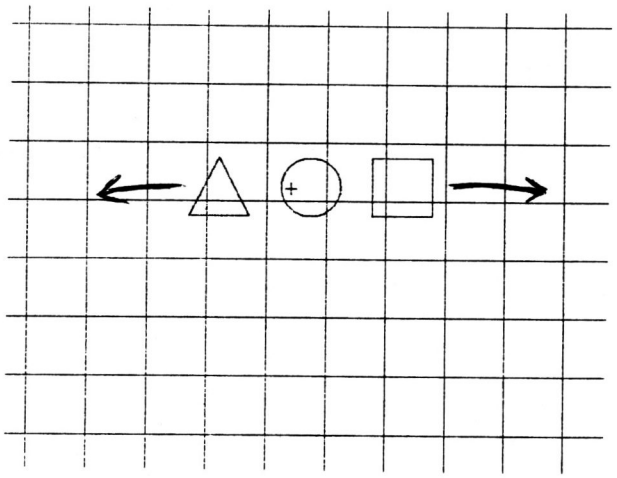

Figure 3. Moving Menu / Stationary Pointer

effective work area.

Having the menu come to us is not new. This is the approach taken by many systems developed at Xerox PARC, for example (Tesler, 1981). The differences are due to the properties of the display technologies used. The Xerox technique is implemented on a raster-scan display which allows the menu to be displayed in the current work region by overwriting a raster or being exclusive OR'd with it. In either case, the menu remains stationary, as in dragging. It just appears closer, and the tracking symbol performs its regular function.

The properties of a vector-drawing display, however, prevent us from clipping a temporary window into which we can place our menu, or performing the equivalent of the exclusive OR function. Consequently, laying the temporary menu down in a stationary position will often result in menu items not being distinguishable from those already entered in the same region. Anchoring the tracking cross means that ambiguities can be easily resolved using cues resulting from manually moving the menu.

A final point has to do with the syntax of specifying the two sub-tasks. Notice that the syntax of the transaction has been reversed when compared to dragging, where items were first selected, then positioned. There is some question as to whether it is more "natural" to perform the selection task first. The question is important in its own right, but there is also a more global issue. Barnard, Hammond, Morton, Long and Clark (1981) give evidence as to the importance of self-consistency of syntax within a system. If, for example, there are several compound tasks that involve selection, the evidence suggests that the selection task should appear in the same syntactic position in each case. The literature is not conclusive and a great deal of work remains to be done. In the meantime, however, these considerations should be kept in mind by designers.

4.2 Redundant Case

The moving-menu/stationary pointer technique can also be implemented to facilitate entering several instances of the same shape. This is accomplished by, on the Z-down, always having the menu appear positioned such that the last selected shape appears centred over the anchored pointer. Thus, for the second instance on, the input gesture is exactly the same as that for the second instance on in the redundant mode of dragging: a simple Z-down/Z-up.

5. CHARACTER RECOGNITION

Much of the preceding discussion has centred on syntactic issues resulting from the compound nature of selection-positioning tasks. It could be argued that if these two tasks were integrated into a single gesture, the simplified syntax would result in a system which is easier to learn and less prone to error. Reisner (1981) gives some experimental evidence in support of this notion.

The character recognition strategy has this integrated property. To input a shape one simply sketches a short-hand symbol at the desired location in the work area. The shape specified is input centred over the starting point of the sketch. The (arbitrary) shorthand symbols used in our example are shown in Figure 4

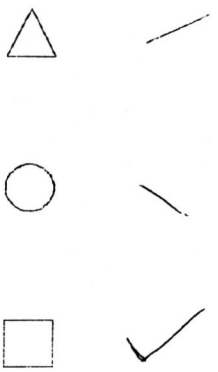

Figure 4. Shorthand Symbols

Like the moving-menu/stationary-pointer technique, this approach requires no display real-estate for control functions. However, it also shares the property of not being self-prompting.

One of the main issues of character recognition approaches has to do with the cognitive burden of

remembering the symbols.[3] One approach is to use a trainable character recognizer based on the premise that it is easier for users to remember symbols which they have designed themselves. Whether this is true or not depends on several factors, including the number of characters and what they have to represent. The training process imposes a cognitive (and temporal) burden of its own, and such character recognizers usually respond more slowly that ones which operate on a predefined set of symbols.

One way which the short-hand symbols can be made easier to remember is to make them representational. Thus symbols would look like transistors, AND gates, or squares, for example. There are problems with this, however. As the symbols become more complex, they take longer to draw, are more prone to error and take longer to recognize. That is, the main benefits of adopting the technique in the first place -- fluency and speed -- are defeated. Consequently, it is often best to adopt a set of pre-defined symbols, each of which can be specified by a continuous line (thereby providing for maximum lexical compactness). One main area for future research concerns how to obtain optimal performance within these constraints.

6. TYPING

Typing a command to select and position shapes was one of the techniques implemented. The syntax used was:

<command> <X val> <Y val>

where the command was one of 's', 'c' or 't' for square, circle, and triangle, respectively. The X and Y values specified the coordinated over which the shape was to be positioned.

One of the reasons that typing is interesting is that it points out that *three* tokens are required to fully specify the task. In the previous techniques we have been implicitly treating the position as a single token. The importance of this observation is to point out how effectively the appropriate interaction can bind elements into a single unit. What we described with the character recognizer with respect to selectioning and positioning, we had been doing all along with the specification of the X and Y values of the position.

While we would sometimes like to believe that graphics solves all problems, the current exercise points out one important thing: sometimes it is better to type. If a user was given a picture made up of squares circles and triangles to reproduce using each of the techniques described, it is most likely that typing would lead to the slowest task performance. However, if the task was reformulated and the picture was presented as a list of numerical coordinates (as is often the case in the real world), the typing technique would clearly be the fastest (given a skilled operator). The reason is that the means of performing the task has a good cognitive "fit" with the task formulation. Similar results would result in cases where a high degree of accuracy was required.

The previous points lead us to a more general comment. *Each* of the techniques described has different strengths and weaknesses, and there most likely exists a task for which each is optimal and each is abysmal. Which technique is appropriate is always a function of the task to be performed *and how it is formulated*.

7. FUNCTION KEYS

The final technique to be presented involves pointing to the position where the shape is to be located and pressing one of three function keys. In our study, the function keys (one for each of the square, circle and triangle) were mounted on the back of the tablet cursor. The result was that, for free-hand layouts, this was the fastest of all of the techniques tested.

From this part of the study, several important questions remain unanswered. Apart from leaving one hand free for other tasks, is there a speed advantage to using the same hand for pointing and selecting? When and under what constraints? Also, when does the speed advantage of function keys break down? Can this be improved by using chording keys, and if so, at what price (for learning and remembering)?

One approach which we have both seen and tried is to use a cursor with only three or four buttons, but change their meaning in different contexts. Our experience is, however, that this is confusing to novice users and leads to a high number of errors. The reason is that "different contexts" means different modes, and as Tesler (1981) has argued, we should be eliminating rather than emphasizing modes. Our conclusion is to fix the function for each key. To get maximum benefit, therefore, we choose the functions to be ones which are both global and frequently used.

One final point, the example provides a good opportunity to point out one difference between a tablet cursor *vs* a stylus: the cursor has function keys and can therefore be used as a selector *simultaneously* with being used as a pointer. As with the character-recognizer, we see how the pragmatic component can lead to a binding of associated functions so as to support a more articulate and fluent means of task performance.

8. CONCLUSIONS

A number of different techniques for performing selection-positioning tasks have been presented. Each was shown to have differing properties in terms of costs and benefits. One conclusion has been that which of these techniques is most appropriate in a given context depends on the task to be performed and how that task is formulated. If it is important to

3. This is of special concern in systems designed for casual users.

understand the differences among the various techniques, then it is perhaps even more important to acknowledge the value of the technique used to bring these properties to light. We believe that creating a test bed where these techniques can be refined and compared in a common, manageable environment has resulted in an acceleration of our understanding of the various issues and their respective importance. Some of the questions which have arisen will now be pursued in more detail. Finally, we intend to use the same test bed to investigate other tasks, and to develop better tools for prototyping such toy systems and evaluating their performance.

9. ACKNOWLEDGEMENTS

The work reported owes a great deal to the work of Sanand Patel, Howie Ray and Anthony Kroll. Patel implemented the menu system which was used, while Ray and Kroll did the implementation of the techniques described. In addition, I am indebted to Alain Fournier for his helpful comments during the preparation of this manuscript. Finally, I would like to acknowledge the financial support of the Social Sciences and Humanities Research Council of Canada and the National Sciences and Engineering Research Council of Canada.

10. REFERENCES

ACMCS (1978). *ACM Computing Surveys*. Special Issue: Graphics Standards, 10(4)

Baecker, R., Buxton, W. & Reeves, W. (1979). Towards Facilitating Graphical Interaction: Some Examples from Computer-Aided Musical Composition. *Proceedings of the 6th Canadian Man-Computer Communications Society Conference*, Ottawa, May 1979: 197-207.

Barnard, P., Hammond, N., Morton, J., Long, J. & Clark, I. (1981). Consistency and Compatibility in Human-Computer Dialogue. *International Journal of Man-Machine Studies* 15(1): 87 - 134.

Buxton, W., Reeves, W., Patel, S. & O'Dell, T. (1979). *SSSP Programmer's Manual*. Toronto: unpublished manuscript, Computer Systems Research Group.

Foley, J. D. & Van Dam, A. (1982). *Fundamentals of Interactive Computer Graphics*. Reading, Massachusetts: Addison-Wesley.

GSPC (1979). Status Report of the Graphics Standards Committee. *Computer Graphics*, 13(3), August, 1979.

Card, S., Moran, T. & Newell, A. (1980). The Keystroke-Level Model for User Performance Time with Interactive Systems. *Communications of the ACM*, 23(7): 396 - 410.

Ray, H. & Kroll, A. (1981). *A Study of Interaction Using Menu-Driven Systems*. Toronto: unpublished manuscript, Computer Systems Research Group.

Reisner, P. (1981). Formal Grammar and Human Factors Design of an Interactive Graphics System. a *IEEE Transactions on Software Engineering* (7)2: 229 - 240.

Tesler, L. (1981). The Smalltalk Environment. *Byte* 6(1): 90 - 147.