A TABLET INPUT FACILITY FOR AN INTERACTIVE GRAPHICS SYSTEM

James E. Curry
Adams Associates
A division of
Keydata and Adams Associates Incorporated
Watertown, Massachusetts

Introduction

Effective man-machine communication in an interactive graphics environment requires a natural and convenient way for the on-line user to input information and issue commands to the computer. Several applications have been described[1] 1,2,3 in which such an interface has been achieved by allowing the user to draw symbols on a tablet. In order to make this mode of interaction generally available to users of the TX-2 graphics facility,* we have augmented the TX-2 graphics programming language LEAP[5] with constructs for handling tablet input and interfacing with a user-trainable symbol recognizer. These extensions to LEAP have made the incorporation of a flexible input technique into a graphics application program as straightforward for the programmer as typewriter input. Experimentation with the man-machine interface has thereby been greatly simplified, resulting in the faster development of useful graphics applications. This paper illustrates an application which uses the symbol recognition facility, and describes the language extensions through which the interactive aspects of such an application are implemented.

Symbol Recognition

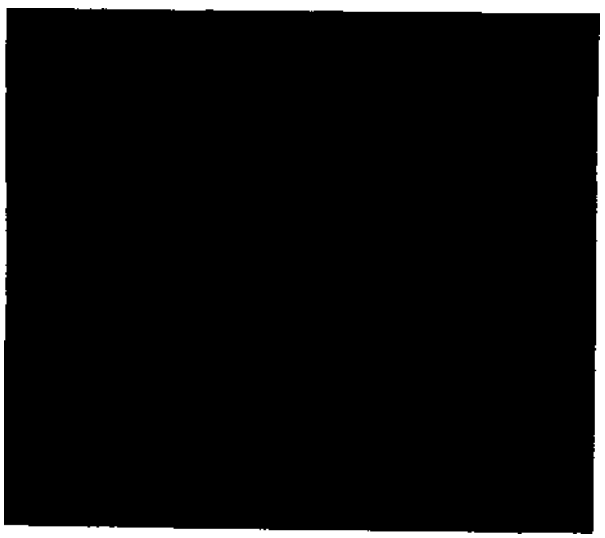
Since the symbol recognition facility is intended for a wide variety of graphics applications, it cannot be limited to a fixed symbol vocabulary. Therefore, we provide a recognizer which can be trained by the individual user to accept whatever symbols he finds convenient. The recognition algorithm is similar to that described by Teitelman.[6]

The recognizer is called from a user program whenever a symbol has been drawn. Properties are extracted from the path of each stroke of the drawn symbol. These properties characterize where the path starts with respect to four overlapping regions of the rectangle enclosing the stroke, and how many times the path crosses the boundaries of these regions. The relative positioning of the strokes composing the symbol is also recorded. The properties extracted from the input symbol are compared to entries in the specified user-defined symbol dictionary, yielding an exact match, a conditional match (most properties of some entry match but not all), or no match (either there is no conditional match or two or more matches are equally close). If the symbol is recognized, either exactly or conditionally, the recognizer returns the character string associated with the matching dictionary entry, and the position and size of the symbol drawn.

This scheme has several deficiencies which stem from its insensitivity to properties such as curvature and occurrence of corners. However, we have found it more than adequate for any application we have attempted; we have used vocabularies of more than eighty symbols without difficulty.

Symbol Definition

The user creates or modifies a named symbol dictionary with a special interactive training program. This program allows the user to enter symbols into a specified dictionary by • simply drawing examples of the symbols and indicating what their defining character strings are to be. For example, in order to define a symbol as 'TRN' (for transistor), the user draws in the writing area of the training program display.

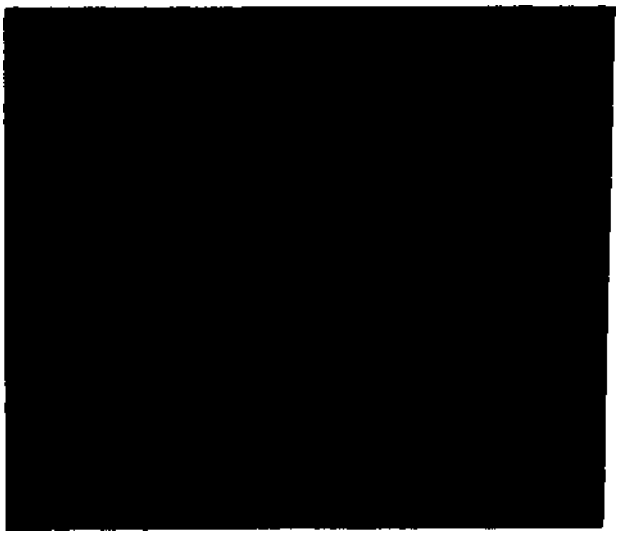


The trainer displays a scaled version of the drawn symbol on the left and calls the recognizer to analyze the symbol. If no match is found, the message •??• is displayed below the writing area.

This work was performed under a subcontract of M.I.T. Lincoln Laboratory which is operated with support from the U.S. Advanced Research Projects Agency.

The user may now define this symbol by touching one of the character strings on the lower part of the screen, or by entering a string (in this case 'TKN') on the typewriter. This causes the properties of the drawn symbol to be entered in the dictionary and associated with 'TRN.' This is indicated by the message 'TRN' displayed below the writing area.

If the user now draws a symbol with the same properties, it will be recognized exactly, and the message 'TRN' will be shown.



A somewhat different symbol might be a conditional match to this dictionary entry. In this case, the message 'TEN?' appears, indicating conditional recognition.

The user may, if he wishes, define this to be a second version of the symbol by touching the •TRN' target. From this point on, the recognizer will indicate exact recognition for either version. (However, it is usually sufficient to define only one version of a symbol, since application programs do not distinguish between exact and conditional recognition.) The user could instead have defined this symbol as a different string; the recognizer would then distinguish the two symbols.

The trainer allows the user to delete dictionary entries with the 'DEL' target appearing on the right of the screen. He can also step through the entries for a given character string with the 'FIND' and 'NEXT' targets, and examine the properties recorded for each entry. These facilities provide the user with the ability to edit a symbol set. This would be done, for example, when the user wishes to replace the set of application-dependent symbols in an existing dictionary which contains symbols for the normal character set.

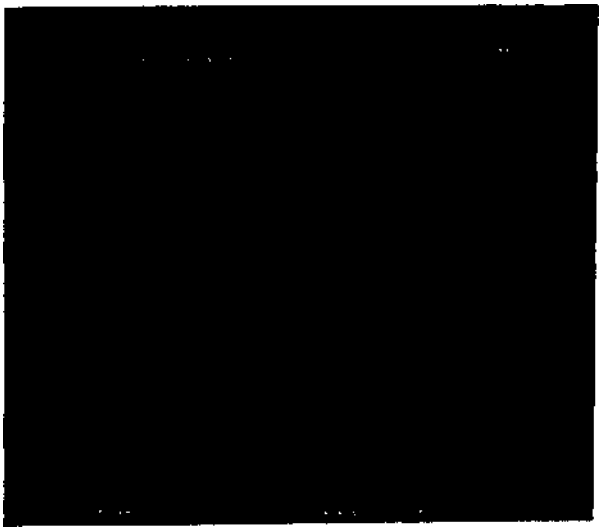
Creating a typical dictionary of about forty symbols takes about one-half hour of console time. Users feel that this minimal effort is a small price to pay for the ability to communicate with the computer in a way which is natural and convenient to them.

An Example

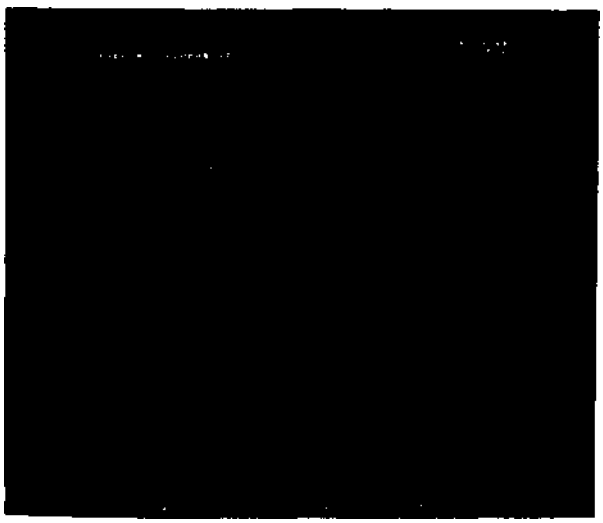
There are a number of graphics applications on TX-2 which use the symbol input facility as the primary user interface. These applications include a graphical (two-dimensional) programming language,[7] a computer-aided logic design system,[8] and a program to assist in the layout of the photo masks required for the diffusion and etching processes involved in the fabrication of integrated circuits.[9,10] A description of the operation of the mask layout program will indicate some of the interactive techniques which users have found convenient•

The mask designer must first create a dictionary of symbols for the commands accepted by the program. He decides what symbol he wants to draw to make a transistor appear, for example, and defines this symbol as the string 'TRN.' The typical dictionary for this application contains about thirty of these symbols.

The user begins layout out a mask with the placement of components on the screen. To place a transistor mask, he draws his 'TEN' symbol where he wishes the transistor to be located.



The symbol disappears and a transistor mask appears in its place.



To attach a resistor to the right of this transistor, he draws a 'RRES' symbol (for right-handed resistor). Resistors are of variable length, so the length of the symbol determines the length of the resistor mask created.

The resistor symbol is recognized, and a resistor mask of the designated length appears.

Various other types of components may be added to the picture in this way.

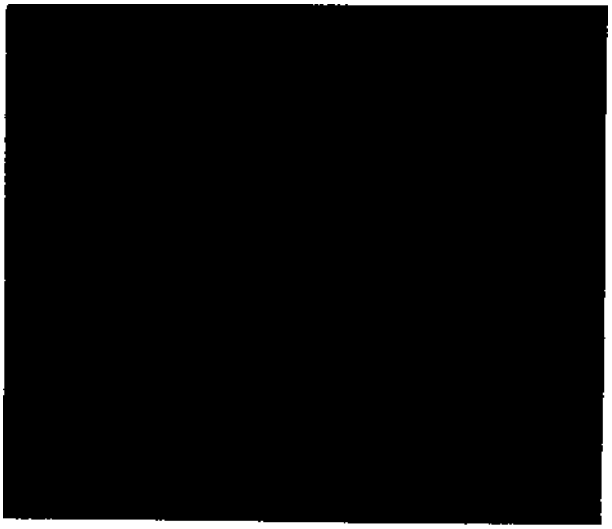
Picture manipulations are also specified by drawing symbols. A component or group of neighboring components may be deleted by drawing a symbol defined as 'DEL' at the appropriate place on the screen.

The components appearing within the bounds of the symbol disappear with the symbol.

An existing component or group of components may be manipulated by first drawing a 'SEL' symbol over the components to select them for some further operation.

Each component within the bounds of the symbol is marked to indicate that it will be affected by the next command.

When all desired components have been selected (which may require several 'SEL' symbols if these components are not adjacent), the user may draw a 'MOV' symbol to indicate the direction and extent of a translation operation.



The selected components are moved by the designated offset.

Other available move commands specify movement to a designated point, and translation in the horizontal or vertical direction only, either through a designated distance or to a designated point. There are also commands for deleting all selected components and for placing a copy of the selected component set at a given point.

The user can also control how much of the mask being designed appears on the scope through commands for scaling and translating the entire picture. One useful command in this class is invoked by the 'FIT' symbol.

This symbol causes the enclosed portion of the picture to be scaled to full screen size for more detailed work.

Other commands give the user the ability to name the current circuit and save it in his filing system, and to recover and merge named circuits. The circuit name may be drawn on the tablet with alphanumeric symbols. However, users generally find it more convenient to enter the name on the typewriter•

It is often convenient to have a symbol which calls the trainer from the application program. The dictionary can be refined or augmented, and layout work then resumed using the new symbol set.

Language Facilities

Most of the graphics programming on TX-2, including the application described above, is done in LEAP,[5] an ALGOL-based language to which constructs have been added for data structure manipulation, homogenous matrix operations, display output, file handling, and other facilities oriented to time-shared graphics. These constructs make the implementation of picture manipulations such as those involved in the mask layout program straightforward for the programmer. LEAP has been further extended to allow simple but powerful language forms for monitoring interactive events and communicating with the symbol recognizer. (LEAP is implemented in VITAL,[n] a compiler-writing system, permitting easy extension of the language.) The skeletion LEAP program below illustrates the level of programmer effort required to incorporate sophisticated tablet input into an application program. (Certain syntactic liberties have been taken in this example to make it more intelligible to the uninitiated reader.)

```
        START (FILE R);
```
Declare an input parameter R of data type 'file name' which will be typed when the program is called.

```
        R→RECOGNIZER;
```
Store the file name into the reserved variable RECOGNIZER to establish the file as the symbol dictionary to be used by the recognition facility.

```
        ACTIVATE TABLET;
```
Request the interactive event monitoring system to display and buffer the pen track and push an event into the input queue when drawing is completed. (Monitoring of other events such as button pushes, target hits, etc., might also have been requested.)

```
NEXT:   GET NEXT INPUT;
```
Pop an event from the input queue (pausing until an event occurs if the queue is empty) and set the reserved integer variable CAUSE to the identification number for the event.

```
        BRANCH ON CAUSE TO...,SYMBOL,...;
```
Branch on the event number to the label of the appropriate processing section for the event.

```
SYMBOL: RECOGNIZE;
```
A tablet action has been completed. Analyze the pen track using the symbol set named in RECOGNIZER. The reserved string variable SYMBOL is set to the associated text string; XCEN, YCEN, WIDTH, HIGHT indicate the center and dimensions of the drawn symbol. The buffer of tablet points is also available in the two dimensional array INK if additional information is needed (for example, the startpoint of the symbol).

```
        CLEAR TABLET;
```
Erase the drawn symbol.

```
        EXECUTE CASE SYMBOL OF
```
Execute the code for the action indicated by SYMBOL.

```
            BEGIN
                CASE ' ': ...
```
The nullstring indicates the symbol was not recognized. Typically a pattern is flashed on the screen or an audible signal given when this occurs.

```
                CASE 'TRN': ...
```
Place a transistor mask at XCEN, YCEN.

```
                CASE 'SEL': ...
```
Record and mark all components whose centers are within the rectangle described by XCEN, YCEN, WIDTH, HIGHT.

```
                CASE 'MOV': ...
                        .
                        .
                        .
            END;
        GO TO NEXT;
        FINISH
```
Move all selected components through the distance indicated by the first and last points of the first stroke as recorded in the array INK.

Conclusion

We feel that the techniques described above are a significant improvement on conventional-interactive input. In most instances, only graphical information is drawn; text and commands must be specified through either targets on the screen (which clutter the picture and increase display load) or auxilliary keyboards or push-button arrays (which are distracting and have only limited mnemonic value). With a symbol recognition facility, the user can input graphical, textual, and command information with a single device which is natural and convenient to use.

The most beneficial result of adding these facilities to the TX-2 graphics system has been that experimentation with the man-machine interface has been greatly simplified. Experience has shown that a major obstacle in developing applications programs is getting a potential user to decide what he wants to say to the computer and how he wants to say it. The flexibility now provided in this area for both the programmer and the user has contributed significantly to the faster development of useful graphics applications.

References

[1] Bernstein, M. I., "An On-Line System for Utilizing Hand-Printed Input," System Development Corporation, Santa Monica, California, TM-3052, July 1966.

[2] Ellis, T, 0., and Sibley, W. L., "On the Problem of Directness in Computer Graphics," Emerging Concepts in Computer Graphics, Benjamin Press, 1968.

[3] Anderson, R. A., "Syntax-Directed Recognition of Hand-Printed Two-Dimensional Mathematics," ACM Symposium on Interactive Systems, Washington, D. C., August I967,

Sutherland, W, R., Forgie, J. W., and Morello, M. V., "Graphics in Time-Sharing; A Summary of the TX-2 Experience," 1969 Spring Joint Computer Conference.

[5] Rovner, P. D., and Feldman, J. A,, "The LEAP Language and Data Structure," IFIP Congress I968, Edinburgh, Scotland, August 1968,

[6] Tietelman, W., "Real-Time Recognition of Hand-Drawn Characters," I964 Fall Joint Computer Conference, p. 559-575.

[7] Rovner, P. D., and Henderson, D. A., Jr., "On the Implementation of AMBIT/G," International Conference on Artificial Intelligence, Washington, D. C, May 1969.

[8] Horbuckle, G. D,, Deiphuis, R., Spann, R., and Thomas, E., "Computer-Aided Logic Design on the TX-2 Computer," 6th Annual Design and Automation Workshop, Miami Beach, Florida, June 1969.

Sutherland, W. R,, "Computer Assistance in the Layout of Integrated Circuit Masks," IEEE International Convention Digest 1968, New York, New York, March 1968.

[10] Richardson, F. K,, and Oestreicher, D. R., "Computer Assisted Circuit Photomask Layout," Illinois Computer Graphics Symposium, University of Illinois, Urbana, Illinois, March 1969.

[n] Mondschein, L. F., "Vital Compiler-Compiler System Reference Manual," M.I.T. Lincoln Laboratory Technical Note 1967-12, February 1967.