SIGNWRITER

Richard Gleaves
Valerie Sutton

Deaf Action Committee for SignWriting
Center For Sutton Movement Writing
La Jolla, California, USA
rgleaves@signwriting.org
sutton@signwriting.org

Abstract

This paper reviews the design history of SignWriter, a word processor for the SignWriting system. While the primary goal of SignWriter was simply to create a word processor for SignWriting, its development and subsequent use had several beneficial effects on the SignWriting system. Various design aspects of SignWriter are considered in the context of current computing technologies and sign processing development efforts.

Background

The SignWriting system [Sutton04] was conceived, developed, and used for many years as a hand-written notation. In particular, its use predated the introduction of low-cost personal computers.

In 1984 Emerson and Stern Associates, a small educational research and development firm, received a grant to develop a word processor for SignWriting. The resulting software, which operated on an Apple II computer, supported only a minor subset of the SignWriting system and was more of a demonstration than a useful tool: it was not subsequently used, and received no further development. The application was notable for displaying the symbols in a virtual "picture frame" around a central editing area, with symbols selected for entry by moving a cursor around the frame until the desired symbol was reached.

Emerson and Stern's software design implied that SignWriting was too complex for the personal computers of the time. Interestingly, their response was to devise an entirely different writing system named SignFont [Newkirk87], which traded computational simplicity - it was designed as a standard Macintosh font - for notational obscurity. SignFont's subsequent nonuse suggests that this design tradeoff was unsuccessful.

SignWriter Apple

It was in this context that SignWriter was conceived in 1986. The intended use for SignWriter was in education and the hardware platform was once again the Apple II, which at the time was an established standard for personal computing. The design goal was to implement the full SignWriting system in a simple but complete and usable word processor.

This more ambitious goal could be attempted on the same hardware because as a former member of the UCSD Pascal project, Richard Gleaves had several years of experience developing system software for the Apple II, and knew how to program in assembly language and make full use of the Apple's 128KB memory. In addition Gleaves' Pascal project colleague Mark Allen provided some high-performance graphics routines that he had developed for writing arcade-style games on the Apple II.

Much of the design effort in SignWriter was spent on two issues:

… Developing a memory-efficient encoding for SignWriting text
… Devising user interface mechanisms for efficiently typing symbols

SignWriting symbols were encoded using a variable-length byte-code system that was introduced in UCSD Pascal p-code [Bowles78] and later adopted for use in Java object code. The SignWriter graphics engine interpreted the byte codes as instructions for drawing symbols on the screen in specific locations and orientations.

Typing was chosen as the input mode for two reasons. First, while mice were available for the Apple II they were an optional add-on and therefore most Apple IIs did not have them. Second, the SignWriting system was receiving criticism at the time for allegedly being a form

of illustration rather than a true writing system. Therefore an efficient typing mechanism would cause SignWriter to serve as implicit proof that SignWriting was indeed a form of writing.

It was evident that SignWriting's complex symbol set would prevent it from being typed as efficiently as the Roman alphabet on a standard keyboard. However, the design that evolved - which involved the context-sensitive dynamic redefinition of the keyboard keys - yielded a valuable tradeoff of efficiency for learnability. The key boxes displayed on the screen highlighted the natural categories of the SignWriting symbols in a manner that allowed the typing mechanism to serve as an implicit learning tool: a crucial property given the symbol set complexity and the application's intended audience. See Figures 1, 2 and 3 from the SignWriter-At-A-Glance Instruction Manual.

The SignWriting symbol images were created by Valerie Sutton using the SignWriter symbol editor program. In addition she defined the mapping of SignWriting symbols to the keyboard keys. As with the key boxes, this mapping emphasized learnability by grouping symbols according to their natural categories. Conversely, the mapping of the key box keys and symbol attribute keys (Arrow, Cursor, Mirror, Size, and Rotate) was determined strictly by typing efficiency.

SignWriter's Find and Replace commands were implemented (at significant expense in memory) both to establish SignWriter as a complete word processor and again to demonstrate SignWriting's status as a true writing system. Unfortunately the search algorithm did not take into account the relative positioning of symbols within a sign, thus making the search feature itself more of a demonstration than a useful tool.

Because SignWriter was developed as a stand-alone application, it was free to possess an application-specific user interface. The interface design was influenced by Tufte's principle of graphical minimalism [Tufte83]: namely, every pixel that was not part of a SignWriting symbol existed onscreen only because it was functionally necessary. While this design approach may seem austere given today's large color displays, it made for a simple and easy-to-use interface on the Apple II, which had a screen resolution of only 560 by 192 pixels.

The major drawbacks to SignWriter's interface design were the inefficient cursor movement commands and the need for a keyboard card showing the assignment of SignWriting symbols and commands to the keys.

The Apple II version of SignWriter supported the full SignWriting system as it was defined at the time (palm orientation had not yet been introduced). The software was quite usable, but was never widely used because experienced SignWriting users had to type in each occurrence of each sign, while for new users typing symbols was relatively inefficient and – in the absence of a system for teaching typing – posed a significant learning curve.

SignWriter DOS

By the late 1980s the IBM PC had replaced the Apple II as the personal computer of choice. SignWriter was ported to the IBM PC with programming assistance from Barry Demchak. We chose the CGA display mode because at the time it was the graphics display mode supported by the most PC models, and because its screen resolution of 640 by 200 pixels was close enough to the Apple to simplify porting the existing symbol graphics to the PC (which is why the SignWriter symbols are so jagged).

The extra memory available on the IBM PC allowed SignWriter to be expanded with additional symbols, a sign dictionary, and support for multiple countries and languages. These features (along with software distribution on the Internet) had a significant impact on SignWriter use, as researchers began using SignWriter to create and publish dictionaries for various signed languages. This is the version of SignWriter that is in common use today.

Effects on SignWriting

The purpose of SignWriter was simply to provide a word processor for the SignWriting system. However, its development and subsequent use had several beneficial effects on SignWriting:

… SignWriter offered a concrete proof of SignWriting's status as a systematic notation rather than an ad hoc form of illustration. This notion influenced the subsequent design of the software.

… The typing mechanism served as an implicit interactive system for learning the SignWriter symbols (an important achievement given the complexity of the symbol set).

… The SignWriter symbol editor was withheld from distribution to ensure the controlled development of the

SignWriting system as it evolved to support more and more signed languages.

… The constraints of computer implementation exerted a positive influence on the subsequent evolution of the SignWriting system.

… The SignWriter software itself served as an efficient means of distributing the SignWriting system, and established a de facto standard for data exchange (an effect greatly amplified by the introduction of the Internet).

Conclusion

Beyond its immediate value as a tool for practical sign processing, SignWriter offers a number of lessons for current and future developers of sign processing software.

The most important is the need to standardize a user interface mechanism for symbol input; just as the symbol set is being standardized across all sign processing programs that use SignWriting, so must symbol entry. Such a standard should be centered on typing, with mouse input as an alternative rather than a replacement. Compelling pedagogical and linguistic reasons exist for providing efficient input mechanisms at the level of symbols rather than signs; while such mechanisms need not supplant text entry at the sign level, the reverse equally holds true.

The diagrams in this paper illustrate SignWriter's typing-based symbol input system as an example of how future typing-centered systems could be designed.

With regards to efficiency, Valerie Sutton has learned to type SignWriting almost as efficiently as English. This suggests that with the proper training (an accepted norm for typing) and appropriate hardware (e.g., a notebook computer with an integrated touchpad for cursor control and fine symbol positioning), typing-centered symbol input may well prove superior to any mouse-based systems.

Finally, SignWriter demonstrated that with the appropriate software architecture a true word processor could be implemented for SignWriting given limited resources for memory, processing power, and display resolution. This in turn suggests opportunities for developing useful sign processing software on the emerging handheld computing platforms such as PDAs and cell phones.

References

[Bowles78]
Bowles, Kenneth L., "UCSD Pascal", Byte. 46 (May)

[Newkirk87]
Newkirk, Don, "SignFont Handbook", San Diego: Emerson and Stern Associates (1987)

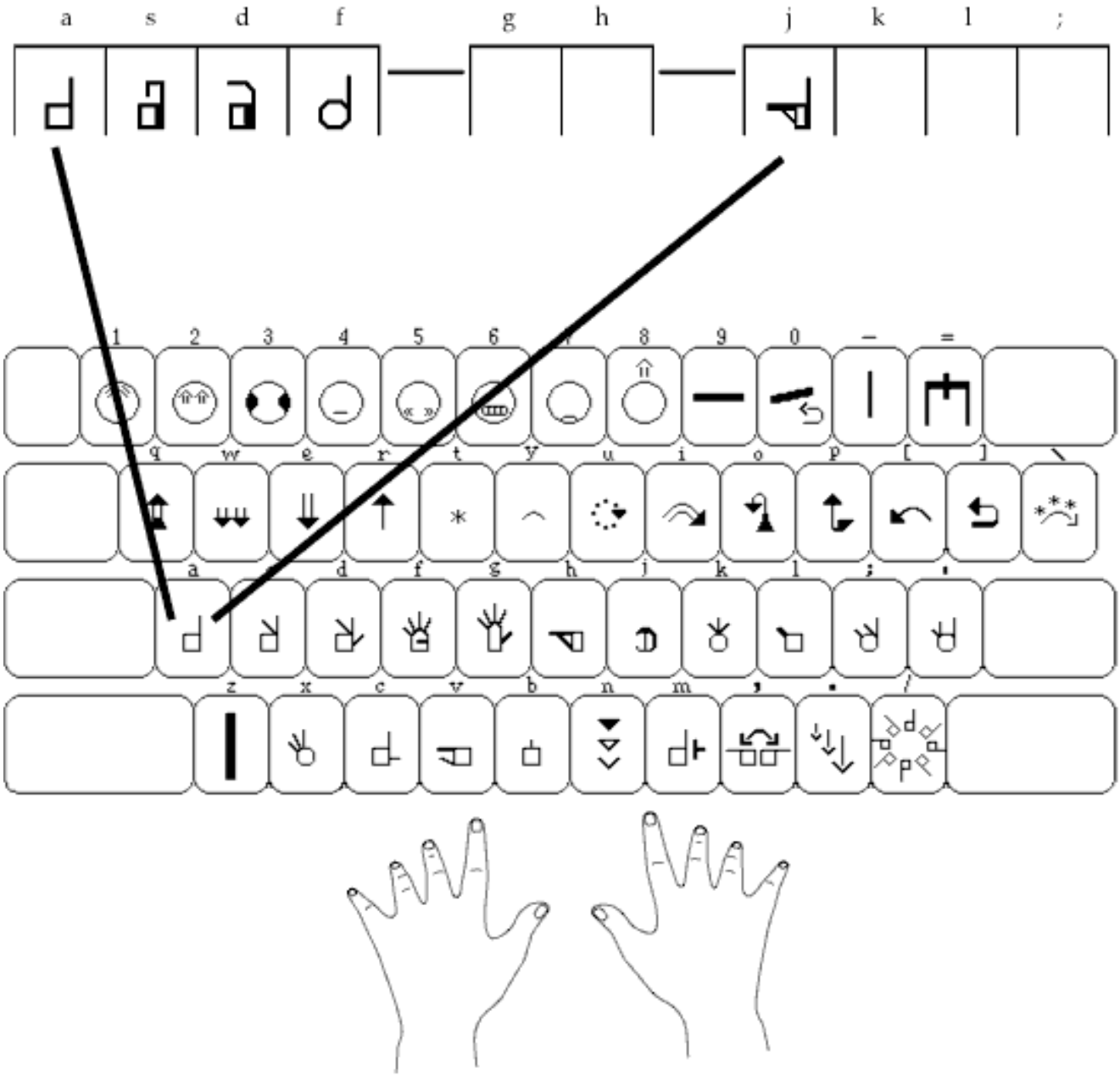
[Tufte83]
Tufte, Edward R., "The Visual Display of Quantitative Information", Graphics Press (1983)

[Sutton93]
Sutton, Valerie. SignWriter-At-A-Glance Instruction Manual, SignWriter Computer Program Notebook, Deaf Action Committee For SignWriting (1993)

[Sutton04]
Sutton, Valerie. SignWriting Site. www.signwriting.org

Typing Group 1

with the **SignWriter®** Computer Program



Group 1 is located on the **a** key.
When you press the **a** key, all of the hands in **Group 1** appear in a row on the screen.
You can then choose the hand symbol you wish, by pressing the **a, s, d, f, or j** keys.

Figure 1: A page from the SignWriter-At-A-Glance-Manual. Symbol groups are under each key.

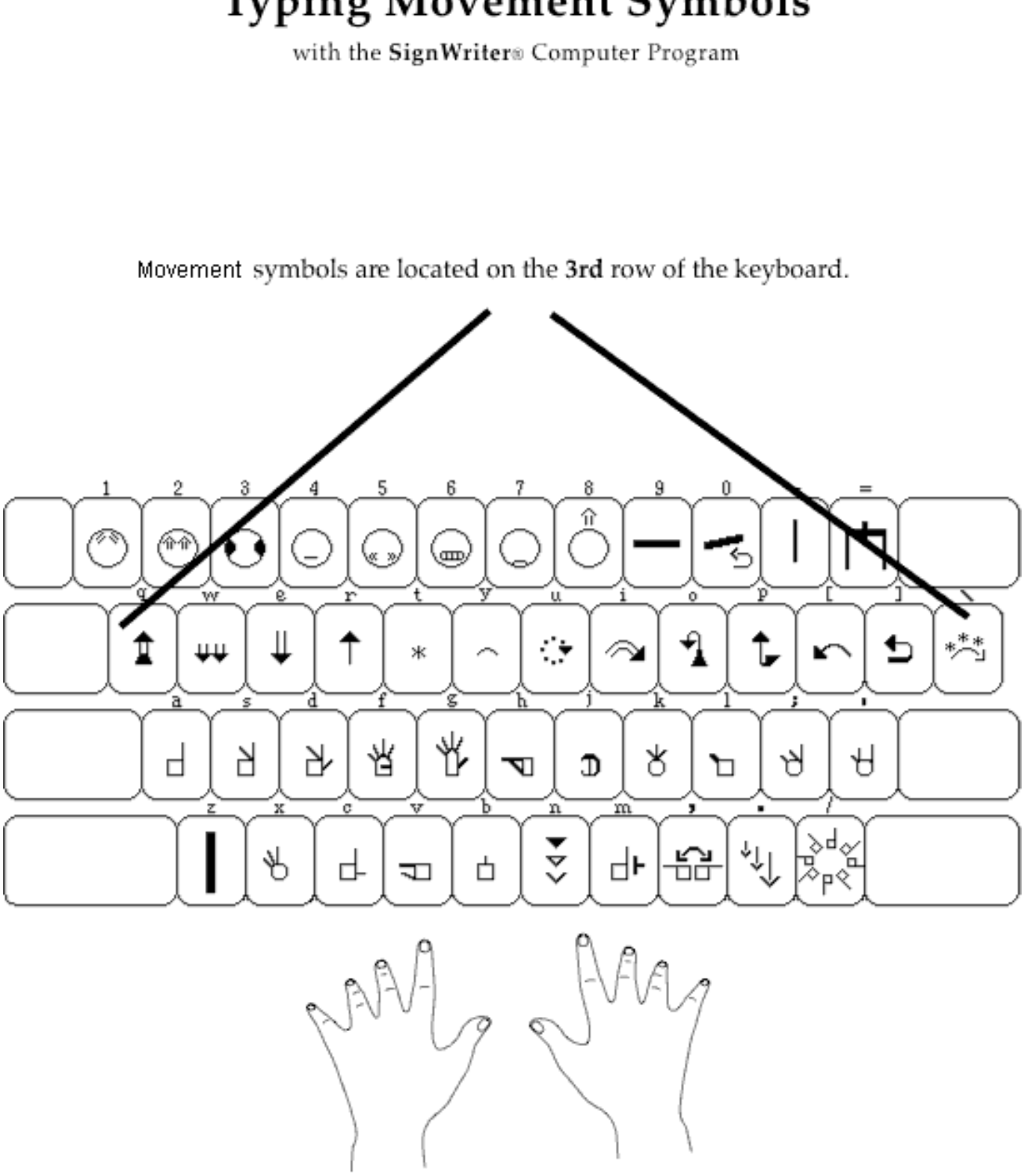
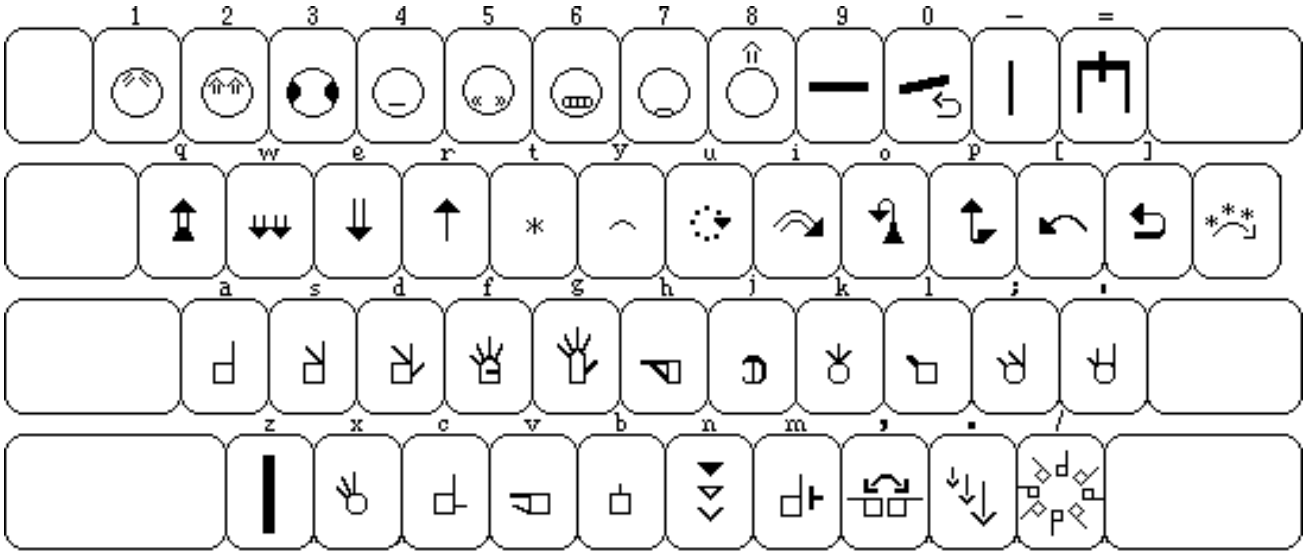
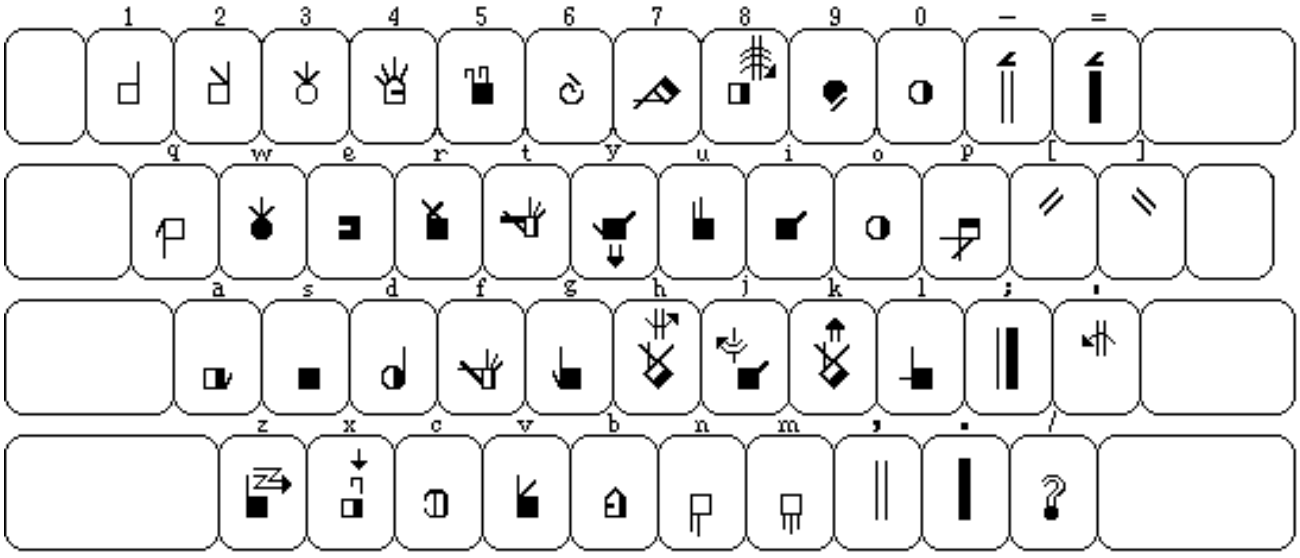


Figure 2: A page from the SignWriter-At-A-Glance-Manual. Symbol categories are placed in rows of keys.



Sign Keyboard
All Countries



Fingerspelling, Brazil
Country Code: 055

Figure 3: A page from the SignWriter-At-A-Glance-Manual. 17 countries with 17 fingerspelling keyboards.