▶ HARDWARE/SOFTWARE/BY DAVID D. THORNBURG

SwyftCard makes an old dog—the Apple IIe—do new tricks

Will Rogers once said that anyone can make things complicated; it takes real genius to make something simple. This line of his came to me as I started using SwyftCard from Information Appliance, Inc.

SwyftCard is a small, multipurpose circuit board that plugs into slot 3 on an Apple IIe, turning it into one of the most useful tools you could ever want for word processing, information retrieval, calculation, BASIC programming, and—if you have a modem—communication. Little did I know that the "lowly" 6502 microprocessor at the heart of the Apple IIe was capable of feats I had to see to believe.

For example, SwyftCard's word-processing program lets you make the cursor jump to any location in a 40K document in a second or two, with only a few keystrokes. It lets you save a document of this size to a completely unformatted disk in less than eight seconds. It lets you perform computations in the middle of creating text without having to pull a calculator down from some menu (or out of a desk drawer). It lets you send documents to or receive them from remote computers with a single keystroke.

The key to this product's utility lies partly in the concern its designers had for the interaction between the computer and its users. Most people who use computers in their work use them to generate text-based documents, so SwyftCard was designed to make this task as easy as possible.

When you turn on an Apple IIe that contains SwyftCard, the disk drive runs for a second to see if it contains a disk. If the drive contains a program disk (your favorite game, for example), it assumes that you want to play that game instead of using SwyftCard, so it boots

the disk, and SwyftCard quietly sits in the background. If the disk contains a SwyftCard document, that document automatically loads into memory instead. If the drive doesn't contain a disk, you get a clear screen for creating a new document.

Documented Evidence

The SwyftCard word-processing program offers no formatting commands—it's actually a what-you-see-is-what-you-get text editor—although it does have automatic word wrap at the end of lines. To add text to the screen, you merely start typing. To correct errors as you make them, you press the Delete key. We don't always catch errors as we make them, though, so we need to be able to find them later to correct them.

If you are used to word processing in AppleWorks or MacWrite, for instance, you are familiar with the notion of using Control-key sequences to manipulate the cursor or the mouse to move the cursor to a desired location on the screen. For particularly large jumps, you may have to enter a Search mode in which you type a representative amount of the text to which you want the cursor to jump. A lot of research on the use of cursor-control keys and mice as vehicles for cursor movement shows that these are effective ways to move around within a document. SwyftCard provides a new method of cursor movement, however, that leaves these old tools in the dust.

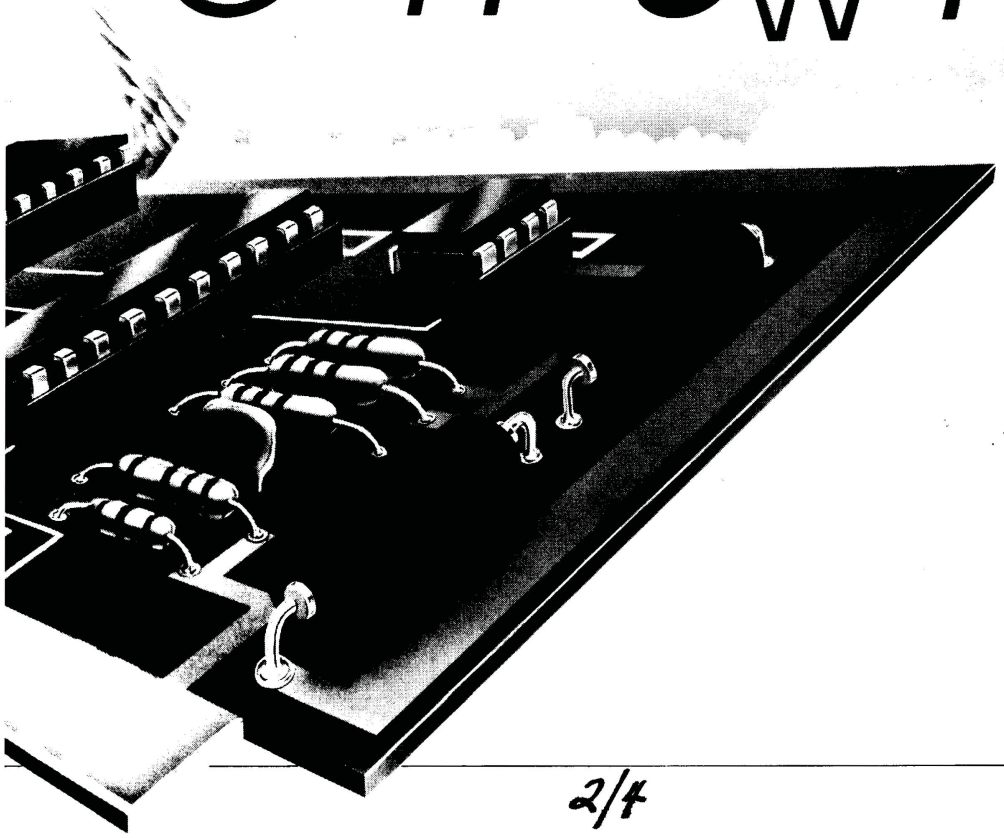
Leaping to Conclusions

The people at Information Appliance have developed a new way to move the cursor to a target, a process they call "leaping." Leaping takes advantage of the keys at each end of the Apple IIe space bar—the open-apple and closed-apple keys. The closed-apple key (on the right side of the space bar) means "leap forward," and the open-apple key (on the left side of the space bar) means "leap backwards."

But "leap forward" needs a target—a place to which to leap. You identify this target by holding down the appropriate leap key and typing the character sequence that starts with the target letter. For example, suppose I want to position the cursor at the letter *u* in the word *But* at the beginning of this paragraph. To do that from this location in the text, I hold down the open-apple key and type the letters *u* and *t* and press the space bar. These three keystrokes are enough to define uniquely the location of the *u* that I want.

What happens as you leap is that the letters you type while you're holding down a leap key form a character string, and the cursor automatically jumps to the nearest occurrence of this

TO The SWYFT



ADRIAN DAY

2/4

string in the leaping direction. For my example, when I typed *u*, the cursor jumped back to the first *u* it could find. When I added the *t*, it jumped to the first *u* followed by a *t*, and adding the space was enough to define the *u* I wanted to reach.

Leaping is easy to master, especially since it's just like typing. For example, if you want to scroll through a long document and you have used double Returns to separate paragraphs, you can leap to this pattern to jump from paragraph to paragraph. You can use the Tab key with either apple key to repeat the last sequence of characters. In SwyftCard terminology, the Tab key (when you use it with the apple keys) is called Again.

Selective Leaping

Leaping may get you from where you are to where you want to be, but sometimes you need to manipulate a block of text. To work with a large amount of text, you have to select it first. Selecting text with SwyftCard is easy. Leap to one end of the selection and then to the other—then press both the open- and closed-apple keys simultaneously.

Once you've selected a block of text, you can delete it with the Delete key. Deleted text is saved in a buffer, and you can reinsert it as many times as you want in any location in the document.

The Insertion command is one of six

THE MAN BEHIND INFORMATION APPLIANCE

Jef RaskiN

Although the SwyftCard is the first product Information Appliance has released, it is far from the first product its founder, Jef Raskin, has designed.

Raskin has been interested in computing since he was a child—one of his high-school science-fair projects was a computer, in fact. He was introduced to Apple Computer in its early days when he worked as a reporter for *Dr. Dobb's Journal*. When he first saw the Apple I computer, he thought that its fledgling maker needed a manual for its new product, so he contracted to write it. The more he got to know his new client, the more he felt that Steve Wozniak was on the right track with his hardware, although Jef thought it was wrong to build a computer that accepted only uppercase letters.

Raskin's own company, Bannister and Crun, wrote manuals and books. His writers used Poly 88 personal computers (which did have uppercase and lowercase characters) with a word-processing program he had written. In 1976, his was probably the first company to use a microprocessor-based word-processing program.

Soon Raskin joined Apple full-time to work on a variety of projects. In 1979 he created a 400-page document that described the hardware, software, and marketing specifications for a computer named after his favorite apple, the McIntosh.

What made Raskin's vision of the Macintosh different from any computer built up until that time was that the software and hardware were designed to augment each other. Earlier computers, in contrast, were designed from the hardware perspective. Unfortunately, once he and the Macintosh project parted ways, the software effort became separate from the hardware. From Raskin's perspective, the result was some problems for the Mac—its sluggishness, the need for a second drive, and the large memory requirements of most applications.

Another Chance

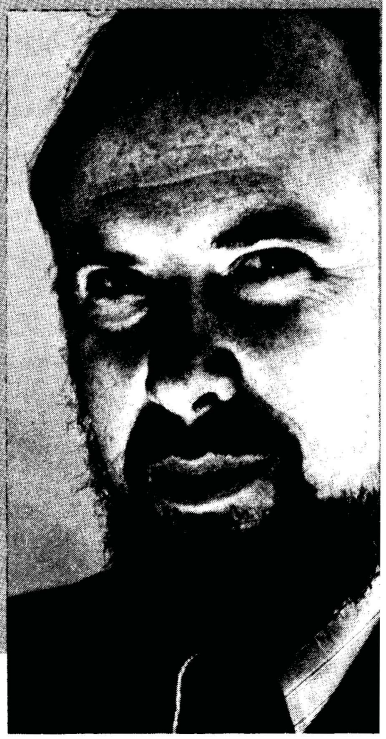
Visions die hard, and Raskin was not one to give up on something he really cared about. He created a new company and called it Information Appliance, a name that implies an ease of use not normally associated with today's computer systems. SwyftCard seems to be a step in that direction, though, and Raskin predicts that within three years some of the ideas behind SwyftCard will be pervasive throughout the industry. The SwyftCard design was influenced strongly by a set of 40 human-factors-design criteria that made this product easy to use.

Although Raskin's professional life seems to center around computers, he has many other interests: He built an electric car when high-power transistors allowed him to try out a new idea for a high-current power supply. He is an avid participant in the field of model-airplane construction and equipment design, and an accomplished musician on a variety of instruments.

Positive Thinking

Looking at Raskin's interests, you wonder how he finds time to fit in all his activities. The key for him is getting over people's negative comments such as "You can't do that" and "It can't be done." He feels that if you accept other people's negative advice, you will only hold yourself back. For example, for 20 years he had wondered what a piano would sound like with only one string per note, rather than the multiple strings that he was told all pianos "must" have. He finally bought an old piano with the goal of having it modified but had a hard time finding a piano technician who was willing to make the modification for him. Finally, he found someone to do the work, and the result was an instrument that produced a delicate and lovely harplike sound. An idea he had been carrying with him for 20 years finally bore fruit.

Jef Raskin keeps good track of his vision. He has eliminated a built-in sense of reluctance many of us have to try things that "can't be done." He believed he could build a high-performance computer system that needed only a handful of commands, and lots of people told him it couldn't be done. The existence of SwyftCard shows that Jef Raskin's positive thoughts were right, after all.



3/4

special commands in the SwyftCard system. One of the more intriguing of these is the Disk command (Control-W). Rather than having separate commands for reading, formatting, and saving information on a disk, SwyftCard has just one command and relies on context for everything else.

Amazingly, this approach works. For example, suppose you have created a document, inserted a blank disk in the drive, and pressed the Disk command. SwyftCard knows that the document

▶ The designers of SwyftCard have enabled a single keystroke to replace a whole family of commands in other word processors.

hasn't been saved yet and assumes that your intention is to do so. It also notices that the disk is blank, so it formats the disk and saves the document—all in about eight seconds!

A Single-Keystroke Family

Now, suppose you place another document disk in the drive and give the Disk command again. In this case, since the system has saved the original document, it assumes that you want to see the contents of the document on the disk, so it loads the document automatically. Through the simple application of common sense, the designers of SwyftCard have enabled a single keystroke to replace a whole family of commands in other word-processing programs.

SwyftCard formats each disk so that it contains only one document (up to about 40K long), and therefore you don't need filenames or a directory. When you save documents, the system saves them along with the location of the cursor and any selected text. For example, when you reload the contents of the document, any selected text remains selected, and the cursor reappears in the position it was in when you saved the document. Another SwyftCard command, Get, retrieves selected text from one document and merges it with text already on the screen.

Farewell, Sweet Prints

To print material, you must select it first (selecting an entire document takes only three keystrokes), and then a single keystroke sends the selected text to the printer. Because Information Appliance could not anticipate all the printers that might ever be connected to the Apple IIe, the designers provided a technique for letting individual users establish their own printer settings. These settings become part of a saved document, so you don't have to reenter them by hand every time you use the system.

Getting Down to BASIC

SwyftCard documents consist of text, but how that text is treated is up to the user. Using text purely as a document is not the only possibility. You can type BASIC commands or programs

▶ SwyftCard was the result of extensive thought about how people might want to use computers if they had a choice in the matter.

into a SwyftCard document and make them execute with the BASIC command. The obvious application for this feature is to use SwyftCard as a tool for writing BASIC programs, but it is much more useful than that.

For example, suppose you want to create a document containing a catch phrase you like. You might assign this phrase to a character string by entering

```
P$ = "the Wonderama Symposia "
```

By selecting this text, entering the BASIC command, and then pressing the Delete key (to delete the string definition from the screen), you can create your document by typing I would like to introduce you to ?P$, the most exciting. . .

Once you've created the document, you can select each of the occurrences of ?P$ and execute the BASIC command, thus causing automatic entry of the desired text.

Another useful application for this command is to perform calculations in the middle of a line of text (for example, ?45.68+25.87). The BASIC command ?, which precedes the numbers, prints the result of the calculations.

Send It Simply

The remaining single-keystroke command that SwyftCard offers is the Send command. If you have a communications interface card (modem) in slot 2 of your Apple, pressing Control-C sends highlighted text to your modem in slot 2. Any incoming text automatically goes into your document. Entire documents can transfer back and forth between users with just a few keystrokes, all without any additional communications software.

The Package

Along with SwyftCard, you receive a floppy disk, a set of decals for labeling the commands on your keyboard's keys, and an instruction manual. One side of the diskette contains a file-conversion utility, similar to Apple's ProDOS-to-DOS conversion program. It allows you to convert a ProDOS file to a SwyftCard file, and vice versa. The other side of the diskette contains an easy-to-use on-line tutorial.

Preferences

SwyftCard was the result of extensive thought about how people might want to use computers if they had a choice in the matter, and the result is a spectacular piece of programming. My only complaint is that there is no mnemonic connection between the SwyftCard commands and the keys that invoke them. You do get decals for the key tops, however, and can even reassign the keys you use for various commands.

Accomplishing the Impossible

SwyftCard has accomplished something that I never knew was possible. It not only outperforms any Apple II word-processing system, but it also lets the Apple IIe outperform the Macintosh. For example, relocating a paragraph in a MacWrite document and saving the result on disk can take 35 seconds. The same task with SwyftCard (saving to an unformatted disk!) takes only 12 seconds.

Will Rogers was right; it does take genius to make things simple. ✚

Dr. David Thornburg has been active in the field of personal computers since 1978. He is the inventor of the KoalaPad touch tablet and is the principal designer of Muppet Learning Keys. He has written more than ten books on computing.

▶ PRODUCT INFORMATION

SwyftCard
Information Appliance
530 University Avenue
Palo Alto, CA 94301
(800) 982-5600
in CA (800) 562-7400
List Price: $89.95 plus shipping and handling
Requires: Apple IIe, one disk drive

4/4