$2.95 US
$3.95 CANADA, FOREIGN
DECEMBER 1986
VOLUME 4/ISSUE 12

Apple II/Macintosh

THE INDEPENDENT GUIDE TO APPLE COMPUTING

IIGS UPDATE
Includes Comprehensive Chart of New Products

THE IIGS COMPARED
How It Stacks Up against the Competition

THE IIGS TOOLBOX
Exploring the IIGs's Mac-Like ROM

OTHER ROUTES

Alternatives to Using AppleWorks

- Stand-Alone PROGRAMS
- Integrated PROGRAMS
- Modular PROGRAMS

ENCHANTED PRINTS
Turn Your Apple II into a Print Shop

APPLEWORKS ADD-ONS
ThinkWorks and ReportWorks

THAT'S ENTERTAINMENT
Special Expanded Holiday Edition

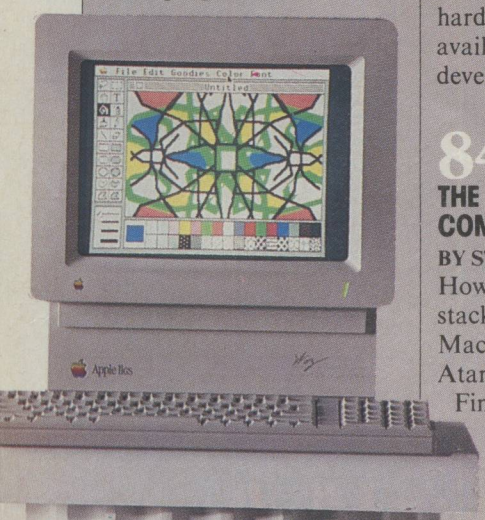
12
0 71486 02401

Contents

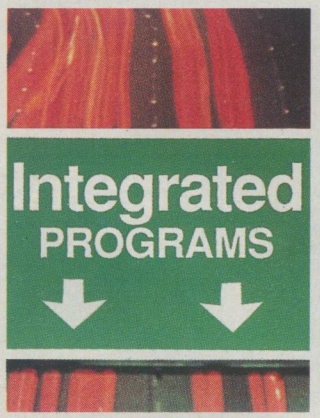
DECEMBER 1986/VOLUME 4/ISSUE 12

A+ THE INDEPENDENT GUIDE TO APPLE COMPUTING

IIGS UPDATE



COVER STORIES



SOFTWARE

BOOK EXCERPT *BY DANNY GOODMAN*

IIGS
UPDATE

USING THE APPLE IIGS TOOLBOX

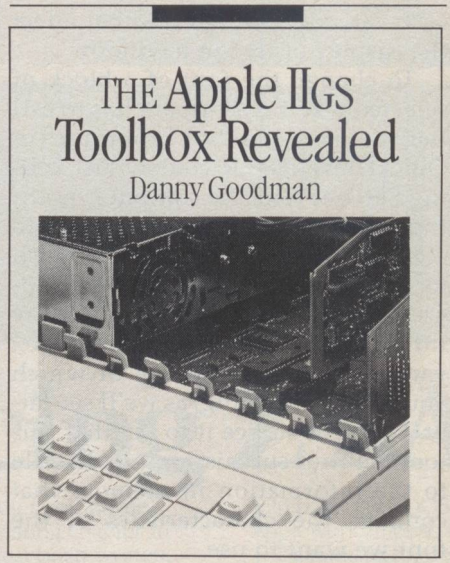
In last month's issue (pages 77–85), we took a first look at the programmers' toolbox of the Apple IIGS. Now we'll discuss the link between your program and the tools. We'll be talking in generic terms because the specifics of incorporating tool sets into programs vary slightly from language to language. Fortunately, the concepts are similar, especially among the languages in the Apple IIGS Programmers' Workshop.

A IIGS language usually comes with two sets of toolbox-related files. One set is source code, the other object code.

The source-code files contain many predefined variables and data structures carrying readily identifiable names that you can begin using in your own source code. They will save you from declaring the same toolbox variables over and over in each application. Moreover, you are assured that they are done correctly.

You can use these predefined variables in your source-code listing, provided you instruct the compiler to incorporate the external source files into your source code at compile time. To do so, you place instructions for the compiler (compiler directives) at the top of your source-code listing to include or use as many of those source files as you need. The files are typically grouped according to tool set, making it easy to specify those files to be merged into your program.

The object-code files that come with the language contain the actual routines that make toolbox calls possible in your program—something the core compiler does not furnish.



Therefore, the linker links your program's object code with as many tool-set object modules as you direct in the command to start the linker. The result will be a load file that makes the appropriate calls to the Tool Locator each time a tool is requested as the program runs.

The Include or Use instructions are simply assembler or compiler directives. They do not represent the calls to the toolbox routines while the program runs. You place actual calls to the toolbox throughout the program as needed.

CALLING A TOOL FUNCTION

Apple has documented the toolbox calls in full detail in a two-volume set called *Apple IIGS Toolbox Reference*. A call to a toolbox routine, as listed in these references, looks like any statement that might be a part of a programming-language vocabulary. Here are some examples of tool calls you might make to the Window Manager:

NewWindow	SetWTitle
CloseWindow	SelectWindow
DisposeWindow	

Most of these statements are in plain language, although occasionally a tool call is abbreviated. SetWTitle, for example, is short for "Set Window Title." Notice, however, that all tool calls are single words. This restriction is for the convenience of the compiler, since compilers find single-word commands easier to recognize than those consisting of multiple words.

Most languages try to adhere to the vocabulary of tool calls defined in Apple's reference material, but that's not always the case. In fact, you may encounter languages, particularly assembly languages, that have different ways of making tool calls. Instead of using the tool-call vocabulary as is, the assembler may require you to precede the call with an underscore character, like this:

```
_NewWindow
```

The underscore is for the convenience of the assembler: It recognizes any word beginning with an underline as being a toolbox call. You may also find languages that use slightly different words for some toolbox calls. When this happens, the new vocabulary words are similar enough for you to make an immediate connection between the new words and the ones defined in Apple's documentation.

Jumping to the Toolbox

In case you're wondering what happens inside the computer when

you make a toolbox call, here is a synopsis of the procedure.

Typically, your program will be following a list of instructions that you write (although converted into machine language). The instruction pointer will be wildly directing the microprocessor to follow instructions from your program loaded in the perhaps tens of thousands of RAM addresses. When the microprocessor encounters a toolbox call, the instruction pointer jumps to the address of the toolbox routine (perhaps in ROM). As soon as the toolbox routine is complete, the instruction pointer returns to its jumping-off spot in RAM and continues working its way through your program instructions (see figure 1).

If you've had experience programming in any language, you will recognize this methodology as that of a simple subroutine from your main program. In this case, however, you don't have to write the subroutines, since they have already been designed and optimized for you. Nor do they take up any disk space in your finished program file.

PASSING PARAMETERS

Toolbox calls are occasionally self-contained, action-oriented functions. An example is one called HidePen. When you issue this tool call, it unilaterally turns off the drawing pen on the screen. When you issue that statement, the tool simply does its action and returns control back to the program—the tool has nothing to report back to

RELATIVELY FEW TOOLS ARE FREESTANDING FUNCTIONS.

the program. The vast majority of toolbox routines fall into one of three categories, however:

1. they require input
2. they generate output
3. they require input and generate output

Let's look at an example from each category, using calls that affect the display of text in a window.

To change the font of a block of selected text, the program has to call the SetFont toolbox routine (in QuickDraw II). Of course, just calling SetFont tells the computer nothing, since somehow we need to convey the particular font we wish to set. That information is considered input to a toolbox call. The way information is passed to the toolbox varies with the language in which you're programming, as we'll see later. For now, suffice it to say that this tool requires submission of a handle to the information in memory that contains the characteristics of the font we want to use.

The opposite occurs when our program needs to know what the current font is. To obtain that information, the program uses the GetFont tool call (also in QuickDraw II). In this case, we have no input for the tool, because it assumes we want to know the current font. When the tool routine has finished, however, the tool will need to give us information, its output. Again, the way this output reaches our program depends on the language in use, as we'll see.

The third category consists of tool calls that require input before running and produce output when they're finished. An example of such a call is StringWidth. This call needs to know the location in memory of the text string that the tool is to measure. When the tool has measured the desired string, the result is then produced in the form of an integer that corresponds to the number of picture elements, or pixels, on the screen the string occupies.

As you begin to program for the IIGS, you'll quickly see that most of the tools fall into the types that involve input, output, or both. Relatively few tools are freestanding functions.

PARAMETERS AND THE STACK

The best way to illustrate the way parameters pass to and from toolbox routines is to examine what happens to the stack during a toolbox call. Only assembly-language programmers will have to bother with direct stack manipulations. High-level-language programmers will have the impression of using other means of passing parameters; in reality, the load file generated by a high-level compiler and linker uses stack mechanics, just as assembly-language programmers do. Everyone, therefore, can benefit from this explanation.

Input Parameters

When we called SetFont, above, we had to pass the handle to the font we wanted as the current font. To do this procedure with the stack, an assembly-language program starts by pushing the handle onto the stack (see figure 2).

Since the stack gets smaller and the contents of the stack pointer (SP) decrement when an item is pushed onto the stack, the illustration makes perfect sense.

Once the parameter is on the stack, the program can call the toolbox routine. When the routine runs,

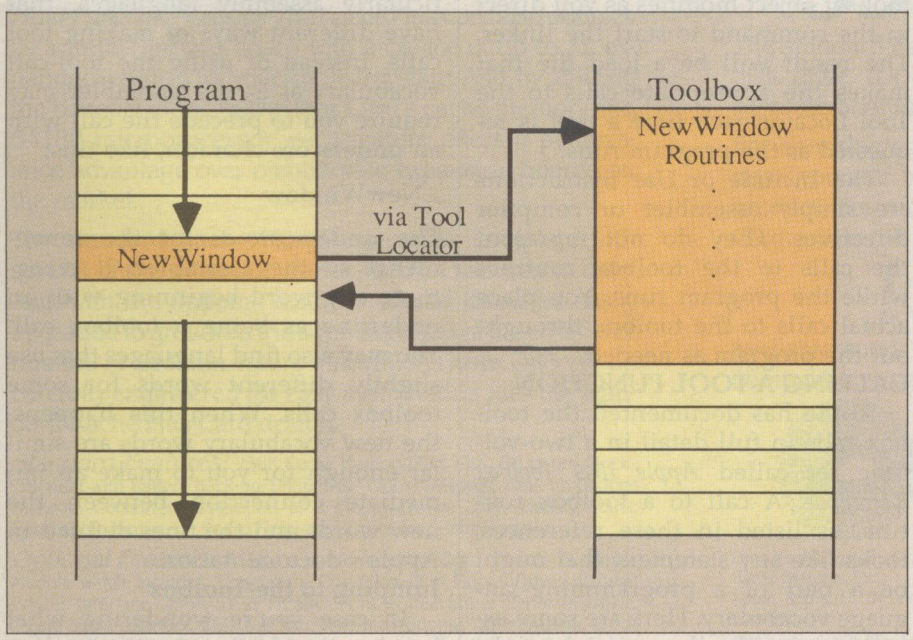


Figure 1: Inside a toolbox call

it automatically looks to the stack for the information it needs—a handle in this case—and pops it from the stack without any intervention from the assembly-language program (see figure 3). Once the routine is finished, control of the program returns to the assembly instructions, and the stack returns to its previous status.

Many toolbox calls require that more than one parameter be pushed onto the stack before being called. Parameters, such as a sequence of pointers and table arrays, can be of unequal length. When a toolbox call expects multiple parameters, those parameters must have been pushed onto the stack in the proper order so that the tool will pop them in the right order. For example, if you push first an integer and then a pointer onto the stack, the tool must expect to pop a pointer and then an integer from the stack—the reverse of the order in which they were pushed onto the stack. If the tool expects a two-byte integer and instead pulls half of a four-byte pointer, then the tool will surely fail and cause a system error. The *Apple IIGS Toolbox Reference* manuals detail the order of multiple parameters for each toolbox call. Observe parameter order religiously.

Output Parameters

For tools that don't require input

OBSERVE PARAMETER ORDER RELIGIOUSLY.

but emit output, we have an entirely different methodology for the stack. Just as a tool knows how to quietly pop an input parameter from the stack, so too does it know how to push an output parameter onto the stack—sort of. The qualification is that the program must specifically request that space be set aside on the stack for the output that is to come from the toolbox call.

For example, if our program is about to call the GetFont tool, which sends as its output the handle to the current font, we must make room for that handle on the stack before calling the tool.

In this case, we must make room for four bytes of data, since the handle coming back will be four bytes long (see figure 4). The empty space usually consists of zeros. The importance of this procedure is that the stack pointer must decrement so that the tool won't overwrite important stack data with the output. If a tool is to output multiple parameters, then you must reserve enough space for them all on the stack.

When a tool supplies output as its result, it is said to return a particular kind of data. For example, in good programmers' jargon, GetFont "returns" a handle to the current font. Get used to hearing *return* as a way of identifying a tool's output.

Input and Output

Stack manipulation for a tool that takes input parameters as well as returning output parameters is only slightly more complicated, in that it combines the actions of the two individual actions. To demonstrate, we'll use the StringWidth tool call described earlier. An assembly-language programmer has to plan the actions of the tool before calling it. Since the tool will return an integer representing the pixel width of a text string, the stack must have enough empty space available for an integer that the tool will push. Of course, the pushing happens after the tool has popped the pointer to the text string from the stack. In other words, the order in which the assembler program must push the space and pointer is (1) the space for the output and (2) the pointer to the text string (see figure 5).

Then the assembler program can call the StringWidth tool. The tool-

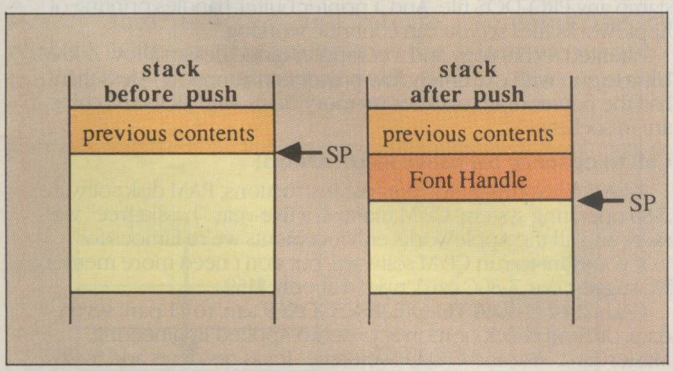


Figure 2: Pushing a handle onto the stack

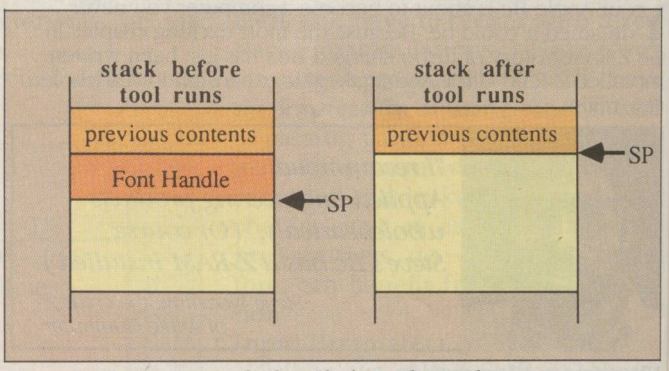


Figure 3: The tool pops the handle from the stack.

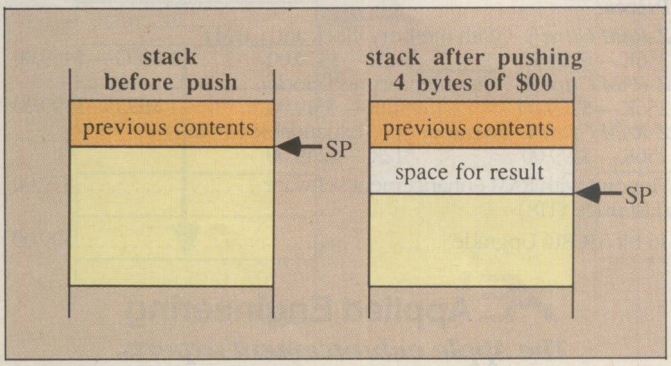


Figure 4: Preparing the stack for output

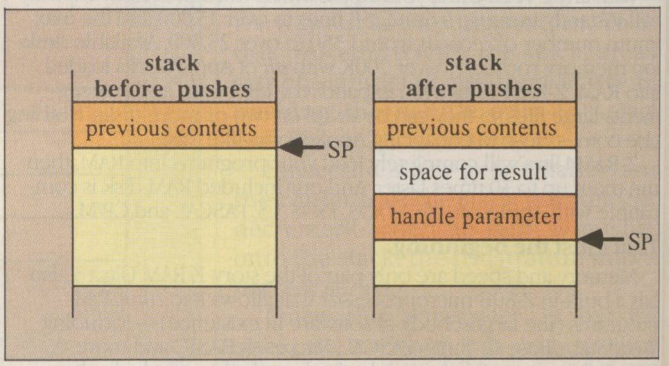


Figure 5: Pushing a space and a parameter onto the stack

box routine automatically pops the string pointer from the stack and writes the resulting pixel-width integer into the space remaining for it on the stack (see figure 6).

Again, by preparing the stack prior to a tool call, an assembly-language programmer (or a high-level-language program under direction of the compiler) is able to lay in any number of blank spaces and input parameters for complex tools that have multiple input and output parameters.

High-Level Parameters

Although a high-level compiled language usually doesn't bother with stack manipulation, you will still have to furnish input parameters and know how to obtain output parameters after IIGS toolbox calls. Pascal and C are remarkably similar to each other in working with toolbox routines that accept and return parameters. Therefore, we'll generalize here a bit to give you an overview of the mechanics of handling parameters in these languages.

Both languages have a syntax that lets programmers assign the value returned by the tool to a variable name. You attach input parameters to the toolbox call by placing them in parentheses immediately following the call in the program listing. Except for minor variations in punctuation, the following listing shows how the StringWidth toolbox call might look inside a high-level-language listing. The call measures the width of a text string whose location in memory has been identified earlier in the program as a pointer called txt:

```
width = StringWidth(txt);
```

Behind the scenes, the language places the pointer (named txt) on the stack, calls the StringWidth routine, and places the width integer on the stack. The language goes one step further by assigning the value of the width integer to the variable called width. Once the language has run this program function, it can use the variable name width in calculations such as determining the center point of the text string so that the text can be centered in the window before display of the string.

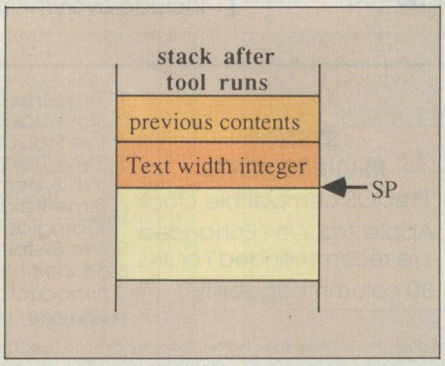
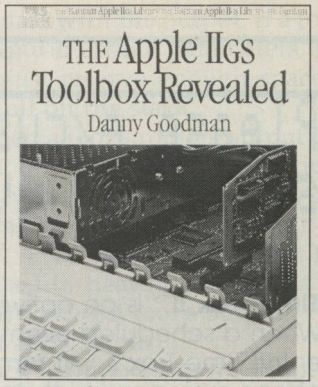


Figure 6: This toolbox routine leaves only its output on the stack.



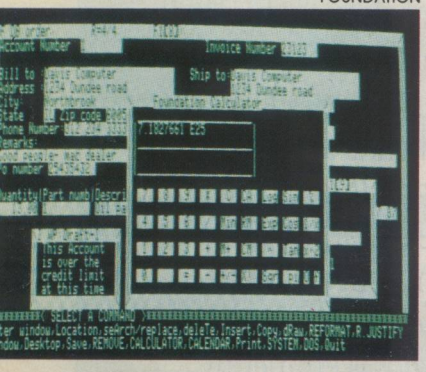
Apple IIGS Toolbox Revealed, *©1986 by Danny Goodman, published by Bantam Books, Inc.*

Toolbox calls that do not return output are often called procedures in a high-level language. Tool calls that return output are called functions. Both procedures and functions can pass input parameters, but only functions get output parameters in return. Your language manual should demonstrate ways of nesting procedures and functions to help reduce the number of lines of source code your programs require. For example, if your program has only one instance in which you need the value StringWidth returned for use as an input parameter in a different toolbox call, you can skip the step of defining the width variable. Instead, you can use the StringWidth(txt) function, itself, as an input parameter. Its returned value (the width) will be passed directly to the other call. ✚

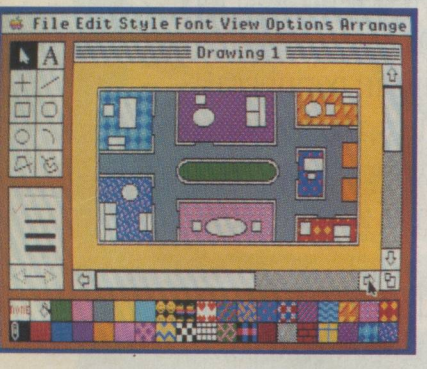
Danny Goodman has been using, programming, and writing about personal computers since the late 1970s. He is the author of nine books, including Going Places With the New Apple IIc, *which has been translated into four European editions, and* The Idea Book For Your Apple II. *"Using the Apple IIGS Toolbox" is adapted from* Apple IIGS Toolbox Revealed, *©1986 by Danny Goodman. Used by permission of Bantam Books, Inc. All rights reserved.*

ECHO IIb

FOUNDATION

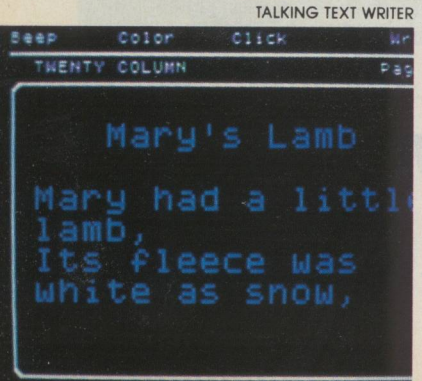


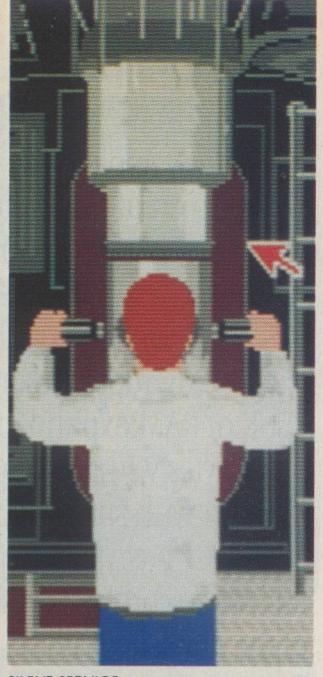
TOPDRAW



PROAPP 40S AND PROAPP 20

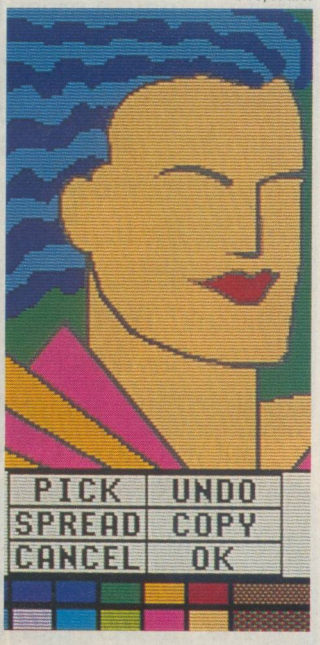
TALKING TEXT WRITER



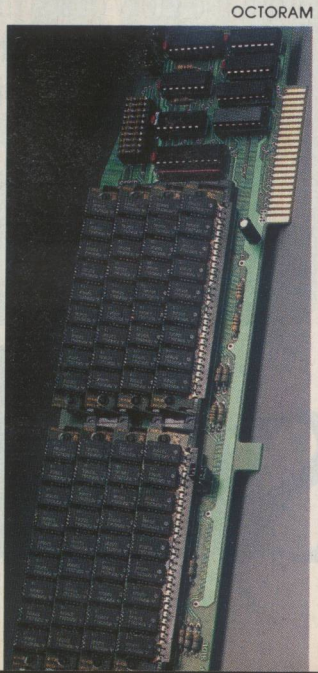


SILENT SERVICE

816/PAINT

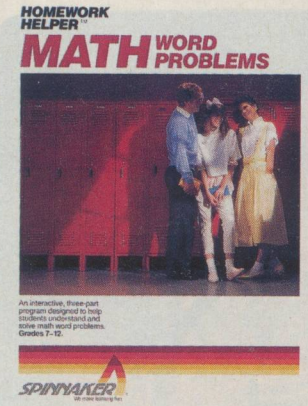


OCTORAM

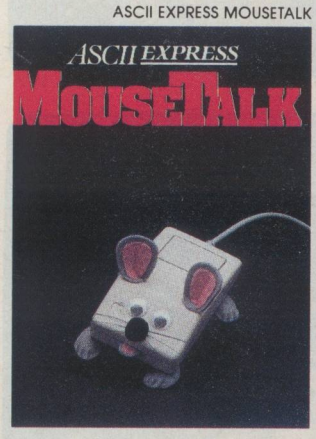


PUPPY LOVE

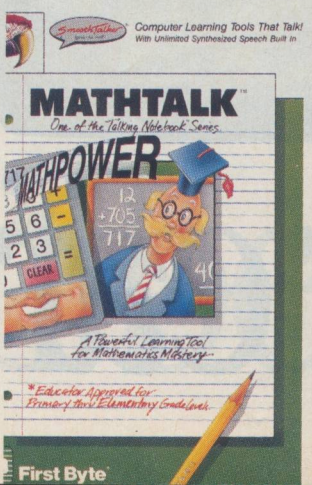
HOMEWORK HELPER

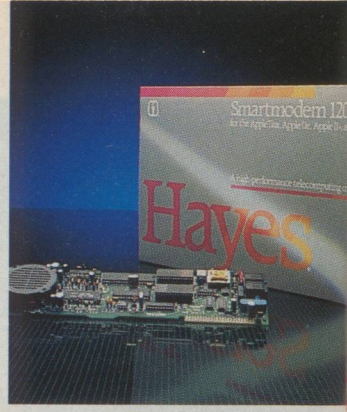


ASCII EXPRESS MOUSETALK



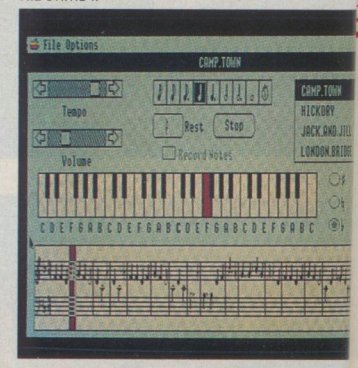
MATHTALK



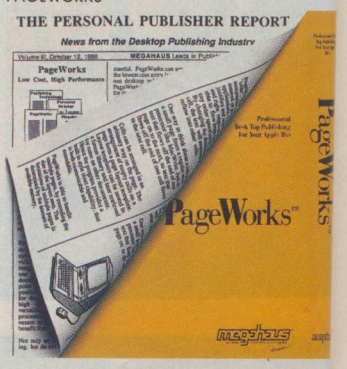


SMARTMODEM 1200A

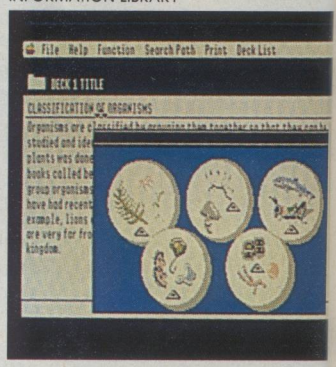
KIDSTIME II



PAGEWORKS



INFORMATION LIBRARY



BOSE ROOMMATE

BY CAROL PERSON AND LISA RALEIGH

IIGS UPDATE

NEW PRODUCTS

A sampling of new products that take advantage of IIGS features—many are available now; others are close behind.

HARDWARE

COMPANY	PRODUCT	PRICE	DESCRIPTION	READER SERVICE NUMBER
Apple Computer, Inc. 20525 Mariani Avenue Cupertino, CA 95014 (408) 996-1010	**Apple 3.5 Drive**	$399	Disk drive with 800K of storage—works with both IIGS and Macintosh	—
	Apple 5.25 Drive	$299	New disk drive that matches "platinum" color of IIGS	—
	Apple II SCSI Card	$129	Interface card—allows IIGS to connect devices using SCSI standard	—
	Hard Disk 20SC	$1299	20-megabyte SCSI hard-disk drive	—
Applied Engineering P.O. Box 798 Carrollton, TX 75006 (214) 241-6060	**GS-RAM**	256K, $169; 512K, $219; 1MB, $299; 1.5M, $379	Memory-expansion card with up to 3.5 megabytes of RAM using 256K chips and piggyback board	—
	GS-RAM Plus	Call co. for price	Memory-expansion card with up to 8 megabytes of RAM using 1-megabyte chips and piggyback board	—
AST Research, Inc. 2121 Alton Avenue Irvine, CA 92714 (714) 553-0340	**AST-2000**	$2795	20-megabyte hard-disk drive with SCSI tape subsystem	502
	RamStakPlus	Call co. for price	One megabyte of memory expansion that uses IIGS built-in RAMdisk features	503
	SprintDisk	$295	Memory-expansion card with one megabyte of RAM	504
	AST-VisionPlus	Call co. for price	Video digitizer that includes special effects such as zoom-in and inverse video	505
Bose Corporation 100 The Mountain Framingham, MA 01701 (617) 879-7330	**Bose RoomMate**	$229 per pair	Compact loudspeakers with built-in amplifiers	506
DataDesk International 7650 Haskell Avenue Van Nuys, CA 91406 (800) 826-5398 In CA (818) 780-1673	**Turbo-101 Enhanced Keyboard**	$149.95	IBM PC-style keyboard	507
Hayes Microcomputer Products, Inc. P.O. Box 105203 Atlanta, GA 30348 (404) 441-1617	**Smartmodem 1200A**	Call co. for price	Internal 1200-baud modem	508
Impulse, Inc. 6860 Shingle Creek Parkway Suite 110 Minneapolis, MN 55430 (612) 566-0221	**(Phoenix)**	Call co. for price	Add-on board with MIDI interface that does audio and video digitizing	509
Kurta Corporation 4610 South 35th Street Phoenix, AZ 85040 (602) 276-5533	**Kurta Graphic Input System**	$495	Digitizing tablet (mouse alternative)	510
MDIdeas 1111 Triton Drive, Suite 205 Foster City, CA 94404 (415) 573-0580	**Conserver**	$129.95	Combination power strip/surge suppressor/fan and monitor platform	511
	OctoRam	1MB, $399; 2MB, $699	Memory-expansion board providing up to 8 megabytes of RAM	512
	SuperSonic	$59.95	Stereo sound and amplifier board	513

This table contains a sampling of available products, as well as some still under development, and the information herein is based on manufacturers' claims. For further information on most items in this table, please circle the corresponding Reader Service Number on the reader-service card in this issue.

COMPANY	PRODUCT	PRICE	DESCRIPTION	READER SERVICE NUMBER
Orange Micro, Inc. 1400 North Lakeview Avenue Anaheim, CA 92807 (714) 779-2772	**JuiceBox**	$119	Power strip/surge suppressor for IIGS and two peripherals, with built-in fan	514
	ProGrappler	$129	Menu-driven parallel printer board. Prints super-hi res screens from IIGS	515
	RamPack 4GS	$259	512K of memory expansion, expandable to 4 megabytes	516
Peak Systems, Inc. 1120 Capital Texas Highway Building 2, Suite 300 Austin, TX 78746 (512) 329-1020	**PLUS20 (30, 45, 65)**	20, $1095; 30, $1395; 45, $1895; 65, $2495	20-, 30-, 45- or 65-megabyte SCSI hard-disk drive	517
	PLUS20T	$895	SCSI tape backup	518
	SIERRA 2040	$2495	20-megabyte SCSI hard-disk drive and 40-megabyte tape backup	519
	SIERRA 3040	$2695	30-megabyte SCSI hard-disk drive and 40-megabyte tape backup	520
ProAPP, Inc. 1475 South Bascom Avenue Suite 101 Campbell, CA 95008 (408) 559-3552	**ProAPP 10, ProAPP 20, ProAPP 40S**	10, $795; 20, $995; 40, $1995	10-, 20- and 40-megabyte hard-disk drives that connect with either the disk port or SCSI interface	521
Street Electronics Corporation 1140 Mark Avenue Carpenteria, CA 93013 (805) 684-4593	**Echo IIb**	$129.95	Speech synthesizer, with external volume control and headphone jack—can be used instead of IIGS internal speaker	522
Thirdware Computer Products 4747 NW 72 Avenue Miami, FL 33166 (305) 592-7522	**FingerPrint Key**	$200	Add-on board that allows users to create macros—includes RS-232 connector into which other devices can be plugged	523
Thunderware, Inc. 21 Orinda Way Orinda, CA 94563 (415) 254-6581	**ThunderScan for the Apple II family**	Call co. for price	Device that scans images into the computer from the printer	524

SOFTWARE

Activision, Inc. 2350 Bayshore Frontage Road Mountain View, CA 94043 (415) 960-6044	**Paintworks Plus**	$79.95	MacPaint-like drawing program	525
	Writer's Choice *elite*	$99.95	Word-processing software with Macintosh-like features	526
	Music Studio	$79.95	Music-composition and -education software	527
	Paper Models: The Christmas Kit	$29.95	Software for making Christmas ornaments and decorations	528
A-Squared Systems 10 Skyway Oakland, CA 94619 (415) 633-0703	**A-LIVE!**	Call co. for price	Plug-in card that captures real-time or still video images	529
Addison-Wesley Publishing Company 2725 Sand Hill Road Menlo Park, CA 94025 (415) 854-0300	**Information Laboratory**	$60	Database of science facts, to be used as research tool for students	530
Addison-Wesley Publishing Company Route 128 Reading, MA 01867 (617) 944-3700	**Puppy Love**	$19.95	Educational/entertainment software featuring a programmable on-screen puppy	531
Baudville, Inc. 1001 Medical Park Drive S.E. Grand Rapids, MI 49506 (616) 957-3036	**816/Paint**	$75	MacPaint-like program that runs in all four graphics modes	532
Brøderbund Software, Inc. 17 Paul Drive San Rafael, CA 94903 (415) 479-1700	**Drawing Table**	$89.95	MacDraw-like structured drawing program—supports LaserWriter and color printing on ImageWriter II	533
	Fantavision	$59.95	Special-effects/animation generator that can incorporate sound	534
	Newsmaker	$89.95	Desktop-publishing program with color drawing—imports text files from AppleWorks and Bank Street Writer	535
	The Print Shop	$69.95	Menu-driven color-graphics editor, for creating cards, banners, letterheads, signs	536
Chancery Software 1120 Hamilton Street, Suite 200 Vancouver, B.C., Canada V6B 2S2 (604) 685-2041	**CSL Marks**	Individual teacher license, $98; school site license, $300	Gradebook program for K–12 teachers that uses a spreadsheet format	537

COMPANY	PRODUCT	PRICE	DESCRIPTION	READER SERVICE NUMBER
Chang Labs 5300 Stevens Creek Blvd. San Jose, CA 95129 (408) 246-8020 (800) 972-8800 In CA (800) 831-8080	**Rags to Riches**	$199 per module; 3-pack, $499.50	Accounting software	538
D.C. Heath & Company 125 Spring Street Lexington, MA 02173 (617) 862-6650	**Explore-A-Story**	School version, $66	Educational software to teach reading and writing for grades K–4	539
DataPak Software, Inc. 14011 Ventura Blvd. Suite 507 Sherman Oaks, CA 91423 (818) 905-6419	**GraphicWriter**	$149.95	Word-processing program that includes MacDraw-like capabilities	540
Electronic Arts 1820 Gateway Drive San Mateo, CA 94404 (415) 571-7171	**Deluxe Music Construction Set**	$49.95	Software for composing, editing, and playing musical scores	541
	Deluxe Paint	$99.95	Painting program	542
First Byte, Inc. 2845 Temple Avenue Long Beach, CA 90806 (213) 595-7006	**First Shapes**	$49.95	Educational program that teaches geometric shapes, for ages 3–8	543
	KidTalk	$49.95	Talking word-processing program for children ages 3–13	544
	MathTalk	$49.95	Math-tutoring program that "speaks"	545
	Speller Bee	$49.95	Talking software that teaches spelling, for ages 5–13	546
Foundation Corporation 506 West Armitage Avenue Chicago, IL 60614 (312) 880-5761	**Foundation**	$250	Integrated software that combines word-processing, spreadsheet, and database applications	547
Great Wave Software 104 Gilbert Avenue Menlo Park, CA 94025 (415) 325-2202	**KidsTime II**	$39.95	Two educational modules: ABKey, a letter-recognition program, and KidsNotes, a music program	548
Haba Systems, Inc. 6711 Valjean Avenue Van Nuys, CA 91406 (818) 994-1899	**HabaCalc**	$49.95	Spreadsheet software that takes advantage of extended memory and speed	549
MECA 285 Riverside Avenue Westport, CN 06880 (203) 222-1000	**Managing Your Money**	$199.95	Financial management for home, professional, or small-business use	550
Megahaus Corporation 5703 Oberlin Drive San Diego, CA 92121 (619) 450-1230	**PageWorks**	$125	Desktop-publishing/page-layout program that can work with LaserWriter and incorporate AppleWorks text	551
MicroProse Software, Inc. 120 Lakefront Drive Hunt Valley, MD 21030 (301) 667-1151	**Silent Service**	$39.95	Submarine simulation	552
Monogram 8295 South La Cienega Inglewood, CA 90301 (213) 215-0355	**Dollars and Sense**	$119.95	Personal-finance software	553
PBI Software, Inc. 1111 Triton Drive, Suite 201 Foster City, CA 94404 (415) 349-8765	**CommWorks 16**	$49.95	Telecommunications program with Macintosh-like features	554
	Visualizer	$99.95	Business-graphics program that converts spreadsheet data into bar, column, pie, line, or scatter charts	555
Roger Wagner Publishing, Inc. 10761 Woodside, Suite E Santee, CA 92071 (619) 562-3670	**Merlin 16**	Call co. for price	65816 assembler	556
	MouseWrite 2.68	$149.95	Word-processing software with Macintosh-like features and print spooler	557
	SoftSwitch	$39.95	Memory-resident desk accessory that allows users to switch between multiple programs	558
Scholastic, Inc. 730 Broadway New York, NY 10003 (800) 325-6149	**Talking Text Writer**	Home version, $249.95; Educator w/Echo+ Board, $199.95; w/o Echo+ Board, $149.95; Lab Pack, $254.95	Word-processing software with built-in speech synthesizer for preschool through 6th grade	559

COMPANY	PRODUCT	PRICE	DESCRIPTION	READER SERVICE NUMBER
Spinnaker Software One Kendall Square Cambridge, MA 02139 (617) 494-1200	**Homework Helper Math Word Problems**	$49.95	Equation-building program for grades 7–12, with built-in calculator	560
	Homework Helper Writing	$49.95	Writing program that guides students through formulating ideas, creating outlines, and writing essays	561
StyleWare, Inc. 5250 Gulfton, Suite 2E Houston, TX 77081 (713) 668-1360	**TopDraw**	$99.95	MacDraw-like structured drawing program that can work with LaserWriter—prints in color on ImageWriter II	562
	MultiScribe GS	$99.95	Word-processing software with Macintosh-like features	563
TML Systems, Inc. 4241 Baymeadows Road, Suite 23 Jacksonville, FL 32217 (904) 636-8592	**TML Pascal**	$125	IIGS version of TML Pascal for the Macintosh	564
Innovision P.O. Box 1317 Los Altos, CA 94023 (415) 964-2885	**Calliope 128**	$59.95	Idea processor that uses a free-form format, rather than an outline approach	565
United Software Industries, Inc. 8399 Topanga Canyon Blvd. Suite 200 Canoga Park, CA 91304 (818) 887-5800 (800) 621-0849, ext. 441	**ASCII Express MouseTalk**	$149.95	Telecommunications software	566
VIP Technology Corporation 2651 John Street, Unit 3 Markham, Ontario, Canada L3R 2W5 (416) 479-1880	**VIP Professional**	$299.95	Integrated spreadsheet, database, and charting software	567
WordPerfect Corporation 266 West Center Orem, UT 84057 (801) 227-4020	**WordPerfect 1.1**	$179	Word-processing software with 115,000-word spelling checker	568

CIRCLE 104 ON READER SERVICE CARD