A2-Central™

formerly Open-Apple

September 1990
Vol.6, No.8

ISSN 0885-4017
newstand price: $2.50
photocopy charge per page: $0.15

A journal and exchange of Apple II discoveries

Straight talk from Apple

Warning: I'm about to give a report on the ***A2-Central Summer Conference.*** I hadn't originally planned to cover the conference in detail; the target audience for the gathering is intentionally small as we couldn't handle all 8,000 of you showing up anyhow. And I had thought the topics discussed would be rather specialized.

But there was a big difference between this conference and the one last year. Not in Apple's participation, because that has been strong both years, but in the attitude of the participants. Last year, many developers perceived they were at odds with Apple, so much so that a few developers formed a committee to lobby Apple for changes they felt needed to be made. This year, Apple had demonstrated that they were listening to the Apple II community during the last year. Apple's Apple II champions Ralph Russo and Jane Lee came to the conference to talk about the recent turn of events and about current plans. And Apple's Apple II engineers arrived loaded for bear.

Warning number two: you are about to get the fairly detailed look at Apple's game plan that many of you are requesting in your letters. But this doesn't include a peek at unannounced products; for this purpose, "unannounced products" means any work-in-progress that Apple isn't talking to the general public about. Anything you read about here will regard items Apple displayed in "open" sessions. Anything discussed during non-disclosed sessions will remain non-disclosed until Apple says differently. Don't bother asking about the things Apple hasn't announced (which is all speculation anyway); it wastes time that would be better spent locating and evaluating existing products.

Marketing the Apple II; what a novel concept! Jane Lee, Apple II Product Marketing Manager in Apple's Apple USA division, spoke about what's been happening recently with regard to the Apple II.

Currently, the majority of Apple II customers are in the educational market, probably due to Apple's recent (and nearly exclusive) marketing emphasis for the Apple II in that area. Apple intends to investigate new target markets as well as find out where it stands among its current customers.

A "partnership packet" was sent out in May, on the heels of John Sculley's address to educators ("Sculley committed to Apple II", ***A2-Central***, June 1990), containing the promised Apple II (and Macintosh) resource guides. In this packet was a response card asking for comments; over 2000 of these cards have been received and reviewed by Apple. Apple intends to send out more such partnership packets in the future.

Additionally, Apple has met with all its resellers, national account representatives, government account representatives, and sales and marketing types to discuss marketing strategy for Apple products, including Apple II products. This was done both to gather input and to communicate continuing support for current products.

In the early fall of this year, the Apple II will be included as part of Apple's corporate advertising strategy (whatever that turns out to be; it may not necessarily include television advertising, for example). That is, Apple II advertising will be done with the "look and feel" of Apple's overall advertising campaign.

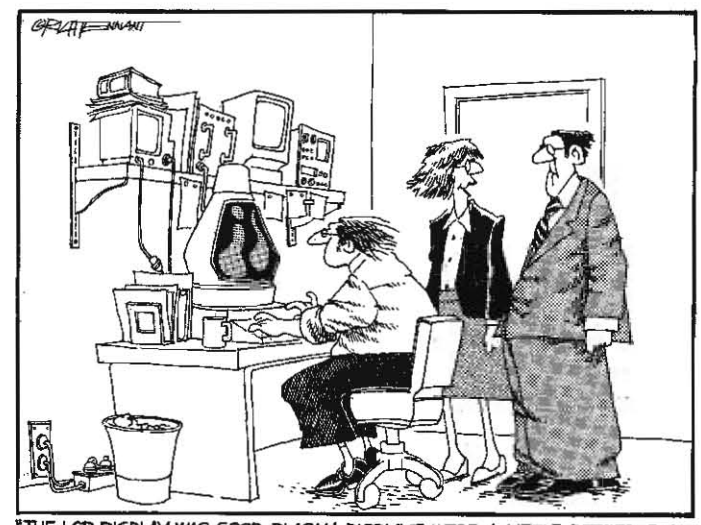
Apple is working on an "Apple Guide" resource similar in concept to Apple France's *Le Guide*. The guide will discuss general topics such as how to choose a dealer (including a listing of selected dealers; user groups are being solicited for opinions on dealers to include), where to find technical and product information, Apple servicing policies, third-party offerings (divided into the categories of user groups, home, home business, and education), and recommendations of customers regarding "dream" systems. User groups may be able to sell the guides at nominal cost as a way to generate some income for the group.

Apple is investigating the feasibility of conducting regional "fairs" (mini-AppleFests, if you will) in conjunction with user groups and regional Apple offices. The Apple Fiesta ("Apple speaks (and listens)", August 1990) is probably a prototype for this type of regional fair, and I look forward to seeing more of these types of gatherings.

Ralph Russo's "charter" has been expanded beyond engineering to look at evangelism, worldwide marketing, and other possible aspects of making the Apple II more profitable. Russo spoke at the meeting of the Apple II Developer's Association at the conference.

Russo was hired 7 years ago by Steve Jobs as the Mac line was starting up. He has served as Apple's Director of Materials as well as under Michael Spindler in International Operations and as Vice President of Apple WorldWide under Del Yocam. That is a strong combination of credentials.

He said his appointment to a full-time Apple II position two months ago was a "very, very good opportunity to do something for Apple as well as for the customer base that has supported us for the last 12 years". He also said that while he could not comment on specific products, he was not going to ask people to be patient. His Number 1 priority is to answer the question "What will Apple do with its Apple II business?" by the end of the summer; "...(the) timing has obviously got to happen fairly quickly". His group (including himself, Jane Lee, and Fern Bachman, head of Apple II engineering) is in the process of putting together a business plan based on the marketing avenues that are available, products that fit the available channels, developers'



"THE LCD DISPLAY WAS GOOD, PLASMA DISPLAYS WERE A LITTLE BETTER, BUT WE THINK THE LIQUID LAVA DISPLAY THAT JERRY'S DEVELOPED IS GONNA ROCK THE WEST COAST."

interest in these products (Apple markets computers, peripherals, and system software and depends on third-party developers to produce other enhancements). The plan will address what can be done with the existing platform. The overall question is "How does Apple Computer, Inc., sell, service, support, and market the Apple II product line?" and Russo wants an answer by the end of this summer.

Russo believes some distribution and marketing ideas for the Apple II haven't been properly evaluated in the last two years. He also related that with Michael Spindler's eye toward global opportunities, the world market is part of the evaluation process. He clarified that a problem in Europe has been that some of Apple's existing channels are through independent marketing companies that are generally not adept at marketing the lower-cost Apple II systems alongside their other clients (including MS-DOS systems). Russo pointed out that the Canadian and Australian Apple II markets, on the other hand, have not been as exclusively education-based as the US market.

He perceives that the strength of the Apple II is that it can "empower individuals." "Individuals" was emphasized (not businesses, educational institutions, and so on). As a marketing strength, he expressed the opinion that "the Apple IIe is basically synonymous with Apple Computer, Inc., ... they appeal to the individual." We perceive part of his task to judge whether Apple's marketing has failed to target the desires of the individual in its attempts to entrench itself in the Fortune 500 business environment.

Given these aims, how does Apple make developers excited about products? How does Apple make individuals excited about the capabilities of its machines? Apple engineers and technical support people as well as other developers spent most of the weekend addressing those questions.

Show developers a rich programming environment. The big news preceding the conference was the release of the third volume of the *Apple IIgs ToolBox Reference*. If you think Apple hasn't been doing anything about the IIgs, notice that this volume has over 1,000 pages of new information.

The *ToolBox References* are just that; references, with a few examples. Apple's Apple II Developer Technical Support group has put together three new disks of sample source code to show you how to create personalized elements for your programs (custom controls, custom windows, and so on) as well as tools showing off other new features of the system software.

One of the more interesting new tools is FakeModalDialog. A IIgs (or Macintosh) "dialog box" is normally employed to interact with the user of a program, either to notify them of some special circumstance (such as verifying that a file should really be deleted), or to gather information before proceeding with an action (such as the "Page Setup" dialog for printout specifications). The IIgs Dialog Manager has been limited in that it doesn't allow the use of resources (one of the radical enhancements to System Software 5.0.2 is the availability of resources—see "GS/OS Resources", Feb. 1990, p. 6.5) and it has only allowed the use of limited types of controls (buttons, check boxes, text fields, and so on).

System Software 5.0.2 added the capabilities of using more varied controls as a part of a new tool call, NewControl2. The Window Manager now also accepts a NewWindow2 command that will allow a broader definition of the window's attributes. If you put these two features together, you can create a window that looks and acts exactly like a "super" dialog box, complete with customized controls. And since the Window Manager supports the use of resources, the "fake dialog" can use resources to define its elements.

System Software Engineer Jim Mensch remarked on this power at the opening of a session on the IIgs tools. One of his opening points was that despite complaints that the Desktop interface is "inflexible," Apple has given developers the tools (and, through sample code, the examples) to customize and personalize the user interface. Mensch said developers haven't taken advantage of their customization options. It is possible to use many of the tools without necessarily binding yourself to the formal Desktop interface. QuickDraw II is designed as a generic graphics toolset. The IIgs Window Manager doesn't care if it's drawing windows with or without a menu bar present. And if the features you want aren't in the system, you can let DTS know that you'd like to see "tool xxx". Apple may not develop it, but if the need is wide enough and it fits within the system software goals, it may be added.

Several of the engineers also emphasized points about compatibility. There are fine points like calling TaskMaster (a Window Manager utility that can automatically handle some common tasks when a "user event" occurs) in your application's event loop; if you don't actually want TaskMaster managing events, then call it with a taskMask of 0. The reason to do this is that TaskMaster might be handling tasks that your application doesn't know about. Calling TaskMaster each time through your event loop assures that TaskMaster will have the opportunity to deal with any pending operations waiting for it. Similarly, it is important to check and make sure that certain features of the IIgs haven't changed before you go charging off to do things on your own. For example, it isn't forbidden to write directly into the Super High-Res screen's video display buffer, but you'd better start QuickDraw and use an OpenPort call to find out if the buffer is really where you think it is. Maybe one day it won't be at $E1/2000. The point is to insure your application will be compatible with future versions of the operating system; this also prevents your application from narrowing the directions in which Apple can enhance the operating system without causing existing software to fail. You can be innovative without being incompatible.

As for future directions for the system software: the engineers feel the implementation of the tools for the Desktop interface is relatively complete. Apple is now looking for more developer feedback on improvements, such as why a tool is not as useful as it could be (why you aren't using it). There will be more enhancements to the toolbox, including as much speeding up as possible without greatly increasing memory demand. It was hinted that future versions of the system software may allow the user more opportunities to configure the system for speed as opposed to minimal memory use.

Features of GS/OS itself that were focused upon are its device and file system independence. This leads to some compatibility issues for applications. GS/OS can use any device that can be electrically connected to the IIgs either through a generated driver or a loaded driver, possibly with a file system translator as part of the interface. That means an application has to avoid assuming anything about the specifics of the device where possible. The file system you are communicating with may not be limited to the ProDOS file naming conventions, volume size limitation, or physical directory layout (this is already true for AppleShare volumes). You can't pre-parse a filename to see if it is legal (you should test the name out on the target volume and see if the FST gags on it) and you can't read directory blocks to get directory information (GS/OS has a GetDirEntry command that should be used for this). The use of native GS/OS Class one calls (as opposed to the ProDOS 16 compatible Class 0 calls) is strongly recommended. Further comments are in GS/OS Tech Note #4.

Show the world the built-in capabilities of the Apple II. Apple's technicians brought an amazing array of "goodies" with them. At this point we don't know when or how any of them will be made available to the public. The impetus seemed to be on the demonstration of the Apple II's capabilities and to let conference attendees know that Apple has some neat projects in progress.

Jim Mensch demonstrated a two-dimensional animation toolkit he is developing for the IIgs. Mensch typified it not as a tool for "Joe Game Designer" (waggishly referring to such programmers, including himself, as being "elitist scum" that always believe they can find a better way to do something), but as a way for the average programmer to add animation that may be just a bit below "state-of-the-art" to their IIgs program. The animation tool eliminates having to sweat out the low-level details such as counting processor cycles (to determine if the routine will draw fast enough) to add good quality animation to a program.

The toolset will be based on a "stage, character, part" metaphor. Imagine a simulation of a frog hopping through a (non-moving) forest. The static image of the forest is the stage, the frog is the character, and the "script" that the frog follows to move ("hop three positions east") is the part. The animation will automatically be clipped to the defined size of the "stage"; you can operate an animation within a document window on the IIgs desktop, for example. Scenery may be defined in the "foreground" and "background", though foreground

scenery (that a character could move "behind") does exact a performance penalty. Tools for editing the components of an animation will be provided.

The metaphor is intended to be generic for people who want to do animation, but may not know the nuts and bolts of how to proceed. There are features of the system tailored to more demanding programmers; an "external services" library exists that can be called to handle certain common operations (fading from one screen to the next, for example). As one of the external services, the toolkit can generate the (*APW* or *MPW*) assembly language source code necessary to "strike" (draw) a character quickly for use in a program. The startup call for the toolset will allow some configuration options so that you can choose not to load and start some functions, saving time and memory if your program does not require them.

The current version being tested (not ready for release and no release date given) supports 16 characters. Mensch hopes to have doubled or even quadrupled that figure by the time the tool is ready to ship.

The strength of the IIgs is the Ensoniq sound chip. Mark Cecys demonstrated his new MIDISynth tool set with a sample application, SynthLab, that allows using the IIgs to act as a MIDI sequencer and sound generation tool.

MIDI stands for Musical Instrument Digital Interface and is a standardized method for intercommunication of electronic musical instruments (such as synthesizers) and sequencers (a device that can send sequences of commands to other devices through the MIDI interface). For the IIgs, the MIDI hardware consists of an Apple MIDI Interface connected through a IIgs serial port, or a peripheral card with MIDI capability such as the Applied Engineering *Audio Animator* or the PassPort Designs MIDI Interface. MIDISynth uses several underlying IIgs tools (the Sound Tools, Note Sequencer, Note Synthesizer, MIDI Toolset, and so on) but presents a simpler programming interface to use the tools in concert (okay, bad pun).

The synthesizer portion of MIDISynth is multi-timbral (it can play different instrument sounds) and multi-sampling (the sound of an instrument can vary as a function of its pitch). MIDISynth supports up to 16 instruments, and can play up to seven of them at one time. If you try to play more than seven at once, MIDISynth uses an intelligent "note-stealing" algorithm to discard the least noticeable sounds.

Each instrument is formed from four of the Ensoniq's oscillators. Two oscillators (one generator) are used as the instrument voice, and each of the remaining two can be used to define an envelope to change a quality of the voice (pitch, volume, and so on) as a function of time. The voices therefore support attributes such as velocity response (reaction to the speed at which a triggering action occurs), pitch-bend (changing pitch with time), and decay (the rate at which the volume of a pitch decreases).

The instrument voices can be controlled by a MIDI device in omni (up to 16 MIDI channels playing one instrument), poly (one MIDI channel playing one instrument), or multi (up to 16 channels each playing a different instrument) modes. The voices are individually tuneable. MIDISynth defaults to equal temperament, but the tuning can be changed.

The MIDISynth sequencer supports up to 16 tracks, measure and beat control (meter changes are supported), and tempo control. A metronome is built in. The sequencer can synchronize to an external MIDI sequencer or to an internal clock. You can specify whether you want the sequencer to play using the internal (IIgs) synthesizer, external MIDI devices, or both. It can record and play as many notes as you have memory for—simultaneously, if you like.

Since MIDISynth can operate in the background, it needs some way of telling a controlling program when it has reached a particular "milestone" in the sequence. It does this by calling a routine you specify when it reaches such a milestone; this effectively happens during an interrupt so your routine is limited in what it can and should try to accomplish (most work should be left to the main program).

Without a MIDI instrument, the IIgs can still serve as a sound generator and sequencer. Add a cheap MIDI keyboard controller, and you have a full-featured synthesizer.

SynthLab is an application built to illustrate the power of MIDISynth. It functions as an 8-track MIDI "recorder" capable of receiving and storing MIDI sequences for playback and as a tool for doing some modification of the sound of existing instruments. Although Mark points out that the program itself is not specifically tailored for a professional musician, features of the program are adequate to demonstrate that the IIgs may be adaptable for professional use.

Show the world the power of Apple peripherals. Matt Gulick, Apple Computer's "Apple II SCSI Guy," showed the now-fabled demo where several hundred frames from *The Empire Strikes Back* are displayed on a IIgs from their digitized form on a SCSI hard disk. Using the new Apple II High-Speed SCSI Card and a command available through the GS/OS driver for the card, the interface is instructed to loop through a file containing the sequential frames from the movie, loading each successive frame into the display buffer of the IIgs using direct memory access. The card achieves this at its full 1 magabyte per second rated throughput, giving a display rate slightly faster than 30 frames per second. The display is quite remarkable; one tip-off is that the sequence is not shown in color (the color palettes would be loaded "out of step" with the images, so a static gray scale palette is used.)

The beta version of an Apple Scanner driver was also demonstrated, as well as a IIe scanner program written by Eric Soldan, Matt Gulick, and Greg Branche in 12 hours the day before the demonstration (this program may not be released, it was primarily a "hack" to show how easy it is to access SCSI devices using the Apple card's firmware support).

Apple is starting to emphasize the utility of CD-ROM as a storage medium. Llew Roberts conducted a session on the basics of CD-ROM production, and *APW* and *MPW* Tools Product Manager Tim Swihart was inquiring whether a disc of IIgs tools (similar in concept to the "Essential-Tools-Objects" release for the Macintosh) would encourage developers to buy into CD-ROM technology. For those products where there is a need to distribute more than 10 megs of software to more than 100 people, the pricing of CD-ROM distribution becomes more attractive than using floppies. It's likely that in the future you may see some software offered on CD-ROM at or below the cost of the same software distributed in floppy format.

Show the world how the two worlds integrate. During a session on using the Apple II computers for media integration, Dan Hitchens demonstrated the use of an Apple II *Video Overlay Card* to display IIgs graphic images on an Apple IIe. He also demonstrated two other features using a IIgs; an as-yet unsupported 640 by 400 display mode (this mode uses the IIgs bank $E1 space in such a way that it is currently incompatible with IIgs software), and the ability to simulate two screens on the IIgs by enabling and disabling the use of the *Video Overlay Card* display when updating the screen.

Dan Hitchens's demonstration focused on media integration (see "Picture this", July 1990), including hypermedia (which refers to media integration involving user interaction). His longest demonstration involved an Apple IIgs connected to a Pioneer LD-4200 commercial laser disc player and running EcoVision, a joint production of Florida State and Houghton-Mifflin. EcoVision uses software in synergy with a supplied laser disc to create a "lab study" exercise. Students are asked to assist in the research of an ecological problem and present a report. The laser disc contains many reference sources that can be accessed through a menu system supplied by the controlling program. The IIgs is used as the tool for accessing these sources, but the student is allowed to wander through the information guided by their own perceptions. The system teaches not only the specifics of the scenarios the disc uses, but also a general methodology of problem solving. It shows the utility hypermedia will have as a flexible instructional aid when the systems needed to use hypermedia start falling within the school and home budget. The IIgs is Apple's prime candidate for that market.

Show the world you mean business. More Apple technical people and engineers attended this year's conference than last year's, including Apple's manager of Developer Services, David Szetela. Apple conducted a one-day IIgs college with an introduction to concepts of IIgs programming and advanced question-and-answer sessions. The original IIgs College, held in Cupertino in 1986, took three days. It may be necessary to return to that format to cover the current state of IIgs programming.

I don't want to underemphasize other participants. Bill Heineman was present demonstrating some IIgs printer drivers for Hewlett-Packard printers he is working on. Barney Stone was present demonstrating his *Programmer's Pak* for *DB Master Professional*. Roger Wagner was still making jaws drop with *HyperStudio* tricks. And there were many other people there doing things that were amazing. My point here is to answer the question paramount in many users' minds: "What is Apple doing?". The answer is "more than you might believe".

You don't get to participate as much in a gathering such as this when you are helping to run it. But I came away with a different feeling this year. Last year at this time, many felt that Apple had spent more time trying to appease Apple II developers than trying to excite them. This year, I think it tried to excite us and I think it succeeded.—DJD

Bihexual programming

We've thrown the words "developer" and "development" around quite a bit in the past without giving good definitions. In the computer world, they refer to the development (production) of software and hardware. Claris is a software developer; Applied Engineering and Cirtech are hardware developers. More often than not, we are talking about software development since its developers tend to outnumber hardware developers.

Back before 1980, many individuals who bought personal computers bought them with the intention of writing some of their own software. Some even published their software; companies tended to appear and disappear monthly.

The situation today is much different. Most personal computer owners buy their systems as appliances, with the intention of using prepackaged software exclusively. Most do not know how to program, nor do they wish to bother learning.

Apple is primarily in the business of selling hardware, and in order for it to sell computers it has to convince potential customers that the computer can do neat things. Apple cannot write enough software to fill the market with attractive programs itself, so it depends upon the assistance of outside (third-party) developers.

Most software development tools are themselves software. The typical components are an editor to enter a text representation of the program logic (the human-readable source code), a translator (assembler for assembly language, compiler for other languages) to convert the source into a form the computer can read (object code), and a linker to combine the object code file with other object files and libraries of "standard" object code to build a complete working program. There may be other utility programs included, and usually the entire system requires a "shell" environment that allows using the components in concert. The entire collection of these programs is referred to as a development environment, or developer tools.

Sometimes these tools are supplied by the manufacturer of the computer; it's in the manufacturer's best interest to make sure it's easy to write programs for its products. Sometimes the manufacturer's development tools aren't the best people can imagine, however, and so third-party companies develop and sometimes sell their own tools, often with the encouragement of the computer manufacturer. Apple has worked it both ways, but the situation now is a little strange, and may be affecting the quantity and quality of programs that you see.

The good news is that Apple is pushing IIgs development tools hard. The other news is that a high proportion of these tools are tied to the *Macintosh Programmer's Workshop* (*MPW*) environment. There are good and bad sides to this, too; let me start by trying to cover the positive aspects, and then follow with my reservations regarding this tactic as opposed to native IIgs development systems.

Tim Swihart, Apple Product Manager for the *APW* and *MPW IIgs*, presents the strongest arguments for *MPW IIgs*. *MPW IIgs* is a set of tools that allows creating complete IIgs applications using *MPW* on a Macintosh. Using a Macintosh to develop IIgs software may sound strange, but there are some considerations.

First is "time to market," the time it takes you to get a program ready to send to customers from the day you decide to start working on it. The speed at which you can compile and test the program significantly affects that time, and a 40 MHz Mac IIfx will create the code much faster than the current IIgs. For a large project where the development time may be a large expense (you have to pay the programmer, among other things) the Mac probably will save you enough time to pay for itself. In such circumstances it is usually advisable to have two systems to develop on in the first place, one to write and compile the code, and one to execute and test the code. The machine you test with has to be a IIgs (and Tim emphasizes that having a IIgs available to test is a requirement if you want your program to run), but the machine you develop with does not. Tim suggests that developers consider making that second machine a Mac.

Not just any Mac, though. The second consideration is the quality of the development environment. Running *MPW* on a 1 meg Mac Plus with a 9 inch screen gives you little advantage in many cases; it may actually be less powerful than using the IIgs. What *MPW* really craves is a Mac system with at least 4 megabytes of RAM, at least 40 megabytes of hard disk space, and the larger video displays available on the Mac II. More memory and disk space is always good; Tim also recommends using a full-page, two-page, or multiple screen video display to see as much of your work as you possibly can. These systems are also MultiFinder capable and can compile one file while you are creating another in the editor. The point is that *MPW IIgs* does not merely require a Mac, it requires an expensive, well-equipped Mac.

Three MPW IIgs languages are available: a C compiler (a native compiler for *APW* is also available), a Pascal compiler, and an assembler. The C and Pascal produce IIgs (65816) code exclusively, but the assembler is actually built on the *MPW* native assembler and can generate code for any of the 6502/65C02/65802/65816 processors.

Eric Soldan from Apple's Developer Technical Support group has written a set of tools for the *MPW IIgs* assembler that simplifies the cross development of programs for 128K IIe (and compatible) systems. These are *Dynamo* (see "Miscellanea", February 1990, p. 1), and a commercial utility called *Pizzazz* ($89.95) that can be used with *MPW IIgs* to design graphics-based screens and user interfaces. Eric used these tools to help bash out the 12-hour scanner application mentioned in the previous article. He is also a strong believer in the values of cross development, but he says he did attempt to make *Dynamo* run under the *APW* native environment. The *MPW* assembly environment is simply more powerful, and a full translation wasn't possible.

I'd like to clarify that these people are not pushing *MPW IIgs* to sell Macintoshes; they each have a strong interest in seeing IIgs applications written. Eric wrote *Dynamo* and is giving it away free; it is only useful in creating Apple II programs. At the conference, Tim Swihart demonstrated *AppMaker GS*, which is a third-party tool that runs on the Mac that helps you design elements of the Desktop interface (even "stealing" them from your Mac program) for the IIgs. Each will also discuss the relative merits of native development; Tim also demonstrated the native user interface design tools *Design Master* and *Genesys* on the IIgs. (In case you're wondering, Tim also goes to Mac shows to demo both native and cross-development tools for the IIgs. How's that for a sneaky way to show off a IIgs to Mac people?)

The way you would normally work in *MPW IIgs* is to write and compile your program on the Macintosh and transfer it to the IIgs to test. Tim provided the following sample timings for a C and assembly compilation (all times in seconds):

APW C	348	on IIgs
	165	with TransWarp GS
MPW IIgs C	41	on Mac II
	<30	on Mac IIci
APW asm	290	on IIgs
	170	with TransWarp GS
MPW IIgs	90	on Mac II
	54	on Mac IIci

On the other hand, in our own tests of compile time alone the *ORCA/C* compiler on a 2.8 MHz IIgs was less than 10 per cent slower than *MPW IIgs* C on our 8-MHz Mac SE: 70 versus 64 seconds. This emphasizes the importance of a fast, expensive Mac system to really see performance benefits in cross development.)

Other than that, it's pretty much the same as writing code on a IIgs. The major advantages of using the Macintosh are speed (for the faster systems) and the ability to get more source code on the screen (using the larger displays available for the Mac II). The *MPW* environment also has the following enhancements.

First, most functions can be controlled by a scripting language which comes with *MPW*. *APW* has a programmable shell, but *MPW*'s script control extends to operations such as controlling the editor without (as APW does) using a separate macro language for the editor. *MPW* does use menu bars and items can be added to the menus with the script language. For example, installing *MPW IIgs* creates new menu items that can be selected to help build your IIgs (as opposed to Mac) program.

Second, *MPW*'s shell supports an interactive command help system called Commando. When you type a command at the shell level and can't remember all the information it requires, you can add an ellipsis ("...") to the command and Commando will pop up a dialog box that shows you your options and allows you to "fill in the blanks". There is no equivalent mechanism under *APW* currently, although *APW* does support on-line help, as does *MPW*.

Finally, *MPW* has Projector, a project manager that manages several programmers working on a common program. Projector keeps tabs on the access and revision of the project's files so that various programmers do not check out and modify the file at the same time, negating each other's revisions. Projector also works across an AppleShare network so that all users can be working with the same collection of source files. *APW* currently does not handle such a mechanism.

There are other differences from *APW*. The most noticeable is that *MPW* uses a variation of the desktop metaphor that includes the use of multiple windows, including a "shell" window. A desktop-based interface (minus the user-customization available in *MPW*) is available for *APW*'s near cousin, the Byte Works' *ORCA* environment. Byte Works has developed a desktop-based environment (*ORCA/Desktop*) for its languages that includes multiple-windows and an integrated debugger. It's supplied with the *ORCA/C* and *ORCA/Pascal* language systems or can be purchased separately.

The above probably most accurately reflect the major differences between the environments.

Cross development simplifies *parallel development,* a concept Apple is also carrying to Mac developers. With the similarities between the Mac and IIgs tool sets, it makes sense to write the same program for both machines at the same time. For a little extra expended effort, you increase your market significantly by also selling a IIgs version. IIgs programmers can use the same logic if they have a Mac for development. Avoiding a duplication of effort in having to maintain powerful development environments for each of Apple's systems is a good reason for Apple to focus on *MPW IIgs*. Having Apple IIgs engineers improving the IIgs system software benefits more IIgs users than having the engineers revising compilers.

Here are the negatives as I perceive them. First, developers who don't spend adequate time actually using the target machine may remain blind to weaknesses in the code they produce with the cross development system. Second, not having the best possible environment for native development hurts non-professionals in their attempts to turn out smaller programs of quality.

***MPW IIgs* C is the biggest argument I can think of against cross development.** Apple is preaching that the payoff in cross development is being able to bring the same program to two different platforms and increase the market. However, unless your program is confined almost solely to using IIgs ToolBox calls, it's not so easy. The *APW/MPW IIgs* C libraries for non-toolbox functions are so poorly written that they heavily penalize the IIgs user through the size of the generated file, if nothing else. Julian Pugh, author of *GS-Numerics*, wrote the program using *MPW IIgs* C and is a staunch supporter of the use of cross development for large projects. Several of the limitations of *GS-Numerics* (such as the relatively awkward way of entering polynomials) came from trying to keep the size of the program small enough to run in a one megabyte IIgs and to fit on a single 800K disk. Pugh is now working on a Mac version of the program with *MPW* C and is finding that the compiled code with similar functionality is almost half the size. He used *MPW* IIgs primarily for the speed advantage in compilation; it looks suspiciously like the advantage he got was producing bloated code much faster.

I have confirmed the code sizes on another large program using native IIgs compilers; the version compiled with *APW* C is 600K, the version compiled with *ORCA/C* v1.1 is less than two-thirds of that size. Larger code penalizes the user with increased disk usage, increased memory usage, and increased load times. C is probably the hottest language for programmers right now, and Sierra On-Line (one of the very companies who can afford the high-dollar cross development systems) has published the opinion that their C-based tools can't be used for IIgs programs due to the poor C compiler environment. This is the exception to the "write system software, not compilers" generalization above: Apple very much needs to fix the libraries for *MPW IIgs* C if it expects parallel development to work.

To be fair, the *MPW IIgs* assembler will obviously not have this problem (assembly translation is straightforward from the source code; the programmer has more control over the quality of the code generated), and the code that *MPW IIgs* Pascal produces is reputably quite good. I am picking on the C compiler because I believe that cross development requires that the developer pay particularly careful attention to the quality of the resulting program. You will be selling the program to someone who may want to use it every day.

Whether for this reason, or others, we also haven't seen the plethora of products that we would like to see from Mac companies for the IIgs. A few, like SoftView (*TaxView*), tested the IIgs waters and got out. Again, the Mac cross development environments are not cheap, and if larger companies are not writing software for the IIgs then the native tools must be as strong as possible to allow IIgs hackers to spread their wings. Native tools empower the *individuals* with the most direct concern for the future of the machine.

Apple has currently left native environments in the hands of third-party companies; for a fast assembly (only) environment, there is *Merlin 16+* from Roger Wagner Publishing; for a powerful multi-language environment, there is the *ORCA* environment from the Byte Works.

Apple itself has been active in bringing tools from the *MPW* system to IIgs; their *APW Tools and Interfaces* v1.1 package ($50 from APDA; part #A0240LL/A) includes such goodies as a resource compiler (REZ) and decompiler (DEREZ) and a port of the powerful *MPW IIgs* linker (LinkIIgs). The *GSBug* debugger is a general-purpose tool not tied to a specific environment. So the problem isn't that Apple is failing to support native development. But programmers have to be very careful to evaluate whether they can afford the type of cross development system that will really allow them to see a benefit, and whether it allows them to produce programs that will actually entice users to buy their products.

Still, the cross development environment is an option that should be considered. If you have the money and the type of project that can benefit from a faster turnaround from your development system, *MPW IIgs* will provide that speed. You can also use the system to develop both a Mac and IIgs version of your software, expanding your potential market. Just don't forget that the quality of the product doesn't depend on how fast you can produce it, but how well you can make it run for the user. Properly applied, *MPW IIgs* may give you more time to work on the latter objective.

Oh, the prices... *MPW 3.1* itself (APDA part #M0019LL/C) is $150. The *MPW IIgs Tools* (#A7Z0012/D) are $50, the *MPW IIgs Assembler* (#A0005LL/D) is $100, *MPW IIgs* C (#A7Z2001/B) is $150, and *MPW IIgs* Pascal (#A7G0032) is $175. A minimum configuration is *MPW*, the *MPW IIgs Tools*, and one language package (assembler, C, or Pascal); all items are sold through APDA.—DJD

Miscellanea

"Boston Fan Knows Apple II Better than Apple Does". Well, that's an alternative interpretation of the title of a copyrighted story on page 1 of the July 24 issue of *The Wall Street Journal*. The story talks about Boston Red Sox fan Charles Waseleski III, who keeps statistics on the Sox that are detailed enough that players have used

them in salary talks. A Boston Globe sportswriter even buys a weekly compilation. The reason we're writing about them here, of course, is that the statistics are kept on an Apple II.

Such usage does indicate that the Apple II is applicable to more than the K-12 educational market, when properly applied.

Applied Engineering is now shipping a $99 heavy-duty power supply for the IIgs. The supply is rated at 60 watts as opposed to the 38 watts of the standard IIgs supply. The individual voltage/power ratings are +5 volts at 6 amps, +12 volts at 2 amps, and -5 and -12 volts at 0.5 amps. The Applied Engineering supply can be converted to 220 volt, 50 Hz operation by a switch on the side of the supply. It comes with a one-year warranty.

TML has shifted its IIgs products to a new publisher, Complete Technology. Complete is owned by former TML Apple IIgs Product Manager Vince Cooper, and will publish and support *Complete Pascal* (formerly *TML Pascal II*), *Complete BASIC*, and the *Complete Source Code Library—Pascal*.

Complete will be mailing details of future product plans and technical support to TML customers. Complete Technology, Inc., offices are open 10 AM to 5 PM EDT daily; the address is 5411 Ortega Blvd., Suite 7, Jacksonville, Fla. 32210, 904-731-7181.

Two innovative Apple II hardware companies aren't answering their phones. According to our best information, Ingenuity, Inc., and Checkmate Technology have closed their doors.

The AppleWorks Educator, P.O. Box 72, Leetsdale, Penn. 15056, is offering a free "AppleWorks Tips Twin Pack" to educators. The package consists of a "Teacher's Best AppleWorks Tips" sheet and "20+ Tips for AppleWorks GS USers". You can receive a single set by sending a letter-size self-addressed stamped envelope.

The AppleWorks Educator is a publication dedicated to educational AppleWorks users. Subscriptions are $25 per year (for $34.95 the subscription includes an annual AppleWorks Teacher Resource Disk); add $3 for purchase orders and $3 for foreign surface mail or $15 for air mail as applicable.

Apple has released a new ProDOS 8 System Disk v3.2 with ProDOS 8 v1.9 and BASIC.System v1.4.1.

The new version of ProDOS 8 has two bug fixes: a problem with loading programs larger than 38K and a problem with calling the "Program too large" error message were fixed. As an enhancement, v1.9 will load an enhanced "quit" routine that is installed on enhanced IIe systems with an 80-column card or equivalent systems. For these systems, you get a mini-selector in place of the "ENTER PREFIX" and "PATHNAME" messages. (Thunderous applause.)

BASIC.System v1.4.1 has been altered to force clearing the previous contents of the pointer to the previous file position before setting the end-of-file. Previously a value remaining in the high byte could prevent setting the correct end-of file.

***ProSel-16* has added two major new features.** The first is an Appointments CDA that you can install in your IIgs boot disks's System/Desk.Accs folder and configure to generate an alarm as appointements approach. The other is a rather spectacular programmable scientific calculator accessible from *ProSel-16* itself. The calculator is user-configurable and can store and run programs using disk files.

AppleFest/West will be December 7-8 at Long Beach Convention Center in Long Beach, Calif. It will be a purely Apple (II and Mac) event with a focus on education in addition to the traditional home and home business flavor. Call 617-290-0420 for more information.—DJD

Ask (or tell) Uncle DOS

The message is...

Regarding "Can the Machine Maim the Message?" ("Miscellanea", ***A2-Central***, Aug. 1990): I only became aware of the controversy through this article. I read it with interest to see if another explanation might surface, but I didn't find it, so I thought I'd offer you my two cents.

Suppose I'm an "average" student and I'm expected to learn a word processor to use for school. I have a choice between a PC and a Mac. I've already heard through the grapevine how difficult it is to learn on a PC. I choose the Mac.

Since a greater number of "average" students pick the Mac, the average writing level of people using the Mac is necessarily lower. Only the "above-average" or tenacious can tolerate the user-rude interface on the PC.

Does this mean something is wrong with the Mac? I shouldn't think so. It means that you've got people actually writing who probably would have done even worse out of frustration or perhaps wouldn't have bothered to do the assignment at all if the only choice was a PC.

The computer doesn't maim the message. If it weren't for the computer, there would be no message.

Brent Thorwall
The Follett Software Company
McHenry, Ill.

I agree. In comparing computers, I hadn't thought it through to the point of realizing that another datum that's missing here is knowing how students with either computer out-produce students of similar abilities with no computer.

If the frustration potential of PC software was extremely bad, it seems that there should have been some moderation of the difference in the PC versus Mac groups. I still think it would be interesting to see what happens if you switch the two groups.

Speaking of maiming the message...—DJD

"It's" not intentional

I know it's a small point but it's beginning to be a habit with you folks. What is it? It's "it's". Please, "it's" = "it is", not the possessive pronoun "its".

Alan G. Hill
Bay City, Mich.

All right, all right, I'll confess..."it's" is my fault. I have three recurring grammatical weaknesses that I know of: "it's" versus "its", "their" versus "there", and the use of "that" and "which". (Also a few structural weaknesses, but those appear to be less offensive.) As the words get shorter and your eyes get blurrier close to deadline, it gets easier to miss that it's "it's" when it should be "its". I'll try to get its (sic) right.—DJD

(Actually, he should blame it on his copy editor.—TW)

No friend of the Mac

Your belief that most Apple II users wish the Macintosh well disturbs me, as does the other news in the June issue; hence this letter.

The Macintosh (black and white version) is almost completely defined by its software: the WIMP (windows, icons, mice, and pull-down menus) interface. The Apple II is almost completely defined by its hardware: the open architecture. This means that the Apple II can encompass the Macintosh—add a faster processor and 142 more lines of resolution to the IIgs: result, a flat-out superior machine—but the Macintosh cannot encompass the Apple II without major surgery for those Apple II slots!

Consequently, the Apple-promised "no compromise" II emulator for the Mac—even if it allows for use of all ports—cannot be other than compromised. Given this and Apple's belief in Mac superiority, it is easy to see that this so-called "no compromise" card is essentially an inverted Trojan Horse designed to kill the Apple II: a device to allow schools to run undemanding Apple II educational software—see the message of Scholastic's Vice President you report in the July issue—while moving into the "superior" Macintosh world.

Apple II fans cannot and must not wish the Macintosh well. Its every success and everything that promotes its success bring the Apple II one step closer to death.

Unless, of course, we all switch to Video Technology's products. Then what Apple does will make no difference: the Apple II will live on indefinitely...just under a different name.

Robert Hardman
Willingboro, N.J.

I didn't comment on the "Apple II emulator" because I think it will be a non-issue from the standpoint of any real effect on Apple II purchases. Look at Apple's pricing structure and tell me if you believe that Apple can sell the card at a price that will actually displace Apple IIe or IIc (or even Laser 128) systems from

schools. I don't think it will happen. I think what will happen is that more Mac users will run Apple II software in their systems, which may actually stimulate Apple II software sales.

What the announcement does do for me is tell me that Apple knows that Apple II compatibility is an issue. And that, to me, validates Apple II technology.

The success of the Macintosh is not germane to the success of the Apple IIe or IIc; the systems are quite different, and the market will decide the future of the systems. The IIgs in "IIe mode" will live or die by the same impetus.

In the "WIMP" mode, the IIgs can benefit by riding on the Mac's coattails if Apple presents the systems as sharing a common user-oriented design philosophy instead of breaking them along hard market boundries. The market is changing as we speak and graphics-oriented interfaces are coming to all personal computers.

If you look back, MS-DOS was little more than the CP/M user interface brought forward. Windows 3.0 will be a graphics interface layered over MS-DOS. You don't see many CP/M machines anymore; they have essentially evolved into MS-DOS machines (some of the same manufacturers have competed in both markets). I think the same thing will happen with the Apple II; you will see the history of the Apple IIe in the IIgs, but the computer will become more and more IIgs-like.

The problem is that Apple calls it "Mac-like", and not everything that the IIgs does is strictly like the Mac. Some of it, like GS/OS, is better. I don't wish the Macintosh ill, and I think other users also need to avoid directing their energy in vain pursuits. What Apple needs to do is to prove it can sell two product lines, and that's done by not trying to typify one product exclusively by its resemblance to another. You don't see General Motors try to sell Corvettes by pointing out their resemblance to Cadillacs. I don't care if Apple sells two million Macs as long as they make the same attempt to sell Apple II systems. So far, they've been found lacking.—DJD

Word processor updates

Since 1983 I have been using DOS 3.3-based *AppleWriter* with an unenhanced Apple IIe for word processing and program writing. What advantages are there for updating *AppleWriter* to ProDOS, etc., and where might I be able to purchase the most recent version of *AppleWriter*?

James L. Wickliff
Fayetteville, Ark.

The major advantage would be ProDOS compatibility and its associated benefits, such as faster loading and saving of files and better and broader hard disk support. And unlike the DOS 3.3 version, the ProDOS version of **AppleWriter** is not copy protected. The ProDOS version also works on enhanced IIe computers without interference from MouseText (if you should decide to install the enhancement kit), and it supports horizontal scrolling of text. This means you can display text wider than 80 columns on your screen and get abetter idea of how it will look when you print it out.

There were only two versions of the ProDOS **AppleWriter**; the original version 2.0, and a minor 2.1 update that corrected some problems with certain printer interfaces. The 2.0 version was sold through dealers and the 2.1 update was distributed as a "service item" for dealers to use to update your disk. Apple itself has not sold the program for some time; we've been told that some remaining copies are available from Alltech Electronics, Inc., 1300 E. Edinger, Suite D, Santa Ana, Calif. 92705. 714-543-5011.

I wrote (and even typeset) **ProDOS Inside and Out** in **AppleWriter** using **AppleWriter**'s Word Processing Language (WPL) command language. Tom and I switched totally over to AppleWorks for our word processing about 4 years ago when add-on macro programs started adding some of the same power to AppleWorks. Still, I felt that AppleWorks came up a bit short and that I was sacrificing some utility as a word processor for its advantage as an integrated environment (my other highly-used application is the database).

Beagle Bros' **TimeOut** applications have been the primary driving force toward AppleWorks. **UltraMacros**, **QuickSpell**, and **Thesaurus** made the writing environment of AppleWorks more tolerable; there are no equivalents to the latter two that can be installed in **AppleWriter**. Utilities like **PowerPack**'s AWP.TO.TXT application made manipulation of text files easier. I began to miss **AppleWriter** less.

Now there are two new packages which I feel will finally eliminate any consideration of **AppleWriter** for me. The first is **TimeOut TextTools**; the second is JEM Software's **Outliner 3.0**.

TextTools ($49.95) contains a utility called Glossary that allows the easy incorporation of "boilerplate" text into a document. If you have certain blocks of text that seem to recur often in your documents, the next time you type them in you can save them with the Glossary function and incorporate them into future documents. That takes care of one of my most common uses of WPL in **AppleWriter**.

Other utilities handle functions that I had to do manually when writing **ProDOS Inside and Out.** There is a Table of Contents generator, an Indexer, a style sheet generator (style sheets allow formatting a block of text according to a pre-defined method), tab ruler generation, an **AutoWorks**-style enhanced mailmerge function, a utility to print multiple files "linked" together into a batch, a column copying and formatting utility, a "find" utility that can locate and replace printer options, and (as they say) more.

Outliner 3.0 ($44.95 from JEM Software, P.O. Box 20920, El Cajon, Calif. 92021) is a utility that installs itself into AppleWorks version 3.0 and allows you to indicate the hierarchial levels of blocks of your document by inserting AppleWorks markers. Then you can call up Outliner to hide blocks at or below a certain level. By hiding more of the low-level blocks, you can get a more accurate overview of the content and structure of your document; then you can expand the lower levels to insert and expand the areas you wish. Additionally, commands are supplied to expand, prune, and reorganize the outline "tree" to aid in organizing and elaborating your thoughts systematically.

Before **AppleWriter** fanatics are offended, I'd like to point out that I'm not claiming the above products make **AppleWriter** obsolete; if it works for you, keep on typing. But the product is not being actively enhanced by its publisher, and my personal view is that if you looked at AppleWorks several years ago and found it lacking, you may find the migration to be less painful today if you are forced to make it.—DJD

More debugging help

Regarding the letter "Debugging Help" on page 6.55: *BugByter* **does** support 65C02 instructions; at least the version I got on the Apple *EDASM* disk did back in 1985 or 1986 (back when they distributed things in a notebook format). I can use it to trace ROM code even in the bank-switched ROM in my revised memory expansion IIc. I don't know if it would work to trace the code in language card memory, and I definitely don't think it could be used to trace code in auxiliary memory (unless you managed to put a copy in the same memory location in aux memory—with all the stack and pointer stuff the same as the copy in main memory—yuck!) *BugByter* is fairly powerful, but I have also wished for a better debugger, one that understands 80 columns better (*BugByter* puts itself in every other column if 80 column mode gets turned on, so must write directly to the 40 column screen).

Steven Weyhrich
Omaha, Neb.

A different type of debugger is available from ProDev (P. O. Box 162, LaSalle, Mich. 48145-0162), the manufacturers of a $189 **DDT8** slot-based hardware debugger for 8-bit Apple II systems. There is also a IIgs version, the **DDT16** (also $189), available. We have not used these, but we've seen them demonstrated at our summer conference and the capabilities seem impressive for the price.

The ProDev debugger occupies a slot in the computer and, as much as feasible, tries to monitor a running machine-language program without intruding on its operation. You can even attach a video monitor to the debugger to observe the debugger's output while your program is displayed on your normal monitor. The DDT will do all the normal debugger tricks (step and trace program execution; monitor the stack, registers, and memory; set breakpoints, and so forth). The debugger contains most of its controlling program in 32K of ROM rather than residing in the computer's memory as software-based debuggers must. And with a hardware debugger you can have the program stop when an address is accessed; this is a very handy tool for determining which part of your program is writing into an area of memory that it shouldn't be. The only other easy way to do this is to use a much more expensive device called a logic analyzer.

Though not literally a debugger, we also overlooked a very handy tool for the IIgs: Dave Lyon's **NiftyList**, which might be described as a "system snooper" CDA. It won't trace your code as it executes, but it lets you examine the software state of your IIgs to find where your program may have gone awry.

NiftyList installs as a Classic Desk Accessory and includes a series of commands to examine IIgs memory use, the contents of addresses (as hex, ASCII, or parameter lists for tool calls), system status (current GS/OS prefixes, memory use, and so forth), tool set versions, and so on.

*The newest version, 3.0, which Dave was distributing at the summer conference, adds the ability to process external commands and to make IIgs ToolBox calls directly from the **NiftyList** prompt. Two sets of grouped extended commands are included; a set of miscellaneous memory examination tools called Goodies (which includes a very useful "\find" command for locating patterns in memory) and a set of utilities to help detect when a program is accidentally changing memory it shouldn't be (a common problem) called Big Brother. Information is included to get you started on writing your own **NiftyList** modules. The data lists used are also provided in a user-modifiable format.*

***NiftyList** is $15 shareware from Dave at DAL Systems, P. O. Box 875, Cupertino, Calif. 95015, and an absolute bargain.—DJD*

AppleWorks GS draft printing

Many of your readers who use *AppleWorks* GS may be interested to know that with version 1.1 they can access the near letter quality (NLQ) and correspondence quality modes of the ImageWriter II by selecting draft in the Print dialog box, then holding down the Option (for NLQ) or open-apple (for correspondence) key as they click "OK" to start printing. If they don't hold down any keys at all they will get the standard draft printout. They cannot get all of the size, style, or font changes that they have made in their text, but they do get the standard ImageWriter II fonts and quicker printing. Claris evidently left this information out of its upgrade package when they sent it out.

David C. Frye
Waldorf, Md.

Claris passed on another undocumented tip that appears in more recent versions of the release notes: when trying to scale a graphic to print correctly in the "condensed" mode, hold down the Option key while resizing to force it to the correct dimensions.—DJD

Sparse files

I discovered a "feature" of GS/OS 3.0 which I consider to be a bug.

So what is it? It appears that the GS/OS "write" command somehow prevents you from writing totally empty blocks, or at least most of them. I encountered it while writing a "no-frills" CDA for saving the super high-res screen. Anyway, when you try to save a screen with one dot on it, the length of the file is only 5 blocks! However, the end-of-file information reads $8000, indicating that GS/OS has created a sparse file.

Now, to me this represents a real obstacle since I'm developing a program that draws Mandlebrot images and thus has to have the ability to save the screen—moreover, in a format that any paint program can use. Any advice on how to save a 32K SHR screen image, regardless of what it contains?

Tero Sand
Helsinki, Finland

A "sparse" file is one that has been stored in a compact format due to the presence of a large sequence of zeros in the file. You can create one under ProDOS 8 by saving one byte, moving the file pointer, and saving another byte. From Applesoft, it looks like this:

```
BSAVE BIGFILE,A$2000,L1          save one byte
BSAVE BIGFILE,A$3FFF,L1,B$2000   move forward and
                                 save another
```

(The "B$2000 tells ProDOS to move forward to the $2000th byte in the file before saving the second byte.)

If you catalog the disk with BIGFILE you'll see an "ENDFILE" (length) value of $2000 (8192) bytes, but the file only occupies three blocks. When the first byte was saved, ProDOS allocated one block to hold the first byte of data. When the second byte was saved, ProDOS noticed that there was no real data between it and the first byte, so it allocated one 512-byte key block to track the blocks where the data is located, marked the location of the first data block (already allocated) and marked the intervening positions as "empty" to account for the distance between the first and second bytes. Finally, ProDOS allocated a block for the data and saved its location in the key block.

The result is that two blocks containing data and one key block have been allocated on the disk; the intervening "phantom blocks" are marked in the key block as pointing to block 0, which ProDOS interprets as meaning that the phantom blocks contain all zero bytes. Therefore, data that contains a large number of sequential zero bytes can be represented by a file on disk that takes up less space than the actual data.

Now I'm going to be really rude and prove that your expensive paint program is dumber than BASIC.System (well, maybe I should say the paint program is "smart" in the wrong way). Use "HGR" to clear the high-res screen and repeat the above instructions to create the sparse file. Then issue:

```
HCOLOR=3
HPLOT 0,0 TO 279,0 TO 0,159 TO 279,159 TO 0,0
```

That should draw enough white lines on the screen for demonstration purposes. Then use:

```
BLOAD BIGFILE,A$2000
```

to read the sparse file into memory. You'll see the screen go completely black, showing that $2000 bytes of the file were loaded, enough to overwrite the entire screen.

*Your paint program probably fails to load the file because it expects a super high-res file image to be a certain size, and it determines the size by checking the block count. That's a bad idea, for the reason we've just seen: the block size is not a true indication of the amount of data in the file. The "ENDFILE" value will be accurate. The program can also use the filetype and auxtype values for the file to confirm the type of data it professes to contain. It should do so (**MiniPaint** and **HyperStudio** loaded a sparse file fine for me; **PaintWorks Gold** wasn't as astute).*

Okay, now you want to know why GS/OS makes the file sparse. The answer is that GS/OS works through several file system translators (FSTs), one of which is the ProDOS FST that we normally use. Not all file systems that GS/OS can access will understand sparse files; one of these is the AppleShare FST, because the disk device it uses is on a Macintosh, and the Mac doesn't use sparse files.

Let's imagine copying a 16-megabyte sparse file from a ProDOS hard disk with 10 megabytes of free space to the AppleShare volume. GS/OS uses the sparse-file-aware ProDOS FST to read the file, and writes it to the AppleShare volume. The AppleShare volume doesn't use sparse files, so it expands the file to use a full 16 megabytes of disk space. No problem, assuming we have enough room on the AppleShare disk.

Suppose that later we want to copy the file from the AppleShare volume to our hard disk. We have ten megabytes of free space. GS/OS is about to read a 16-megabyte file through the AppleShare FST and write it through the ProDOS FST to our disk. I'm sure you've already realized the dilemma; either the ProDOS FST has to make the file sparse again as it writes, or we're going to be unable to put the same file back on the same disk.

Apple elected to give the ProDOS FST the ability make the file sparse, which is the only sane choice. Since the use of an FST should be transparent to the user (accessing an AppleShare disk shouldn't be any different than accessing another type of disk). It was a design decision to support the idea of FSTs.—DJD

A2-Central™

[redacted]

© Copyright 1990 by
A2-Central

Most rights reserved. All *programs* published in ***A2-Central*** are public domain and may be copied and distributed without charge. Apple user groups and significant others may obtain permission to reprint *articles* from time to time by specific written request.

Edited by

Dennis Doms

with help from:

Tom Weishaar	**Sally Dwyer**	**Dean Esmay**
Joyce Hammond	**Jeff Neuer**	**Jay Jennings**
Tom Vanderpool	**Jean Weishaar**	

A2-Central,—titled ***Open-Apple*** through January, 1989—has been published monthly since January 1985. World-wide prices (in U.S. dollars, airmail delivery included at no additional charge): $28 for 1 year; $54 for 2 years; $78 for 3 years. All back issues are currently available for $2 each; bound, indexed editions of our first four volumes are $14.95 each. Volumes end with the January issue; an index for the prior volume is included with the February issue.

The full text of each issue of ***A2-Central*** is available on 3.5 disks, along with a selection of the best new public domain and shareware files and programs. for $84 a year (newsletter and disk combined). Single disks are $10

Please send all correspondence to:

A2-Central
P.O. Box 11250
Overland Park, Kansas 66207 U.S.A.

A2-Central is sold in an unprotected format for your convenience. You are encouraged to make back-up archival copies or easy-to-read enlarged copies for your own use without charge. You may also copy ***A2-Central*** for distribution to others. The distribution fee is 15 cents per page per copy distributed.

WARRANTY AND LIMITATION OF LIABILITY. I warrant that most of the information in ***A2-Central*** is useful and correct, although drivel and mistakes are included from time to time, usually unintentionally. Unsatisfied subscribers may cancel their subscription at any time and receive a full refund of their last subscription payment. The unfilled portion of any paid subscription will be refunded even to satisfied subscribers upon request. MY LIABILITY FOR ERRORS AND OMISSIONS IS LIMITED TO THIS PUBLICATION'S PURCHASE PRICE. In no case shall I or my contributors be liable for any incidental or consequential damages, nor for ANY damages in excess of the fees paid by a subscriber.

ISSN 0885-4017

Printed in the U.S.A.

GEnie mail: A2-CENTRAL
Voice: 913-469-6502
Fax: 913-469-6507