GS+

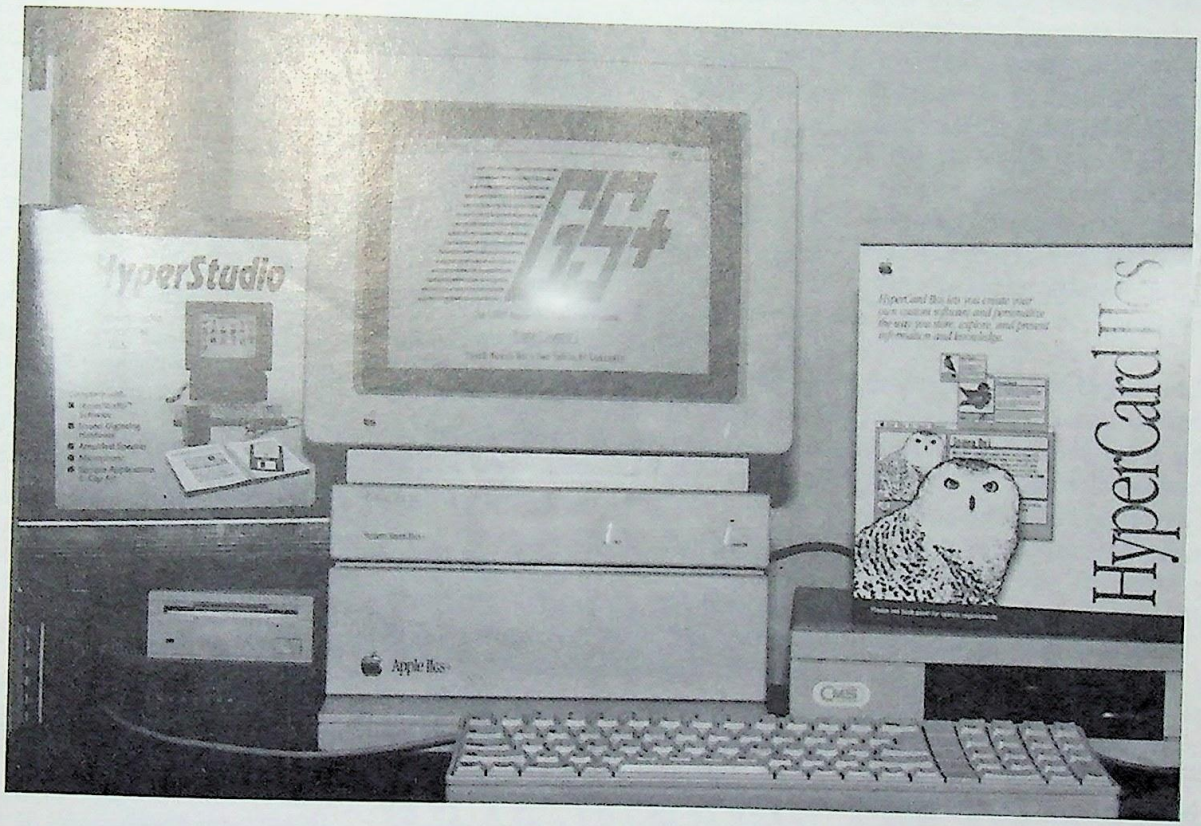
May
June
1991

Volume 2
Number 5

The *First* Apple IIGS® Magazine + Disk Publication!

HyperCard IIGS vs. HyperStudio

HyperMedia Heavyweights Slug It Out For Your Paycheck!



Plus

An Interview With Matt Deatherage Of Apple DTS
Working With The Toolbox Part 2

Programs

Autopilot • EGOed v1.34 • NoDOS v1.5 • Softlock

Reviews

TMS Pro R45 • RamFAST/SCSI • The HyperStuff Collection
McGee At The Fun Fair • Talking Classroom
Talking Multiplication & Division
Bouncing Bluster II • Space Shark • Transylvania III

Writer's Block

By Steven W. Disbrow

1-900-369-AEAE

If you have not heard by now, Applied Engineering has changed their Apple II technical support phone lines over to a 1-900 phone number and are charging $1.50 a minute for technical support. (That's the number up there at the top of this paragraph.) When I first heard about this, I was shocked (to say the least). I was further stunned to learn that this was for their Apple II products only, their Macintosh and Amiga technical support was the same as always.

After a few weeks of soaking up all of the rumors concerning this event, I called AE to get their side of the story. The person I spoke with was Tom Milks, one of AE's Vice-Presidents. After getting past my initial question of "Is this true?" (I honestly could not believe that AE had done this), Tom began to explain *why* AE had taken this step.

If you had ever called the old AE technical support number, chances are you got a busy signal at one time or another (and another and another and . . .). The situation was so bad that here at *GS+* Magazine, we began to poke fun at AE by inventing a fictional spokesperson for them, Mr. Busy Signal (Mr. BS for short). After all, he was the only technical support person *we* ever got through to! According to Tom, AE knew there was a problem and began to research ways to fix it. After performing a "market survey," (in which customers were asked if they would be willing to *pay* for the ability to actually get through to technical support), it was determined that people would indeed pay for better technical support. So, further research was done, and it was determined that a 1-900 number would be the best way to implement this change.

At this point, I told Tom that everyone I had spoken too (via phone, e-mail, etc.) was, um, pissed-off and felt that this was just another example of how AE was dumping the Apple II market (the first examples being their new Macintosh and Amiga products). Tom agreed that not everyone was thrilled, but, then again, a lot of the *facts* had not come out about the situation.

For example, it is not well known that the $1.50 a minute charge includes all long distance charges. So, unless you actually live in AE's local calling zone, you are actually talking about an additional charge of about $1.20 a minute. The further away you live, the more you actually "save." Further, that $1.50 a minute is one of the cheaper 1-900 services around today. A quick check of the "Love Lines" advertised on late night TV verifies this to be true. (I just *love* researching this stuff!)

Also, the $1.50 a minute charges don't start until you actually talk to a technical support representative. If you initially get the call switching box, a tape recording will tell you that you have reached a 1-900 service and you will be charged a $1.50 a minute starting when a technical support representative picks up the phone. If you hang up before then, you won't be charged at all.

By switching to a 1-900 service, Mr. Busy Signal has been, for the most part, retired. You can actually *get through*! Tom says that since they implemented the 1-900 technical support number, they have actually gotten *thank-you* letters from people that had used the service. Here again, the amazing thing is that you can now actually get through to a real person. According to Tom, one of the reasons that the lines were always so busy was that, rather than reading their manuals, people would call AE for answers to questions that were completely unrelated to AE products. Questions like, "What's the Finder?" or "How do I put *Program X* on my hard disk?"

Along those same lines, you may have, at one time or another, called AE Technical support only to get a recording telling you that everyone was at lunch. The lunch hour has been eliminated, so this should not happen any more.

When we began to discuss the perception that AE is "abandoning" the Apple II, Tom became very defensive. I was told that the reason AE's Macintosh and Amiga customers don't have to use the 1-900 number for technical support is that they don't get as many calls from them (these are *very* small markets for AE). According to Tom, they get between four and ten Amiga technical support inquiries each day. They get maybe ten times that many for the Macintosh each day. This is nothing compared to the volume of Apple II technical support calls they used to get. (Tom further explained that the only reason AE had gotten into the Macintosh and Amiga markets was because their Apple II business had been declining. It was either diversify or slowly go out of business.)

Finally, Tom ran down a list of new or recently introduced Apple II products that AE had done: the IIGS high-density 3.5-inch floppy drive, the new TransWarp GS 32K cache upgrade, the Vulcan controller card upgrade, and the send-FAX option for the DataLink external modem. According to Tom, AE has *no* intention of ever abandoning the II, and these products should show that.

What I Think

After talking with Tom, I think that AE honestly feels that they have done the right thing by going to this 1-900 number for technical support. It definitely has the advantage of allowing people that really *need* technical support to get it quickly—and that's good. Beyond that, I can only say that I *personally* disagree with the practice of charging for technical support.

So why did I print all of this? Well, for the last few months, all that I have seen about this situation has been speculation and hateful rhetoric. AE has been a strong supporter of the Apple II for a great many years now and I felt they at least deserved to have their side of the story told.

To me, the real problem is that AE simply didn't tell anybody about this mess! If they had simply done a mass-mailing informing their customers of the change and explaining *why* it was done, a great deal of this speculation and ill-will could have been avoided. Keeping your customers informed is a basic necessity of staying in business and I'm amazed that AE doesn't know that by now.

Let's Talk About Software Theft

A funny thing happened to me a few weeks ago. I received a phone call from someone that has been, by their own admission, stealing *GS+* Magazine and the *GS+* Disk. Now, I get lots of phone calls that lead me to *suspect* that the caller is stealing the *GS+* Disk, but I've never before gotten one where the caller came right out and said, "I get *GS+* from a friend."

Well, that's not exactly true, I have gotten calls from people that wanted to *subscribe* because a friend had faxed them a few pages or had given them EGOed to play with. I can deal with that—I think—but this was the first time the caller wanted not to subscribe, but to register a few *complaints* about the content of *GS+*. (This is roughly equivalent to someone breaking into your house, stealing your TV, and then calling you up to complain that it was not a Cable-Ready model.)

What Would You Do?

Looking back, I am amazed that I did not simply hang up the phone. I guess the same morbid fascination that keeps me glued to the nightly news is what kept me from hanging up. I simply stood there, nodding my head (why *do* people nod their heads when talking on the phone?), and making those polite noises that you make when you want the person at the other end of the phone to just go away.

However, the person would not go away, and I felt myself slowly but surely losing my temper. (At one time or another, I am sure we have all tried to imagine what the person at the other end of the phone looked like. To keep my temper under control, I was doing the same thing—only with weapons of mass-destruction added.) Fortunately, before I said any of those words which public relations nightmares are made of, the caller reached the end of his complaint list and decided to end the conversation. As I hung up the phone, all I could manage was a very weak, "I hope we have you as a subscriber soon."

As If That Weren't Bad Enough!

A few days later, one of the regulars on our pro-gsplus BBS uploaded a new program to the data library. He described it as a public-domain text editor that looked quite a bit like EGOed. So, I took a look at it and, indeed, it did look a lot like EGOed. EGOed was mentioned in the program credits and it even had the exact same "Info" dialog as EGOed. Certainly, I was flattered. Here was a subscriber that was actually learning from what we were doing and putting that knowledge to good use! (This is one of the main reasons that I started *GS+* Magazine.) Since his phone number was not included in the program or documentation, I decided to look him up in the subscriber database and give him a call. The funny thing was, he was not in the subscriber database. The other funny thing was, we don't sell *GS+* Magazine on newsstands. So, I began to wonder exactly *how* this person got the EGOed source code. I certainly did not like the answer I came up with.

At This Point

Some of you are probably thinking, "Oh, great. Diz is just another of these self-righteous pinheads that's gonna threaten to fold his IIGS support and move on to greener pastures unless the universe straightens up and flies right!" Well, I'm not.

The reason I told you these stories is that, usually, when you read about software theft, all you hear are *generalities*. What I wanted to do is to give some *concrete* examples of how software theft actually affects the attitude of a publisher. Each of these incidents was *very* depressing and left an incredibly bad taste in my mouth. For a couple of days following each incident, I was less than enthusiastic about what I am doing here and I spent more time trying to figure out what to do about the problem than I did getting any work done. And, yes, some of my time was spent pondering the question, "Is it worth it?"

Software theft hurts. It hurts the IIGS, it hurts us, and it hurts you. Please don't do it.

Diz **GS+**

We're Working On It!

Features

Working with The Toolbox Part 3
Protecting Your Investment - A guide to surge suppression

Programs

Transfusion v2.0 - support for internal modems and a few other tricks. . .
Texture - You'll just have to wait and see what this one is!

Reviews

The Software Bargain Bin - A series on the best IIGS Shareware
Space Shuttle Word Problems
Jungle Safari
Photonix II
ProSel 16
ProDev DDT16

GS+ Magazine

May-June 1991
Volume 2, Number 5

STEVEN W. DISBROW
Publisher, Editor, Layout

NOREEN RIBARIC
Associate Editor, Layout

JOSEF W. WANKERL
Technical Editor

SUSAN THOEMING
Production Manager

GS+ is published bimonthly by:
EGO Systems
3535 Mountain Creek Road #A-17
Chattanooga, TN 37415-6734
(DO NOT SEND MAIL TO THIS ADDRESS—USE FOR UPS AND OVERNIGHT DELIVERIES ONLY!)

Subscription rates - Magazine only:
1/2 year (3 issues) - $8
1 year (6 issues) - $15
Subscription rates - Magazine w/Disk:
1/2 year (3 issues) - $20
1 year (6 issues) - $36
Tennessee residents add 7.25% sales tax.
Add $1.50 per issue if you want First-Class delivery.
Canadian and Mexican orders add $1.50 per issue.
Other foreign orders add $1.50 per issue for surface delivery or $5 per issue for Air Mail.

Send subscription orders, ads, inquiries, and address changes to:
GS+ Subscription Services
c/o EGO Systems
P.O. Box 15366
Chattanooga, TN 37415-0366
or call (615) 870-4960
Monday-Friday 9 a.m.-6 p.m. EST
pro-gsplus BBS (615) 875-4607
2400 Baud/8 data bits/no parity/1 stop bit

GS+ can also be contacted on these online services:
America Online: GSPlusDiz
Delphi: GSPlusDiz
ProLine/InterNet: diz@pro-gsplus.cts.com

If you have a submission for *GS+*, send it to:
GS+ Submissions
c/o EGO Systems
P.O. Box 15366
Chattanooga, TN 37415-0366

GS+ is produced on an Apple IIGS using GraphicWriter III, EGOed, AppleWorks GS, and an Apple LaserWriter IINT.

CONTENTS

ARTICLES

PROGRAMS

Products marked with an asterisk (*) were review copies provided by the publisher.

REVIEWS

DEPARTMENTS

Letters

Dear Steve,
I'd like to comment on your interview in your last issue [*GS+* V2.N4] with Dave Hecker of Seven Hills Software and his response to your question "There has been a bit of controversy about how Independence and Harmonie (from Vitesse) printer drivers came into being. Give us your side of the story." Dave responded "Our record of integrity speaks for itself. Let's just say that our business philosophy differs 180 degrees from theirs."

As you know Steve, Vitesse is well respected in the Apple II market. We have not sat on our development laurels, nor do we have a reputation to bury our heads if someone launches a missile. I don't quite understand what Dave Hecker's comment implies, but I think I am obligated to let you hear the verifiable story of how Harmonie came about.

Bill Heineman was developing printer drivers for the Apple IIGS for Virtual Realities. Bill approached Kevin Johnson (of Vitesse) and said that he was not getting the cooperation he had hoped and that Virtual Realties had made a deal with Seven Hills Software. He said he wasn't sure the drivers would ever get to market. He *asked* if Vitesse was interested in marketing the software. Kevin asked me to meet with Bill.

Bill told me that he could not get a commitment from Virtual Realties. He also said that Seven Hills had approached him directly and said they may be interested in having him write drivers for them. Bill asked me if Vitesse would market the printer driver software. I asked him if he had signed any contracts or made any verbal commitments. He stated that he hadn't. I told him I would draw up a development agreement and meet with him the following week. When we met he said that Seven Hills had sent him a royalty agreement. I told him that he had to choose, but if he was going to develop printer drivers for Vitesse, it would have to be more than just the HP compatible. He said he wanted to work with Vitesse because of the excellent reputation we had [gotten after being in the Apple IIGS market] for such a short time. Bill made this decision.

Steve, you as much as anyone should know that Vitesse's business philosophy is nothing other than "honorable." So again, I don't know what Dave Hecker means when he says "Let's just say that our (Seven Hills Software) business philosophy differs 180 degrees from theirs." I am under the impression that Seven Hills Software is a very fine company, and to my knowledge they have always had a unquestionable integrity.

I just wanted to express my view for the record.

Jim Carson
President
Vitesse, Inc.
LaPuente, CA

Dear *GS+*,
I just got through reading my first issue of *GS+* [V2.N3] and was it ever timely! Especially the interview with Jim Carson of Vitesse.

But, I felt compelled to respond to some minor errors. For instance . . . Names and their correct spelling are very important to the egos of the people being reported on. For instance, [in the portion of the conversation referring to InWords], it's "John Obberick," not "John Oberick," "Alan Bird," not "Alan Berg" and "Rob Renstrom," not "Rob Renson." Other than that, your conversation with Mr. Carson about InWords and WestCode software is fairly accurate . . . .

Chuck Newby
San Diego, CA

Thanks for setting us straight. We certainly know what it's like to have your name mispronounced and misspelled. (Can you say "Disbrow," "Ribaric," "Thoeming," and "Wankerl"? Neither can we.) It should be noted that the fault is mine and not *Mr. Carson's. The interview was conducted over the phone and I simply neglected to follow up on finding the correct spellings of these names. My apologies.*

Dear Steven,
We [Roger Wagner Publishing, Inc.] have a new policy for disk-based publications; our [HyperStudio] licensing fee for them is $100/year. This is much more reasonable, we feel, than $1200 per year (if you were paying $100 each for a monthly magazine).

Additionally, though HyperCard IIGS doesn't have a licensing fee, it also doesn't have a run-time version of the program: it's impossible to make a HyperCard-based shell for a publication unless the subscriber already owns HyperCard IIGS, due to the fact that a HyperTalk program can access any of the menus, anyplace in the HyperCard program.

Finally, you have mentioned that you tried producing *GS+* volume one, number two in HyperStudio and ended up with a stack over 300K in size, and without the entire contents of the magazine! This problem can be overcome with a little work in HyperStudio—by using several stacks, common backgrounds, and disk-based text items, it can be done. (Witness *HyperBole*, a HyperStudio-based literary magazine published by A2-Central, for example.)

Eric C. Mueller
Manager, Software Engineering
Roger Wagner Publishing
El Cajon, CA

Dear *GS+*,
I love your magazine, and eagerly await its arrival. I hope you are considering publishing it monthly in the near future! Also, EGOed has been a very useful program. Enough of the praise, now my complaint. *inCider* had given a so-so review of 2088: The Cryllan Mission, and

so I decided to pass. Then I read your review (actually it was a review of 2088 part 2). Your last line, "I highly recommend both the first and second scenarios of 2088" prompted me to go out and buy them, since previous reviews by the *GS+* staff had up until now been very consistent with my own evaluation. But boy was I disappointed with 2088. I realize that everyone has different tastes, but such an argument is no defense for your review. No IIGS-specific program with sound and graphics far inferior to many IIe programs should be rated better than +/- [sic]. If you liked the game because personal preference for that style of game, then it is OK; but your review should objectively state the poor quality of the game's graphics and sound (as in Apple *GS*). Sorry for the long complaint. Overall, your magazine is great, and I enjoy it over *inCider*, *Nibble*, etc. I'm glad you are adding advertisements—I enjoy reading about new products.

Rodney J. Avilla
Hilmar, CA

Dave Adams replies:
I am sorry that you did not agree with my review. I still feel that both scenarios of 2088 are great games and are worth buying. I inferred that you have not read my first scenario review. In it I stated that the dungeon graphics "are not really that great" and "could be improved upon." I stated in the second scenario review that the graphics "have been improved over the first scenario" and "are more detailed." I agree that neither Scenario is likely to win an award for graphic excellence. I do not think that the graphics take away anything from the game play. I didn't mention anything in the sounds department simply because there aren't that many sounds in the game. I guess that I never noticed that as a deficiency although I see now that it could be. I understand what you mean about the poor graphics. That is one reason why I didn't buy the game for a long while when I saw it at my local software store. It didn't look that interesting and I assumed (there is a good saying about assuming things that might apply here to me) that the game was not worth buying. At AppleFest I saw the 2088 demo and talked with the programmers from Victory Software. They gave GS+ *a copy of the demo and Diz let me take it home to play with until I bought a copy of 2088. I played the heck out of that demo. When I finally received the actual game, I threw myself into it and had a roaring good time. I scraped together enough money to buy the Second Scenario as soon as I could. You will notice that these are not review copies. I paid for them out of my meager paycheck. I still recommend them as great games.*

Which brings me to my point. I recommended these games based upon the strengths of the gaming system *and the* fun *I had playing them. I still rate the desktop interface used by these games as a major asset. Compare restoring the health of a player in 2088 and in the IIe version of Pool of Radiance. All "housekeeping" tasks in each 2088 scenario are incredibly simple. I also feel that each game's weaknesses are more than compensated for in the actual game play and contributions to the game genre. Quite simply, you are going to be hard pressed to find many games that are easier to play than 2088. It is a well-designed and well-executed program. Although neither game is perfect (read my remarks about the end of the first Scenario and my gripes about the second) they are solid games. I never noticed these deficiencies as extremely major faults, simply because I was having too good a time* playing the game.

I understand your frustration. I am tired of having to play the IIe version of Bard's Tale III, and all of the SSI AD&D games while most IBM users are enjoying VGA graphics. I hate seeing rows full of IBM and Amiga game software at the mall and one section of 5.25-inch stuff in the discount rack or clearance table for Apples. That's why I'm thrilled when I see a game written specifically for my IIGS that takes advantage of my machine's capabilities. In this day when IIGS developers can be as hard to find as Iraqi armor, I appreciate that. I stand by my recommendation.

Dear *GS+*,
I find myself using EGOed more and more. The idea of a full fledged word processor as an NDA is intriguing. I have a couple of questions though . . . How can you set bottom and top margins? How can you set right and left margins? Can you print in columns?

I figured that you must be able to do these things as you use it to edit the magazine.

Charles E. Garrett
Hanford, CA

Actually, EGOed can't do any *of those things! We use EGOed only to edit the* text *of the magazine. We save the text out as TeachText and then use our TeachText Translator for GraphicWriter III [GS+ V2.N3] to import the TeachText files into GraphicWriter III with their font, size and style information intact. GraphicWriter III is where we set up all of the margins, columns and other elements of the* GS+ *page layout.*

Dear *GS+*,
I have all of your issues except V1.N4, which features a desktop color CDev. I understand the magazine is sold out, but is there any way of getting a copy of the CDev please?

Maziar Maniei
Morley, W. A.
Australia

Even though the magazine is sold out, we still sell the disks. The price for a single disk is $6.50. For more information, see "GS+ Back Issues" in this issue.

If you have a question, comment, or criticism about *GS+* Magazine, we want to hear it! Due to space limitations, we cannot answer every letter here in *GS+* Magazine. If you want a personal reply, please enclose a self-addressed, stamped envelope. Please address all letters to:

GS+ Letters
P. O. Box 15366
Chattanooga, TN 37415-0366 GS+

Interview With Matt Deatherage of Apple DTS

Believe it or not, Matt Deatherage started at Apple Computer as an summer intern in 1988. During his brief stay, he did such a good job that Apple asked him to stay on permanently. Matt went to work for Apple Developer Technical Support (DTS) and, since then, his name has become a fixture in the Apple II development community. If you pick up an Apple II Technical Note, or look at the liner notes in an Apple II reference manual, chances are Matt's name will be there somewhere.

Before we begin the interview, Matt asked us to remind you of two things:
1) These are exclusively Matt's opinions and *not* the opinions or policy of anyone at, near or who has heard of Apple Computer, Inc., much less those of the company itself.
2) Send your résumés to Apple's Human Resources Department—not Matt.

GS+ How and when did you first get involved with the Apple II?

MD I started getting interested and involved with microcomputers when I was in high school, believe it or not. Our school's computer lab had TRS-80 Model III computers, but our counselor (who had been my speech teacher the previous year and was a good friend) got an Apple II Plus for his office (this was in the fall of 1982). I spent a lot of time on both computers when our family decided to get a computer (mainly for me) the following Christmas.

The reason I chose the Apple IIe ([which] I still have upstairs) over the TRS-80 Model III? I was (and remain) heavily involved with music, and I could hook a synthesizer to an Apple II and play and print music. No other computer could do that. It also had color, and more memory, and a better keyboard, and more software. It just turned out to be the better deal.

It's kind of my equivalent of Mike Westerfield's chess program (legend has it the original ORCA/M was written so Mike would have the tool he wanted to write a chess program). Today I have that IIe, a IIc Plus, a very well-stocked IIGS, a Mac SE and a Mac IIfx as well as two printers, a modem, assorted disk drives and an AppleTalk network. I also have a clarinet, a bass clarinet and a grand piano—but I never bought a synthesizer or music transcription software. Maybe someday.

GS+ Tell us about your job at Apple. What exactly does a Developer Technical Support person do?

MD The definition of a DTS engineer has changed over the years. The current definition involves answering technical questions that Apple Partners send in through electronic mail, writing technical notes and sample code, reviewing technical documentation (like *GS/OS Reference* or *Toolbox Reference Volume 3*) and sitting on project teams to represent developers (and DTS's) technical needs to the engineers creating products.

I believe sitting on project teams is one of the most important functions DTS engineers have. Most of DTS is Macintosh-oriented because that's where most of the work is, and that leads to an overall DTS focus on e-mail. That means we have to be extra careful to make sure the Apple II gets the attention it deserves, since Apple II developers have never been big users of DTS support through e-mail. But we still get a lot in for Apple II developers by sitting on project teams.

For example, when Standard File was being rewritten, even though developers had been assigned file type and auxiliary type combinations for a long time, the new Standard File was still only going to match by file type—if you wanted to check the auxiliary type, you'd have to write a custom filter procedure. I didn't think this was right, and I worked with the engineering team to change the definition to include not only file types and auxiliary types, but a new kind of flag word that lets you specify whether to display, display dimmed or not display all files with those combinations. This new behavior in Standard File drastically reduced the number of instances where filter procedures are required, and it's an example of how DTS can help make Apple's products better for developers.

DTS also assigns all file type and auxiliary type values in the ProDOS file system (and file/creator types in HFS). We host debugging rooms at major conferences (like the Apple Worldwide Developer's Conference, the A2-Central Summer Conference and AppleFest), and some of us hang around on online services in our spare time.

GS+ What is the most frequent problem that developers report?

MD We try to make sure that there aren't frequent problems for developers to report. When lots of developers ask the same questions, we try to publish an answer in a public forum, such as a Technical Note, Q&A in *develop* or the new AppleLink Developer Technical Assistance library (DTA), or maybe a brief discussion in a developer mailing.

About the only things we know about that we don't publish are bug reports—with few exceptions, we don't publish lists of known bugs in the system as a matter of policy. We'd have to update it every time we found another bug, and most of the bugs that we know about are really minor and trivial, yet their presence in a published list would lower people's confidence in the system. So occasionally we'll get the same bug report from a bunch of developers, but even that doesn't happen very often.

GS+ In what areas can we expect to see the "synergy" between the Apple II and the Macintosh increased?

MD I think we'll see more leverage off those standards that Apple has pioneered and driven throughout the industry, such as AppleTalk. There are also Apple-

specific things which are still pretty neat (some peripherals and file systems) which we haven't seen on the Apple II yet, or in a while, that would be useful. There are also many common peripherals between the product lines, and we can expect that strategy to continue.

As for "synergy," hopefully that will come for free. As long as the left hand knows what the right hand is doing, Apple should make enhancements to our *entire* computing platforms that can benefit all users. It doesn't always work that way, but it's very nice when it does. Macintosh System 7.0 includes the ability to share files on your hard disk with other people on an AppleTalk network—and since it works by making part of your hard disk function as a miniature AFP server, it's fully accessible from any Apple II AppleShare client (IIe or IIGS). That's the way it works, and that's the way it ought to work.

GS+ IIGS software releases have slowed during the last few years. In your opinion, what is the current state of the Apple IIGS market?

MD You have to remember that I'm not in marketing, I'm an engineer. I know a lot about what goes on in the Apple II world, but I'm not an Apple II marketing expert. Fair warning.

There are two basic kinds of software sales—sales to new computer buyers and sales to the installed base. Many people think the former is the most important, since new computer buyers basically have nothing to do with their expensive home companion and will buy a comparatively large number of software packages in a relatively short time. These sales have suffered as new Apple II sales have declined over the past few years, and this is the main reason why people are constantly yelling at Apple to move more systems—the more systems we sell, the more software is sold and the more viable the platform is as a whole.

But there's an entire other arm to software sales—the installed base. That's us—the approximately one million Apple IIGS owners who have purchased systems over the past four and a half years. This software market isn't doing so well, either.

Why? I really don't know. On other personal computer platforms, a software title is considered a raging success if it can get a 25% penetration (where 25% of the installed base has purchased that software). On the Apple IIGS, with the exception of AppleWorks and AppleWorks GS, a software program is extremely lucky if it can get one or two percent of the installed base.

It's an exceptionally low figure, and it has not gone unnoticed by software publishers. Even when Apple was doing a better job of selling the IIGS back in 1987 and 1988, software publishers weren't doing so well. Think back to the classic Apple IIGS programs that were raging successes a few years ago when the IIGS was introduced, and think about how many of those companies are still producing IIGS software. If they were making money on it, they'd still be doing it.

The basic fact is that Apple IIGS owners don't buy software, and we all suffer for it. It could be because they just don't get around to it, or it could be because there's still an all-too-thriving pirate community out there, patting themselves on the back for getting the work of others for free while driving the creators out of business. I don't know. I do know that Apple selling more systems alone wouldn't be enough to turn the Apple IIGS software business around—more people than the newly initiated have to want to buy the product.

The exception to this market picture is the education market, where Apple II software and hardware sales remain strong.

GS+ What markets do you think Apple IIGS developers should concentrate on? Games? Productivity? Business?

MD Games and productivity software are both widely wanted by Apple IIGS owners, but their desire seems to evaporate when it comes time to purchase the software. If you want games or productivity software, take the time to write a letter to companies that have published the stuff you've liked before and tell them you want to purchase more of their IIGS software. If they've already decided that people won't buy their IIGS offerings, they're not going to change their mind without good reason, like lots of people telling them they're wrong.

Business software just isn't a big market for Apple IIGS computers these days, with the marginal exception of some business-at-home applications. It's a market that the Apple II doesn't play in very well, and that's not likely to change any time soon. It may not be what we want to hear, but it's true, and the amount of effort that would be required to change that perception is so herculean that I can't believe it wouldn't be better spent on other pursuits.

The fact is that most of the Apple IIGS systems being sold today go into schools, and educational software remains in strong demand. Schools particularly want software that's written *today* and takes advantage of things like ProDOS, AppleTalk networks, video overlay and SCSI peripherals like CD-ROM and scanners. I think every school and every Apple IIGS owner has had enough of old software that doesn't work correctly with some peripheral or another because the author didn't think it was important to him, even though compatibility guidelines were clearly and easily available.

GS+ What advice do you have for beginning Apple IIGS programmers?

MD The first thing any prospective Apple IIGS programmer must remember is that the Apple IIGS is not programmed like an Apple II—there is no simple solution built-in.

If you like playing around in Applesoft BASIC, the best choice for you to enter IIGS programming may well be HyperCard IIGS. It's easy to use, it gets good results quickly, and it's quite approachable. It's the same thing to the Apple Desktop Interface that Applesoft BASIC was to programming in 1977—it's the way that you, the non-guru, can get to the power locked inside that little platinum box.

If you do have a fair amount of programming experience, you might want to dive in with an advanced BASIC, or Pascal, or C compiler and write stand-alone Apple IIGS programs. Be prepared if you want to do this—you won't learn it quickly and it won't necessarily be cheap. Not only will you need a compiler, you'll need the reference manuals (not someone else's interpretation of them, but the real manuals) and a suitable system to develop on, and that might include more memory and a hard drive.

The one thing I hear from beginning programmers all the time is "Why do I have to buy *Toolbox Reference Volume 3*?" Or *GS/OS Reference*, or the *Human Interface Guidelines*, or any other reference book that's inconvenient this week. To them, I ask in return, "Would you trust your system and your data to a program written by someone who wouldn't even bother to read the manual?"

Remember, anything you want to do on your own system is your business. Break any rule—heck, break *all* the rules! Learn whatever you want. That's part of the fun of programming. But once you give that program to *anyone* else, you're no longer playing with your computer. You're playing with someone else's computer, and you're expected to behave yourself and do everything you can to make your program work correctly. If you've ever had a program crash on you at a bad time, you know exactly what you hope to avoid by following the rules.

And then, when Apple does revise the system software to add whizzy new features, if you followed the rules you won't have to revise your software to take advantage of the new stuff. The people who wouldn't use the tools or follow the rules won't be so fortunate.

GS+ Recently you worked on the Apple II Guide. Tell us a little about that project and your involvement with it.

MD The Apple II Guide was started and written the first few times by Emile Schwarz of Apple France. Last spring Jane Lee and Rajiv Mehta (Apple II product manager at the time) decided to make it an international project—Cupertino would arrange for most of the book to be written, and the international divisions (Apple USA, Apple France, Apple Australia) would insert the portions about dealers and other items that were country-specific. I volunteered to write the article on the history of the operating systems.

I got that written (and only munged one major thing—I attributed Wendell Saunder's daughter, Sara, as Dick Huston's offspring. Oops!) and also helped review the rest of the articles in the Guide for technical accuracy, relevance and consistency. It was an interesting project and I'm glad it was well-received—I think the book contains lots of useful information that should be easily available.

GS+ Is there anything else would you like to say?

MD Sure. How many pages do I get?

I've been doing Apple II DTS for over three years, and for the most part it has been a lot of fun. It's never pleasant to explain something in plain English (so it won't be misunderstood) and get in return someone's pent-up hostilities over Apple's treatment of the Apple II, but it happens. I'm willing for it to happen to me every now and then—hopefully everyone who vents to me is someone who doesn't vent to Jane [Lee] or Ralph [Russo], who need the time to get something done instead of replying to apologize for past mistakes (some of which continue to ripple through all our lives).

A lot of times I think it would be easier to just keep a lower profile—don't be on online services, attend conferences, etc. But that's not an effective way to communicate what's going on. It gets *really* frustrating trying to tell people that the world is not coming to an end when you can't reveal anything that could affect Apple's Apple II business proposition.

I personally even dislike pre-announcements—all that happens when something is pre-announced is that people flood the community with questions about how it works and when it will be released. They're honest questions and they deserve honest answers, but what winds up happening is that the DTS and engineering folks wind up trying to explain documentation that just isn't available (or isn't written yet), or explaining why schedules won't permit something to be added even though release is a few months away. It's frustrating all around, and works much better if we have documentation available when the product is ready, which is what we target. I spent a lot of time in 1989 and 1990 reviewing our new Apple II documentation (*Toolbox Reference Volume 3*, *GS/OS Reference* and *GS/OS Driver Reference*, mainly) to make sure they were complete, accurate and helpful. It's enough work that it has always really bugged me that people won't buy these books and instead want other people to type in function descriptions—if we didn't want you to have the information, we wouldn't have made the book available.

Even though the job is often incredibly frustrating (sometimes for reasons that can't be discussed, which means I can't even vent my anger to anyone), it has also been rewarding. It's been my pleasure to work with an incredible group of people, both in DTS and in Apple II System Software. I've been proud of the things I've been able to accomplish, such as reorganizing the Apple II file type situation, revising the Apple II Technical Notes in 1988, and I have really enjoyed helping to improve Apple's system software and manuals through my input (at least, I hope it's improved).

I'm as ambivalent about what the future brings as anyone else I know. I do Macintosh support these days because it's part of my job in DTS—I don't know whether I'll continue down this path or change jobs somehow to be a full-time Apple II engineer in one aspect or another. I think I'm ready for broader and newer challenges, and I'd like them to be on the Apple II platform. I think I still have the capability to bring good things to Apple's Apple II developers and customers, and as long as Apple will pay me to do it I probably will. It's not often you have a chance to make a real difference in the world, so change everything you can while you have the chance. **GS+**

Working With The Toolbox

By Josef W. Wankerl

Part 2: The Memory Manager

Memory management is a big part of correctly creating an application on the Apple IIGS that will work today and, better yet, tomorrow as well. Memory management is controlled by a tool set called the Memory Manager. In the "old days" of the Apple II, each application had total control of memory. The application didn't have to worry about stepping on something it shouldn't. Now, because there are multiple programs in memory (applications, NDAs, CDAs, and others), and each of those programs are competing for memory, the Memory Manager steps in to arbitrate who gets what. If the Memory Manager is not used prperly, it is almost certain that the machine will crash.

Using the Memory Manager is quite easy. There are only a handful of calls that can be made. But before I describe the calls, I will talk about when to use the Memory Manager and describe Apple IIGS memory. I won't be totally detailed in my descriptions and duplicate *Apple IIGS Toolbox Reference: Volume 1*—you should already have that book and you should read the Memory Manager chapter carefully.

When Should You Use The Memory Magager?

Most of the information presented in this article is to get you familiar with Apple IIGS memory management. If you are programming in a high-level language like C or Pascal, you shouldn't have to worry about the majority of these calls since the compiler's run-time package will handle all your memory allocations for you. If, however, you are working with assembly language, understanding memory and the Memory Manager is vital for your programs to survive. Be aware that the only time you will ever need to allocate memory is when you are using dynamic data structures, or if you are using certain Toolbox calls that require memory to operate on. For example, some tool startup calls require some direct page memory space—it's the application's job to allocate that memory and tell the tool where it is. The Memory Manager is invaluable in such a situation. When your program needs memory for another item in a linked list, you should probably use the conventions of the language you are working in to get it (like `New()` in Pascal, or `malloc()`/`calloc()` in C). If you are working in assembly language, use the Memory Manager's `NewHandle` call.

Blocks Of Memory

If your application needs some memory, it should ask the Memory Manager for it. Memory is allocated for you in *blocks*. A block is a contiguous space of memory which your program is allowed to manipulate in any way. Every block of memory has a *size* (how large you want the block to be in bytes), an owning *user ID* (which tells who owns the block), some *attributes* (describing various properties that the block has), and a *starting location* (which tells where in memory the block starts).

Be The First Kid To Own Your Own Block

The user ID assigned to a block of memory tells who owns the memory. Your application can obtain its user ID by making the `MMStartUp` call. In most every case you should use the user ID returned by the `MMStartUp` call (or a derivative of it) to allocate new memory. There is a Miscellaneous Tool Set call, `GetNewID`, which generates a new user ID, but this call should be used with discretion. You can make a `MMStartUp` call *at any time* to get the current user ID.

User IDs are fully described in *Apple IIGS Toolbox Reference: Volume 1* on page 12-10 (be sure to note the correction on page 36-2 in Volume 3). Basically, a user ID is comprised of three parts: its *type*, *auxID*, and *mainID*. The type describes what kind of application owns the block. The mainID is used to uniquely identify which application owns the block. The auxID is used to classify memory within a mainID. Note that all memory is always uniquely identified by the type and the mainID; the auxID is used for distinctions within this unique identifier.

Auxiliary IDs

The use for auxIDs might not be immediately obvious, so I'll use an example to help clear things up. Note that you don't even have to fool with auxIDs, but they make some situations a bit easier to deal with. If you have an application which is a desk accessory, its type will be $05, its mainID will uniquely identify the memory from all other desk accessory memory (for this example we'll assume that the mainID is $xx), and the auxID will further break down memory within your allocations. You can have auxID $1 be for control information, auxID $2 be for linked list data structures, and auxID $3 be for error message data. Without an auxID, if you suddenly wanted to get rid of all the linked list memory blocks, you would have to go through them one by one and dispose of them. If you used an auxID, you could dispose of all blocks with type $05, mainID of $xx, and auxID of $2. The Memory Manager would go through its list of memory blocks, and everything that matched your specifications would be disposed of. An auxID of $0 acts as a *wildcard* that matches *every* auxiliary ID. So, if our hypothetical desk accessory tried to dispose memory of type $05, mainID of $xx, and auxID of $0, *all* of its memory would be disposed of—no matter what the aux ID is! This would also dispose of the memory that the desk accessory lives in!

Attributes

Each block of memory has some attributes associated with it. You can get more information about attributes from page 12-12 of *Apple IIGS Toolbox Reference: Volume 1*. The attributes describe certain properties that the block has. There are eight different properties that a block can possess: *attrLocked*, *attrFixed*, *attrPurge*, *attrNoCross*, *attrNoSpec*, *attrPage*, *attrAddr*, and *attrBank*. The simple attributes of attrNoCross, attrNoSpec, attrPage, attrAddr, and attrBank are

described fully on page 12-13 of *Apple IIGS Toolbox Reference: Volume 1*. I'll only describe the attrLocked, attrFixed, and attrPurge attributes, as they are the ones that cause the most confusion.

Fixed Memory

A fixed memory block *cannot* move. It will always stay at the same location. Using fixed memory, however, makes memory management a bit more difficult. The more fixed memory there is hanging around in the system, the higher the chance of memory fragmentation. The only reasons I can think of to use fixed memory is when you have program code or you need to allocate a direct page. Using fixed memory for data is *not* a good idea. Also note that attrFixed, attrAddr, and attrBank attributes are independent, which means that you can allocate fixed memory, but not have it start at any predetermined location. Also, to help clarify the next sections, a "locked, fixed" memory block and an "unlocked, fixed" memory block are pretty much the same unless the unlocked block also has a purge status—then the block can be purged.

Locked Memory

When a memory block is locked, it *will not* move. What makes this different from a fixed block? Well, when the block is unlocked, it *can* move. This makes memory management *much* easier for the Memory Manager because memory fragmentation can be reduced. Virtually all memory that your program allocates should not be fixed, but should be locked instead. You should keep the memory block unlocked until you need to use it, then you should lock the block, work with its contents, then unlock it again.

Purgable Memory

The attrPurge attribute gives unlocked blocks the chance to be purged. When a block is purged, its size is set to zero and the contents of the block are lost. There are four purge levels: 0, 1, 2, and 3. Purge level 0 means that the block cannot be purged, no matter what the locked attribute is. Levels 1, 2, and 3 mean that the block can be purged only if the block is unlocked. Blocks with level 3 will be purged first, and blocks with level 1 will be purged last. The reason to make blocks purgable is to keep memory that isn't needed anymore, but may be later, around as long as possible. If the memory is needed for another purpose, it is allowed to be purged. Blocks are purged only when a request for more memory is made and there isn't enough memory left to satisfy the request. Page 36-2 of *Apple IIGS Toolbox Reference: Volume 3* describes how low- and out-of-memory situations are dealt with.

Size

The size of a memory block describes how large the block is in terms of bytes. A memory block can have a size of zero bytes, meaning that the block is empty. An empty block doesn't have much of a use until it is resized. When a block is resized, its starting location remains the same. Only blocks that are unlocked can be resized. There are two cases to consider when resizing a block: making it smaller or making it larger. Making the block smaller is the easiest case—the size is reduced so that any information that was previously at the end of the block is lost. Both fixed and unfixed blocks can be reduced in size easily. The difficult case is when a size is enlarged. When the size increases there may be a conflict because the end of the block would overlap an already existing memory block. If you are resizing a fixed block and the conflict occurs with another fixed block or a locked block, then nothing can be done and the resize will return with an error. If you are resizing a nonfixed block, much can be done to prevent the out-of-memory situation.

Starting Location

The final attribute that a memory block has is a starting location in memory. Although this may seem ridiculous to talk about, it's actually very important. Blocks that are not fixed and unlocked may move around in memory. When your application needs more memory, or when a handle size needs to be increased, there may be a situation in which nonfixed, unlocked blocks of memory will move. Because blocks of memory may move, your application needs a way to figure out exactly where in memory the block starts before it can use the block. The Memory Manager does this by maintaining a *handle* for every block that is allocated. A handle, basically, is a pointer to a pointer. A handle is *guaranteed* to remain at the same memory location—it never moves. The handle contains a pointer to the start of the memory block. Whenever a block of memory moves, the pointer contained in the handle changes to reflect the new position of the memory block. Before your program can use the memory, it must *dereference* the handle. Dereferencing simply means to find the start of the memory block by looking at the pointer contained in the handle. Note that you should always lock the memory block before you dereference the handle and work with the memory. This prevents the memory block from moving while you are working with it. The following code snippets shows how to dereference a handle in various languages:

C:

```
Pointer = *Handle;
```

Pascal:

```
Pointer := Handle^;
```

Assembly:

For assembly language, I've written a short routine (shown in Figure 1) which dereferences a handle for you—call it with two words of result space and the handle you want to dereference on the stack and when the routine returns simply pull the dereferenced pointer off the stack.

Making Memory Manager Calls

Now that you know just about everything there is to know about blocks of memory and memory handles, it's time to describe some of the basic Memory Manager calls. As with every tool set, `MMStartUp` should be called at the start of your application and `MMShutDown` should be called when your application finishes. Desk Accessories, CDevs, and other programs that aren't real applications need only make the `MMStartUp` call to obtain their user ID—`MMShutDown` is not necessary. I am not going to describe every Memory Manager call—*Apple IIGS Toolbox Reference*, Volumes 1 and 3, have already done this for me. The calls are fairly straightforward and you should be able to figure out exactly what they do now that you know about handles and

```
* == Dereference A Handle == *
Deref       START
DataBank    equ         $01
ReturnAddr  equ         DataBank+2
Handle      equ         ReturnAddr+2
Pointer     equ         Handle+4

            phd                         ;Save current direct page
            tsc                         ;Get stack pointer in accumulator...
            tcd                         ;...then make the stack the direct page

            ldy #$0002                  ;Prepare to access the high word
            lda [<Handle]               ;Get low word of master pointer
            sta <Pointer                ;Save low word of master pointer
            lda [<Handle],y             ;Get high word of master pointer
            sta <Pointer+2              ;Save high word of master pointer

            pld                         ;Restore old direct page
            plx                         ;Get return address off stack
            pla                         ;Remove handle low word from stack
            pla                         ;Remove handle high word from stack
            phx                         ;Put return address back on stack
            rts                         ;Return to caller
            END
```

Figure 1 - Dereferencing a Handle In Assembly Language

memory blocks. Some calls may be a little misleading, though, so I'll walk you through them.

NewHandle

To get some memory for your application, you issue a `NewHandle` call. The call then finds some free memory and assigns it to your application. The `NewHandle` call returns a handle to the memory it allocated. Most Memory Manager calls take a handle to identify the block to work with. The terms *block of memory* and *handle* are very interchangeable since the handle describes the block of memory. In fact, the handle contains more than a pointer to the block of memory, it is a data structure in which the very first element is a pointer to the block of memory. Other elements in the handle structure contain the attributes, user ID, and size of the block; however, this data structure is not documented by Apple because it is subject to change in the future. You should strictly use Memory Manager calls to get information on handles and not access the individual fields of the handle record.

When you are totally done with a block of memory, you should make a `DisposeHandle` call on it. The `NewHandle` and `DisposeHandle` calls are probably the most frequently used Memory Manager calls. The other methods of dealing with handles that are no longer needed are discussed in the "Keeping The Memory Manager Happy" section, although the most common way of dealing with unwanted handles is the `DisposeHandle` call.

Compacting Memory

The `CompactMem` call is an attempt to reduce memory fragmentations. `CompactMem` doesn't purge any blocks or free any memory, it simply rearranges things so that free memory is grouped into the largest blocks possible. The call can only move nonfixed, unlocked memory blocks, hence the reason to keep as many blocks of memory unlocked as possible. People sometimes think that the `CompactMem` call actually invokes the set of steps described on page 36-3 of Volume 3, but that is simply untrue.

Working With Purged Handles

Page 12-8 of Volume 1 describes purged handles and how to recognize them. There are two calls to get back memory that has been purged: `ReAllocHandle` and `RestoreHandle`. Neither of these calls can get back the data that was purged—that data is gone. The `RestoreHandle` call is an easy way to completely restore the memory block to the way it was before it was purged with the exception that the data is not valid. The `ReAllocHandle` call is basically the same as the `NewHandle` call, except that you must provide the handle (and the handle must have been purged).

Moving Memory

The Memory Manager provides four nice routines that copy bytes from a source location to a destination: `BlockMove`, `PtrToHand`, `HandToPtr` and `HandToHand`. In most cases, these routines will be adequate—you don't need to write your own memory copy routines. I've seen quite a few applications that use their own memory copying routines when they didn't need to.

Keeping The Memory Manager Happy

There are a few things that you can do to

keep the Memory Manager happy. First and foremost, when you are done working with some memory, unlock it! If you are through with the memory, but you might need it later, make the block purgable (but be sure you can recover the data if the block actually gets purged). If you are done with the memory for good, dispose of it. Don't forget to dispose of all your memory when your application shuts down, either. The easiest way to do this is to allocate all your memory with an auxID that is not $0 and then do a `DisposeAll` call on it. If you call `DisposeAll` with an aux ID of $0 then not only would all your data memory be disposed of, but the memory your application resides in will be disposed of as well and that would be harmful if somebody else came along and managed to acquire it before your application was completely through.

May Your Memories Be Pleasant!

That's all there is to it. If you have any specific questions about the Memory Manager (or any other tool set that's been covered) let me know what they are and I'll try to answer them. Also, don't forget to let me know what language I should use to give examples with. This time I used C, Pascal, and Assembly because the example was small—I won't be able to do this in the future. So until next time, have fun exploring the Toolbox! GS+

Advertisers Index

By Steven W. Disbrow

Golden Delicious Computer Services - Page 32

Free Memory? Really? Well, yes and no. It's not free memory *chips*—it's a Classic Desk Accessory that compacts the memory in your IIGS. Ahem. That's all it does. $15? Buy this only if you can't scrounge up one of the dozen or so public domain or shareware utilities that do the same thing. This isn't a *bad* piece of software—it seems to be quite well done in fact—it just does not do $15 worth of stuff.

Learning Experiences - Page 40

It seemed kind of silly to continue to use a RGB monitor on our BBS, so I began to hunt for a monochrome monitor to use instead. Our good friend Greg Zimmerman put me in touch with Rick Slone and the rest of the fine folks at Learning Experiences. A few days later, a Macintosh IIsi box arrived at my door. Ripping open the box, I found that, not only does Learning Experiences have great prices, they have a great sense of humor too! The monitor has worked flawlessly and I couldn't be happier with it.

Raptor, Inc. - Page 45

The products advertised here are *image enhancers* for your black-and-white 320 mode graphics. Although I have not had much time to play with the review copies that Raptor sent us, they seem to do exactly what they promise—and they do it very well! The only *real* problem is that they do it *very* slowly. However, if you need the ability to perform the same kind of processing that NASA does on satellite photos (and then some!), there simply aren't any other products like these for the IIGS.

TMS Peripherals - Back Cover

If you need a hard drive, I *highly* recommend that you buy it from TMS. Several months ago, I desperately needed a 45 MB removable to do backups with (see review in this issue). TMS got it too me in less than 24 hours and I paid about $100 less (shipping included) than the "base sticker price" of just about any of the big name 45 MB removable drives advertised in the MacRags. Since then, the drive has not given me a single problem. Even if the drive *had* been bad, TMS has a Toll-Free support line, a 30 day Money Back Guarantee and a 2 year Warranty!

Part of our job here at *GS+* is to find the best people in the business and point them out to our readers. I was so impressed with TMS' products and service that I practically *begged* them to advertise with us. These guys are some of the best and they truly deserve your business. For more on TMS Peripherals, see the review of the TMS Pro R45 in this issue.

ToolBox - Pages 52, 53

Photonix II is the commercial descendant of the shareware disk-copy utility Photonix. Unlike the original, Photonix II can be installed on a hard disk and can be run from the Finder. Unfortunately, the only way to quit is to reboot. If you can live with that, and you thought the original Photonix was the cat's pajamas, you should get Photonix II. Look for a full review in the next *GS+*.

Space Shark is billed as "the fastest IIGS game ever." See Dave Adams' review in this issue.

Bouncing Bluster II is the commercial descendant of the shareware game, Bouncing Bluster. See Dave Adams' review in this issue. GS+

Autopilot

By Josef W. Wankerl

Description

A "Start" program is the first program run after the boot process is done. On a normal system disk, the start program is the Finder. Autopilot is an enhancement, not a replacement, to your normal start program. When Autopilot is installed, it is the first program to get control of your system when the boot process is done. Autopilot does one of two things: it either automatically launches an application of your choice or it presents a dialog with a list of applications for you to choose from.

I wrote Autopilot because I have more than one environment which I like to work with. When I'm not programming, I like to boot into the Finder. When I'm programming, I like to boot into the ORCA shell. Originally I had a small alternate start program that launched the ORCA shell and I could switch between it and the Finder start program, but I wanted a bit more flexibility. Autopilot was the result. After Autopilot was functional, I found myself booting straight to it instead of having it launch other programs (like the Finder). The number of programs I usually work with is rather small, and Autopilot takes less time to load than the Finder and it provides an easier selection method than the Finder. But notice that Autopilot is *not* meant to replace the Finder! The Finder is still, in my opinion, the best file launching and utility program around.

Limitations

Autopilot can only launch S16 and SYS files. It cannot, at this time, launch EXE files or BASIC files. (I told you it isn't a replacement for the Finder!) I would like to add this capability eventually, but I think this version, even though it only launches two types of files, is beneficial enough to distribute.

Installation

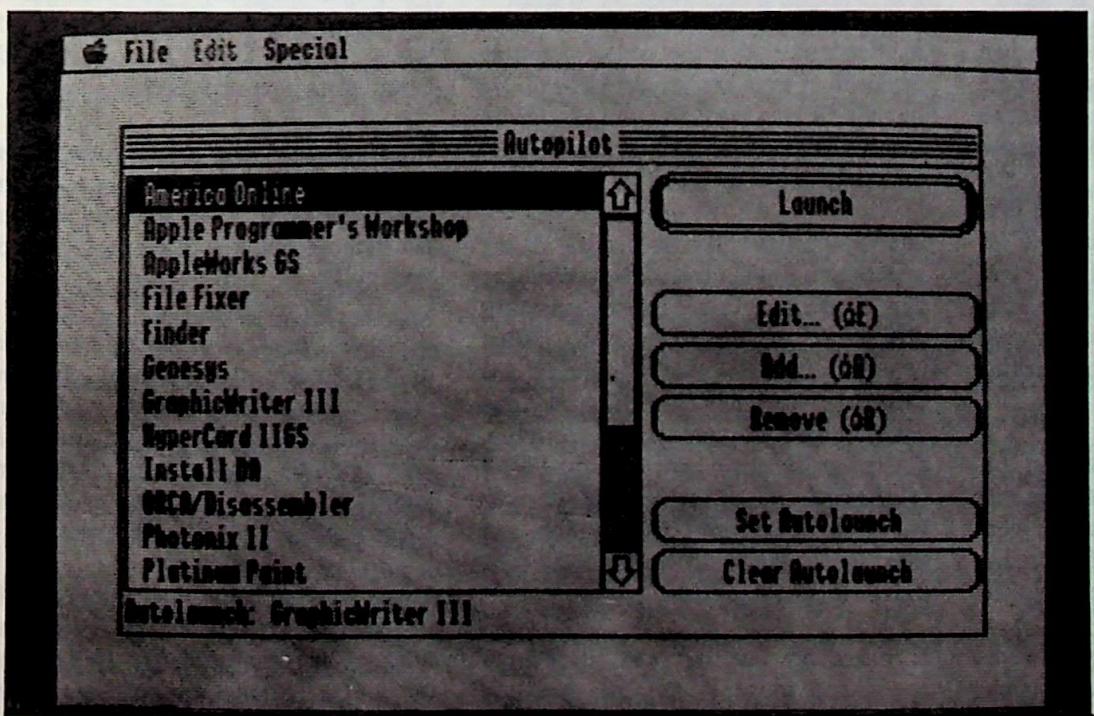
There is *not* an Installer script for Autopilot. The reason for this is that I cannot reliably determine what the current start program is on your boot disk. To install Autopilot, all you have to do is to copy the Autopilot file from the **Autopilot** folder on your *GS+* Disk to your ***:System** folder (where ***:** is the name of your startup disk), then rename your current **Start** program to something else (like Finder), then rename **Autopilot** to **Start**. Unfortunately, you can't rename files when they are open. Applications that use resources, like the Finder, are *always* open. This means that you can't use the Finder to rename itself! Instead, you have to use some other utility, such as NoDOS (see "NoDOS v1.5" in this issue). So, to install Autopilot, follow these five steps:

1) Launch the Installer from your *GS+* Disk and use it to install NoDOS and the Autopilot file (the Autopilot Prep script.)
2) Reboot your computer and run the Installer program from your *GS+* Disk again.
3) Select NoDOS from the Apple menu and pick the Rename option.
4) Rename the ***:System:Start** program to something else (like Finder or Wings.)
5) Rename the ***:System:Autopilot** file to **Start**.

Now Autopilot will be correctly installed and the next time you boot your computer you will come up in Autopilot.

User Documentation

The Autopilot desktop is composed of a standard menu bar and a window containing a list and a few buttons. The menu bar is the easiest to describe, so I'll start there. Under the multicolored Apple menu you will find the usual About menu item as well as all of your NDAs.



The File menu has the normal items of Quit, Close, and Open. The Open menu item displays a standard file dialog box allowing you to select a program that you would like to launch. Clicking on the Open button will then quit Autopilot and launch the application that you selected. The Close item, as you would expect, closes the front window. The main Autopilot window cannot be closed, however.

The Quit item shuts down Autopilot and returns to the application that launched it, just like the Quit item in any other program. It seems rather silly to have a quit option for the start program because it would just quit to itself, but if Autopilot was indeed launched by another program the Quit item is useful.

The Edit menu is provided for NDAs—Autopilot does not use this menu.

The Special menu allows you to either restart or shut down your system. When you select the Shutdown item, all removable disks are ejected from their drives and a restart dialog is presented. When you select the Restart item, your system is simply restarted—no disks are ejected.

The main Autopilot window is dominated by the Application list. You can add applications to the list with the Add button, remove an application from the list with the Remove button, and edit how the application name will appear in the list with the Edit button. Selecting an application from the list and clicking on the Launch button will quit Autopilot and launch the application that you have selected. Alternately, you can double-click on an application in the list to launch it.

Selecting an application from the list is what makes Autopilot easier to use than other program launchers. You can use the mouse to click on the item you want to select, you can use the up and down arrow keys to move the selection bar, or you can press the letter corresponding to the first letter of the selection that you want. If there is more than one application starting with the same letter, you can press your letter again and the selection bar will advance to the next application starting with the same letter.

Below the list control is the name of the application that will automatically be run the next time you boot your system. You can change the autolaunch application by selecting an application from the list control and clicking on the Set Autolaunch button. If you want Autopilot to be the program you boot into instead of something else, you can click on the Clear Autolaunch button and NONE will appear as the autolaunch application name.

If you are booting your system and you don't want the autolaunch application to be run, simply put your caps lock key in the locked position. This will override the autolaunch application and Autopilot will take control. You can then select another program from the application list, or you can use the Open menu item to select a different application to launch.

An Example

To make things a bit clearer, I'll present an example of how to use Autopilot. To follow along, install Autopilot using the instructions given above and reboot your system. When you arrive in Autopilot, the list will be empty and there will be no autolaunch application selected. You're not very comfortable, are you? This is, after all, a new environment to you. So let's go somewhere to feel a bit more at home. Click on the Add button and when the standard file dialog appears, go to your ***:System** directory and select the application named **Finder**. Now click on the Accept button. The application that you just selected will be added to Autopilot's main application list and the Standard File dialog will reappear allowing you to select some more files to add. If you're daring, go ahead and add a few more applications. You can select multiple applications from the Standard File dialog by shift-clicking and Command-clicking on items. Once you've added enough items to the list, click on the Cancel button.

Autopilot's list will now display all the applications you just added. Now select the **Finder** and click on the Launch button (or double-click on the Finder item). What happens? Well now you're in the Finder! That wasn't so bad, was it? Now let's quit back to Autopilot and play some more. Select the Shutdown item from the Finder (Command-Q), select the "Return to launching application" radio button, and then click on the Shutdown button. You'll then be back in Autopilot.

If you don't like the way the Finder application looks in the list, you can change it by using the Edit button. Select the Finder application from the list and click on the Edit button. An edit dialog will appear. Yes, that's right, I said a dialog. But it looks like a window, right? It's actually a *movable modal dialog box*. You can move the dialog anywhere on the screen just as you would a normal window, but you can't do anything else until you click on the Okay or Cancel button. Well let's get on with what we're here for: change the LineEdit control's contents to be the way you want to see your Finder application displayed in the list. Note that the change is just for Autopilot's list control, it *doesn't* rename the application. You can use special characters and spaces all you want. Click on the Okay button to accept the changes or the Cancel button to disregard any changes that you made.

Now it's time to play with autolaunching an application. Select the Finder application from the list control again and then click on the Set Autolaunch button. You'll notice that the autolaunch string below the list control changes from NONE to whatever you named the Finder. Now select Restart from the Special menu and confirm that you really want to restart your system. Your computer will go through the boot process again and, instead of appearing in Autopilot, it will come up in the Finder. You could have, of course, had the caps lock key down and that would have forced Autopilot to load even if there is an autolaunch application.

Removing an item from Autopilot's application list is easy. Simply select an application from the list and then click on the Remove button. It will be removed from the list.

If you ever get tired of having the Finder as your autolaunch application, you can either change it by selecting another application from the list control and clicking on the Set Autolaunch button, or you can click on the Clear Autolaunch button to clear any autolaunch application.

That's all there is to using Autopilot! If you have any problems, or if you can think of any enhancements, be sure to contact me. A good way to let me know

what's really going on with some of those problems is for you to fill out that nifty problem form that's provided on your *GS+* Disk.

Programming Tricks And Treats

Even though it's not Halloween, Autopilot packs a few groovy programming tricks. The first obstacle I had to overcome was how to know whether to launch the autolaunch application or not. This was solved by posting a message to the message center. The first thing that Autopilot does is it checks to see if the message is there. If it is, it knows not to launch the autolaunch application. If the message is not there, Autopilot assumes that the system was just booted. So, it posts the message and then checks to see if it should autolaunch an application. Note that if Autopilot is run without the message present, it will attempt to autolaunch an application. This means that if you use a different start program and then launch Autopilot, you might end up in an autolaunch application instead.

Autopilot also makes use of `FillReplyRec` from my OS Library routines presented in *GS+* V2.N4. I could have also used the `TraverseNames` routine, but I wrote Autopilot before I wrote the `TraverseNames` routine so I left Autopilot alone. I might use the `TraverseNames` routine in the future, though. For more information on OS Library, refer to the "OS Library" article in *GS+* V2.N4 and the "NoDOS v1.5" article in this issue.

Program Specifications

Autopilot is written completely in assembly language using ORCA/M. There is one REZ source code file, one file of equates, and five source code files. I turned the case directive on because some of my equates (`ListHandle` comes to mind) conflicted with some equates in the standard equates for the tool sets. I'll now describe what each routine of Autopilot does. I find it easier to document this here instead of in comments within the source code—that just takes up space. I do, however, comment as to what is going on in the source code, you just don't get the big picture from it.

In the **Auto.ASM** file there are routines to start up and shutdown Autopilot, including the code to autolaunch an application. The **Events.ASM** file contains the routines to handle events such as the main event loop and keypress handling for the list. The **Menus.ASM** file contains the code to handle each of the menu items. The **Buttons.ASM** file contains the code to handle each of the main window's buttons. The **Data.ASM** file contains global data that is used by each of the source code files. The **Fake.ASM** file contains the code that replaces some of the Apple DTS Fake Modal Dialog routines. An explanation for the **Fake.ASM** file is below under the "Fake.ASM" section.

The Auto.ASM File

`Autopilot`: Autopilot starts by starting up some basic tools and checking to see if an application needs to be autolaunched. If an application needs to be autolaunched, the basic tools are shut down and the application is launched. If an application does not need to be autolaunched, the rest of the tools are started and the main event loop is called.

Whenever an application is launched, the name of the application is displayed at the top of the screen by using the GS/OS `.CONSOLE` driver. If an error occurs when Autopilot attempts to quit, then Autopilot is restarted by loading in all the tools again and displaying an error window.

`StartBasics`: This routine starts up the Tool Locator, Memory Manager, and the Resource Manager. Autopilot's resource fork is opened as read-only so there can be a check for an autolaunch application.

`StartUp`: This routine first shuts down the Resource Manager (which closes Autopilot's resource fork) and then the rest of the tools are started. Since the `StartUpTools` call only opens Autopilot's resource fork in read-only mode, and it should be opened in read/write mode, the Resource Manager is shut down yet again to close the resource fork, then started up again. Autopilot's resource fork is opened in read/write mode and the new fileID is stored in the start/stop record structure so when `ShutDownTools` is called the file is closed properly. Finally the FakeModalDialog library startup call is made so future calls can be made to the library.

`ShutDownBasics`: This routine simply shuts down the Resource Manager, Memory Manager, and Tool Locator.

`ShutDown`: This routine simply calls the FakeModalDialog library shutdown call and then shuts down the rest of Autopilot's tools.

`CheckAutoload`: This checks to see if an application needs to be autoloaded. First the `QuitGS` pathname parameters are zeroed for the default case of there being no autolaunch application. Next, the message center is checked to see if the Autopilot message exists. If the message exists, then no application needs to be autoloaded, so the routine exits. If no message exists, it is automatically added and the keyboard modifier register is checked to see if the caps lock key is down. If it is down, then no application needs to be autoloaded, so the routine exits. If it is up, then the autoload ID is loaded. If the ID is zero, then that means that there is no autoload application so the routine exits. If the ID is nonzero, then the pathname of the application to autolaunch is loaded and the `QuitGS` pathname parameter is set to launch the application. The name of the application is loaded and prepared to be displayed on the text screen.

`SetFilename`: This routine prepares the filename to be displayed on the text screen. First, 80 space characters are shoved into the display buffer. Next the filename is moved to the display buffer so it will be centered when it is displayed.

`Initialize`: This routine initializes the desktop for Autopilot. First the menu bar is created and the desk accessory items are added to the Apple menu. The main window is opened and the handle to the

list control is determined and the first member of the list is selected. The button states for the window are then set, the menu bar is drawn and correctly set for the main window. Finally the current window pointer and the main event loop quit flag are zeroed and the pointer is switched from the watch to the pointer.

The Events.ASM File

`EventLoop`: This is a fairly standard main event loop.

`InMenu`: This handles the selection of menu items.

`InControl`: This handles the selection of a control.

`CheckFrontWind`: This checks to see if the front window has changed, and if it has, the menu bar is adjusted to the new window.

`Ignore`: This is called for events which require no action.

`DrawWindow`: This draws the controls in a window.

`ListKey`: This handles key-down events for the list control. If the key is an up or down arrow, the previous or next item in the list is selected. If the key is not a control key (greater than the ASCII space code) then the list item that has a first character corresponding to the key is selected. The algorithm used to select the item is if the current selection doesn't start with the key pressed, a search starts from the first list entry. If the next selection starts with the key pressed, it is selected. The search starts at the top of the list and compares the first letter of the entry with the key and if the first letter is greater than or equal to the key then it is selected. All comparisons are case insensitive, which means that letters are first converted to uppercase before comparisons.

`GetFirstChar`: This takes a list entry number and returns the first character of the entry's string after converting it to uppercase.

`CustomSort`: This compares two items and returns the carry clear if the first item is less than the second item; otherwise, the carry is set on return. The comparison routine compares two Pascal strings with a case-insensitive compare. This routine is called by the List Manager.

The Menus.ASM File

`DoAbout`: This displays the About window.

`DoOpen`: This displays the open Standard File dialog which allows the selection of an application to launch. If an application is selected, the `QuitGS` pathname parameter is set to launch the application. Then the length of the name is shortened to be less than 80 characters so it will fit on the screen and then the name of the application is prepared to be displayed on the text screen. The quit flag is finally set so the event loop will be exited and the application will be launched.

`DoCloseW`: This closes the front window, if it exists.

`DoQuit`: The quit flag is set so the event loop will be exited and Autopilot will return to the application that launched it.

`DoRestart`: This routine sets the restart flag and then calls the routine to display the restart dialog.

`DoShutdown`: This routine sets the shutdown flag and then calls the routine to display the shutdown dialog.

`KillOSDialog`: This displays either the restart or shutdown dialog. If cancel is selected from the dialog, the routine exits; otherwise, the flags for `OSShutdownGS` are set. If shutdown was selected, then all removable disks are ejected. Next all the windows are closed to let desk accessories prompt for loss of work, the Resource Manager is shut down to write out all resources changes, and finally the `OSShutdownGS` call is made to either restart or shut down the computer.

The Buttons.ASM File

`DoLaunch`: This loads in the pathname of the application that was selected in the list and sets the `QuitGS` pathname parameter to launch the application. The name of the application is prepared to be displayed on the text screen, and the quit flag is set so the event loop will be exited and the application will be launched.

`DoEdit`: This opens the edit window, sets the TextEdit control to the application pathname, sets the LineEdit control to the display name, and then begins the FakeModalDialog event loop. If the Cancel button is selected, the window is closed and no changes are made. If the Okay button is selected, the new display name is retrieved from the LineEdit control and the display name resource is marked as changed. The autolaunch application is checked to see if it was the one edited, and if it was, then the autolaunch name is changed as well.

`DoAdd`: This displays a Standard File multi-get dialog allowing the selection of only S16 and SYS file types. If the dialog is cancelled, nothing happens. If something was selected from the dialog, the multi-file reply record is stepped through one entry at a time and the OS Library is used to copy the entry from the multi-file reply record to a new-style reply record because information is easier to retrieve that way. A unique resource ID is obtained, and the pathname for the application is added to the resource fork. The filename is converted to a Pascal string and added to the resource fork as well. The listRef is loaded, expanded in size by one entry, and the new entry is added. (For a better description of how to add an entry to a list, see "The New Order" in *GS+* V2.N4.) After the entry is added, the listRef resource as marked as changed and then the list control template is loaded. The listSize field (number of entries in the list) is incremented and that resource is also marked as changed. Finally, after all items have been added, the changes made to the list are told to the List Manager, and the Standard File dialog is displayed again.

`DoRemove`: This first checks the autolaunch application to see if it is the one being removed, and if it is, the autolaunch application is cleared. Next the pathname resource and the display name resource are removed from the resource fork. The list control template is loaded and the listSize field (number of

entries in the list) is decremented and the resource is marked as changed. Finally the listRef resource is loaded and all the entries past the entry to be deleted are moved up one space. The listRef memory is then resized to remove the dead entry at the end and the resource is marked as changed. The List Manager is then notified of the changes to the list.

`ChangeList`: This first updates the resource file to write out all resources that have been marked as changed. Then the list control template is loaded and the listSize field (number of entries in the list) is read. Then any changes to the list are told to the List Manager with the `NewList2` tool call. The list is sorted using a custom case-insensitive sort, the first item in the list is selected, the new sorted listRef resource is marked as changed, and the members of the list are redrawn. Finally the button states are set in case any changes to the list require a button state to be changed.

`UpdateButtons`: This sets the states of the buttons depending on the state of the list control and the autolaunch application. If there are no items in the list control, the Launch, Edit, Remove, and Set Autolaunch buttons are made inactive; otherwise, they are made active. If there is no autolaunch application, the Clear Autolaunch button is made inactive; otherwise, it is made active. Note that if there aren't any items in the list, then there won't be an autolaunch application either.

`DoList`: This handles any clicks in the list control. If there is a double-click, the `DoLaunch` procedure is called to launch the application that was double-clicked on.

`DoSetAuto`: This gets the resource ID of the selected application in the list control and sets the autolaunch resource to that ID. The custom autolaunch string is then loaded, and the display name of the application is converted from a Pascal string to a LETextBox2 string and the autolaunch string is marked as changed. The custom autolaunch string control template is loaded and the resource is changed to display the new LETextBox2 string. The new autolaunch name is then displayed in the main window and the button states are set in case setting an autolaunch application requires a button state to change.

`DoClearAuto`: This sets the autolaunch application resource to zero, which means that there is no autolaunch application. The custom autolaunch string control template is loaded and the resource is changed to display the string "NONE". The new autolaunch name (NONE) is then displayed in the main window and the button states are set because clearing an autolaunch application always requires a button state to change.

`ChangeTheName`: This first updates the resource file to write out all resources that have been marked as changed. Then the new autolaunch string is displayed by using the `SetCtlTitle` tool call.

`FindMemberPtr`: This gets the number of the selected list item and decrements it by one so the first item is number zero. The list number is then multiplied by the size of each listRef entry to get the offset from the start of the listRef memory to the entry field. Then the offset is added to the start of the listRef memory to find the absolute location of the selected member's entry in the listRef.

The Fake.ASM File

I wrote the **Fake.ASM** file because of the letters "GS." Yes, it's the name conflict with Apple that prompted me to write these routines. I had originally used the Apple DTS Fake Modal Dialog library routines in Autopilot, but we didn't want to release the program (it was actually ready to go back in February) unless we could provide a way for you to assemble the program yourself. So, we wanted to provide the Fake Modal Dialog libraries as well. For that, we need a Licensing Agreement from Apple. As has been detailed elsewhere, we can't license things from Apple until we change our name! So, these routines are the workaround. They are called exactly the same way that the Fake Modal Dialog routines are called so I didn't have to change my program any. The routines that get and set LineEdit text probably aren't as robust as the "official" Fake Modal Dialog routines, but they work for this application. My `FakerModalDialog` routine is really skimpy compared to the DTS routine, but it works. If you have a copy of the "real" Fake Modal Dialog libraries and you want to see what the "real" version of the edit dialog operates like, simply take out the comment character in front of the `fmdStartUp` and `fmdShutDown` lines in the **Auto.ASM** file. Then, in the **Buttons.ASM** file, find the `LEGetText` line, comment it out, and remove the comment character from the `fmdLEGetText` line. Do the same for the `LESetText`/`fmdLESetText` and `FakerModalDialog`/`fakeModalDialog` lines. Then recompile.

`LEGetText`: This routine takes the text within a LineEdit control and returns it to a buffer as a Pascal string.

`LESetText`: This routine takes a Pascal string and puts it in a LineEdit control.

`FakerModalDialog`: This routine handles events for the edit dialog.

The Wrap Up

Autopilot is by no means a short and sweet program. It will take a bit of studying to figure out exactly what is going on, but with the synopsis of each routine, you can get the basic feel of how things are working. There are *lots* of improvements that I want to make to Autopilot—this has been just a scratch on the surface. If there is something you would like to see in the next version of Autopilot, please let me know what it is. **GS+**

Softlock

By Jeff Hartkopf

Do you ever have trouble with people using your computer against your wishes when you are not around? Here is a new desk accessory (NDA) that provides a solution. Softlock is a password-protection system for the Apple IIGS. Its purpose is to keep unauthorized people from using your computer. Once installed, it brings up a dialog box when you start up your computer and asks for a password. No one is allowed access until the correct password is typed. In addition, you can bring up Softlock at any time while your computer is running to protect it until you return.

It is important to realize, however, that Softlock is intended to protect a computer from casual users only. A knowledgeable user can get around the password-protection system, but you can minimize this risk by taking a few precautions.

Installation

Although Softlock is most beneficial when used from a hard disk, it can also be installed on a floppy disk, provided the disk boots to a desktop program that supports NDAs. If you do install it on a floppy, be sure you do not write-protect the disk because Softlock must write to it to change your password.

To install Softlock, simply launch the Installer from your *GS+* Disk. For more information on installing Softlock, see "Using The *GS+* Disk" in this issue.

Using Softlock

Once Softlock is properly installed, simply restart your IIGS for it to take effect. After the "Welcome" thermometer bar goes away, Softlock's main dialog box will pop up. Type your password into the Password field. If you have just installed Softlock, the initial password is "pass" and it must be typed exactly as shown (except you should not type the quotation marks), using lowercase letters since Softlock is case sensitive. As you type, the password will appear as a string of asterisks so that someone looking over your shoulder can't read your password.

Until you enter the correct password, the only option available to you will be to shut down the computer. As soon as you have typed the correct password, however, the other features of Softlock will become available.

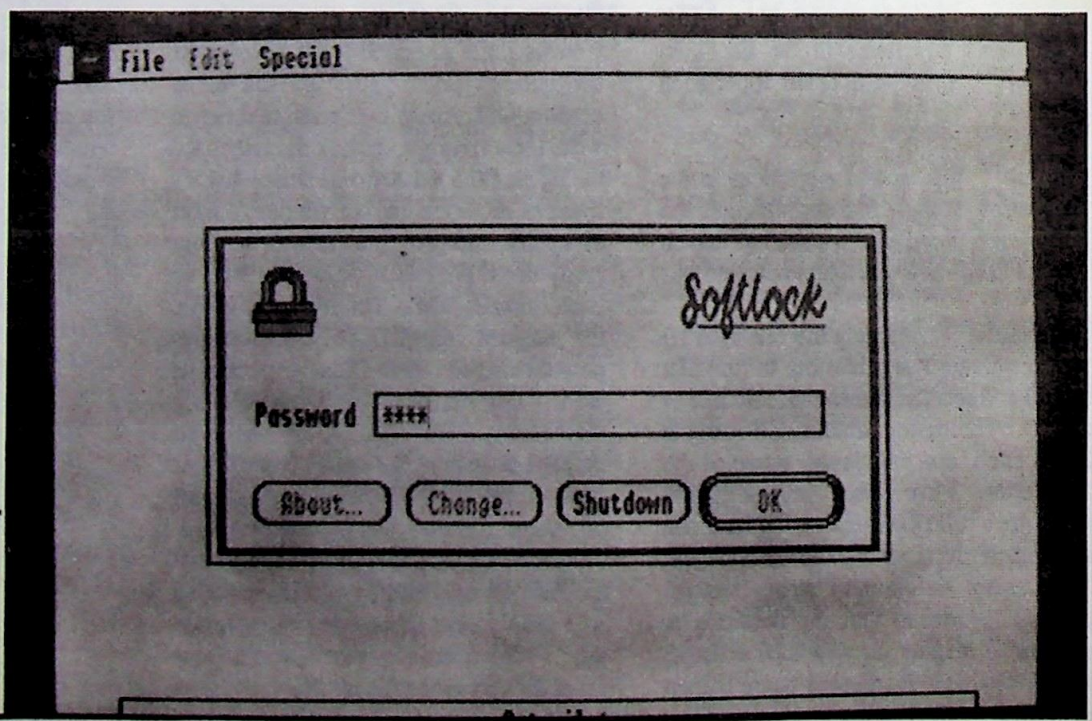
After you have entered your password, you can click the OK button to continue with your application. You can also shut down the system by clicking the Shutdown button, view information about Softlock by clicking the About button, or change your password by clicking the Change button. The latter process is described in the following section, "Changing Your Password."

Once your computer is running, you can choose Softlock from the Apple menu of the current application to bring up the Softlock dialog box. As before, no one will be able to use the computer until the correct password is entered. This can be a handy feature if you need to leave your computer unattended for a while, but do not wish to shut it down. To get back to work, simply type your password and click the OK button.

Changing Your Password

You should change your password the first time you use Softlock, and any time thereafter when you deem it necessary. The password you choose should be something you can easily remember, but not something others are likely to guess. Your password can be up to 30 characters long, with each character being any lowercase or uppercase letter (case is significant), a digit, a symbol, or any other character you can type on the keyboard. With such a large variety of available characters and long maximum password length, there are an enormous number of possible passwords. With a little thought, you should be able to choose a password that no one will ever be able to guess.

To change your password, you must first correctly type in your *current* password, and then click the Change button. A new dialog box appears, into which you need to enter your new password twice, once in



each of two line edit fields (you can press the tab key to move from field to field if you don't want to use the mouse). Since asterisks again hide what you type, entering your password twice ensures that you don't accidentally type something you did not intend. You will not be allowed to click the OK button until both password fields are identical and nonempty. If you decide you don't want to change your password, click the Cancel button. Otherwise, when you click the OK button your new password will be permanently saved and remain in effect until you decide to change it again.

Disabling Softlock

If you wish to temporarily disable Softlock (for example, if you need to reboot often and don't wish to type your password every time), you can enter the Finder and click on the Softlock icon located in the **Desk.Accs** folder. Choose the Icon Info option from the Special menu and click the Inactive check box to disable Softlock. To reenable Softlock, repeat the same process, unchecking the Inactive box.

Removing Softlock

If you decide you no longer wish to use Softlock, simply delete the file from your **Desk.Accs** folder. Be sure to use a program that can delete files with resource forks.

Troubleshooting

If you run into problems using Softlock, check this troubleshooting section first. If you still have problems, or if you find a bug, give us a call here at *GS+* or fill out the *GS+* Problem Form (which is on your *GS+* Disk) and send it in to us.

Your password doesn't seem to work. This problem may happen because you are typing your password incorrectly. Be sure the caps lock key is up (unless your entire password is in uppercase), since Softlock considers "Password," "password," and "PASSWORD" to be three different passwords. If all else fails, see the following section entitled "You forget your password."

You forget your password. If you forget your password, there are two ways to get around it. The simplest is to remove Softlock (see section entitled "Removing Softlock" above), and install a fresh copy with the original password ("pass"). To do this you will need to boot off of a copy of the System Disk that does *not* have Softlock installed; you *cannot* delete Softlock from a ProDOS 8 program because Softlock has a resource fork. Note that for this method to work successfully, you must have both a copy of the System Disk without Softlock installed and an unaltered original copy of Softlock. Be sure you have both of these items before attempting to use Softlock, in case you run into problems or forget your password.

The second solution does not require you to have an unaltered original copy of Softlock, but does require you to have access to a resource editor (such as Genesys) as well as a copy of the System Disk without Softlock installed. Boot the System Disk, then run your resource editor. Open the Softlock NDA file located in your **Desk.Accs** folder and edit the single C String resource (named PASSWORD) that it contains. This will display your current password and allow you to change it. To be safe, clear the password string completely and type in a new password. This technique can always be used to change your password; however, it is recommended that, whenever possible, you do so by using the Change feature of Softlock.

Be warned that while the above methods tell you how to get back into your system if you forget your password, they also tell others how to break into your system. Be sure not to leave these instructions in a visible place. To further protect yourself, especially if the unauthorized parties are sufficiently familiar with the IIGS to figure out the above methods by themselves, it is a good idea to lock up any copies of the System Disk that do not have Softlock installed. This is only necessary, of course, if you have a hard disk, which is vulnerable to these sorts of attacks.

Hints and Tricks

If you are paranoid or want to give yourself even more protection than Softlock provides, you can install multiple copies of Softlock. Each will have to be given a different name, such as **Softlock1**, **Softlock2**, and so forth. Then when you start your computer several password dialog boxes will come up before you can get into the system, each possibly with a different password.

This concludes the section on using Softlock. The rest of this article deals with the programming aspects of Softlock. If you are interested only in using Softlock, and not in learning how it is programmed, this is where you can stop reading.

If you have any suggestions or find any bugs in Softlock, please be sure to notify me in care of *GS+* Magazine. I'm

Shortcut-Key Reference

The following reference lists all the controls in the Softlock dialog boxes and the keyboard shortcut for each. Keyboard equivalents are available for all relevant controls, so you can, if you like, use the keyboard exclusively when operating Softlock. The Command key is the key with an apple and a propeller on it; pressing Command-x means to hold down the Command key while you press the x key.

Main Dialog Box

About button	Command-A
Change button	Command-C
Shutdown button	Command-S
OK button	Return

Change Password Dialog Box

Cancel button	Escape
OK button	Return

About Dialog Box

OK button	Return

always happy to hear from the people that use my programs!

Softlock Programming Issues

Softlock is not a large piece of software, and as such it can be easily understood in fairly complete detail with a little study of the source code and the information presented here.

As previously mentioned in this article, Softlock has a resource fork, created with Genesys, and uses resources for the dialogs, alerts, and to store the password. You can look at the resources and customize Softlock with a resource editor if you desire. All of the resources are named so that they are easy to identify, and also so that a list of resource names defined as the appropriate numerical IDs can be easily generated and used in the program to refer to individual resources (see the file **softlock.h** for this list).

Softlock is written in ORCA/C in a highly modular fashion: it is divided into four separate, independent modules, two of which do not even apply specifically to Softlock and can be used without modification in your own programs. Each module consists of a source (.cc) file and a header (.h) file; to use the features of a given module, the programmer need only read the interface specification given in the appropriate header file, and then include that header file where needed to use the functions and constants it declares. In the source files, functions internal to the module are declared as static so that no external module can reference them. The modules are: *softlock*, the main module for controlling basic functions; *controls*, to handle the user's actions in the Softlock dialog box; *status*, which takes care of starting up and shutting down necessary tools; and *fakemodal*, to process a window with extended controls and an alert-type frame as a modal dialog box. Each of these modules is explained in detail below.

The Softlock Module

This module contains the four basic NDA functions: `init`, `open`, `close`, and `action`. Since none of the functions in this file need to be accessed externally, all are declared static.

The `init` function is called by the system when you start up the computer, as well as every time an application starts up and shuts down. We do not need to do anything except when the start application runs for the first time, so all `init` does is to determine if we are starting up and, if so, calls open to bring up the enter password dialog box. The `init` routine is called twice on start-up: once when the NDA is loaded (about halfway through the boot process), and once when your start application (usually the Finder) is run. The first time `init` is called, the start parameter is zero, so nothing is done. The second time, start is nonzero, so `open` is called and a flag, called "done," is set so we ignore all future calls to `init` until the system is restarted.

The `open` function, which is called by `init` and also whenever the user chooses Softlock from the Apple menu, calls `handle_tools` (discussed below) to make sure all necessary tools are loaded, and then creates the main dialog box. Next it initializes the controls module by calling `ctl_init`, and goes into a control-process loop which terminates only when the OK or Cancel button is clicked, in which case the shutdown variable will accurately reflect whether or not to shut down the system. After shutting down the tools we loaded and closing the window, we call `shutdown_system` if necessary, and if not, return to the caller.

Neither the `close` nor `action` functions do anything, but their empty stubs are still needed to create a desk accessory.

When `handle_tools` is called with a nonzero start parameter, it calls some status module functions to make sure the QuickDraw II Auxiliary and Font Manager tool sets are started, starts up the Resource Manager, and loads Softlock's resource fork. `LGetPathname2` is a GS/OS loader function used to get the correct pathname of the NDA. When called with start equal to zero, `handle_tools` cleans up by closing the resource fork, shutting down the Resource Manager and restoring the other tools to their initial state.

Finally, `shutdown_system` does exactly that: it calls `OSShutDown` to bring up the dialog box saying it is safe to shut down your computer.

The header file for the softlock module, **softlock.h**, simply contains a list of all the ID numbers of the resources in the resource fork. The two modules that need to access resources (softlock and controls) include this file.

The Controls Module

Once the main Softlock dialog is displayed on the screen, we want to keep all controls except the Password line edit and the Shutdown button disabled until the correct password is typed. At that point, we want the line edit to become disabled since no more typing is needed, and we want all buttons to become available.

The controls module contains two entry points: `ctl_init` which must be called first, and `ctl_process` which handles a main-dialog control event. See the file **controls.h** for a complete description of these functions' interface. Also included in this module is an update function which just calls `DrawControls` on the current port, and is used to update all of Softlock's dialogs.

Inside `ctl_init` we set up a vector containing the handle of each main dialog control, indexed by control ID (which is the same as the resource ID). We also deactivate the About, Change, and OK buttons.

The `ctl_process` routine is called when an event occurs in the main dialog. It first checks which control ID it has been passed, and then calls the appropriate handler routine for that control. After the handler returns, `ctl_process` sets the shutdown parameter to be true if the Shutdown button was clicked and the user confirms his or her wish in the alert dialog that comes up. Finally, `ctl_process` returns a value telling the caller whether to continue processing the dialog by determining if we are going to shut down or if the OK button was clicked.

The control handling routines come next. First is `process_pass`, which

processes an event in the Password line edit. If the user has typed the correct password, the line edit control is disabled and all buttons are activated. A `LoadResource` call is first made to load the C-string resource containing the correct password, and then the appropriate calls are made to extract the text from the line edit control and compare it to the password.

The `process_about` procedure brings up a dialog to display information about Softlock, and exits when the user clicks the OK button. The About dialog is a resource like the other windows.

When the user clicks the Change button to change his or her password, the `process_change` function is called. It first brings up the new password dialog box, saves some commonly referenced control handles in local variables, and inactivates the OK button, which will remain inactive until two identical passwords are typed. Next, `process_change` enters a loop until the user clicks either the OK or Cancel button. Each time through the loop, it checks to see if the passwords are identical (and not empty), disabling or enabling the OK button appropriately. If the Cancel button is clicked, control is returned to the caller without further ado. However, if the OK button is clicked, the old password resource is loaded, its handle is resized to fit the new password, the new password is copied over, and finally the password resource is marked as changed so it will be written back to disk when the softlock module closes the resource fork.

The last two control processing routines, `process_shutdown` and `process_ok` for the Shutdown and OK buttons respectively, are currently empty. They are left as stubs in case they are needed in the future.

The Status Module

The status module contains three useful functions, whose purposes are described in **status.h** (which you should study before proceeding). Two of the functions in **status.cc**, `tool_loaded` and `tool_started`, are very simple functions that just call the desired tool set's status function to determine their respective results.

The `tool_status` routine is somewhat more complex, in that it needs to make sure the desired tool set is not only started, but also loaded, and then when called later, restored to its original state. The original state is stored in local static integer variables, which retain their values even outside of the function, named `da_load_xxx` and `da_start_xxx`, where `xxx` is the name of the tool set. A nonzero value indicates a true condition, and zero indicates false. These variables are set when the state parameter equals OPENED, and checked later when state is CLOSED to restore each tool set to its initial loaded/started state.

One other thing that should be pointed out about the `tool_status` function is that some tool sets, such as the Font Manager, need memory allocated for them (which is done with a `NewHandle` call), and freed (with `DisposeHandle`) when `tool_status` is later called with the state parameter equal to CLOSED.

The Fakemodal Module

Only one function, `fake_modal_dialog`, is contained in this module, whose purpose and usage is again completely described in **fakemodal.h**. [This is *not* the same as the Fake Modal Dialog Tools provided by Apple. These are Jeff's own Fake Modal Dialog tools. - Ed.] Softlock cannot just use `ModalDialog` because the Dialog Manager does not support the extended controls which Softlock uses.

After setting a rectangle the size of the entire dialog window for the current port, `fake_modal_dialog` enters a loop calling TaskMaster to process the dialog until an event (mouse click or key press) occurs in a control, in which case that control's ID is returned to the calling program. Before calling TaskMaster the task mask is set up to process the front window, but nothing else. The task mask is not set to handle update events (which are not allowed for NDAs that use resources), so when an update event occurs, and if a content draw routine is defined for the current port, the routine is called. If a mouse click occurs outside of the window rectangle, `fake_modal_dialog` sounds a beep.

Note that this module is completely generic, and can be used to process any alert-type window containing any combination of extended controls.

That's about all there is to say about Softlock! I hope you enjoy it and that it proves useful to you. **GS+**

Errata

In last issue's (V2.N4) reviews of Harmonie, Independence and Talking Speller II, we neglected to mention that the products were review copies provided by the publishers.

Going *way* back to *GS+* V1.N6, we neglected to mention that Disk Access was a review copy provided by the publisher. Our apologies to Vitesse, Seven Hills Software and Orange Cherry Software.

Starting with this issue, review copies will be identified as such in the table of contents. **GS+**

EGOed v1.34

By Steven W. Disbrow

New Features

This version of EGOed has one bug fix and only two new features. It doesn't sound like much, and these features caused an increase in the size of EGOed, but I think you'll find them most useful.

The Bug Fix

Several folks reported that when using the Save As option in EGOed v1.33, EGOed would sometimes crash. I myself had had similar problems but was at a loss to explain what was going on. Fortunately, Joe was able to track the problem down to a very odd compiler bug in ORCA/C v1.2. This version of EGOed works around this compiler bug so that the Save As dialog once again works as it should. For more information on this bug and how to work around it in your own programs, see "Programming Considerations," below.

New Feature #1

The first new feature of EGOed v1.34 is that when you open an AppleWorks Classic word processor file, EGOed retains as much of the actual formatting in the file that it can. Basically, this means that if your AppleWorks Classic file has any **boldface** or underlined characters in it, they will appear as boldfaced and/or underlined when you open the file with EGOed. Unfortunately, those are the only two styles that AppleWorks Classic supports that EGOed can display. (AppleWorks Classic does not support Italics or Outline or Shadowed text.)

At this point, I need to warn you that this is *read-only*! If you attempt to save the file back out as an AppleWorks Classic file, *all of the formatting will be lost*!

It also needs to be pointed out that EGOed uses the default font that you have selected (the default font selection is in the Prefs menu) to display AppleWorks Classic files. If your default font is 8-point Shaston, *underlining will not show up*! Eight-point Shaston simply does not support underlining.

New Feature #2

The second new feature in EGOed v1.34 is that you can now read AppleWorks GS v1.1 word processor files with most formatting intact! Just as with AppleWorks Classic files, this is a *read-only* translation and only those styles that EGOed can actually display are supported. Superscripts, subscripts, rulers, headers and footers are not yet supported.

As I said earlier, this is a read-only translation. If you load an AppleWorks GS file into EGOed, make a change and then try to save the file, you will be presented with the Save As dialog and you will have to save the file as one of the four supported file types (TeachText, AppleWorks Classic, ASCII Text or APW Source). Pick TeachText if you want to retain the formatting shown in the EGOed window. Be sure to give the file a new name so that you don't destroy your AppleWorks GS file.

If EGOed seems to have a problem reading an AppleWorks GS file (the formatting is wrong or it takes a long time to open or the file contains gibberish), the file was probably saved under AppleWorks GS v1.0 or 1.0v2 and not v1.1. If you can, resave the file using AppleWorks GS v1.1 and EGOed should then be able to open it correctly. Another problem that you may run into is that the fonts and styles are correct, but they are applied to the wrong text! There is a bug in AppleWorks GS v1.1 that sometimes causes text to appear in large, blocky letters that are completely unrelated to the font you have selected. Another symptom of this bug is that the text will appear in the correct font on screen, but the AppleWorks GS Font menu gives an incorrect report as to which font is actually applied to the text. Although I can't be sure of *exactly* what is going on, it seems that, when one of these things occurs, AppleWorks GS is saving style and size information in the wrong place, usually on top of the font information! If you use EGOed to open a file that was saved while this was wierdness was going on inside AppleWorks GS, the formatting information is likely to be quite screwy.

Why Read Only?

Now, some of you are probably thinking, "Why read only?" Well, the main reason is a lack of time. The second reason is a lack of space. This version of EGOed is a whopping 35K—the biggest yet. Adding the ability to write AppleWorks GS files would probably increase that size to about 40K—if not larger! (For more information, see "File Format From Hell" below.) But, if you don't mind the increasing size, I'll be happy to put it in there–let me know what you think.

That's it for this version of EGOed. For complete EGOed documentation, see the file **EGOed.Docs** which is in the **EGOed** folder on your *GS+* Disk. For information on the programming aspects of this version of EGOed, read on. And don't forget to send in the Problem Form on your *GS+* Disk if you find a problem.

Programming Considerations

Before we get into TEFormats and other heavy stuff, let me give you a description of the compiler bug that we found in ORCA/C v1.2. If you compile with all optimizations on (`#pragma optimize -1`) and you have a global structure and you assign a value to one of its fields from a value specified in a handle, ORCA/C v1.2 generates incorrect code which corrupts the stack and causes a crash on return from the function. A more visual description can be found in Figure 1. Once you know what the bug is, the work around is simple; it is also shown in Figure 1. Under ORCA/C v1.1, this seems to have been an intermittent bug that might or might not end up in the final code and might or might not cause a crash (which explains why the Save As dialog in EGOed v1.33 seems to have crashed at random). Under ORCA/C v1.2 however, it seems to end up in the code every time.

TEMagic!

Moving on to something a bit more fun, lets talk about how we get all of that formatting out of those AppleWorks files. EGOed uses the TextEdit tool set for all of its editing chores. To display formatting

(fonts, styles, etc.) the TextEdit tool set makes use of a data structure known as the *TEFormat* structure. When you add text to a TextEdit record using, say, the `TEInsert` tool call, you specify the formatting to be used by passing a TEFormat structure to the tool call. TextEdit uses the information in the TEFormat structure to set the appropriate formatting in the text.

So, the key to getting the formatting out of other files and presenting it in the EGOed window is figuring out a way to turn the formatting into a TEFormat structure. Basically, a TEFormat is made up of a list of rulers (each entry in the list is called a *TERuler*; at this point, TextEdit only supports one ruler), a list of the unique styles that are in the text (each entry in the list is called a *TEStyle*) and a list detailing which characters in the text that those styles are to be applied to (each entry in the list is called a *StyleItem*). The TEFormat structure is explained in detail in the TextEdit chapter of *The Apple IIGS Toolbox Reference: Volume 3*, beginning on page 49-31.

So, what this boils down to is three variable-length structures that you have to create on the fly and then combine into one structure. Ordinarily, this is not a fun thing to do. So, to take some of the sting out of the process, I have written four routines to aid in this task and have put them all in the file **TEMagic.h**.

The first routine, `FindTEStyle`, is used to search a list of TEStyle structures to see if a particular style is already in the list. If it is, `FindTEStyle` returns the offset, in bytes, of the style from the start of the list. If it is not in the list, `FindTEStyle` returns a -1.

The next routine, `AddTEStyle` is used to add a TEStyle structure to the list that you just searched with `FindTEStyle`. It places the TEStyle structure at the end of the list, and returns the new size of the list of TEStyles. If something goes wrong, `AddTEStyle` returns a zero.

The third routine, `AddStyleItem` adds a StyleItem structure to the end of a list of StyleItems. If the function fails to add the item, it returns a zero. Otherwise, it returns the number of items that are in the list.

The last routine, `MakeTEFormat` takes handles to the TERuler list, the TEStyle List and the StyleItem list and combines them into a TEFormat structure. When it finishes, it returns a handle to the TEFormat structure that you can then pass to various TextEdit tools.

What About The Ruler?

Oh, yeah, the ruler. Well, since TextEdit only supports one ruler at a time, I simply use the `TEGetRuler` tool call to get a copy of the ruler that is currently defined in the EGOed window. I then pass this ruler to `MakeTEFormat` for use in the new TEFormat. Not fancy, but it gets the job done.

The point of writing these routines was to have a generic means of building TEFormat structures. This way, I can easily give EGOed the ability to read—with most formatting intact—just about any file format I want. For example, the AppleWorks Classic and AppleWorks GS file formats are as different as night and day, but, once I had the TEMagic routines working for AppleWorks Classic files, it took only about a half a day to develop the routine to read in the AppleWorks GS files. For just about any other file format, it would have only taken an hour or so, but AppleWorks GS uses the . . .

File Format From Hell!

Before you get the idea that I'm going to explain all of the nuances of the AppleWorks GS word processor file format, I'm not. If you really want to know about the format in detail, grab a bottle of aspirin and a copy of the Apple II File Type Note for file type $50, auxiliary type $8010. If you think your SAT's were a nightmare to get through, you ain't seen nothin' yet!

Basically, the file is made up of a ton of variable length structures and a host of variables that were in memory when the document was saved. As with any other conversion process, the idea is to skip over all of the flotsam that the designers of the format thought was important and dig out what you want—in this case the text and formatting codes. Due to the number of variable length structures in an AppleWorks GS file, simply *finding* the actual text is the hardest part of the process! So, how do you find that text? Ugh. I was afraid you would ask that! Take a look at the source code on your *GS+* Disk, it's in the file **AWGS.h** in the **EGOed** folder. That and the tech note will tell you everything you need to know.

After you actually find the text, it's a simple matter of putting the TEMagic routines to use. Again, the source code on your *GS+* Disk will explain things better that I can here.

What's Missing?

Well, most of the stuff that was left out was due to a lack of support in the TextEdit tool set. For example, TextEdit does not support superscript or subscript text and, as I mentioned before, it only supports one ruler. Other things, like the lack of support for headers and footers, are due to limitations of the original design of EGOed. It was never intended to be a word processor, just a simple text editor. Headers and footers are in the domain of the word processor.

However . . .

People keep asking for new features in EGOed and I keep plugging them in there. As I said earlier, if you don't mind the size, I'll keep adding features. If you want to see EGOed become *Super*EGOed, or if you think a separate product would be justified, let me know. **GS+**

Figure 1

With all optimizations on, statements of the following form will trash the stack:

```
aGlobalStructure.aField = **handleToAValue;
```

To avoid this, use the form:

```
aLocalVariable = **handleToAValue;
aGlobalStructure.aField = aLocalVariable;
```

NoDOS v1.5

By Josef W. Wankerl

NoDOS is a New Desk Accessory (NDA) which provides the ability to delete, rename, move, and get/set file information for GS/OS files. NoDOS v1.0, written by Steven W. Disbrow, first appeared in V1.N1 of *GS+*. It was a great utility—I used it all the time—but it had some drawbacks. Namely, it was a modal dialog and it used ProDOS 16 calls instead of GS/OS calls. This new version fixes these problems and adds a bit more functionality—like being able to move files and being AppleShare aware (although we don't have an AppleShare network to test this on.) The origin of the NoDOS name is rather obscure, so if you want the complete story, check out *GS+* V1.N1.

Installing NoDOS

To install NoDOS, use the Installer program on your *GS+* Disk. If you need help using the Installer, see the "How To Use The *GS+* Disk" in this issue. After you install NoDOS, you must reboot for it to be available.

Using NoDOS

Once you have NoDOS properly installed, you can select it from the Apple menu that is available in most desktop programs. A tidy little window will then appear on your desktop with four buttons: Delete, Rename, Info, and About. To select a file utility, simply click on the appropriate button with the mouse. Each function will repeat until you click on the Standard File cancel button. Each button also has a key equivalence. If you want to Delete files, press D. If you want to Rename files, press R. If you want to get/set Info on files, press I. If you want to see the about dialog, press A. I always get frustrated with NDAs that can't be closed using the keyboard, so I made sure that NoDOS also supports the Command-W sequence to close itself.

Prefixes

NoDOS follows our proposed standard for saving prefixes (see "EGOed v1.33" in *GS+* V2.N4). What this means is that NoDOS remembers the folder you were in the last time you used NoDOS and goes there the next time you use it. When you are finished with a NoDOS function, NoDOS restores the current prefix to where it was before you used NoDOS. This makes it a bit easier for you to use your host application because the folder you are using with your host application is almost always different from the folder you want your NDAs to play with. If, for some reason, you don't want NoDOS to use its saved folder when you invoke a NoDOS function, simply hold down the option key when you select that function. When you finally cancel the NoDOS function, you can also hold down the option key to keep NoDOS from saving the current directory (i.e. NoDOS won't remember where it was).

Selecting Files

When you select an option from the NoDOS window, you will be prsented with a Standard File dialog to choose files from. Since NoDOS uses the Standard File tools, you navigate through your disks and select files just as you would with any other Standard File dialog.

To select a single file, simply click on it.

To select a group of files; click on the first file, hold down the *shift* key and then click on the last file in the group. All of the files in-between the first and last file will also be selected.

To select specific files (i.e. files that may or may not be grouped together), hold down the *Command* key (also known as the Open-Apple key) as you select your files. As long as you hold down the Command key, each file you click on will be selected.

Once you have all of your files selected, click on the Accept or Delete button, and NoDOS will perform the action you have requested.

Delete

With the old version of NoDOS, you could delete only one file at a time and you were prompted each time to make sure that you really wanted to delete the file. As mentioned above, the new delete lets you select multiple files and folders to delete. It also allows you three levels of prompting: prompt on locked files, prompt on all files, and no prompting. If you select a folder to delete, *everything* within that folder is also deleted—*so be careful*!!

Rename

With the old version of NoDOS, you could rename only one file at a time. The new rename lets you select multiple files and folders to rename and/or move. Simply select the files you want to rename or move and a Standard File dialog box will appear letting you either type in a new name for the file, select a new directory for the file, or doing both renaming and moving. Note that you can only move a file to a different location *on the same disk*. NoDOS *can not* move a file to a different disk.

Info

Again, with the old version of NoDOS, you could only get/set info on one file at a time. The new info procedure lets you select multiple files and folders to get/set info on. Simply select the files you want to get/set info on and a dialog will appear showing you information on the files you selected, one at a time. You are allowed to change the file type, auxiliary type, and access switches for the file.

There are key equivalences for each of the file access switch check boxes. They are all Command key combinations because, if they weren't, they would conflict with some hexadecimal digits (namely D and B.) To toggle the Delete box, press Command-D. The Rename box is Command-N (for Name.) The Backup box is Command-B. To toggle the Read and Write boxes, press Command-R and Command-W respectively.

The file type is displayed as a three-character abbreviation (the old NoDOS knew 43 file type abbreviations, the new

NoDOS knows 96. For a complete list of the file types that NoDOS knows, see "What NoDOS Knows" on the next page.) If the type is not known, it is displayed in hexadecimal. Also, if you would rather always see the actual hexadecimal file type number, you can hold down the shift key when you confirm the files to get/set info on and also when you cancel or confirm the info dialog box to move on to the next file. If you try to change the file type to a new three-character abbreviation that NoDOS does not know, the type will actually be changed to zero. Additionally, you can change both the file type and auxiliary type of a file by typing a hexadecimal number (preceded with a $) or a decimal number. To make the changes permanent, click on the Okay button. To discard any changes, click on the Cancel button.

I've found NoDOS to be an invaluable aid. I can go for weeks without using it, but then there is always an occasion that pops up where I am grateful that it's still in my desk accessory folder. I hope you will get some good mileage out of NoDOS, too. If you find any bugs, be sure to fill out the problem form supplied on the *GS+* Disk and send it in to help me figure out what's going on.

Programming Considerations

NoDOS v1.5 was written using ORCA/C v1.2 and my OS Library functions. There are two source code files for NoDOS: **NoDOS.CC**, which contains the NDA shell information, and **Info.CC**, which contains the code necessary to get/set file information.

NoDOS.CC

This file is the framework for the NoDOS NDA. It can also be the framework for a lot of other NDAs that use resources. Reading the comments in the source code will easily reveal what's going on at each stage of the game.

Info.CC

This file contains the code to get/set information for a file. Some tricks it uses are the `FakerModalDialog` function (similar to the procedure used in Autopilot, also in this issue, except that update events are not handled) and the `AppendString` function. The `FakerModalDialog` function handles events within the info dialog. The `AppendString` function takes a LETextBox2 string and appends a GS/OS input string to it. This allows the StatText controls in the information dialog to be set.

OS Library

The delete and rename/move functions are taken from the new version of the OS Library. They are called, and return, with no parameters. If you want to use the enhanced delete and move functions within your programs, simply make sure that Standard File is started and then make the OS Library call. The documentation for all the OS Library functions is on your *GS+* Disk.

There really wasn't too much involved with programming NoDOS v1.5. Like the original version, it's an incredibly simple (and useful) NDA. If you want to use resources with a NDA, conform to our proposed standard of prefix saving, or use my OS Library, NoDOS is a good example to base your work on. I hope you find it useful. **GS+**

Warranty Disclaimer

APPLE COMPUTER, INC. ("APPLE") MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE APPLE SOFTWARE. APPLE DOES NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE APPLE SOFTWARE IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS OR OTHERWISE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE APPLE SOFTWARE IS ASSUMED BY YOU. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME STATES. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU.

IN NO EVENT WILL APPLE, ITS DIRECTORS, OFFICERS, EMPLOYEES OR AGENTS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, AND THE LIKE) ARISING OUT OF THE USE OR INABILITY TO USE THE APPLE SOFTWARE EVEN IF APPLE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU. Apple's liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort (including negligence), product liability or otherwise), will be limited to $50.

What NoDOS Knows

Here is a list of all the three-character file type abbreviations that NoDOS v1.5 knows:

Code	Decimal	Hex	Description
UNK	0	$00	Unknown
BAD	1	$01	Bad blocks
PCD	2	$02	Apple /// Pascal code
PTX	3	$03	Apple /// Pascal text
TXT	4	$04	ASCII text
PDA	5	$05	Apple /// Pascal data
BIN	6	$06	Binary
FNT	7	$07	Apple /// Font
FOT	8	$08	Apple II or /// Graphics
BA3	9	$09	Apple /// BASIC program
DA3	10	$0A	Apple /// BASIC data
WPF	11	$0B	Apple /// Word Processor
SOS	12	$0C	Apple /// SOS System
DIR	15	$0F	Folder
RPD	16	$10	Apple /// RPS data
RPI	17	$11	Apple /// RPS index
AFD	18	$12	Apple /// AppleFile discard
AFM	19	$13	Apple /// AppleFile model
AFR	20	$14	Apple /// AppleFile report format
SCL	21	$15	Apple /// screen library
PFS	22	$16	PFS document
ADB	25	$19	AppleWorks Data Base
AWP	26	$1A	AppleWorks Word Processor
ASP	27	$1B	AppleWorks Spreadsheet
TDM	32	$20	Desktop Manager document
8SC	42	$2A	Apple II Source Code
8OB	42	$2B	Apple II Object Code
8IC	43	$2C	Apple II Interpreted Code
8LD	44	$2D	Apple II Language Data
P8C	45	$2E	ProDOS 8 code module
FTD	66	$42	File Type Names
GWP	80	$50	Apple IIGS Word Processor
GSS	81	$51	Apple IIGS Spreadsheet
GDB	82	$52	Apple IIGS Data Base
DRW	83	$53	Drawing
GDP	84	$54	Desktop Publishing
HMD	85	$55	Hypermedia
EDU	86	$56	Educational Data
STN	87	$57	Stationery
HLP	88	$58	Help File
COM	89	$59	Communications File
CFG	90	$5A	Configuration file
ANM	91	$5B	Animation file
MUM	92	$5C	Multimedia document
ENT	93	$5D	Game Entertainment document
DVU	94	$5E	Development utility document
BIO	107	$6B	PC Transporter BIOS
TDR	109	$6D	PC Transporter driver
PRE	110	$6E	PC Transporter pre-boot
HDV	111	$6F	PC Transporter volume
WP	160	$A0	WordPerfect document
GSB	171	$AB	Apple IIGS BASIC program
TDF	172	$AC	Apple IIGS BASIC TDF
BDF	173	$AD	Apple IIGS BASIC data
SRC	176	$B0	Apple IIGS source code
OBJ	177	$B1	Apple IIGS object code
LIB	178	$B2	Apple IIGS Library code
S16	179	$B3	GS/OS application
RTL	180	$B4	GS/OS Run-Time Library
EXE	181	$B5	GS/OS Shell application
PIF	182	$B6	Permanent initialization file
TIF	183	$B7	Temporary initialization file
NDA	184	$B8	New desk accessory
CDA	185	$B9	Classic desk accessory
TOL	186	$BA	Tool
DVR	187	$BB	Apple IIGS Device Driver file
LDF	188	$BC	Load file (genaric)
FST	189	$BD	GS/OS File System Translator
DOC	191	$BF	GS/OS document
PNT	192	$C0	Packed Super Hi-Res picture
PIC	193	$C1	Super Hi-Res picture
ANI	194	$C2	Paintworks animation
PAL	195	$C3	Paintworks palette
OOG	197	$C5	Object-oriented graphics
SCR	198	$C6	Script
CDV	199	$C7	Control Panel document
FON	200	$C8	Font
FND	201	$C9	Finder data
ICN	202	$CA	Icons
MUS	213	$D5	Music sequence
INS	214	$D6	Instrument
MDI	215	$D7	MIDI data
SND	216	$D8	Sampled sound
DBM	219	$DB	DB Master document
LBR	224	$E0	Archival library
ATK	226	$E2	AppleTalk data
R16	238	$EE	EDASM 816 relocatable file
PAS	239	$EF	Pascal area
CMD	240	$F0	BASIC command
OS	249	$F9	GS/OS System file
INT	250	$FA	Integer BASIC program
IVR	251	$FB	Integer BASIC variables
BAS	252	$FC	AppleSoft BASIC program
VAR	253	$FD	AppleSoft BASIC variables
REL	254	$FE	Relocatable code
SYS	255	$FF	ProDOS 8 system application

GS+ User Group Connection

By Steven W. Disbrow

Newsletter Facelifts

We get a lot of user group newsletters here at *GS+* Magazine. Some of these are printed on laser printers, but the vast majority are printed on the dependable old ImageWriter. Over the last few months though, I have noticed that an increasing number are printed with Hewlett-Packard ink-jet printers (the DeskJet and DeskJet 500) using either Harmonie or Independence (see reviews in *GS+* V2.N4). If you don't read the fine print, it's very difficult to tell these newsletters apart from their laserwritten cousins. If your newsletter is not as attractive as you might like it to be, you might want to suggest that your user group purchase a DeskJet. Not only is it fairly inexpensive (about $700 total—most user groups can handle that if it's spread out over a few months), but if you arrange things so that the person that prepares the newsletter gets to keep the printer (for as long as they continue to prepare the newsletter that is), you might just find yourself with a couple of new volunteers for the job!

Speaking of Volunteers

User groups need volunteers. In some newsletters, I have seen the presidents of user groups literally *beg* for volunteers. The most common plea is for volunteers to contribute to the newsletter. (Many user group newsletters are done *entirely* by one person!) In several cases, the situation has become so bad that presidents have to threaten to cease publication of the newsletter unless volunteers present themselves.

Volunteers are also needed to make presentations during meetings. I know that not everyone is comfortable with public speaking, but here again, in some user groups, *one person* does all of the speaking! It's no wonder that many user group officials suffer extreme burnout.

As an example, consider something that happened in my own user group. After I became president, I began to make a lot of presentations on my IIGS. After a few meetings, someone came up to me and said, "I have a IIe, not a IIGS. This junk is doing me absolutely no good! When are you going to have some IIe presentations?" All I could say was, "I'm sorry, but I don't have a IIe and I only buy IIGS-specific applications. We ask for volunteers at each meeting, but no one with a IIe has volunteered to do anything. Would *you* like to make a presentation?" Of course, they stamped away in a huff. Several times after that, I made it a point to ask for IIe/IIc demonstrations, but I only got one or two responses.

Apathy also raises its ugly head whenever user group elections come around. In most cases, only one person "runs" for an office. In many of the newsletters I've seen, several of the offices are listed as being filled by "Mr. Vacant - Faithfully serving since 1989."

How can *you* help? Simple. *Volunteer!* Ask your group president what you can do to help, and then, *do it!* Even if it's something as simple as sticking stamps on the newsletter, it will be appreciated. Do yourself a favor and get involved.

It's A Hit!

Last issue's announcement of our expanded *GS+* User Group Connection benefits has certainly proven to be popular! For those of you tuning in late, here's the deal: We are making available several of our more popular articles and reviews in plain ASCII and TeachText formats. These items, along with our demo programs and some other goodies are on our *GS+* User Group Connection Disk. To get the disk, there are only 6 rules that you have to follow:

- Your user group *must* be involved in our newsletter exchange program.
- You *must* send us a *blank, formatted* disk and a *self-addressed* return envelope with *return postage.*
- You may only print *one* article and/or review in each issue of your newsletter.
- You may not change the content of the articles in *any* way.
- You may not distribute the articles and/or reviews any farther or by any means other than printing them in your newsletter.
- You *must* print copyright information along with each item you print. (The appropriate copyright information is included in each file on the disk.)

The first disk (**:GSP.UGC.1**) contains the following items:

Beginner's Guide To The Finder - Parts 1, 2 and 3.
Beginner's Guide To System Disks - Parts 1, 2, and 3.
Trash Can Award: Inner*Express*

The disk also contains our EGOed and Transfusion Demonstrations and the HyperStudio stack version of *GS+* V1.N2. These files (and only these files) are public domain and can be placed on your club's Disk O' The Month.

That's seven months' worth of articles and three months' worth of demos (if you put only one demo on each disk). Not a bad deal for the price of a disk and a few stamps! At this point, we are trying to decide what should be on our second disk. If you have any favorite articles or reviews that you would like to see included, let us know! **GS+**

We Want You!

If your IIGS user group or IIGS special interest group (SIG) is not a member of the *GS+* User Group Connection, we want to change that. To become a member, simply have one of your club officers contact us. All we need is the name and address of your group. However, if you give us a free subscription to your group's newsletter, we'll give your group a free magazine-only subscription to *GS+* and access to the *GS+* User Group Connection reprints! Send that information and/or newsletter subscription to:

GS+ User Group Connection
P. O. Box 15366
Chattanooga, TN 37415-0366

How To Use The GS+ Disk

The first thing you need to do is ***make a backup copy of your GS+ Disk with the Finder!!!*** Next, put the original in a safe place. If you are having a problem making a backup copy, give us a call at (615) 870-4960. If your disk is damaged, let us know and we'll get a new one to you as soon as possible.

Installing The Software

To install the software on this issue's *GS+* Disk, start up your computer using System Software v5.0.2 or later (preferably v5.0.4), and then place your *backup* copy of the *GS+* Disk in a drive (You *did* make a backup didn't you?) Now run the Installer program that is on your *GS+* Disk. (From the Finder, you would double-click on the Installer icon.). When the Installer window appears, select the update you want to install from the left-hand window, and the disk you want to install it on in the right-hand window. Then click on the Install button. If you are installing Autopilot, be *sure* to refer to the "Autopilot" article in this issue for more detailed installation instructions. For more information on how to use the Installer, refer to your IIGS owner's manual.

For More Help...

Remember, each folder on the *GS+* Disk contains its own **a.Read.Me** file that describes the contents of the folder in detail.

What's On The Disk

There are 11 items in the root directory of this issue's disk. They are:

a.Read.Me

A lot can happen from the time we send this magazine to the printer and the time we get ready to mail them out. If anything does happen, we will put everything we can find out about it in this file. This is a plain text file.

Autopilot

This folder contains the Autopilot program launcher as described in the "Autopilot" article. Autopilot can not be properly installed using the Installer alone, so be *sure* to follow the installation instructions that are in the "Autopilot" article!

EGOed

This folder contains EGOed v1.34. This folder also contains complete documentation for EGOed v1.34 in the plain text file, **EGOed.Docs**. EGOed must be installed on a startup disk.

Icons

This folder contains the Finder icons discussed in the "Icons" article.

Installer

This is the Apple IIGS Installer. Run it to install the other programs on this issue's disk. For more information on the Installer, refer to your IIGS owner's manual.

NoDOS

This folder contains NoDOS v1.5 as described in the "NoDOS v1.5" article. NoDOS must be installed on a startup disk.

OS Library

This folder contains the OS Library routines which first appeared in *GS+* V2.N4. Documentation for the OS Library is in the file **OSLib.Docs**. When you install the OS Library, be sure that you have your main ORCA directory selected.

Problem.Form

This is the standard *GS+* bug report form. If you have a problem with one of our programs, fill out this form and send it to us. This is a TeachText file. You may use EGOed to view it.

Scripts

This folder contains all the scripts that are used by the Installer in order to automate the installation process.

Softlock

This folder contains the Softlock NDA described in the "Softlock" article. Softlock must be installed on a startup disk.

Writers.Guide

This is a TeachText file that explains what you need to do in order to write reviews, articles, programs, etc. for *GS+*. You may use EGOed to view it.

Please Remember...

The contents of the *GS+* Disk are *not* public domain or shareware! We depend on *your* honesty to stay in business. Please do not give away copies of the *GS+* Disk or any of the programs on it. If you do, we will not be able to stay in business. It really is that simple! ***GS+***

HAVING PROBLEMS?

If you are having a problem with your GS+ subscription, we want to help! But we can't help if we don't know about it! You can call us at (615) 870-4960, Monday through Friday between 9 a.m. and 6 p.m. Eastern Standard Time. Or, you can write to us at:

GS+ Subscription Services
P. O. Box 15366
Chattanooga, TN 37415-0366

Icons

By Steven W. Disbrow

This issue we have a bunch of nifty icons that were sent to us by one of our Australian readers, Maziar Maniei. There are so many cool icons in this file (**Maziar.Icons**), that it would be impossible to describe them all in the half-page that this column gets! However, to hit a couple of the highlights, let me just say that I especially like the "Unknown" icon and the icon for "Bobby." Some of the icons in this file will replace some of your current icons, so I strongly recommend that you use an Icon editor to pull out only the ones you want as opposed to copying the whole file to your **Icons** folder. Thanks for the icons Maziar!

The second icon file on this issues disk is called **Recycle.Icons**. These icons are intended to be used as replacements for your Trash Can icons. However, *don't* delete your old Trash Can icons! To use these icons, copy them into your **Finder.Icons** file *before* your regular Trash Can icons. To create these icons, I used the Quickie hand scanner to scan in a recycling symbol from an office products catalog. I then copied the graphic out of the Quickie application and pasted it into IconEd. The rest was just a matter of touch-up work. (I freely admit that I stole the idea for this icon from NeXT.)

That's all for this issue. Remember: we need your help! If you have any Finder icons that you would like to share with the rest of our readers, please send them on in! ***GS+***

IMPORTANT!
Use scissors or a knife to open disk bag!
Do not attempt to pull bag away from magazine!

Rumors, Wishes & Blatant Lies

By Prof. G. S. Gumby

ROM 2x2

Well, blow me down! Once again, sources *very* close to Apple are telling us that within the next few months, Apple will unleash upon the world a new Apple II CPU! This new CPU, which will be called the IIGS Plus (no kidding!) is a direct result of the embarrassment Apple suffered when it became evident that the current IIGS could not run HyperCard IIGS without a significant investment in additional hardware (more memory and a hard drive).

This new CPU, will have 2 MB of memory, a built-in SuperDrive, a new 640x400 graphics mode (please God, let the pixels be square!) and an optional 40 MB internal hard drive. (Perhaps this means that there will also be a built-in SCSI port.) Of course, all of these internal changes means that the case will have to be redesigned (and you'll probably need a new monitor), so it will probably end up looking like a brick or—even worse—a sawed-off Mac Classic.

Speaking of which, the price structuring for the IIGS Plus should be similar to that of the Mac Classic (i.e. the hard drive and keyboard will cost extra). At this point, it is not known if the price will be more or less than that of a comparably equipped Classic.

If you want more speed, forget about it. The IIGS Plus will continue to plod along at only 2.8 MHz. However, it should work with both the Zip GS and TransWarp GS.

If you want to see this new CPU before the rest of the world, be sure to go to KansasFest. That's where the unveiling is supposed to be. When the IIGS Plus ships, HyperCard IIGS will be included.

System 6.0

Speaking of unvielings, KansasFest will also be the place Apple will be showing beta versions of IIGS System Software v6.0. Included in this new system will be:

- A new Finder—written in part by Andy Nicholas (the author of ShrinkIt and ShrinkIt-GS)

- Read-only File System Translators for old DOS 3.3 and Apple Pascal disks.

- Read-Write File System Translators for MS-DOS and Macintosh HFS disks.

- The long-awaited animation and MIDISynth sound tools.

- A host of other great stuff, including an increase in overall speed!

1-900-HOT-STUFF

Of course, we all know that AE started it's 1-900 technical support service to *improve* support for it's Apple II customers. But, did you also know that to further cut costs, they decided to share the line with another service company? You didn't? Well, from 9 a.m. to 5 p.m. CST, the line belongs to AE. But, if you call between midnight and 6 a.m. CST, you get *Wicked Wanda's Wet World.* (No expansion slot jokes, please!)

The Hunt Is On!

First of all, let me say that *I* had nothing to do with it and I thought it was a stupid idea from the very start. It was all Steve's idea and you should hunt *him* down, not me. What am I talking about? Why, last issue's "Easter Egg Hunt" of course. The cover of last issue proclaimed that it was our "Special Easter Egg Edition" and asked, "How Many Can *You* Find?" A great many readers called up, completely baffled. "Where are the Easter Eggs? I've searched the entire magazine and can't find a single one!"

Well, since Einstein the publisher is too cowardly to tell you the truth, it's up to me to reveal the number and location of all of the Easter Eggs in the last *GS+*.

There were 12 of them. They were on the cover. In the photo. (For those of you that are still searching, here's a hint: they are next to the inflatable rabbit.)

Speaking Of KansasFest . . .

Even as I type this, our fearless leader is trying to line up the very first "Girls Of KansasFest" photo shoot. According to that peerless pinhead of publishing, "Our readers want to see what *really* goes on at these trade shows. They want hard-hitting facts and only the best in-depth investigative reporting. Barring that, of course, they want to see babes in skimpy bathing suits. Here at *GS+* Magazine, we aim to please!" Stay tuned for more on this rather improbable story!

15 Minutes

If you have not heard by now, "Burger" Bill Heineman has contacted the CBS newsmagazine *60 Minutes*, in an attempt to get them to investigate Apple Computer, Inc's lack of attention to the needs of their Apple II customers. Bill is requesting that if you have a (true) story to tell about bad dealings you have had with Apple or their dealers, you write to:

60 Minutes
Attn: Story Editor
524 W 57th Street
New York NY, 10019

It is also requested that you use *polite* terms and state only the *facts.* Remember, the objective here is to have *60 Minutes* make Apple sweat, not to have them broadcast a story titled, "Fanatical Apple II Pinheads—How Far Will They Go?"

Hey You!

We want your help! Send your Apple IIGS rumors, wishes and blatant lies to us at:

America Online, Delphi:
GSPlusDiz

ProLine/InterNet:
rumors@pro-gsplus.cts.com

US Mail:
GS+ Rumors
P. O. Box 15366
Chattanooga, TN 37415-0366 GS+

TMS Pro R45 Hard Drive

Price from manufacturer: $499
Additional 45MB cartridges: $67

TMS Peripherals
23123 SW 58th Avenue
Boca Raton, FL 33428-2036
Orders and technical support:
(800) 626-MEGS
Product information: (407) 482-3821

Reviewed by Steven W. Disbrow

Every computer owner *needs* a hard drive. On systems like the Macintosh and IIGS, with their massive system software configurations, this is doubly true. But do you need a *removable* hard drive? And, if you do, what *is* a removable hard drive and why should you give your money to TMS?

What A Removable Hard Drive Is

To pull a quote from my review of the CMS SDRM 45 Removable Drive (*GS+* V1.N5), "What [removable hard drive] means is that the hard disk platters are stored in a cartridge that you can actually remove. Sort of like a very expensive, very fast, and very high capacity floppy drive."

Why You Need One

If you don't already have a hard drive, the main reason to buy a removable hard drive is that *it's a hard drive*. With a hard drive you get the ability to store large amounts of data in a single place and the ability to access that data at speeds that will make long time floppy-drive users weep with joy. A *removable* hard drive has the added advantage of providing virtually unlimited amounts of storage. Filled up the 45 MB cartridge that came with your drive? Buy another one! Since you've already got the drive *mechanism*, you only have to pay for the cartridge! This drives the old "bucks per megabyte" ratio down very quickly and is, as you can imagine, much cheaper than buying a whole new drive. Another advantage is the high degree of data security that removable media has. Got a bunch of really inquisitive (i.e. dangerous) kids? Take your data-you-can't-live-without cartridge out of the drive and lock it up. Get a separate cartridge for the kids and put games and educational stuff on it. If they wreck *that*, make them buy their own cartridge!

If you *already* have a hard drive, there are several reasons that you should consider the purchase of a removable drive. The first three reasons are the same as for people that don't already have a hard drive. The fourth is backups. Now we all know that we should backup our hard drives every day, but who has the time to do it? Programs like Salvation: Bakkup make the job a lot easier by ejecting disks automatically, but it's still a pain having to sit there for 45 minutes pulling disks out and sticking them in until the backup is complete. But, if you back up your primary drive to a removable hard drive, you simply give it the destination pathname, start the backup and leave. No more disk swapping! Not only that, but it's *fast*! Using a Rev C Apple SCSI card (the slow one), it takes about 30 minutes to totally back up *both* partitions of my nearly full 60MB hard drive. Partial backups (modified files only) take between 1 and 5 minutes for each partition, depending on how much actual work I've done that day. With a Apple High-Speed SCSI or RamFAST SCSI card, it would be even faster!

Why Buy TMS?

Of course, the TMS Pro R45 has all of the features mentioned above. These are the basic things that you expect from *any* removable hard drive. The Pro R45 is also a very reliable drive. I've had mine for about four months now and have not had a single problem with it. This is also something that I expect from *any* removable hard drive. If I *had* had

problems, this review would have ended after about two paragraphs. So, if the drive has all of the things that you expect, what does it have that sets it apart from other removable drives?

Support

If you have reliability, you should not need service. But, if you *should* need service, there are a few little things that set TMS apart. First of all, TMS has a *toll-free* technical support number. (I think this speaks for itself.) Second, when you say, "Apple IIGS" they don't hang up, laugh, or treat you like a leper. If that doesn't convince you, how about a two-year limited warranty? Still not enough? Then how about the fact that if you don't like your drive, you can use the TMS 30-day Money Back Guarantee to get your money back? Two-year warranties, and 30-day Money Back Guarantees are the standard in the Mac hard drive market, but they are something of an oddity in the Apple IIGS world. This is especially nice to see happening after Applied Engineering's recent decisions to change their technical support over to a 1-900 service (see "Writer's Block" in this issue) and cut their warranties to only 1 year.

I have used TMS tech support several times and each time I have been very pleased with the level of support I have gotten. Thinking that I might be getting "special treatment" because of *GS+* Magazine, I did some snooping around in the TMS customer support areas of various online services. Without fail, the comments I read were positive. TMS goes out of their way to please their customers and their customers are very vocal about letting people know it.

Little Things Mean A Lot

Beyond the reliability of the drive and the service of TMS, the Pro R45 has a couple of other little features that make it an excellent buy. First is the push-button SCSI ID select switch on the back of the drive. This allows you to quickly and easily set the SCSI ID of your drive (each SCSI device must have a unique ID so that it can be identified and given the appropriate priority in your chain of SCSI devices). Push one button, and the number goes up. Push the other button and the number goes down. My other removable hard drive (a CMS drive, see review in *GS+* V1.N5), uses a set of dip switches for its SCSI ID select. To set it, you have to refer to the manual for the correct dip switch settings. I don't even know where that manual is anymore!

Another nice little feature of the Pro R45 is its "little" size. At only 2.5-inches high, the Pro R45 fits very nicely in the scheme of my things. (See photo).

What's Missing?

The only things that are missing from the Pro R45 are things that you find on more expensive removables. For example, there is no extra fuse or fuse storage "box." (What a loss.) About the only thing I *really* wish that the Pro R45 had would be the additional electrical outlets that you find on some of the more expensive drives. Most of these drives come with two additional power outlets that you can plug your other peripherals into. However, considering that the cost of a good, surge-suppressing power strip is about $15 (and that it will provide you with 6 outlets) this is not a major loss.

Conclusion

If you are the cynical type, like me, you are probably thinking, "He's just saying all this because of the TMS ad on the back cover!" Well, I'm not. I was a TMS *customer* before they were a *GS+* advertiser. In fact, after I bought my Pro R45 and had called their technical support line, I was so impressed with the product and the company, I practically *begged* them to advertise with us. It took some work, but this is the kind of company that IIGS owners need to know about.

So, if you need a hard drive (and you do—trust me), the TMS Pro R45 is a *very* good choice. The price is right, the drive is fast (as removables go) and reliable, and the service is second to none. **GS+**

RamFAST/SCSI

Retail price: $229
Typical mail-order price: $174

C.V. Technologies
1800 E. Whipp Road, Suite 200
Kettering, OH 45440
(513) 435-5743

Reviewed by Joe Wankerl

Well, I finally did it. I bought *yet another* hard drive! Why, you ask? Well, because I decided I needed a good way of backing up the information on my other hard drive, a 40 MB Vulcan internal hard drive. So I got a TMS Pro R45 removable SCSI drive (see review of the TMS Pro R45 in this issue). To go along with the drive, I also needed a SCSI interface card. Steve has the Apple High-Speed SCSI card and recommended it, but I had heard rave reviews about the RamFAST/SCSI card, so I decided to try it. And boy am I glad I did!

Installation

Needless to say, I was very anxious to get my system up and running when the package arrived at my door. I read the introduction to the manual (come on, folks, *read the manual*—it saves headaches later—*really*) and slipped the card into slot 1 of my system. I would have used slot 7, but I am also on an AppleTalk network and AppleTalk requires slot 7 on a ROM 01 IIGS (which is what I have). It *barely* squeezed in there. In fact, it was resting against the Vulcan. Just to be safe, I put in that little blue divider card that came with my Vulcan to separate the card in slot 1 from the drive. Next I plugged the SCSI drive into the RamFAST/SCSI card's connector and then turned on the power. Easy, eh?

Well, before I go into any details of the many features of the RamFAST, I want to give you an idea of what it can do. I restarted my computer, and the time it took to boot and display the Finder menu bar was 35 seconds. The same task from my Vulcan took 50 seconds (I have lots of Inits/CDevs/DAs). Not that impressive you say? Well maybe not *too* impressive, but check this fact out: I selected the 32 MB partition of the removable SCSI drive from the Finder and did a disk verify on it—it took 1 minute and 14 seconds to complete the verify. The 32 MB Vulcan partition took 7 minutes and 16 seconds. My 1 MB RAM disk took 10 seconds—multiplying by 32 gives us an approximate 320 seconds (5 minutes and 20 seconds) if the RAM disk was 32 MB. Not bad, eh? Well, just to ram (ha ha) the fact home, I ran DiskTimerGS, a freeware utility by Joe Jaworksi designed to measure hard drive performance. The Vulcan had a read performance of 54, seek of 104, and adapter of 25. The lower the score, the faster the drive is. Now those values are pretty darn good compared to the list of other hard drive values that came with DiskTimerGS. My removable with RamFAST/SCSI card came in with a read performance of 12, seek of 6, and adapter of 6! Those are the best values of any I've ever seen! (For explanations of those values, pick up a copy of DiskTimerGS—it's available on our pro-gsplus BBS [(615) 875-4607] as well as many other online services.)

Built-In Software

Well now that you have a general idea of what the RamFAST can do, I'll tell you about some of the card's features. One of the outstanding features of the RamFAST is that you don't need a disk containing utilities to format and partition your SCSI drives. I know that I can never find that one configuration disk at the crucial moment that things go haywire in my system. All the software needed to format drives, partition drives, and configure the RamFAST is contained on the RamFAST card itself—instantly available. The RamFAST card installs a ROM disk into your system: this ROM disk contains the RamFAST configuration utility and a GS/OS device driver that improves performance of the RamFAST from GS/OS. Simply copy the driver to your ***:System:Drivers** folder and forget about it.

The Settings

The RamFAST card has 8 dip switches on it that you set to configure the card. The switches control whether terminator power is supplied or not, which machine you are using (IIe or IIGS), whether your IIGS is DMA compatible or not, whether you have a ROM 01 or ROM 03 motherboard, whether or not the ROM disk is available, and how long the card will wait on startup until it times out (3 or 30 seconds). Most of these settings can be overridden by the utility program.

The Utility Program

The utility program that comes with the RamFAST/SCSI card is very easy to use. It is a text-based application (since you can use the RamFAST on a IIe as well.) The main purpose of the program is to format and partition your hard drive. You can format your drive at an interleave of between 1 and 9—yes, that's right, the RamFAST card can keep up with a 1:1 interleave! The utility program can also tell you what kind of devices you have hooked up to it—it can display the name, serial number, number of cylinders, number of heads, number of sectors per track, and current interleave factor for any drive on your SCSI chain.

Another feature of the RamFAST card is it allows ProDOS 8 to see *eight* hard drive partitions instead of two. It does this by "phantom mapping" additional slot and drive pairs. You can set which slots are used for the phantom mapping with the utility program. Some ProDOS 8 programs can't handle that many drives all at once, so you also have the option of allowing six, instead of eight, additional phantom drives.

Partitions

The utility program sets parameters for each partition. You can change the name of the partition, whether the partition is locked or not, whether the partition is active (online) or not, and a lookahead factor. The lookahead value is used to speed up sequential reads. The manual recommends a default of a 12K lookahead, so whenever you perform a read, 12K of additional data is read and placed in the RamFAST's cache memory. The next time you perform a sequential read, the data has already been read and all that needs to happen is to transfer the data from the cache to main memory. No disk access is required!

The Hardware

The RamFAST card has 256K of dedicated

cache memory and an internal processor that runs at 10 MHz. The fast processor speed allows the RamFAST to keep up with a 1:1 interleave and transfer data at 1 megabyte per second. The RamFAST card transfers data to memory via Direct Memory Access (DMA). With DMA, your computer's processor is not busy transferring data—the RamFAST card knows where it should go and puts it there as fast as possible. This is how the card can attain such fast transfer rates. With the fast processor, the RamFAST can handle a 1:1 interleave for both reads *and writes*! If you have an old Sider hard drive, you're in luck—the RamFAST supports both SASI and SCSI hard drives! Just get yourself the appropriate cable (Larry Beyer of B&D Computer Repair can set you up with the correct cable. Give him a call at (312) 735-9010), plug it in, and go. If you ever run into trouble, don't worry—the RamFAST is covered by a limited lifetime warranty.

ROM Limitations

So far, I've only found two limiting factors on the RamFAST/SCSI card. The first is with using the ROM disk. Although it's a brilliant concept, it has its drawback. When the ROM disk is active, it takes up a place in your SCSI chain—limiting you to 7, instead of 8, active partitions. I can live with this, though, because I rarely have the ROM disk active, and you can copy all the essential files off the ROM disk so you never need it active past the initial setup.

Removable Cartridges

My removable drive works with a SyQuest mechanism, probably the most common removable drive mechanism around. I have transferred cartridges between drives that have a RamFAST/SCSI, Apple High-Speed SCSI, and Apple Rev. C SCSI card with absolutely no problems. If you ever have the need to transport cartridges from one environment to another, it's nice to know that everything is compatible.

The other limiting factor of the RamFAST is its internal ROMs. The current ROM version for the RamFAST card is v2.00, however if you buy a RamFAST, you'll probably get ROM v1.11 or v1.12. The v2.00 ROMs are deemed an optional luxury by C.V. Technologies at this point and you need to send them a check for $15 to get the new ROMs. They are not set up to handle credit card orders, so mailing a check is the only way you can get the ROMs. The standard ROMs work just fine—if all you need is hard drive support. If you want to use a tape backup system or CD-ROM drive, you'll have to spring for the v2.00 ROMs. Eventually (a C.V. Technology representative said in a few months) the RamFASTs will be made with the v2.00 ROMs, but the retail price will go up $15 as well.

v2.00

Although it took a while, I finally got my v2.00 ROMs and installed them quite easily. Within a few minutes, my machine properly supported my removable drive by recognizing cartridge swaps. The v2.00 update also inproved upon the built-in software by providing a much more friendly interface. If you have a SCSI tape drive, you're also in luck. The new v2.00 ROMs will allow you to backup your harddrive to tape *in the background*! This option is really slick. All the processing takes place within the RamFAST/SCSI card—the CPU isn't involved at all so you won't notice any speed loss. The backup is an image backup, so if you need to restore data, you have to restore an entire partition. You can, optionally, access the tape drive to get the file but it will be very slow as tape drives use sequential, rather than random access. (This is the same as the difference between your cassette tape player and your audio CD player.) C.V. Technologies also has a patch to their software to backup to a hard drive instead of a tape drive. This won't make much sense unless you're backing up to a removable hard drive. I don't have a tape drive to test, so I tried backing up to my Pro R45 drive. I found out that there are bugs with this backup option. Whether the bugs exist with backing up to a tape drive or not I can't say, but my best guess is that the bugs are there since the patch to backup to a hard drive simply disables the check to see if the destination drive is in fact a tape drive. The bug I found was that I was unable to backup anything but my first hard drive partition. I reported the bug to C.V. Technologies and so far they don't know what's wrong.

Another nifty feature of the v2.00 ROMs is the ability to set a password. No, it's not a password to the hard drive, it's a password to let you use the partitioning software. It's a feature that allows you to keep nosy people from formatting and partitioning your drive. Also, if you don't want a password, but are afraid that someone will come along and set one so you can't get to the partitioning software, you can disable passwording. If you ever forget your password or you have disabled passwording and want to reenable it, you can call C.V. Technologies and they will get you around it.

Perhaps the best feature of the v2.00 ROMs is the flexibility you have when you boot. If you hold down the zero key whenever your computer restarts (power up, after a reset, etc.), you will automatically come up in the configuration program. This feature is nice if you have turned off your ROM disk, don't have a copy of the configuration program anywhere, and you wish to backup a drive or just change things around. The new configuration program lets you select which partition you boot from as well. And if that isn't enough, you can hold down the number keys (1 - 9) during a restart and you will boot from that number partition, overriding your default boot partition. This is really handy if you boot into both GS/OS and ProDOS 8 in equal frequencies.

Recommendation

The RamFAST/SCSI card is to be used mainly for mass storage devices such as hard drives. Additional SCSI hardware, such as scanners, will not work with the RamFAST. If you want to use that kind of SCSI equipment, your best bet is to get the Apple High-Speed SCSI card. If you want blazing speed from your hard drive, the RamFAST can't be beat, especially with the v2.00 ROMs. GS+

HyperCard IIGS vs. HyperStudio

By Jonah Stich

HyperCard IIGS v1.0
by Andy Stadler and a whole host of other people
Retail price: $99
Typical mail-order price: $80
Not copy protected
Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, CA 95014
(408) 996-1010
Requires System Software v5.0.4 or later, 1.5 MB of RAM, and a hard drive.

HyperStudio v2.01
by Michael O'Keefe, Dave Klimas and Eric Mueller
Retail price: $149
Typical mail-order price: $85
Not copy protected
Roger Wagner Publishing, Inc.
1050 Pioneer Way, Suite P
El Cajon, CA 92020
(619) 442-0522
Requires System Software v5.0.2 or later, 768K of RAM, and one 3.5-inch drive.

Recently, Apple released its long-awaited version of HyperCard for the IIGS. With the release of this product, the IIGS has been reaffirmed as a powerful hypermedia platform. There are now two major hypermedia products available for the IIGS: HyperStudio from Roger Wagner Publishing, Inc., and HyperCard IIGS from Apple Computer, Inc. However, with this freedom of choice comes also the burden of decision—you now must decide which program is right for you. In this article we will explore the similarities and differences between the two programs, pointing out their respective strengths and weaknesses along the way.

What Did You Say?

First off, lets define some of the common 'hyper' terms. Hypermedia is the name given to the form of communication and information exchange that HyperCard and HyperStudio provide. Its main claim to fame is that instead of the usual, passive modes of acquiring information, such as reading a book or watching a documentary on TV, hypermedia is interactive. In its purest form, hypermedia is also unstructured, rather than the standard linear nature of a book or documentary. Finally, hypermedia tries to involve as many of the senses as possible (using many different "media"—ironically, "hyper" is defined as "excessive, overmuch," which may be a subtle form of warning). In both HyperStudio and HyperCard IIGS, a hypermedia presentation is called a "stack." The intended analogy is a stack of index cards, each with one piece of information (another hyper concept is that there should, ideally, be only one main piece of information per screen). You can instantly move to any card in this stack of index cards, allowing you to access this information in any order you like. Similarly, a hypermedia stack is made up of "cards" (also known as screens) from which all of the other cards should be readily available. One central stack, which is called the "home stack," is used as a starting point from which you can reach all of your other stacks.

Outside In

HyperStudio has a retail price of $149, but you can usually get it from a mail-order house for about $85. HyperStudio comes with a single manual and a few pages of release notes, totalling about 85 pages of documentation. It comes on four 3.5-inch disks, one containing the main program and several auxiliary programs (such as a digitizing program, allowing you to use the included audio digitizer, and a program to show a picture and/or play a sound as the computer starts up), one with several demo stacks, and the final two containing a clip-art and sound library. As mentioned previously, HyperStudio also comes with a "slotless" (it doesn't plug into a slot, though it does block one) audio digitizer and a microphone.

HyperCard IIGS lists for $99 and is available from a few mail-order dealerships for about $80. HyperCard IIGS comes with three manuals, totaling about 750 pages of documentation to wade through. Luckily, much of this documentation is also available in the included help stack, which makes it a bit easier to assimilate the relevant information. HyperCard IIGS comes on 6 disks: a program disk, an installation disk, and four stack disks. One of the stacks is a clip-art stack, and there are some sample sounds available as well.

HyperStudio occupies 2.3 MB on my hard drive. Of this, about 600K is essential to program operation (the HyperStudio program itself, plus the "home" or main stack, plus some other little goodies). The rest of the room is used by the clip art/sound libraries and the demonstration stacks. HyperCard IIGS resides on a hefty 3.2 MB, of which about 1.5 MB are necessary for program operation. This means that HyperStudio can actually be run from a IIGS with only a single 3.5-inch drive (with a bit of disk swapping), whereas HyperCard IIGS requires a hard drive. Furthermore, with HyperStudio's home stack open, about 785K of memory was in use. From HyperCard IIGS's home stack, 1190K was in use (1265K from the help stack). As you have probably guessed, the HyperStudio package calls for a minimum of a single 3.5-inch drive and at least 768K of memory. HyperCard IIGS requires a minimum of a hard drive and 1.5 MB of memory. Keep in mind that these are *minimum* requirements for both programs—they'll work, but you aren't left with much elbow room.

No Guts, No Glory?

If you believe that more is better, then HyperCard IIGS is the program for you. Almost everything that HyperStudio can do, HyperCard IIGS can do more of, though it doesn't always do it better. HyperCard IIGS simply has more guts than HyperStudio does. A comparison of the two is somewhat like a comparison of a sumo wrestler to a marathon runner.

The main difference in bulk is the result of the different focuses of the two

In Defense Of HyperStudio

By Dave Adams

Ok, let me start this by saying that Diz is going to try to bury me with all sorts of technical mumbo jumbo and make me look like a fool. But that's why he asked me to write this. Diz is a programmer, and as a result of that he sometimes doesn't see the world in the same way that the rest of us do. He's wild about Hypercard IIGS (HCGS). I'm a HyperStudio (HS) fan. Here's why:

1. HyperStudio runs easily on a stock IIGS.

All you need is 1.25 MB of RAM. Most ROM 01 IIGS users have a full RAM card, if nothing else, in their slots. HyperStudio was written to run on the stock IIGS and actually contains features to squeeze as much RAM as possible for the program. HCGS requires 3 MB of disk space and 1.5 MB of RAM to operate. This is obviously not able to run on a stock IIGS. Well, you can run really small stacks, but your arms will fall off from disk swapping. Let's face it—the vast majority of IIGS's out there don't have hard disks, accelerators, or mega RAM. The main reason why Mac HyperCard was so popular was that it came free with every Mac, and every Mac could use it. Unless you have a hard drive, HCGS isn't for you.

2. HyperStudio is easier for a novice to use.

I'm speaking from experience here. There are HS stacks created by first graders in the public domain. I myself have taught all sorts of students to use HS. They take to it quite easily. I know that HCGS has many of the same features, but I've seen it and played with it, and I prefer HyperStudio. Sure, HCGS has scripting, but if I wanted to

In Defense of HyperCard IIGS

By Steven W. Disbrow

Oh, *really* Dave! I didn't ask you to write your essay because I knew I could take you apart. I asked you to write it because I knew you would give the best argument possible—and you did! However, I think that you are simply missing the point in comparing the two products. That point is: there is no comparison!

You see, regardless of what you may have heard, HyperCard IIGS and HyperStudio are completely different types of programs. Oh sure, they look similar, and they both have the same basic premise of allowing you to organize information in any fashion that suits your fancy. The point where they diverge is when you begin to try to *manipulate* your information. With HyperStudio, data manipulation is pretty much limited to an interactive slide show. HyperCard IIGS on the other hand, is actually a *programmable database* in hyperclothing. Oh sure, HyperCard IIGS can do exactly the same things as HyperStudio, if that is all you want. However, if you really need to *work* with all the data you've squirreled away on those cards, HyperCard IIGS leaves HyperStudio in the dust.

For instance, let's take the simple example of a stack containing a comic book collection. Both HyperStudio and HyperCard IIGS are up to the task of simply listing all of your comics. If you want, you could type in a plot synopsis for each one and even scan in a small picture of the cover for each issue. Both programs. Both programs have tools (a collection of those commonly found in painting or desktop publishing programs, plus some extras specific to stack making) to allow the user to use graphics, text, sounds, and animation to create the cards in their stacks and add data to them. HyperCard IIGS, however, goes a step further—it has a built-in programming language, *HyperTalk*. You don't need to be able to program HyperTalk to use HyperCard IIGS, but if you do use it, you can add a whole layer of complexity to your stacks that HyperStudio simply can't touch. For instance, with HyperTalk it is possible to add your own menu to HyperCard IIGS's menu bar; there is no provision for this in HyperStudio. On the other hand, because HyperStudio has no programming language, it is often easier to use, particularly for beginners or people with a strong aversion to programming. For instance, if you have a text file that you would like to let the user of your stack read, you have to dip into HyperTalk in order to keep the file on disk and have HyperCard IIGS load it when it needs it. HyperStudio, on the other hand, has automatic provisions for loading text (as well as graphic, sound, and animation) files from disk.

Another major difference in the orientation of the two programs is their use of resources. HyperCard IIGS makes extensive use of resources, whereas they are barely used at all in HyperStudio. A few examples: the HyperStudio program file has 5K of resource data; the HyperCard IIGS file has 73K. HyperStudio stacks make no use of their resource fork, whereas much of the data for a HyperCard IIGS stack is stored in the resource fork: the home card has 9K of resources, and the ButtonIdeas stack (which contains a plethora of buttons for you to copy and paste into your own stacks) has a whopping 192K! Again, this adds another possible layer of sophistication to HyperCard IIGS stacks, but also another layer of complexity.

If you are intimidated by an abundance of options, HyperCard IIGS is not the program for you. For every tool, there are more options than most people will ever need. This makes it a very powerful program, but for some people can lead to confusion or frustration. HyperStudio, on the other hand, gives you all of the tools you need, but doesn't force you to choose from a multitude of options at each step along the way. For instance, both HyperStudio and HyperCard IIGS have a polygon tool—the standard paint tool that allows you to draw multiple connected line segments to form one shape. HyperCard IIGS, however, also has a "rotating regular polygon" tool. This tool automatically draws one of six regular polygons and allows you to rotate it about its center. You can accomplish the same thing with HyperStudio's polygon tool, but it's easier in HyperCard IIGS.

HyperCard IIGS's interface is simply much fancier than HyperStudio's, there are no two ways about it. At first, this gives the impression that HyperCard IIGS is a better program than HyperStudio. This is not so. Although some of the features of the interface are very convenient, such as the extensive use of tear-off menus (Tear-off menus allow you to "tear" a menu from the menu bar. The menu then occupies its own window which you can move around the screen.), they are also merely

learn a programming language, I'd learn a programming language. As it is, HS lets you create stacks quicker and easier than HCGS. For those of us that still haven't figured out the spreadsheet in AppleWorks, that is a big factor in favor of HS. Let's face it, the majority of people that buy IIGS's never use them to their full potential. I teach high school. Educators are scared to death by computers. They don't understand them. But give them a 10-minute course in HyperStudio, and they are creating stacks and are extremely impressed by how easy to use computers are. I use HyperStudio as a way to get people excited about the IIGS. Most people are not power users—they want simplicity, speed, and power. HyperStudio provides all three.

3. HyperStudio comes as a complete package.
With every copy of HyperStudio you get a sound digitizing board and a microphone. You get lots of clip art to use in the program. You get XCMDs that expand your power as you learn the program. You get sound files and sample stacks. Sure, HCGS comes with many of the same items, but it doesn't give you the hardware to actually digitize your own sounds! It doesn't come with programs like Sound Shop and Sight and Sound. Sure, HCGS lets you use sounds, but it doesn't give you any way of making your own. If you don't have a modem, then you can't download them. Guess you better call some public domain distributor, because you're on your own, buddy.

4. The manuals for HS are better for beginners.
The first thing Diz showed me when he got HCGS were the manuals that came with it. They must weigh about five to nine pounds. Any programs even have a 'find' option that will let you find your X-Men® comics in the wink of an eye (just in case She-Hulk® is on her way over to tear them up). But what if you wanted to, say, get a *count* of all of your X-Men? What about an average of the prices for all of those X-Men? How about finding the issue that has the highest resale value? (To convince your spouse that your aren't wasting your money!) You could even use HyperCard IIGS to figure out the interest on that loan you'll have to take out to buy a copy of Superman® number 1! I've looked all through that wonderfully simple HyperStudio documentation and I can't seem to find any way to do any of those things, Dave.

And what about reports? Let's suppose the sad day comes that you have to part with your comic collection. If you have your collection stored in HyperStudio, you have three choices: you can print out each card of the stack (taking up about 1 page per card), you can lug your IIGS down to the comic shop to show the buyer your collection, or you can (horrors!) take your comics out of their boxes and down to the shop to raffle off.

You can do all of these things with HyperCard IIGS, of course. Or, you can print an honest-to-gosh *report* containing only the *relevant* information that you need to get the job done. If you were also using HyperCard IIGS to keep track of your favorite comic dealerships, you could use it to print a set of mailing labels (3-up if you want) and you could then send a copy of your report to each prospective buyer. Or, if you want, you could

conveniences. Their absence from HyperStudio is not in the least detrimental to the program's usefulness.

HyperCard IIGS does, however, seem to be somewhat better planned than HyperStudio, and for this it picks up a slight edge in usability. In HyperStudio, for example, there is a white border around the pencil tool that blocks out the background underneath the pencil, making it impossible to do detail work without using the zoom mode; HyperCard IIGS does not have this restriction. Also, in HyperStudio, picture buttons (buttons that are represented by a picture) are not highlighted when clicked. This can lead to a bit of disorientation, particularly if the next card is slow in appearing, as you become unsure of whether you actually pressed the button. In HyperCard IIGS, all buttons can be highlighted when clicked (the author of the stack decides whether or not a button is highlighted—you can usually change it if you disagree with the author's decision). HyperCard IIGS also allows you to change all attributes of objects, whereas HyperStudio is somewhat restrictive of what parts of an object are editable after the object has been created. For instance, once a button is created in HyperStudio, only the name and style can be edited. To change one of the other button parameters, you must *delete* the button and make a new one. HyperCard IIGS allows you to edit all aspects of a button at any point in time.

On the other hand, HyperStudio has a "run-time" version that you can legally give to people who don't own HyperStudio so they can use your stacks (although they won't be able to edit them). For someone else to use a HyperCard IIGS stack that you designed, they also have to have a copy of HyperCard IIGS—a major limitation for people hoping to create stacks for others to use. HyperCard IIGS is also the first IIGS program that I know of that automatically saves files that have been modified. In some instances, this is a great thing, as it

program that requires that much reading can't be "intuitive." Then he bought a book to learn scripting better. He says that you really need this book to get the most out of HyperTalk. Wow, four books for one software package. What next? A quick reference poster? If HCGS is so easy to use, how come they rewrote *War and Peace* to show you how to use it? HS comes with a simple to read and excellent manual. It gets you creating stacks right off the bat. It doesn't beat around the bush or waste your time. And as a tribute to HS's simplicity, you don't need to buy anything else.

5. HyperStudio lets other people view your stacks.
Let's face it, you need HCGS to see a HCGS stack. That's nice. Let me send you my latest stack on 7 disks so you can look at it. The stack only takes one disk but you'll need the others to view it. On the other hand, I could give you a HS stack with the run-time module and all you'll need to see it is a program launcher. I guess Apple never learned to share in Kindergarten.

6. Roger Wagner
Let's face it. Roger supports the IIGS. He makes his living off of it. You can order clip art disks, XCMD libraries, and other fine products from his company. Want to conduct a hypermedia workshop? Roger will send you all of the materials necessary for a complete workshop. Who cleaned up at the first Apple II awards? Guess. Rog knows the IIGS.

Sure, HCGS comes directly from Apple. Maybe someday they'll let all the Apple dealers know that it exists. Maybe they'll dump it on Claris even have HyperCard IIGS dial up the dealerships for you so you could speak to your prospective buyers in person. Gee, Dave, I can't seem to find a section on reporting in the HyperStudio manual.

Of course, to do most of those things with HyperCard IIGS, you are going to have to do a little bit of programming in HyperTalk. So what? I, for one, don't see what all the fuss is about. Did people complain when Apple decided to put Applesoft in ROM? No. Those that needed it used it to make their Apple IIs more productive. Those that didn't need it didn't use it. Like Applesoft, HyperTalk is simply there for those that need (or want) to use it. If you don't want to use HyperTalk, you don't *have* to use HyperTalk. You can still create good HyperCard IIGS stacks without typing a line of HyperTalk. And as far as programming difficulty goes, Applesoft is about 10 times harder to program in than HyperTalk, and the results are nowhere near as impressive.

For example, during the last year we have used DB Master Professional to manage our customer database. Without going into detail, I'll just say that I was less than 100% pleased with DB Master. (My definition of a relational database is somewhat different than DB Master's.) So, when I got HyperCard IIGS, one of my first projects was to convert our customer database from DB Master. Using HyperCard IIGS, I created three stacks that contain our customer information, sales information, and all of the custom reports that I need to keep Noreen happy at tax

makes it easy for beginners to use, and allows you to personalize stacks painlessly. On the other hand, it can't be turned off! So, if you make major modifications and then realize that you shouldn't have, there's no "old" version on disk that you can drop back to. Unless you made a copy of the stack before editing it, the deleted data is gone for good.

With all of these extra features, however, comes the two main curses of HyperCard IIGS—it's big and it's slow. The lesser of the two evils, the size, can be dealt with by the purchase of more memory and bigger hard drives. The slowness is much harder to control. I have a ZipGSX accelerator in my IIGS, and even though my computer is now running at 8 MHz (instead of the usual 2.8 MHz), I still feel HyperCard's lack of speed more often than I would like. Most things run at an acceptable clip—many of the tools are even a little on the speedy side. Some things, however, such as detecting clicks in buttons and moving from one card to another are often painfully slow. HyperStudio, on the other hand, suffers very little from speed problems. Although there is sometimes a slight delay when moving between cards (particularly if the current card has been modified, and HyperStudio has to remember the modifications), its speed is for the most part completely acceptable.

Other Stuff

HyperCard IIGS can, through the use of a stack called HyperMover (which is not included in the HyperCard IIGS package), load and use Macintosh HyperCard stacks. There is already a huge library of Macintosh HyperCard stacks, providing HyperCard IIGS with a large collection of possible stacks from the word go. HyperStudio, on the other hand, has been available for over two years now, and there are hundreds of stacks available specifically for it. These stacks have a slight advantage over Macintosh HyperCard stacks that have been converted to HyperCard IIGS stacks because they were made on the IIGS and, therefore, use all of its available resources, including color graphics (Macintosh stacks are usually in black and white).

A quick trip to my local bookstore revealed seven books on Macintosh HyperCard. Because of the amazing similarity between the IIGS and Macintosh implementations of HyperCard (though this has changed with version 2.0 of Macintosh HyperCard) all of these books apply to HyperCard IIGS as well. These books (including two from Apple) show you how to get the most out of HyperCard. There is also an Apple book specifically on programming HyperCard IIGS in HyperTalk. On the other hand, there are, to the best of my knowledge, no books on using HyperStudio (there definitely weren't any at the local bookstore). Although some of the ideas in the HyperCard books apply to HyperStudio as well (good stack design is good stack design, regardless of the hypermedia program used), much of the information is too specific to HyperCard to be of much use to HyperStudio users. There are, however, at least two HyperStudio newsletters and one magazine, *Hyperbole* (published by ResourceCentral), so support for HyperStudio is not completely lacking. Excellent sources of support for HyperStudio can be found at your local user group, as well as online services such as America Online and GEnie. HyperStudio has been around for long enough that many people have had a chance to learn how to

and you can get an upgrade right after the next AppleWorks GS upgrade . . . right. Maybe someday they'll advertise the IIGS. Until then, look for more "synergy."

7. We got more stacks than you do! (Nyah!)

At least for now. HyperStudio has been around for a few years and, consequently, there are more HS stacks in the public domain. Sure, you might argue that there are thousands of HyperCard stacks for the Mac that can be ported over. Go ahead. Unless you are a developer or a user group, you can't get HyperMover. Oh yeah, it only runs on a Mac. And it only works one way, so don't plan on sending your HCGS stacks to the Mac owner down the street.

In summary, I feel that HyperStudio is the best hypermedia product for the average IIGS user. Although I made some rather nasty comments about HyperCard GS, it is an excellent program. But, I don't think that it is the be-all-to-end-all program that many people are saying it is. Scripting is nice, but many of us simply don't have the time to learn it. I want results quickly. I get them with HyperStudio. If I need a custom sound, I grab my microphone and record it. I can share my stacks or other stacks with non-HyperStudio owners with the run-time module. I can download hundreds of public domain stacks. Most importantly, when I have a question about HyperStudio, I call up America Online or GEnie, and I get it answered. Quickly. And almost always by Roger Wagner himself. HyperCard IIGS owners . . . who you gonna call? Sculley? GS+

time. Using Hyper*Talk*, I'm able to link those three stacks into an *almost* relational database system that is ten times easier to use than DB Master. HyperStudio simply can't touch that. And I didn't have to log on to America Online and send anyone an e-mail to figure out how to do it. Everything I needed was in the manuals. Those *big, complete* manuals.

Actually, now that I think about it, I didn't really even need the manuals that much. I was able to find most of what I needed in the online help. Hmmm, I can't seem to find any help online in HyperStudio.

In summary Dave, I agree with you 100% when you say that HyperStudio is the best *hypermedia* program for the average IIGS user. I hope you can agree with me that Hyper Studio and HyperCard IIGS are as different as night and day. HyperStudio is a great way to display information. I also admit that HyperCard IIGS is *not* for your average IIGS user. However, if you need to do more with your information than just *look* at it, HyperCard IIGS is simply lightyears ahead. GS+

What Do *You* Think?

Now that the "experts" have had their say, we want to know what you think of HyperCard IIGS and HyperStudio. We also want to know what you are doing with these two programs. Your comments will be published in our new "HyperActivities" department starting next issue.

use the program well, so if you can contact such a person through a user group or online service, they can probably answer your questions about how to use HyperStudio. Roger Wagner Publishing also has a customer support line and maintains an active presence on most on-line networks. HyperCard IIGS, on the other hand, is so new that the only people who are pros at using it at this point are Apple employees. This means that for help using the program, you need to contact Apple, either through a user group connection or by a phone call. You might not get to speak to the president of the company, as you sometimes do with Roger Wagner Publishing, but you can probably talk to someone who can answer your questions.

And The Winner Is . . .

And the winner is . . . no one! Neither of these two excellent programs is a clear leader in the hypermedia field. For some people, HyperStudio is the only choice because it will run without a hard drive and without huge amounts of memory. For many schools, this is an important concern. Beginners will probably prefer HyperStudio because it is somewhat more straightforward in its operation. There is no need to know what resources are or to learn a programming language—you simply draw your backgrounds, write your text, make your buttons, and away you go. HyperCard IIGS is more suitable for a "power user" with a lot of memory, a large hard drive, and some interest in programming. Although programming skills aren't necessary to use HyperCard IIGS, they can be a great help.

I feel that HyperCard IIGS stacks look a little better than HyperStudio. Perhaps this is because most of the stacks I've seen for HyperCard IIGS are done by Apple employees and professional artists, whereas most HyperStudio stacks are done by average people in their spare time. However, my stacks seem to look a bit better with HyperCard IIGS, due, I think, to the abundance of features and the availability of a programming language. By using HyperTalk, a HyperCard IIGS stack can open windows and menus, conducting a dialog with the user that is a level of sophistication higher than that attainable with HyperStudio. Both programs, however, are excellent programs. If you're new to hypermedia and don't have a fully loaded IIGS, HyperStudio is probably the program for you. It's small, fast, uncomplicated, and easy to use. If you are a hypermedia addict, or a IIGS junkie, then HyperCard IIGS is probably the way to go—the plethora of features allows you to really flex your creative muscles. Luckily, this is one of those decisions for which there is no wrong answer—no matter which program you decide to purchase, you'll have bought a powerful hypermedia tool that has the ability to drastically increase the usefulness of your IIGS. GS+

THE FINE PRINT IN OTHER ADS:

"Not responsible for product compatibility"
"Add 3% (minimum $4.00) for each shipment"
"Personal and company checks allow 3 weeks to clear"
"20% restocking charge on returned items"
"All sales final."

THE FINE PRINT IN OUR ADS:

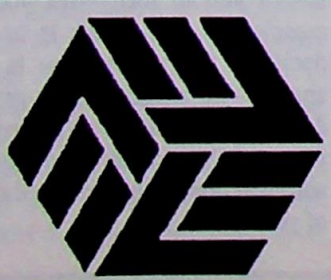
"We Pamper Apple Computers"
"Trade in credit for old hardware"
"85% of our business is the Apple II line"
"Hardware - Software - Repairs - Training"
"We use what we sell"
"Free shipping"

Some of the company we keep:

American Micro Research ✦ Applied Engineering ✦ Appoint ✦ Apricorn ✦ Beagle Bros ✦ Broderbund ✦ Brother International ✦ CH Products ✦ Data Spec ✦ Electronic Arts ✦ Learning Company ✦ Logitech ✦ Olympia ✦ Orange Micro ✦ Qtronix Corporation ✦ Roger Wagner ✦ Seikosha ✦ SoftSpoken ✦ Supra Corp ✦ TimeWorks ✦ Vitesse ✦ Westcode Software ✦ Wordperfect Corp ✦ Zip Technology ✦ Zoom ✦

Some of the items we carry:

cables ⊄ chips ⊄ computers ⊄ cooling fans ⊄ data switches ⊄ disks ⊄ dot matrix printers ⊄ drive boards ⊄ fax & modem switches ⊄ 5.25 drives ⊄ hard drives ⊄ interface cards ⊄ joysticks ⊄ keyboards ⊄ laser printers ⊄ mice ⊄ modems ⊄ monitors ⊄ mouse pads ⊄ network connectors ⊄ power supplies ⊄ printer cards⊄ ram boards ⊄ scsi cards ⊄ software ⊄ surge protectors ⊄ 3.5 drives ⊄ track balls ⊄



LEARNING EXPERIENCES
New Milford, CT 06776-1537
Phone: (203) 354-3669
Fax: (203) 355-1900
America Online: PAX5

The HyperStuff Collection

Programmed by Michael Nuzzi
Music by Gregory Dib
Art by Mike Monastero

Retail price: $39.95 each
Typical mail-order price: $30
Not copy-protected
Requires 1.5 MB of RAM and HyperCard IIGS or HyperStudio

Reviewed by Jonah Stich

Only minutes after HyperCard IIGS was released, Triad Venture, Inc. released the first two volumes of its HyperStuff Collection—a series of HyperCard IIGS and HyperStudio add-ons. These volumes are ClipTunes—which uses Apple's new MIDISynth tool to play songs, and ClipArt Plus—a package with four HyperCard IIGS XCMDs (short for eXternal CoMmanDs), four NDAs, an application, and 20 screens of 640-mode clip-art.

These two packages are designed to work with both HyperCard IIGS and HyperStudio, but they work a bit better with HyperCard IIGS. This is due mostly to HyperCard IIGS's built in scripting language (HyperTalk) and its better interface with external commands. The four XCMDs that come with the ClipArt Plus package work only with HyperCard IIGS, and the song XCMD that comes with ClipTunes works much better with it. There is a song XCMD for HyperStudio, and although it works fine (you must first patch the HyperStudio program file, but there's a program included that does this for you), it's not as well integrated into the HyperStudio environment as the HyperCard IIGS XCMD is. (Keep in mind that, although HyperCard IIGS and HyperStudio perform many of the same functions, they are completely separate programs, and cannot share data files such as stacks or XCMDs. Therefore, a separate XCMD is needed for each program.)

Both products come with manuals of about 15 pages, which provide enough information to get you started without going to great depths. Both packages come with a license that grants the original purchaser use of the Songs, Artwork, and XCMDs in their public domain or shareware stacks as long as the Triad parts of the stack are not resold for their intrinsic value. Furthermore, HyperStudio users must imbed the XCMDs in their stacks using the Master XCMD package available from Roger Wagner, Inc., and HyperCard IIGS stacks must be "Protected" with a user level of Authoring or lower and have a protection password set. Stacks also cannot have a method of copying the XCMDs to other stacks, nor may they provide information on the syntax or parameters used to invoke the XCMDs. There is no information provided on the use of the HyperStuff products in commercial stacks—you'll have to contact Triad Venture and discuss it with them.

ClipTunes

ClipTunes comes with XCMDs for both HyperCard IIGS and HyperStudio, a copy of Apple's new MIDISynth tool, and 30 songs to play with it. The XCMDs can, of course, also play any other songs created for the MIDISynth tool, meaning that if you have a MIDI card and Apple's synthLAB program, you can create your own songs to play. These songs are really *songs*, not just digitized sound files. This means that they can be quite long without using hundreds of kilobytes of RAM.

The HyperCard IIGS XCMD is installed in your stack by the included installation stack. The only mildly tricky thing here is that you have to install some of the XCMDs included with HyperCard IIGS into the installation stack to allow it, in turn, to install the Triad XCMDs into your stack. This isn't at all hard, but you have to read the manual first to know that you have to do it.

Once the XCMDs are installed, you can call them from a script using "Synth <parameter>". If there is no parameter, you are presented with a Standard File dialog box and can select the file you would like to play. If the parameter is a file name, the file is played (or an error message is returned in the message box if the song can't be played). Finally, the Stop and Check parameters are used to stop the currently playing song and to release memory used by the XCMD if the song has stopped playing. Also, a global variable "Music" is made available to scripts. It is equal to empty when no song is playing and to the string "Playing" when music is being produced. This combination of functions and variables makes it extremely easy to write a script that uses the Synth XCMD (and for those lazy people who would just like to listen to the songs without having to create a stack, it can also be invoked from the message box). There are examples of how to use all of the different modes of the XCMD in the manual, and it's easy enough that even a new "HyperTalker" can use it.

The HyperStudio XCMD is not as well integrated into HyperStudio, but this is mainly the fault of HyperStudio's lack of a scripting language. Before using the XCMD you must patch your copy of HyperStudio (the patch also works with the run-time version) and copy the XCMD into the same folder as the stack (or imbed it in the stack with the Master XCMD package from Roger Wagner). Once this is done, the XCMD is invoked by clicking the "Trigger XCMD" check box when defining a button and then entering a string to pass to the XCMD. This string can be one of three things: the pathname of a song to play, the word "Stop" to stop the song, or the word "Choose" to open a dialog allowing you to select the file to be played. A slight flaw in the design of the XCMD here is that, when a song name is passed to the XCMD, the song, which is made up of three files—a sequence file, a wavebank file, and an instrument file—must be in the same folder as the stack. For people with hard drives who like to keep their music files in one folder and their stacks in another, well, they're out of luck. Also annoying is that no message is reported when a song can't be played. Granted, HyperStudio has no message box, but it would be nice if the XCMD opened an alert box telling the user that an error has occurred.

ClipArt Plus

ClipArt Plus is a package of clip art, plus some other neat utilities to make your "HyperLife" a bit easier. The clip art is well drawn and covers many topics: there

are over two hundred images, split into 20 files. It is all 640-mode clip art, which means that it will look good in both HyperCard IIGS and HyperStudio and doesn't require any conversion or color fiddling—just cut out the picture you want and paste it right into your stack.

Included in the package are four New Desk Accessories. *ClipIt* allows you to "clip" a piece of the currently displayed SHR screen onto the clipboard (using cross-hairs to draw a rectangle around the part you'd like). This image can then be used in any program that supports the standard IIGS clipboard. One problem with this NDA is that it opens an invisible window in front of all other windows, causing the front window's title bar to become unhilighted. For most people this will not matter, but if you're creating a stack that shows people how to use your program, it can be a bit strange to have all of the windows in the example pictures unhilighted. *PrintShr*, another NDA in the package, prints whatever is on the SHR screen to the printer—it'll even print the cursor if you don't hide it in the upper right hand corner of the screen. *Icon Buttons* is an NDA much like the ClipIt NDA. This one allows you to select a portion of the screen and then save it as a HyperCard IIGS icon in the stack of your choice. Here again the NDA opens a window in front of all others, causing the front most window to become deselected. Also, after you have clipped the portion of the screen, the NDA opens a window asking you to enter a name for the icon. When this happens, the cursor completely disappears and you have to remember that you need to type return at the end of the name to make the window go away. *Instant Icon*, the last of the four NDAs, is quite similar to Icon Buttons, but it saves its icons as Finder-type icons (those that you put in the **Icons** folder of your disks). Here again the deselected window problem is present. After you select a section of the screen to use as an icon, a window opens (the cursor doesn't disappear this time) allowing you to set the file type and auxiliary type, filename, and program filename that this icon will be used for. Also provided is a short list of filetype names for you to select from.

ClipArt Plus also comes with an application, Convert Icon, that allows you to convert Finder-type icons into HyperCard IIGS icons and attach them to HyperCard IIGS stacks. This is a little, no frills program that does what it is supposed to, but not a lot more.

Finally, ClipArt Plus comes with four HyperCard IIGS XCMDs. Unlike all of the other parts of the package, these XCMDs work only with HyperCard IIGS—there is no HyperStudio version. When *VClip* is invoked from a stack, it opens a window with the contents of the clipboard in it for you to look at. You can print the contents by pressing "P" while this window is open. Clicking the mouse will return you to your stack. *MoveIt* is an XCMD that copies icons from the current HyperCard IIGS stack into other stacks. When it is invoked, it opens a window with a list of icon names, a Move button, a Quit button, and an area for display of the currently selected icon. You select an icon name from the list on the left, make sure it's the right one by looking at the picture area on the right, then click the Move button to select a stack to copy the icon into. Note that the icon is only copied—it is not removed from the original stack, even though the button is labeled Move. When you're finished copying icons, you click the Quit button to return to your stack. The *GetPrefix* XCMD allows the user to select a stack and the complete pathname of the stack is returned in the "The Result" variable. (Note that the entire pathname, not just the prefix, is returned. This XCMD would probably be better named GetStackPath XCMD.) The final XCMD is the *GetPic* XCMD. This command allows the user to select a graphic file (either a standard SHR picture or a PrintShop GS picture) and then displays it on the screen. The user is then presented with a cross-hair cursor, and can select a portion of the picture by dragging a rectangle around it. When the selection has been made, the relevant piece of the picture is copied to the clipboard.

The Good, The Bad, and The Messy

The ClipTunes collection is an excellent package for users of either HyperCard IIGS or HyperStudio, allowing them to play long musical pieces from within a stack. This is a utility that can be found nowhere else and is extremely useful. Hopefully Triad will release more songs for the MIDISynth tool, as well as a SoundSmith playing XCMD for both HyperCard IIGS and HyperStudio in the near future. The ability to easily include complex musical scores in stacks really sets the IIGS apart from other computers.

The ClipArt Plus collection is slightly less wonderful. All of the utilities have somewhat sloppy interfaces—they're not unusable by any means, but they don't quite look like programs that you would expect to pay $30 for; I got the feeling that, they had been quickly put together, which is not surprising considering the extremely short time between the announcement of HyperCard IIGS and the release of these programs. Several of the commands are quite useful, especially those that manipulate HyperCard IIGS icons, as there isn't currently any other easy way to do this. Some of them, however, such as the GetPrefix XCMD, don't really seem to have a home. Still others seem as if they would be better if combined into one larger program. Topping all of this off is the fact that the clip art, while of high quality and extremely useful for fools like me who can't draw a straight line with a ruler, is not significantly better than that which can be obtained through user groups or on national online services such as America Online and GEnie. It is not a bad product—for those who need to play with icons in HyperCard IIGS, or those that are always looking for that perfect piece of clip art, this collection is quite useful. It's not, however, as novel and full of usefulness as the ClipTunes package.

GS+

McGee at the Fun Fair

Programming by Frank Andrews
James McCarthy

Retail price: $39.95
Typical mail-order price: $25
Not copy protected
1 MB Required

Lawrence Productions Inc.
1800 South 35th Street
Galesburg, MI 49053-9687
(800)-421-4157

Reviewed by Greg Zimmerman

McGee at the Fun Fair (Fun Fair) is the third Apple IIGS software offering from Lawrence Productions aimed specifically at preschool children.

The program follows the same successful "NO WORDS" menu selection concept as the previous McGee offerings, and is easily navigable by a two-year-old child using only the mouse.

Fun Fair takes the child along on a trip (to guess where) with McGee and his friend Tony. Each screen consists of a large colorful screen graphic consuming about three quarters of the screen area, and four small graphics at the bottom of each screen. The graphics at the bottom are miniature pieces of the larger screen graphic, or of other screen graphics in the program. The cursor is "locked" into this lower screen area. The child positions the cursor over one of the four graphics at the bottom of the screen, and clicks the mouse to select that graphic. Depending on the selection, the program then animates the existing screen graphic in some way, or moves to a new screen graphic with four new action choices for the child.

All of the reactions that the child gets with the click of the mouse are high quality animations of the screen graphics, many of which are accompanied by sound. In each of the four selections per screen is at least one which is linked to another graphic screen and another set of choices for the child to explore.

Activities the child sees and hears include a sketch artist, a music man (Remember Dick Van Dyke in Mary Poppins?), a clown, playground activities, kite flying, a drink at the drinking fountain, and much more.

Details

Fun Fair is a faithful addition to the McGee series. It allows, through the point and click at the pictures concept, extremely young children to use the Apple IIGS. It is so simple that a person that has never touched a computer before could be navigating through the program in less than five seconds.

Fun Fair comes on two 3.5-inch disks, is System Software v5.0.4 compatible, will work on either a ROM 01 or ROM 03 IIGS, is hard drive installable, and is not copy-protected.

The manual is short and well-written, but completely unnecessary, as is the keyboard to your computer. All the child needs to use is the mouse. If anything, the keyboard will just get in the way.

Lawrence Productions has a toll-free number for customer support. I have called this number many times to discuss both Fun Fair and its forerunners (McGee and McGee visits Katie's Farm). The people who answer are friendly, knowledgeable, and helpful.

Is It Good?

Fun Fair is really good. The graphics are high quality, the animations are high quality, the sound is high quality, and the concept is high quality.

This is software that really lets a two-year-old child operate the Apple IIGS without constant parental involvement. One important aspect to the ease of use is that the cursor is locked into the lower screen area. It will not move above the four possible selections. As a result, wherever the cursor is at the time the mouse is clicked, a reaction will occur. And the reaction is not always the same when selecting the same menu choice for a second time. These two programming aspects really help keep the child motivated. This helps the child build self confidence, and gives the child a chance to be in control on his (her) own. At the young age for which this program is intended, these are two very important developmental areas that young children are constantly fighting for. They want a chance to make decisions and to do things on their own, and this software lets them do just that while at the same time gaining a noncompetitive sense of accomplishment.

My three-year-old son (the beast) sat for over 30 minutes exploring Fun Fair the first time he tried the program. And then he came back to it a couple of hours later for another session.

One more thing I really like about Fun Fair is that it is not copy protected. This is an important consideration when investing in software designed for use by young children. It is a lot less worrisome while the child is at the computer knowing that if he tries to eat the program disk, that it is only a backup, and not a $25 original.

Any Bad Points?

Fun Fair is a slow loader. On a stock ROM 01 IIGS, booting from the floppies, using two drives, it takes one minute just to get to the "Hi, I'm McGee" screen, and another twenty seconds before the child can make a selection.

Of course this time is considerably shortened when launching the program from a 100 MB Toshiba hard drive attached to an Apple DMA SCSI card under System Software v5.0.4 on a ROM 01 IIGS equipped with a 7 MHz TransWarp. I happen to have a IIGS equipped just like that (what a coincidence), and on my machine, the loading time was only 23 seconds.

Should I Buy It?

I highly recommend Fun Fair for any family that has children under the age of four. It is one of very few pieces of software on the market today that allows the youngest of children to use the Apple IIGS.

McGee at the Fun Fair is enjoyable, it's easy to use, it helps the child get comfortable with the computer at a very early age, it builds self-confidence in the child, and gives the child some pride in using the same computer that mom and dad use. It's simple, but it is also effective and attractive. GS+

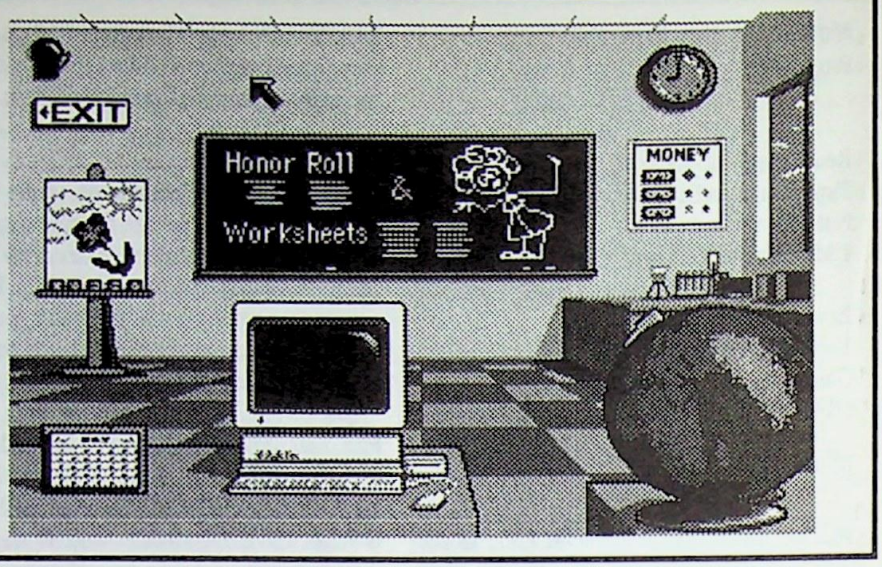
Talking Classroom

Program by Richard E. Dye

Retail price: $49 ($59 with backup disk)
Typical mail-order price: $38
Not copy protected
1 MB of RAM required

Orange Cherry Software
Box 390 Westchester Ave.
Pound Ridge, New York 10576-0390
(800) 672-6002
(914) 764-4104

Reviewed by Greg Zimmerman



Talking Classroom is another in the long line of educational programs made specifically for the Apple IIGS by Orange Cherry Software. It is part of the Talking Schoolhouse Series which consists of approximately 20 programs currently available, and which is targeted at the school and home education markets.

Talking Classroom is intended by Orange Cherry for children ages 4 to 7 years old, an age group that appears appropriate for the activities contained in the program.

Talking Classroom consists of two 3.5-inch disks, which contain a total of eight activities. The program is System Software v5.0.4 compatible, it is not copy protected, it is hard drive installable, and it will run on either á ROM 01 or ROM 03 Apple IIGS.

Although the manual says differently, Orange Cherry informed me over the phone that the disks are covered by their lifetime replacement policy. Should anything happen to a disk, just return the non-working disk, and it will be replaced. Obviously, this is a great selling point for software that is intended for use by young children.

The manual itself is short, concise, and well-written. In addition to explaining the features of the software, it also contains hard drive installation instructions as well as suggestions for activities that will enhance the educational process when performed in conjunction with the use of the software.

What It Does

Talking Classroom greets the user with a nice high-quality graphic screen containing a picture of part of a typical school classroom. There are no instructions given or otherwise indicated by the program, but it doesn't take long to figure out. In this fast moving world of hypermedia, it's reasonable to assume that clicking with the mouse on one of the objects in the classroom may get some reaction.

There are maybe sixteen or so choices of things to click on, but the classroom window, the wall, floor, ceiling, desk, and table will not elicit any response. As one may deduce from perusing the accompanying picture of the Talking Classroom startup screen, this leaves the calendar, the computer, the globe, the easel, the chalkboard, the science area, the clock, the money poster, the exit sign, and the bell. Clicking on any of these ten pictures will begin an activity or elicit a reaction that is associated with the selected picture.

For example, clicking on the computer brings up a screen of a keyboard entitled Talking Computer. The 26 letter keys are all labeled (the others are blank), and a large hand style cursor appears. The user is to click on one of the letters shown, which will elicit some animation as the key containing the selected letter appears to be pushed down, and the computer says in a human voice the name of the selected letter.

Selecting the easel brings up a miniature paint program wherein the user may select from among five black and white line drawings that may then be colored in using the paint tools, and then printed. The paint options are limited (13 colors instead of 16 colors, undo, clear, several paint brush sizes), and the drawing cannot be saved.

Selecting the globe brings up a split screen, with the top containing a miniature of the globe with some adjacent space for text, and the bottom containing an outline map of the world with a sailboat (kind of) cursor. If the user clicks on one of the continents on the map, the miniature globe spins until it highlights the selected continent, at which time a short paragraph of information about that continent appears in the text area.

Selecting the chalkboard allows the user to print out one of four pre-made worksheets relating to subject matter covered in the program, such as a calendar for a selected month, or a quiz to identify the correct time shown on 10 clocks, as well as an award certificate.

The clock brings up a pretty neat screen for kids to improve their clock reading ability (both digital and regular), and the science area brings up a screen from which three high quality and somewhat instructive animations can be seen. The money poster yields a short instruction on the identity and relative value of four coins and two bills, and the calendar attempts to teach though voice and graphics a recognition of the days of the week.

Of the two activities for which I did not give any details, the bell merely brings up

a screen with which the user can adjust the sound volume, and the exit sign allows the user to depart the program.

The graphics range from good to outstanding, and the voice and sounds are excellent. If the user is at all familiar with hypermedia selection techniques, it is easy to get around the program and figure out what is to be done.

The Good Points

Talking Classroom has good graphics, some good use of sound, and point-and-click ease of use. Several of the activities are high quality, particularly the clock and science modules, though there is a lack of depth even in the these two areas.

Also, the opening screen will engender some interest in first-time users as they click around the classroom without instruction and explore on their own what the program will do.

The Bad Points

Before casting the stones here, I tried this software out on three of my own kids ("Let Mikey try it, he'll eat it.").

The eight-year-old played with the painting module for five minutes and got frustrated with the lack of features she is used to using in other paint programs. Not being able to save the work was especially annoying, but missing features such as flood fill, spray paint, zoom, and patterns, as well as the lack of ability to draw from scratch without one of the five pre-made line drawings, turned her off quickly. The six-year-old went through the modules, checked them all out, and ejected the program in about ten minutes. The four-year-old played with the drawing program for 15 minutes or so, but never finished a drawing completely without interruption, and of course, this program has no save feature for the drawings.

I wanted to like the program, but after seeing the kid's fleeting use, and talking to them about the software, I concluded that the shallowness of the educational material brought on boredom relatively quickly.

The program also loads pretty slowly. On a stock ROM 01 IIGS equipped with 1.25 MB of memory and two 3.5-inch drives, booting from the floppy took one minute and forty five seconds. It took over 1 minute just to get to the screen that says "please wait." To be fair, those users booting from a 40 MB InnerDrive, with the Inner*Express* chip installed, on a ROM 01 IIGS equipped with a 7 MHz TransWarp, under System Software v5.0.4 will find that the program loads in 20 seconds.

The Wrap-Up

This program, as with several others of the Talking Schoolhouse Series, left me with some mixed feelings. The program content is varied and overall shallow, however there is some attractive material, and some learning concepts that could be expanded upon and made really useful. But in its present state, it is more a mixed bag of entertainment than it is education, and it won't entertain the kids for too long. I don't believe it is worth the roughly $40 investment. **GS+**

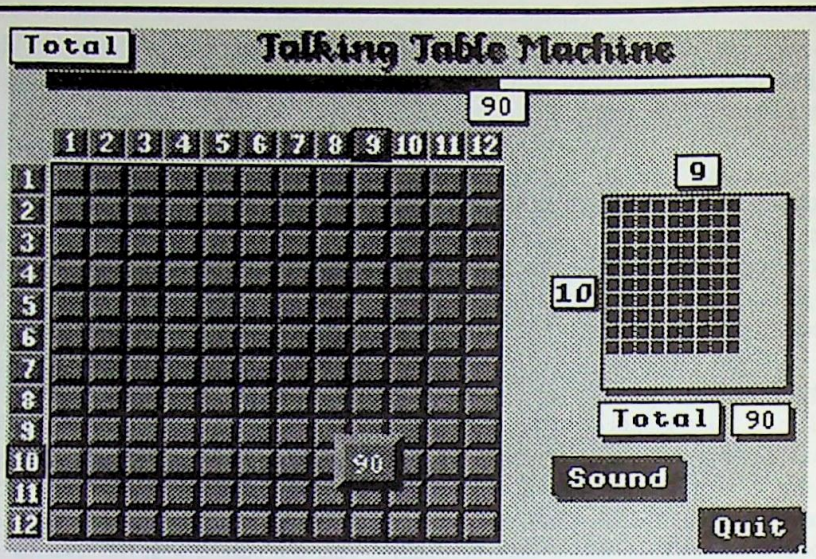
Talking Multiplication And Division

Program by Richard E. Dye

Retail price: $49 ($59 with backup disk)
Typical mail-order price: $38
Not copy protected
Requires 1 MB of RAM

Orange Cherry Software
Box 390 Westchester Ave.
Pound Ridge, New York 10576-0390
(800) 672-6002
(914) 764-4104

Reviewed by Greg Zimmerman



Introduction

Talking Multiplication and Division is one of the Talking Schoolhouse Series of Apple IIGS software published by Orange Cherry. It is intended to teach multiplication and division to a recommended age group of ages eight to eleven.

Details

Talking Multiplication and Division comes on two 3.5-inch disks that are not copy protected. The program is hard drive installable and will run on either a ROM 01 or ROM 03 Apple IIGS. Talking Multiplication and Division is compatible with System Software v5.0.4, and is accompanied by a well-written manual which both explains the program's features, and offers suggestions for other activities related to the program material.

The program disks are covered by Orange Cherry's lifetime replacement policy. Though the manual says different, the people at Orange Cherry say they will replace any disk that does not work if the user sends it back to the company.

Orange Cherry has supported the Apple IIGS market with more than 20 software offerings. They have a toll-free number for users to call for tech support and questions, and the people answering the phone are well-informed, polite, and helpful.

What It Does

Talking Multiplication and Division is a five-part program which teaches the basics of multiplication and division of the numbers 1 through 12.

The user is greeted with a main menu screen listing the five selections as well as the option to quit the program.

Part One is the Talking Table Machine. In this module, the user is presented with a large graphic square which has the numbers 1 through 12 across the top, and down the left hand side. By clicking anywhere in the large square at the intersection of two of the numbers, those two numbers are multiplied by the program and the answer is exhibited at the point that was clicked by the user. The problem and the answer are also recited by a human voice. Two other graphic representations of the particular problem are also displayed in a good attempt to give the user a better understanding of the relative quantities involved in the particular multiplication problem.

Part Two is Facts and Factors. In this part, 10 jigsaw puzzle pieces are displayed, each containing a number against a white background. A multiplication problem is displayed at the bottom of the screen and at the same time is recited by a human voice. The user is to click on the puzzle piece that contains the correct answer to the problem. All the problems are missing factor problems such as "2 x ? = 4". As each correct answer is given, the white background of the puzzle piece transforms into a colored graphic that is part of the puzzle picture. When all ten problems are correctly solved, the entire puzzle graphic is complete, and a short animation sequence is displayed.

Part Three of Talking Multiplication and Division is the Dividing Machine. In this segment a division problem is presented as part of an attractive animated screen graphic (the Machine). The problem is also spoken by the human voice. The user inputs the correct answer (with unlimited incorrect tries permitted) by clicking on the appropriate number on the Machine, the voice informs the user that the answer is correct, and the module moves on to the next question. After 10 problems are solved, a high-quality animated reward graphic is displayed, and the program returns to the main menu screen.

Part Four is Solving Word Problems, a module which presents both division and multiplication problems in word format. Each problem is accompanied by high-quality graphics. Prior to solving a problem, the user must first indicate whether division or multiplication is required, after which the answer is input by clicking on an on-screen calculator. After two unsuccessful tries at the answer, the correct answer is given, and a new problem appears. The module records both the number and percentage of correct answers.

Part Five, Banker's Run, is a maze (an underground vault) through which the user must move a would-be bank president by

clicking directional arrows on the screen. At many points in the maze, multiplication and division problems must be answered correctly to continue the journey. Those reaching the end of the maze find themselves in an animated graphic of the bank's money room. The user can choose from among several levels of difficulty, and as with the rest of the program, the quality of the graphics is very good.

All the modules let the user adjust the sound level on screen, and each module contains a clearly marked way to quit the current activity and return to the main menu.

Is It Any Good?

Yes, it's pretty good. The program operation is smooth and easy to follow. The program delivers the educational content it promises, and it does so in a way that makes it somewhat interesting.

The recommended age group of eight to eleven may be somewhat high, as my six-year-old (brag brag) had no problem using the program.

Bad Points?

It takes one minute and 15 seconds to load on a stock ROM 01 IIGS with two 3.5-inch drives attached. However, with the 40 MB InnerDrive/*InnerExpress*/7 MHz TransWarp/System Software v5.0.4 configuration, loading time is only 17 seconds.

Also, while the program does have five different modules, it lacks the depth of other programs, such as Math Blaster Plus GS (see review in *GS+* V1.N6), that are already available for the Apple IIGS market that have a similar purpose.

For example, there is no provision to change the problems (and therefore the difficulty level), no overall score-keeping, no fractions or addition or subtraction modules (well, if there were I guess they wouldn't call it Talking *Multiplication and Division*), and no way to print out any of the subject matter. All of these features can already be found in a product such as Math Blaster Plus GS that is already available for the Apple IIGS market. So while Talking Multiplication and Division does have some very good learning modules, it has a limited purpose, limited features, and it is neither expandable nor adjustable.

Should I Buy It?

Talking Multiplication and Division does a credible job of making the unavoidable drill and practice of learning multiplication and division a little easier to swallow and a little more interesting than just a list of problems on a math worksheet. It does not have as many features as I would like to see in a program with this price tag, but is worthy of consideration for parents with children ages six to eleven that need help in the specific subject matter covered by the software. If I were in the aforementioned circumstance, the purchase of this program would be a really borderline call. But because it was for my kids, it would probably be a decision to buy.

What's contained in the software is good, intuitive, and easy to follow. I just wish there was more. GS+

Bouncing Bluster II

Programmed by Jean Francois Doue and Jean Michel Valaat

Retail price: $59.95
Not copy protected
Requires System Software v5.0.2 or later

Toolbox
6, Rue Henri Barbusse
9500 Argenteuil, France
FAX: 0 11 33 1 39 47 44 08

Reviewed by Dave Adams

Just When You Thought It Was Safe To Go Back To The Paddle . . .

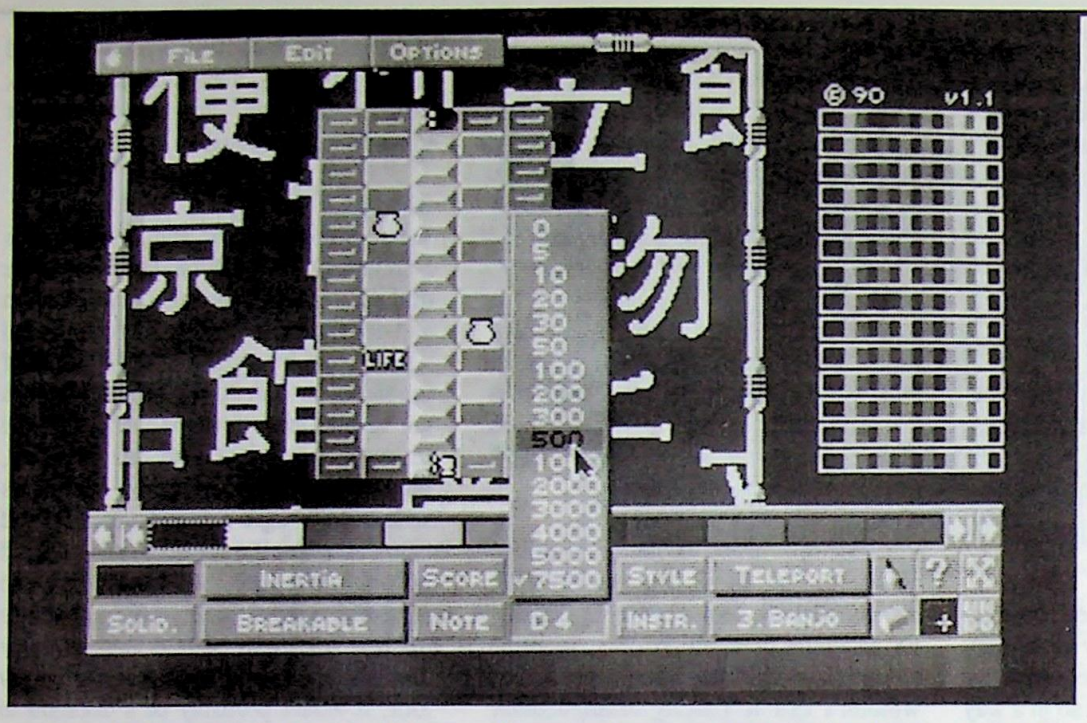
Bouncing Bluster II is the latest evolution in the "Breakout" series of games. You know the type—you have a paddle at the bottom of the screen that slides to the left and right. You use that paddle to bounce a ball up into a row of colored squares/bricks and when the ball hits one, the brick disappears. New twists on the game involved catching capsules that give your paddle unique effects such as catching the ball and holding it or arming you with lasers. Ho hum, here we go again . . .

Bouncing Bluster II is a sequel to Bouncing Bluster (a shareware game) and a cousin to Arkanoid. When I first got this game, I had to wonder, "Just how much can you improve on this type of game?" I am happy to report that Bouncing Bluster II has made some nifty improvements and put some more luster on Bouncing Bluster. (OK, OK, no more bad puns . . .)

Bouncing Bluster comes on two 3.5-inch disks and is *not* copy protected. The game also comes with a user's manual (in English) that clearly and succinctly explains every aspect of the game. You must use some sort of program launcher to run Bouncing Bluster II—the disks are not self-starting. The game easily installs on a hard drive and rarely crashes—unlike the shareware version. In fact it has *never* crashed on me. Bouncing Bluster II comes complete with a construction set that lets you edit every feature of the game. This is a godsend for uncoordinated persons like myself that have poor hand and eye motor skills.

The More Things Change . . .

. . . the more they stay the same. Bouncing Bluster II has the same look and feel of the original and uses many of the boards from Bouncing Bluster. While it doesn't have the garish and bright colors of Arkanoid, it does offer more intriguing and subtle backgrounds. Some screens are simply wonderful to look at. Many of the screens are designed to make following the ball more difficult. While some screens are familiar, there are plenty of new boards to keep you from getting jaded. However, the opening has changed a great deal. The introduction screen and SoundSmith soundtrack give it an entirely different atmosphere. SoundSmith instruments are also incorporated into the game. Each time you strike a block, it sounds off with one of 10 different instruments (from a banjo to a barking dog) and chimes a note from a scale of about 5 octaves. With some creative planning, you can create some interesting musical effects on your



boards. In short, Bouncing Bluster II has great graphics and great sounds. It is what a IIGS game should be—a feast for the eyes and ears.

If you've never played the original Bouncing Bluster and you've gotten tired of Arkanoid, then this game should be a real treat. Even if you have the original Bouncing Bluster, the newest version makes it look pale in comparison. If you are looking for a challenging arcade game, then look no more. Bouncing Bluster II is by far the most challenging game of this type. For many people, it may be a bit *too* challenging. I myself had to edit a few boards so I could continue playing the game without ripping my hair out. I haven't felt this way since I played Arkanoid and the original Bouncing Bluster. Bouncing Bluster II has far more features than either of those fine games. To best discuss this, we should refer to the Construction set.

I've Been Working On The First Board, All The Live Long Day . . .

If you've ever gotten so frustrated playing an arcade game that you've wanted to throw your IIGS out the window, then you will soon learn the ins and outs of the Bouncing Bluster Construction set. Mastering this program will allow you to keep your temper under control and save on home repair bills. You can edit everything to your satisfaction. Having trouble getting through the first board? Load it up with extra lives so you can play with almost unlimited chances. Stuck on that one level? Fix it so that it clears more easily. Some people might call this cheating. I prefer to think of it as maintaining one's sanity. The Construction set is simple to use. You can change everything from the color and type of the bricks to the sound that they make when they are struck. You can load in backgrounds from any decent paint program to assist or hinder the player. You can change the order of the boards by altering the script (the game's method of displaying the boards in order). You have total control over the game.

Bricks can be of many types. They can break with anywhere from 1 to 14 hits or be unbreakable. They can reappear after a short delay. They can be one of many different colors or "invisible." They can have point values ranging from 0 to 7500.

There are even transporter bricks that will randomly teleport the ball to another transporter brick across the screen. Unlike Arkanoid, you can have any type of brick contain any type of capsule that you wish. Add in some sound effects, choose the type of monster to ruin the player's day, and you're finished. If you get tired of playing the same boards over and over, edit them to make them more or less difficult. The Construction set is your key to happiness.

The capsules range from the ever popular extra lives to the dreaded time bomb. Some of the capsule types and their effects are:

3 or 5 balls - Pretty self-explanatory
Ghost - Five balls appear, only one of which is real (formerly called Stealth)
Iron ball - Goes through anything
Leap Ball - Jumps over obstacles (and your paddle if you aren't careful)
Inertia - Amplifies paddle movements and is terribly difficult to control
Magnet - Allows you to hold onto the ball for a short time
Bomb - Gives you five seconds to grab another capsule or your paddle explodes
Small - Reduces paddle size
Large - Increases paddle size
X-Y axis - Allows you to move your paddle up and down the screen
Mystery - Randomly selects an effect
Extra Lives - Everybody's favorite
X- Reversal - Your paddle travels in the opposite direction of the way you move it
Slow - Slows down the ball
Bonus - Adds extra points to your score
Exit - Lets you to skip to the next level
Laser - Allows you to blast bricks with your paddle
Fans - Blows the ball to the opposite side of the screen
Random - Randomly selects an effect

Another innovation in the game is the ability to commit "Paddle suicide" if the ball gets stuck in a spot on the board. No more rebooting or eternal waiting for the ball to free itself. One of the best things about the game is that they got rid of the annoying synthesized speech! No more of that "Get ready player one" drone to distract you.

But What Of The Bad Stuff?

There really isn't that much to complain about with this game. The only feature that annoys me is that you don't have the total control over the path of the ball that you get when you play Arkanoid. In Bouncing Bluster II, the ball tends to drift a bit when it travels in a straight line over a distance. In Arkanoid, the ball goes where you want it to. You can pretty much line your paddle up and put the ball where you want it to go. In Bouncing Bluster II, the ball seems to have a mind of its own. You simply cannot always determine whether the ball will get to the precise spot that you aim it towards. This can be pretty frustrating when you are after that one last brick in a tight spot.

The levels that come with the game range from simply easy to the four-letter-word-inciting difficult. There are some that are simply maddening. Fortunately, you can edit them. The game comes with two scripts. Personally, I felt that the "Easy.Script" was much harder than the "Hard.Script" that comes with the game. Of course, the editor let me combine them into my own "Easy to play and not too hard to finish" script.

The Bottom Line . . .

If you like this type of arcade game, then you are going to love Bouncing Bluster II. If you have never liked this type of game because it is too difficult, then you still ought to give it Bouncing Bluster II a try. The Construction set was made for people that like to "fiddle" with the game. You have to love the wonderful graphics, sounds, and flexibility that Bouncing Bluster II offers. It's a great game. **GS+**

Space Shark

By Olivier Phillip, Pascal Watel

Retail price: $59.00
Copy protected
Requires 768K RAM
Joystick recommended

ToolBox
6, Rue Henri Barbusse
95100 Argenteuil, France
FAX 0 11 33 1 39 47 44 08

Reviewed by Dave Adams

Space Shark is an arcade game that is similar to the old arcade game Defender. It is billed as the fastest arcade game ever produced for the IIGS. I, for one, will not dispute that claim. If you ever felt a "need for speed," then this game will definitely meet that need. Quite simply, *this game is fast*!

Space Shark comes on one 3.5-inch disk and is (unfortunately) copy protected. The game instructions are printed on two sheets of paper. Although this is a French product, the instructions are printed in English and are very succinct and complete. You can play with either the keyboard, joystick, or the mouse. Your best chances of survival lie in using a joystick. Like all good arcade games, it will run a demonstration if left unattended.

The Good Stuff

Space Shark is a game that makes you glad you bought a IIGS. The graphics are excellent and the SoundSmith sound track and the special effects sounds are incredible. The action is fast and furious, and I have caught myself actually wincing when my Space Shark was hit and destroyed.

The premise behind Space Shark is simple: blast aliens . . . lots of aliens . . . lots of really fast aliens . . . lots of really fast aliens that have nothing better to do than to ram your ship and blast you out of existence. There are two types of Space Shark pilots: the Quick and the Dead. I, myself, am intimately familiar with the dead . . . .

In Space Shark your official mission is to stop an alien invasion of the solar system. The aliens are from Iru Bom Maz, the only planet that has refused to sign the International Peace Treaty. (Yeah, I know that it should be Inter*stellar* Peace Treaty, but that's what the game says . . .) They are headed for our solar system and are mad as all get out. Your job is to stop them. There are 15 different waves (levels) attacking—each one deadlier then the preceding wave. To make things worse, there are only six Space Shark fighters left to stop them. Each Space Shark has two Mega bombs and lasers. The lasers fire to the front of the Space Shark and will destroy any craft they hit. The Mega bomb destroys every enemy vessel on the screen. You must clear each sector of aliens and then proceed to the next level. You can do this by killing all of the aliens in the sector (very deadly) or by running out of space time. Space time is indicated by a bar along the bottom of the screen. Killing aliens earns points; running out of time doesn't. This is important because you can get a new Space Shark every 20,000 points.

Each new level brings on a new and nastier type of alien. It is possible for you to begin playing at higher levels. You can begin on any level up to level twelve. Other than that there isn't that much to the game—except surviving.

The Bad Stuff

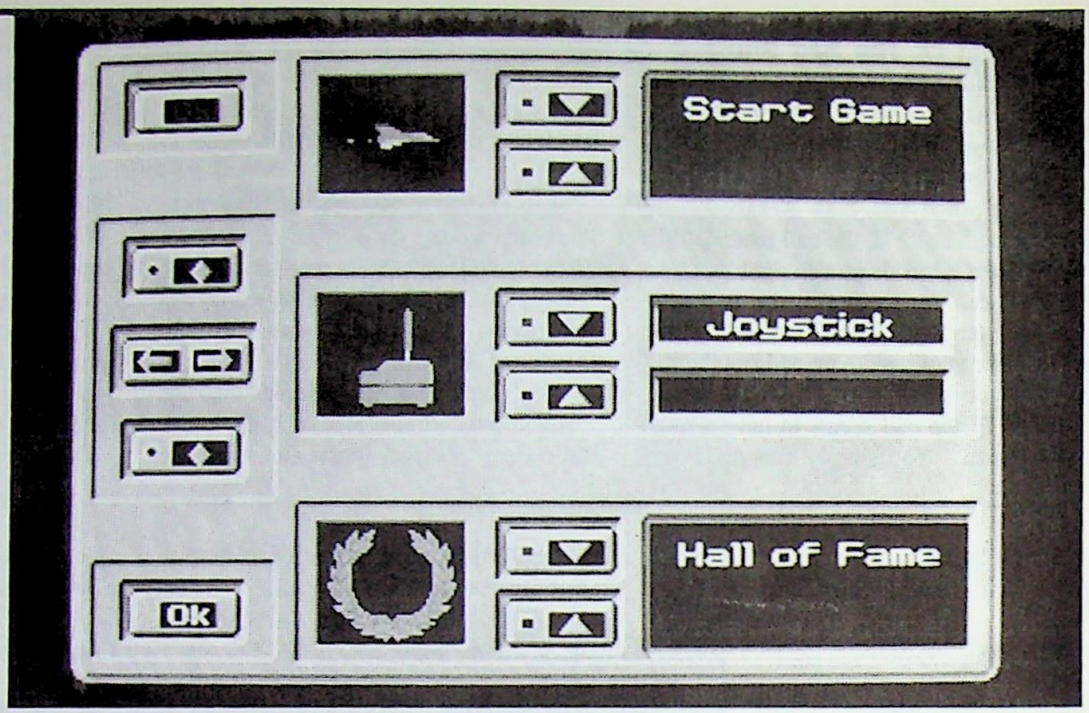
Overall there isn't that much to complain about. A lack of copy protection would be a vast improvement. As it now stands, the disk is not recognized by GS/OS and you cannot install it on a hard drive. When you are through playing the game, you must reboot your system. Using the standard Control-Command-Reset crashes the computer, forcing you to power down to restart the system. Space Shark is not exactly what I can call a GS/OS-friendly program.

The biggest overall problem in Space Shark is simply surviving. This game is not for those people that like easy wins and high scores. You will spend a lot of time getting frustrated before you get the hang of the game. It is quite difficult to win. I have not gotten past the 6th level. It is a fast and furious game and if you liked the arcade game "Defender" then you will probably like Space Shark. If you hated that game then you will probably hate Space Shark. If you like incredible speed in your games then try Space Shark. If you can't play regular arcade games very well you probably shouldn't buy this. Space Shark is for the dedicated arcade gamer looking for a good

challenge. It is not for the easily frustrated.

The Bottom Line

I think Space Shark is an excellent "Defender-style" game. However, the game is not for everybody. It carries a pretty steep price and there are very few mail-order houses that carry it. The best advice I can give is to go to the local arcade and see if they have the "Defender" game sitting in the corner. Plop a few quarters in it and see how you like it. If you don't like it, then don't buy Space Shark. If you are a Defender freak, then get your credit card ready and call somebody to order this game. The graphics and sound are superior to many games. The actual game play is fast and furious. Perhaps too fast for the average Joe. Of course you could always turn your kids loose on it. I let my students at school have a go at it and they enjoyed it immensely. For me (Mr. Poor Motor Skills and lame eye-hand-coordination), it got to be a frustrating experience. It is a very nicely done game—it's just not for everybody. **GS+**



TRANSYLVANIA III: VANQUISH THE NIGHT

Programmed by Antonio Antiochia and Veronika Slintak

Retail price: $39.95
Typical mail-order price: $15
Not copy protected
Requires 512K RAM

Merit Software
13635 Gamma Road
Dallas, TX 75244-9920
(800) 238-4277

Reviewed by Noreen Ribaric

Transylvania III: Vanquish the Night is a graphics adventure—or, more specifically, an interactive novel with graphics. You become the main character in a story and control what happens by typing in commands. Transylvania III: Vanquish the Night is the third in the Transylvania series from Polarware. It is the first of them, however, to have a IIGS-specific version released.

In the first game in the series, Transylvania, the Princess Sabrina (daughter of King John the Good of Wallachia) mysteriously disappears. Your search for her leads you to the dark forests of Transylvania, and to the castle of a murderous Vampyr. In the second adventure in the series, Crimson Crown, the Vampyr steals the Crimson Crown—which possesses great magical powers—from King John the Good. You must journey back to Transylvania, accompanied by Princess Sabrina and Prince Erik (heir to the throne), and recover the Crown.

Now, in Transylvania III, you find yourself in a place called Slavaria, and learn from Prince Erik that the three of you had been overpowered by a tidal wave while returning the Crimson Crown to its rightful owner. You must search for Princess Sabrina once again, and also help to defeat the unjust King Boleslav of Slavaria who—along with the evil Vampyr, Lord Drakul—is wreaking havoc all over Slavaria in search of the Crimson Crown.

Your commands are interpreted by COMPREHEND, a parser that can recognize over 1,000 words. Your commands can be as simple as one word, or as complex as a compound sentence. With the help of a program called The Graphics Magician, Transylvania III displays detailed illustrations for each scene in the story. The graphics are very nice, and although I have not played the Apple IIe/IIc versions of the first two games in the series, I have played them on another computer, and the graphics for the IIGS version of Transylvania III are much better. There is also a neat feature in Transylvania III (and the others) that allows you to turn off the graphics, if you wish. This can become handy if you find yourself backtracking a lot and do not want to wait while the graphics are drawn (which can sometimes be a little slow). In addition to the graphics, Transylvania III also has some nice sound effects—such as dragons breathing fire and Death arriving on a galloping horse.

As you travel throughout Slavaria, there will be many puzzles to solve and obstacles to get past. Some clues are presented along the way, and some in the extra materials provided with the game. Some of these "extras" include the journal

of a mad scientist (which provides some very useful information that he discovered during his experiments), and a Slavarian phrase book (of course they don't speak English in Slavaria) to help you get along with the locals. Some of these puzzles and obstacles are fairly simple to get past, and others require a bit of thinking (and maybe a little research if you didn't pay attention in your high school literature classes)! But never fear, if you can't solve all the puzzles or get past all the obstacles, you can send off for a hint book, *free of charge*! (Just send them a Self-Addressed-Stamped-Envelope.) The hint book is designed so that it doesn't give away everything—it has three levels of hints (which must be unscrambled) for each of the obstacles, so you can't accidentally read too much and spoil the game.

Some of the obstacles that you must get past take the form of evil beings that will kill you if you make one wrong move. Some of the not-so-nice creatures that will impede your progress are mummies, dragons, killer moths, and other assorted monsters (including vampires, of course). But there are helpful creatures, too—be sure to check out the magic elk! In case you find yourself running into the not-so-nice creatures more often than you'd like, don't worry—you have the ability to save your game. You can only save three separate positions in the story, though, and you must save them on the Transylvania program disk labeled "disk 2" (although nothing is stopping you from making multiple copies of "disk 2" with a different set of saves on each one)!

There are a few annoyances, though. The first is that Transylvania III is not installable on a hard drive, even though it is not copy protected. This is probably because it runs under an old version of ProDOS 8, not GS/OS. You have to boot with the Transylvania III disk to run the game, and have to reboot your IIGS when you are finished. And booting the game is slow! It takes 30 seconds just to hear the introductory sounds, 60 seconds to get to the title screen, and a total of 2 minutes and 40 seconds before the game is finished loading—and this is with an accelerator board running at 8 MHz! Another thing I did not like is that it does not support the Apple Human Interface Guidelines—it does not use the mouse or menus, and does not support desk accessories. The third thing is that the instructions included in the box were for the Apple IIe/IIc (the instructions on how to run the program referred to side 1 and side 2 of the disk).

Transylvania III is a good program. The story is interesting, and the graphics and sounds add to an already enjoyable game. It does have a few minor annoyances though—although they don't really interfere with the play of the game. But if these things were changed, Transylvania III would be an even better program! [Just before we went to press, we were informed that Merit Software is working on an update to Transylvania III that will allow it to be run from a hard drive under GS/OS.] If you like mystery, fantasy, adventure, and controlling your own destiny, you will enjoy Transylvania III. GS+

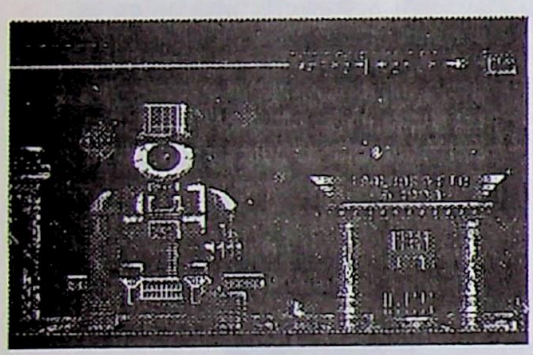
ATTENTION!!! La FRANCE arrive...

3 New and Amazing softwares for your Apple IIGS

PHOTONIX II - BOUNCING BLUSTER II - SPACE SHARK

Now!!! for the American users of Apple IIGS

PHOTONIX II The fastest copier for reproducing Apple IIGS and Macintosh diskettes.

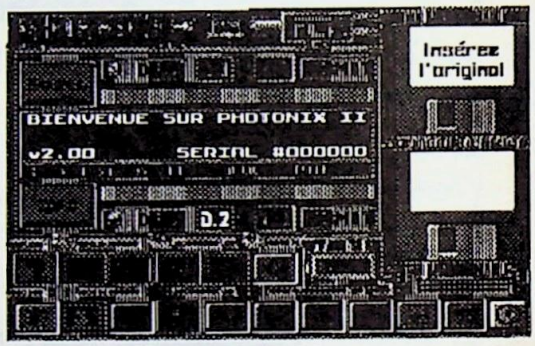


Graphical Interface

Mouse Support

Stereo Sound

Runs on Apple IIGS with Roms 01 or 03



By Pierre Raynaud-Richard - Olivier Goguel - Denis Barritault - Daniel Bar - Olivier Bailly-Maitre

FUNCTIONS : Quick copy - Mass copy - Anti-Virus - File Re-organization - Rapid Format
Directory Reading - Parameters saved in BRAM...

PHOTONIX II can be installed on a hard disk, it is reliable, and allows the verification of copies made.

PHOTONIX II is a program in English and is delivered with user manual in English.

PRICE : $45.00 + $5.00 for shipping by airmail

TOOLBOX

6, Rue Henri Barbusse - 95100 Argenteuil - France
Fax : 0 11 33 1 39 47 44 08

BOUNCING BLUSTER II

Breakout style game with screen editor

GS/OS 5.0.2 (or later) compatible
Installable on Hard Disk

Terrific patterns, magic capsules, psychedelic monsters. You will need a lot of skill and dexterity to get to the end of **Bouncing Bluster II**.

The game exploits all of the graphical and musical capabilities of the Apple IIGS.

It is delivered with user's manual in English.

PRICE : $59.00 + $5.00 for shipping by airmail

SPACE SHARK

A Defender style arcade game for the IIGS

The object of the game is to destroy all of the enemies who want to invade you. You pilot a space ship around with a laser cannon forward and aft and bombs. **Space Shark** consists of 15 levels of play that are more and more crazy... The game is very fast and exploits all of the graphical and sound capabilities of the Apple IIGS. It can be played using a mouse or a joystick.

Space Shark is delivered with user's manual in English.

Free demo disk on request !

PRICE : $49.00 + $5.00 for shipping by airmail

PHOTONIX II + BOUNCING BLUSTER II + SPACE SHARK
for only $120.00 (+ $5.00 for shipping by airmail)

All orders must be prepaid in U.S. funds drawn on an U.S. bank. We accept checks in U.S. dollars.

Send us your address

you'll receive a free bonus disk
for your APPLE IIGS Computer !!!

PROGRAMMERS !!!
IF YOU HAVE SOME PROJECTS CONTACT US...

TOOLBOX

6, Rue Henri Barbusse - 95100 Argenteuil - France

GS+ Back Issue Information

Sep-Oct 1989 (V1.N1)
• $4.50 mag • $5.50 disk • $6.50 both

- System Software 5.0 Compatibility Chart
- NoDOS - A file utility New Desk Accessory complete with ORCA/C source code on disk
- Graphics Galore - Drawing "how-to" with 3 pictures on disk
- Reviews of Arkanoid II (new custom levels on disk), Crystal Quest, ORCA/C, Rocket Ranger, Silpheed, Test Drive II, TransWarp GS, Turbo Mouse ADB
- PLUS: Graphics, rumors, and the most over-hyped product of the year!

Nov-Dec 1989 (V1.N2)
• $6.50 disk (magazine is sold out!)

- EGOed - An NDA text editor (TML Pascal II source code on disk)
- Brush with Greatness - Tips on drawing faces (pictures on disk)
- PLUS: Original icons and new levels for Laser Force on disk

Jan-Feb 1990 (V1.N3)
• $6.50 disk (magazine is sold out!)

- Rotator - A beginner's desktop programming tutorial and program w/source code written in ORCA/C
- Winning Arkanoid II Levels
- Brush with Greatness - Space graphics (pictures on disk)
- HyperStudio stack version of GS+ V1.N2 on disk.

Mar-Apr 1990 (V1.N4)
• $6.50 disk (magazine is sold out!)

- All About Control Panel Devices - with Desk Color CDev and ORCA/C source code on disk
- Random IIGS Programming Notes - An EGOed update
- Brush with Greatness - Architecture on your IIGS with pictures of the CitiCorp building and Frank Lloyd Wright's house on disk

May-Jun 1990 (V1.N5)
• $4.50 mag • $6.50 disk • $9.50 both

- AppleFest Report
- Beginner's Guide to System Disks - Part 1
- GS/OS prefixes - PreFixer CDev and ORCA/Pascal source code on disk
- Brush with Greatness - How your IIGS makes colors
- Reviews of CMS SDRM 45 Megabyte Removable Hard Drive, S&S-RAMCard, DataLink Express modem, Visionary GS digitizer, GraphicWriter III, ZapLink, McGee, Math Blaster Plus IIGS, The New Talking Stickybear Alphabet, a sneak peek at the ZipGS

Jul-Aug 1990 (V1.N6)
• $4.50 mag • $6.50 disk • $9.50 both
(Only a few left—call for availability!)

- KansasFest Report
- Beginner's Guide to System Disks - Part 2
- Transfusion - An NDA terminal program (ORCA/C)
- Reviews of AMR AS800K 3.5-inch drive, Salvation: The Exorciser, Disk Access, MD-BASIC, Katie's Farm, Task Force, BLOCKOUT, OMEGA, 2088: The Cryllan Mission, Hunt for Red October, Revolution '76, Where in the U.S.A. is Carmen Sandiego?

Sep-Oct 1990 (V2.N1)
• $4.50 mag • $6.50 disk • $9.50 both

- Brush With Greatness - making the most of your digitizer
- Interview with Brian Greenstone (programmer of Xenocide)
- PING - video table tennis program (Merlin assembly)
- Shuffle - an Init file that allows you to move desktop windows from the foreground to the background (ORCA/M)
- Battery Brain - CDev saves BRAM parameters to disk (ORCA/C)
- Reviews of GS Sauce memory card, Salvation: Wings, World GeoGraph, Orange Cherry Talking Schoolhouse series, QIX, Solitaire Royale, InnerExpress

Nov-Dec 1990 (V2.N2)
• $4.50 mag • $6.50 disk • $9.50 both

- Interview with Bill Heineman (programmer of Dragon Wars)
- Beginner's Guide to System Disks - Part 3
- LaserWriting - a guide to using an Apple LaserWriter with the IIGS
- Christmas Buyer's Guide
- TeachText Translator - import and export TeachText files in GWIII
- Reviews of Quickie Hand Scanner, AE 3.5" Disk Drive, Salvation: Renaissance, USA GeoGraph, Rastan, Captain Blood, HOSTAGE, Questmaster, Pipe Dream, The Immortal, PIRATES!

Jan-Feb 1991 (V2.N3)
• $4.50 mag • $6.50 disk • $9.50 both

- AppleFest/Long Beach '90 and the Apple II Achievement Awards
- Interview with Jim Carson of Vitesse, Inc.
- Introduction to System Software v5.0.4
- RAM Namer - a CDEV that can rename your RAM disk at boot time, with ORCA/C source code on disk
- *GS+* program updates - Battery Brain v1.1, EGOed v1.32c (now written in ORCA/C), GWIII TeachText Translator v1.1
- Reviews of ZipGSX, LightningScan, Design Your Own Home, Print Shop Companion IIGS, Your IIGS Guide, Dragon Wars, 2088: The Cryllan Mission - Second Scenario, Space Ace, Sinbad and the Throne of the Falcon

Mar-Apr 1991 (V2.N4)
• $4.50 mag • $6.50 disk • $9.50 both

- Interview with Dave Hecker of Seven Hills Software
- Working with the Toolbox - Part 1: The Tool Locator
- Quick NDA - an init that can assign control-keypad equivalents to your New Desk Accessories, with ORCA/M source code on disk
- The New Order - a NDA that can reorder the contents of your directories, with ORCA/C source code on disk
- GS+ program updates - EGOed v1.33, Transfusion v1.1.1
- Reviews of Harmonie, Independence, InWords, Allison Digitizing Software, MAX/Edit, Software of the Month Club, Super GS Award Maker, Talking Speller II, Halls of Montezuma

All prices include $1.50 postage and handling (orders will be sent First-Class to the U.S., Air Mail to Canada and Mexico, and surface to all other countries. For Air Mail to all other countries, add $5 per issue). Tennessee residents add 7.25% sales tax. Mail back issue requests to: *GS+* Back Issues, c/o EGO Systems, PO BOX 15366, Chattanooga, TN, 37415-0366; or call (615) 870-4960, Monday through Friday between 9 a.m. and 6 p.m. EST, to verify availability. Please include your phone number on all orders placed by mail (in case we are sold out of an issue)! For MasterCard or VISA orders placed by mail, also include your card number, expiration date, and signature.

Buying Ad Space In GS+ Magazine

1/3 page - $30
1/2 page - $45
2/3 page - $60
1 page - $75
2 pages - $140

If you want your ad to appear on the back cover or inside back cover, add an additional $75.
Save an additional 20% by placing the same size ad in more than one issue!
Prices good through December 15,1991.
Request your ad space today to guarantee these low prices!

DEADLINES:
July-August 1991 (Volume 2, Number 6) June 15, 1991
September-October 1991 (Volume 3, Number 1) August 15, 1991

All ads must be camera-ready copy. Payment must accompany ad. Make checks payable to **EGO** Systems, or call us to charge it on your credit card. For more information contact:

GS+ Magazine
c/o **EGO** Systems
P.O. Box 15366
Chattanooga, TN 37415-0366
Voice phone: (615) 870-4960

If you need to ship your ad to us using a service other than the U.S. Postal Service, please call to make arrangements.
If you wish to place an ad for a product we have not reviewed, we request that you include a review copy with your ad.

GS+ Ordering Information

GS+ is published bimonthly and sold for $3.00 an issue for the magazine only, and $8.00 an issue for the magazine + disk. But, if you sign up for a 1-year subscription (six issues) or a 1/2-year subscription (three issues), you can save 11-25%! To sign up, send this completed form (or a photocopy) along with a check or money order (**payable to EGO Systems**), or your credit card number, to: *GS+* Subscription Services, c/o **EGO** Systems, P.O. Box 15366, Chattanooga, TN 37415-0366.

Name: _____ Phone: ( ) _____
Address: _____
City: _____ State: _____ Zip: _____

- ❑ 1-year subscription - magazine + disk - $36
- ❑ 1/2-year subscription - magazine + disk - $20
- ❑ 1-year subscription - magazine only - $15
- ❑ 1/2-year subscription - magazine only - $8
- ❑ Sample issue - magazine + disk - $8
- ❑ Sample issue - magazine only - $3

- ❑ Check or money order enclosed
- ❑ Bill my MasterCard
- ❑ Bill my VISA

Card No.: _____
Expiration Date: ____/____
Signature: _____

If you prefer to use your credit card to order by phone, give us a call at (615) 870-4960, Monday through Friday, between 9 a.m. and 6 p.m. EST. All subscriptions will start with the next issue published. Please allow 2-8 weeks for delivery of first issue. Tennessee residents add 7.25% sales tax. Add $1.50 an issue if you want First-Class delivery. Add $1.50 an issue for delivery to Canada or Mexico. Add $1.50 for surface mail to all other foriegn countries. Add $5 per issue for Air Mail to all other countries.

Contest #4 Update

By Steven W. Disbrow

The Envelope Please . . .

Well, the deadline for contest #4 has passed and, quite frankly, I just can't decide on a winner! So, here's what we are going to do: below, you will find the three finalists, simply let us know which one you like best. The name that receives the most votes will be the new name for *GS+*. (For those of you that are just joining us, we are going to change the name of this magazine. For full details as to *why* we are doing this, see "Contest #4" in *GS+* V2.N3.)

The Finalists

Granny Smith's Almanac
Suggested by Keith Bekofske.
Super II
Suggested by Joe Wankerl.
ROM II
Suggested by yours truly.

The deadline for your votes is July 15, 1991. The winning name will appear on the cover of (what would have been) *GS+* V3.N1. Send those votes to:

GS+ Contest #4
P. O. Box 15366
Chattanooga, TN 37415-0366

You can also send your votes to me via America Online: GSPlusDiz
proLine/InterNet: diz@pro-gsplus.cts.com

But I Like The Name *GS+*!

Me too. So, I wrote a letter to Mr. Sculley asking if we might continue to use the name *GS+*. Thus far, I have gotten no response. If you would like to see us continue with the name *GS+*, you can send your vote to Mr. Sculley in care of:

Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, CA 95014-6299 *GS+*

GS+ Classifieds

1/2 PRICE SOFTWARE

The Hunt For Red October
(novel not included) - $ 20
Questmaster - $20

Contact: *GS+* Magazine
c/o EGO Systems
P.O. Box 15366
Chattanooga, TN 37415-0366
(615) 870-4960

FOR SALE

Apple 3.5-inch 800K disk drive - $240 or best offer.

Contact: Eli Weissman
4454 Van Noord Avenue
Studio City, CA 91604
(818) 906-2860

WANTED

Kensington System Saver GS

Contact: Eli Weissman
4454 Van Noord Avenue
Studio City, CA 91604
(818) 906-2860

Readers can place an ad in the *GS+* Classifieds for only $5. This cost buys 25 words in one issue of *GS+*. Additional words are just 10 cents each. The *GS+* Classifieds are the perfect way to contact all of the other IIGS owners out there. The deadline for inclusion of a classified ad in the next issue (Volume 2, Number 6) of *GS+* is June 15, 1991. Simply fill out a photocopy of the coupon below; or send your ad along with your name, address, phone number, number of issues to run, and payment (made payable to EGO Systems) to us here at *GS+*; or call us at (615) 870–4960, Monday through Friday between 9 a.m. and 6 p.m. EST, to place an ad with your MasterCard or VISA.

GS+ Classified Ad Order Form

Ad copy: _____

_____

_____

_____

_____

_____

Number of issues to run: _____ Number of words: _____ Total enclosed:$_____

Name: _____ Phone: _____

Address: _____

City: _____ State: _____ Zip: _____

Feedback

How did you first hear about *GS+* Magazine?

Are you a subscriber to *GS+* or was this a sample issue?

If you are a subscriber, do you receive the Magazine and Disk? If you do not receive the disk, why not?

How would you describe your level of computer experience?
- ❑ babe in the woods
- ❑ novice
- ❑ I get by
- ❑ fairly proficient
- ❑ experienced
- ❑ power-user
- ❑ digital deity

Are you a member of a user group?
- ❑ Yes
- ❑ No

Please tell us a little about your IIGS system. Do you have a:
- ❑ Hard drive
- ❑ Modem
- ❑ Dot Matrix Printer
- ❑ Laser Printer
- ❑ 3.5-inch drive
- ❑ 5.25-inch drive
- ❑ Scanner
- ❑ Digitizer
- ❑ Accelerator card
- ❑ Fan
- ❑ Extended Keyboard
- ❑ Anything else?

How much memory do you have in your IIGS?

Is your IIGS part of an AppleTalk network?
- ❑ Yes
- ❑ No

Do you have a SCSI card in your IIGS?
- ❑ Yes Which one?
- ❑ No

Which name do you like best?
- ❑ *Granny Smith's Almanac*
- ❑ *Super II*
- ❑ *ROM II*

On a scale of 1 to 5 (with 1 being "poor" and 5 being "excellent"), how would you rate the following items from this issue of *GS+*?

Matt Deatherage Interview ____

Working With The Toolbox ____

Autopilot ____

EGOed v1.34 ____

NoDOS v1.5 ____

Softlock ____

How would you rate the reviews in *GS+* Magazine? ____

How would you rate *GS+ Magazine*? ____

How would you rate the *GS+ Disk*? ____

How would you rate *GS+* overall? ____

How often do you think we should update our programs?
- ❑ Every issue
- ❑ Every other issue
- ❑ Only when the update is a significant improvement
- ❑ Never

We want to put more stuff on the *GS+* Disk. Would it be OK with you if we began using ShrinkIt to compress the *source code* on the *GS+* Disk?
- ❑ Yes
- ❑ No
- ❑ Drop the source code entirely

Should we review more:
- ❑ Educational Software
- ❑ Games
- ❑ Hardware
- ❑ Productivity Software
- ❑ Utilities

How would you rate the technical content of *GS+* Magazine?
- ❑ Child's play
- ❑ Just right
- ❑ I'm drowning in jargon!

What would you like to see in the next issue or two of *GS+*?

We want to have the best, most reliable advertisers in the business. Who would you recommend that we try to get? Why?

If there was one thing you could change about *GS+*, what would it be?

Do you think you will renew your *GS+* subscription? If not, please tell us why.

What do *you* think about Applied Engineering's new technical support policies?

Anything else you want to say? Feel free to add additional sheets.

Send this sheet to:
GS+ Feedback
P. O. Box 15366
Chattanooga TN, 37415-0366 GS+

***GS+* Magazine**
P. O. Box 15366
Chattanooga TN, 37415-0366

Last Issue Will Be: V3.N4*
DONALD COHEN - 100CHE205W
205 W 95 ST #3E
NEW YORK NY 10025

BULK RATE
U.S. POSTAGE
PAID
Chattanooga, TN
PERMIT NO. 616