$3.00
(magazine only)

$8.00
(magazine + disk)

September
October
1990

Volume 2
Number 1

GS+

The *First* Apple IIGS Magazine + Disk Publication!

Gala First Anniversary Issue!

IN THIS ISSUE:

WRITER'S BLOCK

HAPPY BIRTHDAY!
On September 1st, 1990, EGO Systems and *GS+* Magazine both turned 1 year old. On September 26th, 1990, the Apple IIGS turned 4 years old. On September 30th, 1990, I turned 25. Happy Birthday! Now, le'ts get down to business.

NO ADS
There are no ads in this issue other than our usual Classified Ads. This is no problem for us financially, but it did have an adverse effect on the size of the magazine. This issue is only 40 pages (including covers) as compared to last issue's 48 pages. We had about 45 pages of material ready to go, but, since our printer requires that we lay this thing out in 8 page increments, we had no choice but to pull the extra pages.

AD PROBLEMS
Speaking of ads, several of you have called to say that you have had problems with an advertiser of ours, LRO Computer Sales. Basically, some of you ordered some things from them, and the items never showed up. I have spoken with Larry O'Conner at LRO about the matter and was told that this was the result of a burglary that took place at LRO. I was also told that UPS had actually lost several of the shipments. Mr. O'Conner was very helpful, and has promised to send me documentation supporting these claims. At this time, however, I have not received these documents. Until I receive these documents, there will be no ads from LRO in *GS+*. When I do receive these documents, and if they support LRO's claims, I will have no qualms about continuing to run their ads.

WE ARE LATE
As you have no doubt noticed, *GS+* has been slipping further and further out of sync with the months on the cover of each issue. This was due to a number of things, all of which were ultimately my fault, and all of which are totally irrelevant now. The fact is that we are way behind schedule. I don't like it, and you probably don't either. So, we are going to change it. Expect your next issue of *GS+* around the middle of November.

Sorry to be so abrupt, but, as you might imagine, there's lots to be done!

Diz

LETTERS

Dear *GS+*,

I have just finished reading my second issue of *GS+* Magazine and looking over my *GS+* disk. I would like to commend your staff on distributing an excellent source of information on the Apple IIGS. I have owned my IIGS for a little over a year now, and despite the recent troubles of the Apple II market, I would never even consider using another computer. I only wish your magazine was monthly instead of bimonthly. Your magazine seems to be one of the very few reliable sources to obtain up-to-date information on how Apple is planning to breathe new life into the Apple IIGS market. I am planning to attend AppleFest this winter now that the location has been switched to my city. I mean, how could I pass up this opportunity? I hope to possibly meet some of you there. Until then, keep up the excellent work and I'll be anxiously awaiting my next issue.

Carl A. Cartwright
Longbeach, CA

Thanks for the positive comments! We would love to meet all of you as well, but... the sudden increase in airfare has made it next to impossible for us to send anyone to the show. We will have an "official" representative there (more about him next issue), and we hope to have plenty of free magazines at the show for folks to take, we just won't be there to see Apple introduce the new... whatever it is.

Which is why we need your help. If you are going to AppleFest this December, we want to borrow your eyes and ears. If you see or hear anything that you think your fellow GS+ readers should know about, write it down and send it to us. If we use your tidbit in our AppleFest report, we'll extend your subscription for an issue. If more than one person sends in the same tidbit, we'll give the free issue to the one with the earliest postmark. If they come in on the same day, we'll pick one at random. If it is a really hot tidbit, you can call us.

Dear Steve,

I have a suggestion—allow programs written in AppleSoft BASIC to be submitted (the Writer's Guide only specifies C, Pascal and Assembler). AppleSoft BASIC is still alive and well, and with such GS-specific accessories as the Call-Box Toolbox Programming System and MD-BASIC, it is still a viable language.

Keep up the good work!

John K. Estell
Champaign, IL

Well, I knew that, someday, someone would bring this up! I didn't think it would take a whole year though! While I do admit that AppleSoft BASIC has been, and still is, a very useful thing to have built into the IIGS, I don't think that AppleSoft BASIC programs belong in GS+.

continued on page 2...

CONTENTS

ARTICLES

PROGRAMS

REVIEWS

DEPARTMENTS

GS+ is produced on an Apple IIGS using GraphicWriter III, EGOed, AppleWorks GS, and an Apple LaserWriter IINT. FingerPrint GSi is used to freeze the screen so the screen photographs can be taken.

STEVEN W. DISBROW
Publisher, Editor

NOREEN RIBARIC
Associate Editor, Layout

JOSEF WANKERL
Contributing Editor, Programmer

GS+ is an independent publication, not affiliated in any way with Apple Computer, Inc.

GS+ is published bimonthly by:
EGO Systems
3535 Mountain Creek Road #A-17
Chattanooga, TN 37415-6700
(DO NOT SEND MAIL TO THIS ADDRESS!)

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

Subscription rates for one year (six issues) are:
Magazine only - $15
Magazine + Disk - $36
Tennessee residents add 5.5% sales tax.
Chattanooga residents add 7.25% sales tax.
Canadian and Mexican orders add $1.
Other foreign orders add $5 for surface delivery or $25 for airmail.

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

pro-gsplus BBS (615) 875-4607
2400 Baud/8 data bits/no parity/1 stop bit

The reason I started GS+, and the reason that I don't publish AppleSoft programs, is that I want to help establish the IIGS as more than just a educational computer that has a library of 10,000+ ancient programs. I want to show people that this is a serious computing machine that you can use to do things that the "experts" will tell you can only be done on an IBM PC or Macintosh. GS+ is one of those things I was told could not be done on the IIGS.

Another thing I was told was that there would never be any new and truly useful programs written for the IIGS. I think EGOed and Transfusion pretty much blow that theory out of the water. They could not have been done in AppleSoft.

What we are trying to accomplish here is to show folks how to write new and useful programs in high-level languages that look good on a résumé! In my opinion, AppleSoft just won't cut it.

Of course, it is entirely possible that my opinion is dead wrong. If you think it is, prove it! Use AppleSoft BASIC to write us a desktop *program that* follows all the rules *and does something* useful. *If you can do it, I'll be more than happy to publish your program!*

Dear *GS+*,

I would like to take this opportunity to thank you for your review of *2088: The Cryllan Mission* [*GS+* V1.N6, page 40].

I found the review fair, honest, and unbiased. It is refreshing to find a magazine that pays more attention to the product than the hype or name recognition. Thank you once again.

There is one minor point that needs correction though: the 800 number for orders was disconnected some time ago. All calls should be directed to our (713) 493-3232 number.

Vivek Pai
Vice-President, Development
Victory Software

Steve,

Thanks for the great review of *Disk Access!* [*GS+* V1.N6, page 29] We already had your two objections on our "to change" list: the 64 character pathname limit seemed reasonable at the time (especially with System Software v4.0) but it really [isn't.] [We] also wanted to have the feature of "opening to the current prefix," but had to drop it in the interest of time. ...This one will definitely be in an update.

You mentioned leaving Disk Access open, but it being so big... I don't know if you mean big memory-wise or screen-space-wise. If it's screen space, there is a nifty solution: [Click on] the Zoom Box in the upper-right of the Disk Access window and the window will zip to the bottom of the screen, just showing the window's title bar. Click it again and it will zip to its normal position. (The keyboard shortcut for this is Command-Z.)

Dave Hecker
Seven Hills Software

Thanks for the tip Dave! Of course, that is in the manual... I just overlooked it. Now, where's that GraphicWriter III update? Hmmm?

If you have a question, comment or criticism about *GS+* Magazine, we want to hear it! Please note that due to space limitations, we can not answer every letter here in *GS+*. If you want a personal reply, please enclose a self-addressed, stamped envelope, a phone number, and the best time to call. Please direct *all* letters to:

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

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, from 8 a.m. to 6 p.m. Eastern Standard Time. Or, write to us at:

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

PROGRAMMER'S QUEUE & A

This time around, all of our questions are in regard to our NDA terminal program, Transfusion. The questions come from Ken Moordigian of San Fernando, CA. The programs author, Josef W. Wankerl, supplies the answers. [For more information about the latest version of Transfusion (Version 1.0.1), see "Random IIGS Programming Notes" on page 16 - Ed]

Queue:
In the Transfusion NDA program you call, `DoAbout()` which displays the about screens as an `AlertWindow()`. My questions are:

1) How do you change the text style?
2) Where can I find the control codes for changing the text styles (Bold, Shadow, Underline, etc.)?
3) Can these styles be used in a StatText item?
4) Can they be used with a `DrawText()`, `DrawCString()`, or `printf()`?

I looked through most of the Toolbox, firmware, hardware, and GS manuals and came up with zero!

A:
The codes for changing the text styles are, believe it or not, in the LineEdit chapter of *The Apple IIGS Toolbox Reference: Volume 1*. In my copy, they are on pages 10-43 and 10-44. The way you code a LETextBox2 string embedded style change is to first use what the toolbox calls a "delta flag byte" or \0x01. Following the delta flag byte, you put the change flag value (S for style, F for Font, J for justification... kind of intuitive, eh?) and then a word (this is two bytes in the integer format, so if you're coding them in C, don't forget to put the low order byte first) specifying what changes to make. `DrawText()`, `DrawCString()`, or `printf()` can not use this, though. You first have to manually change the style with a `SetFont()` call or the like, then call `DrawText()`. If you want to use a LETextBox2 string, you can use the `LETextBox2()` tool call to draw it for you. StatText items can not use these type of embedded changes either. The only kind of text that will allow the embedded style changes is the LETextBox2 string.

Queue:
I'm using a Terminal Node Controller (TNC) with packet radio as one of my 'modem devices'. The TNC will spit out incoming streams of data continually when in the command mode during peak traffic hours on a busy frequency. This is normal for a TNC. A good terminal program for packet radio will receive the data packets, parse them for the identifier codes, then route them to the proper destination (window, in this case), then add the text to the TERecord. All of this is irrelevent to a program that just has one window/destination like Transfusion. A split screen for outgoing and incoming data would be nice but that's another problem, along with sending a whole line of text to the modem port instead of a character at a time.

The point of all that drivel above is that if you open a model dialog box, specifically 'Font', with the incoming data running in the background (continually) the time lag can be great, or worse. After selecting a slow or large writing font and clicking OK the menu bar does not always unhilight! If I again select Choose Font, I can get the whole program to hang... as in, [I have to] reboot to escape!

I haven't gotten deep enough into the code to find the reason, but I thought you might like to know. Maybe if you XOFFed the serial port before bringing up a dialog then XONed after closing it would eliminate the problem.

A:
I've not used Transfusion over 2400 baud. I assume that your TNC sends you bursts of data faster than that. A good fix for that without using the XON/XOFF method would be to increase the internal buffer size from the system's default 4K to something a bit more manageable—say, the maximum 64K. To do this, you'd issue a `SetInBuffer` extended serial card call documented in the *Apple IIGS Firmware Reference* by Addison-Wesley. I might add this ability to Transfusion if I ever switch to using resources.

The problem with using XON/XOFF is having one system supporting XON/XOFF and the other one not—thus producing nice garbage characters on the system that doesn't support it. What I do now in version 1.0.1 is to just not read from the buffer at all.

Implementing a split screen option wouldn't really be that difficult. I'd just have two TextEdit controls in the window and route outgoing info to the top one, and the incoming info to the bottom one. With most applications, however, the remote computer will echo your transmissions back to you, so if you really wanted to get picky, you'd have to weed out the echos—not a very fun task.

Now for your font problems, well, why in the world would you want to pick a large and/or slow font? The GS is already slow enough as it is! When Transfusion is running, it's sharing cycles with a host application—slowing things down even more. It really doesn't make much sense! Basically, the only reason the font option is in there is so you can have fast text (Shaston 8) or nice, mono-spaced text (Courier or Monaco). I didn't intend people to go online with the NASA 48 font! The GS seems to have a problem with larger fonts anyhow (here I go passing the buck to Apple—"It's not my fault that the Toolbox is buggy"—don't you hate it when people say that?). Nevertheless, this was a potential danger, so I fixed it. For more on this, see "Random IIGS Programming Notes" on page 16.

Once again, thanks for the questions. I hope to update Transfusion as time allows. If anyone has any questions about, or suggestions for Transfusion, be sure to send them in!

BRUSH WITH GREATNESS

Making the Most of Your Video Digitizer
by Michael J. Quinn

How many times have you pulled the hair out of your head trying to get a decent picture from your video digitizer? How about a decent *color* picture? Well, believe it or not, it *is* possible to do. If you've ever tried doing a color digitization with the default settings, you've found out that the default is nowhere near what the best possible settings should be. In this article, I hope to help you achieve the best possible pictures with your video digitizer.

BLACK & WHITE

Black and white pictures are, by far, the easiest images to make with good results. The key thing to remember is *high contrast*. In normal light settings, it is usually best to put the settings of your digitizer software at a high contrast. You want to use as many of the 16 shades of grey as possible to reduce the *banding* effect. The banding effect is what happens when you are limited to the amount of colors (or shades of grey, in this case) available for you to use. The human eye can distinguish between about 40 shades of grey, so with only 16 available to us on the GS, you can definitely see the borders of adjacent shades of grey, so you can see why you would want to use as many shades as possible. The more shades you use, the harder it is to see the borders of those shades.

I have provided two black and white digitized pictures showing the effects of contrast (**Low.Contrast.BW** and **HighContrast.BW**) to show what can happen when your settings are off just a little. In the **Low.Contrast.BW** image, the contrast was set too low while digitizing and resulted in an image that was missing the four brightest shades of grey, which resulted in a low contrast and highly–banded image. In the picture **HighContrast.BW**, all 16 shades of grey were used, which results in a balanced intensity image which is close to the best that can be produced. While you are digitizing, once you are sure that all 16 shades are being used, adjust the brightness to make sure the lightest shade does not overpower certain details of the picture. For example, suppose you are digitizing someone's body from the waist up with a completely black background and a bright light aimed right in their face. It would be difficult, in this image, *not* to use all shades; but since there is a bright light on the face, if your brightness is just a little too high, you may not see much detail in this persons face, even though you are using all 16 shades of grey. So to put it in a nutshell: use the highest possible contrast. Then when you are satisfied with the contrast, adjust the brightness so that you can see as much detail as possible.

COLOR

Color images are, by far, the most difficult to achieve with decent results. Two key things to remember for color digitizing are *medium-low contrast* and *medium-high brightness*. The default settings for most digitizers are to have the contrast for all colors (red, green, and blue) set as high as possible, and brightness settings for all three RGB values set to the middle (about 7). The results of these settings, after digitizing, are similar to the picture **Default.Colors**. For this image, I used the default settings to digitize. As you can see, the results were not very good! There is a bright orange color that dominates the picture. The image is also spotted with this orange. The "spotting" is cause by the digitizer's color separation technique. The video digitizer digitizes three images before you see the final result. It digitizes a red image first (the circuitry in the digitizer separates the red, green, and blue light) which is similar to a black and white image, except that it is digitizing the shades of red, and only parts of the image that are supposed to be red, then stores that image in the computer's memory. Next, a green image is captured and stored in memory, then a blue image. The problem with this is that the color separation produces stray, random, highly contrasted dots on the screen. This is where the orange dots in the **Default.Colors** picture came from. By lowering the contrast of the color, you can keep this spotted effect to a minimum, as well as get rid of the problem of having super-bright colors on the image. I usually keep my color contrast settings at about 5 for all three red, green, and blue colors. By lowering the contrast like this, you will also decrease the brightness of the picture. All you have to do to fix this is to increase the brightness of all the colors. I usually set the brightness on my digitizing software to about 8.

The settings that are best for one image are not always best for another image. You may fix the settings and get a great result on one picture, then, using the same settings, get a purple, red, green, or bluish image on a different picture. All you need to do is slightly readjust the settings. If your image comes out with a bluish tinge, you should try turning the blue brightness down one notch and digitize again. Sometimes by moving the settings only one notch, it will be too far. When this happens, you should try moving the contrast for that color one notch in the opposite direction that you just turned the brightness, but remember: the higher you make the contrast, the more spots you will have in your final image. To be able to get the best results for color pictures, you need to know how the mixtures of red, green, and blue primary colors produce a certain color. If you don't understand the mixing of the primary colors of light (red, green, and blue), you should refer to my previous "Brush With Greatness" article in *GS+* V1.N5 (May-June 1990).

IMAGE SOURCE

Software settings are not the only things that can effect the final result of a digitized image. The source of the video signal has as much or more to do with the quality than everything else.

VIDEO CAMERA

If you are using a home video camera as a source for your images, there are four things you should keep in mind: *brightness*, *contrast*, *focus*, and *movement*. All of these things can affect

the final result of your digitized image. You must always be sure that you have enough light. If you don't, your pictures will either come out dark or with low contrast, and sometimes both. The brighter the source is, the higher the contrast will be; and the higher the contrast is, the better the image will *usually* be. Some digitizers have problems with high brightness levels. The AST VisionPlus, for example, doesn't like bright lights. Images that are too bright will cause portions of the digitized image to be pulled to the right and leave a black blob on the left side of the screen. If your video camera has a contrast knob, set it high. It's fairly obvious what will happen if your image is not in focus, so I will not waste any space talking about it. Movement is a major factor to consider. As of this writing, all video digitizers for the GS take time to capture an image, even the Visionary GS (also known as the Enhanced VisionPlus). Computer Eyes GS takes six seconds for the fastest digitization rate, which is not very fast at all. If there is any movement while the image is being digitized with Computer Eyes GS, the final image will have wiggles in it. The way a video signal works is, the video source (camera, VCR, Video disc, computer video out, or anything that sends a standard NTSC signal) transmits still images. It sends these still images so fast (60 of them per second) that it looks like there is actual motion. This is similar to the way a film projector works. Different digitizers digitize at different rates, but none of them (for the GS) digitize as fast as the images are being sent to them. The fastest digitizer is the Visionary GS/Enhanced VisionPlus, which can digitize up to 15 frames per second. Even though that sounds fast, it's only 1/4 the speed of the actual video signal being fed to it. To get one complete image, the digitizer has to compile that image out of four frames from the live video signal, so if someone happened to be waving their arms at each of those four frames, their arms will be at a slightly different place and the resulting image will have vertical lines running through it. If at all possible, keep the object in front of the video camera as still as possible. Computer Eyes GS would have to compile an image out of 360 frames (60 frames per second times 6 seconds = 360). You can see how an object moving with this one could cause problems.

If you're digitizing from a source that you cannot control the lighting from (such as a TV broadcast or a video tape), then the only thing you can do (without extra video processing equipment) is adjust the brightness and contrast settings in your software. If you cannot control the motion of objects in the video (such as my cat), then the best thing to do is to tape the video, play it back and use the still image or freeze-frame/pause button on your VCR to digitize it. Freeze-framing an image on tape also has its limitations. For one, most VCRs will only hold an image for a few minutes and it's very difficult to pause on that same image later with a normal home VCR. Another limitation of freeze-framing is that a lot of VCRs do not have very good pausing results: the image usually has a lot of snow or horizontal lines near the top or bottom of the screen and this will usually show up on your digitized image. If this happens, try doing a frame advance until you find an image without much snow on it, or just press stop and try that same frame again if you can find it, because sometimes the snow is gone or at a different place in the picture the second or third time. If the snow is in a different place in the picture, you could save several different digitized versions of that image to disk. Then, using a paint program, you can use the cut and paste features to paste all the good parts into one final picture.

SPECIAL EFFECTS

Both digitizers (Computer Eyes GS and Visionary GS/Enhanced VisionPlus) have their own uses for special effects. The Computer Eyes GS can be used to create some really funky-looking distorted images. Since it takes so long (6 seconds) to digitize an image, you can take advantage of that. Computer Eyes GS digitizes from left to right, so you could sit in front of your video camera and slowly move your head to the right (the video camera's right, not yours). This will make your head look stretched out. You could also do the opposite and move from right to left while digitizing to make your head looked squashed. You could also stand in front of the camera with your arms sticking straight out to your sides and move towards the right while digitizing, until you see that your arm is finished being digitized, then stand still for the rest of your body. The resulting picture would be a normal-looking you, except that your right arm would only be about a foot long. You could move to the right instead and make your arm look twice as long as normal. Here's something we tried at the *GS+* headquarters: get two or three people and pick up another person, holding them sideways and facing the camera, then move to your right while Computer Eyes GS is digitizing you. This will make the person you are holding look less than half their height. You could do the opposite to make a short person look tall. Here's something else to try: you can make your arms look like rubber by slowly flapping your arms while being digitized.

The Visionary GS/Enhanced VisionPlus cannot do the previous special effects, but they can do other effects that Computer Eyes GS cannot do. This is something you'd probably rather save to video tape than to disk, because it has to do with watching live video. Even though you get wiggles when something is moving while you are digitizing, the effects are not so bad that you can't tell what's going on, so you can run your video camera through the digitizer, then plug a standard video cable from the Apple IIGS's video out port on the back of the computer and plug it to a VCR. You can record yourself in live video as a negative image. One of the options in the software will let you invert an image, then continue to digitize in that same mode. You may also try digitizing in a two, four, or eight grey scale mode. Another fun thing to try is digitizing a 16 grey scale black and white image, then changing the palette from the standard black and white to different colors such as red, green, blue, purple, yellow, or whatever other colors you can imagine by replacing the grey values. After you have changed the colors on an image in this way, you can continue to digitize (in black and white mode) but preserving these new colors, which gives sort of a psychedelic look. You could try making your own music video using this technique and playing air guitar to one of your favorite songs, or even play one of your own songs.

INTERVIEW WITH BRIAN GREENSTONE

By Brian M. Winn

Brian Greenstone, a young man in his senior year of college at the University of Texas, has been an asset to the Apple IIGS game market for a few years now. He, with the aid of his company, *Pangea Software*, has designed the ultimate GS commercial game, *Xenocide*. They have also produced six excellent shareware games (*Grackle*, *Copy Killers*, *Quadronome*, *Orbizone*, *Senseless Violence I*, and *Senseless Violence II*) that have a shareware fee of only five -dollars each! Pangea Software is currently working on a new commercial game called *Cosmocade*.

The following is an interview that Brian M. Winn conducted with Brian Greenstone during the month of August, 1990:

GS+
When, and why did you initially get involved in the Apple II world?

BG
Well, I first got involved in eighth grade when my mom enrolled me in a summer night course being taught at my high school. Two other friends of mine went also. Most of the people in the class were in their forties, and they didn't know a resistor from a battery, or a disk from a drive. I read all that I could find and only two weeks after learning my first 'X=1' command in BASIC, I started work on my first game. I can't remember the name of the game, but it was similar to Lunar Lander, but much more complex. From there, it just evolved. I read more books and taught myself assembly and Pascal. The following summer I took an Apple graphics course which got me into doing more complicated and professional animation.

GS+
During this time, was there any one person or anything that gave you inspiration in your programming?

BG
Well, as for inspiration, I guess you could say that I always wanted to "one up" one of my friends in that night class. I didn't even own a computer until about a year after I took that class. I used to write programs on my typewriter and file them away. When I finally got my II+ with 64K, I typed in those programs which I had in my desk for a year. I just enjoyed programming, and I was better than anyone else I knew at the time, so it was sort of a unique "attention-getter" hobby. It wasn't until five years later when I came to the University of Texas that I found other Apple programmers who knew what they were doing, and that's what pushed me to write Xenocide. I wanted to prove to them I could "cut the mustard" since all I had to show at that point were cheesy high school quality programs.

GS+
When did you first begin to program specifically on the Apple IIGS?

BG
I first began to program the IIGS about three years ago. Xenocide was the first program I wrote for it. I had the GS for over one half a year before I attempted to program it. Actually, Xenocide started out as a two-player race car game, but since I had never done a GS program before, I didn't know if it was fast enough to handle it... it wasn't fast enough.

GS+
Software development takes a lot of originality and organization (not to mention patience). What is your method of forming an idea and then developing it into a final product?

BG
Actually, I don't form ideas, rather I let ideas evolve. Like I said, Xenocide started as a two-player race car game. Look what happened. Most of my games start out as an experiment to see if a new idea will work. If it works, then I just keep adding on to it until it's a game. The only game I've done that was fairly well planned out was Copy Killers.

GS+
New game releases for the GS have slowed over the past year and a half. What do you personally feel is the current state of the GS game market?

BG
I personally think that the current state of the GS market is one of rigor mortis. All of the big publishers have formally dropped the GS, no one is buying GS software. I don't think that the GS will last much longer, perhaps another year before even the little guys aren't supporting it.

GS+
But, it would seem that Apple has finally awakened to the needs of the GS community. They seem to be working hard to reestablish the GS as an attractive computer for first-time buyers. With products such as their new DMA SCSI card, speculations of System Software v6.0 (which should include the new sound and animation tools), and the growing possibility of a ROM revision 04 machine, do you think the GS game world can take a turn for the better?

BG
New ROMs won't solve anything for the IIGS. You have to remember that about 2,000,000 people out there own current IIGS ROMs, and even if there was a free upgrade, only a very small percentage of those people would have the new ROM. Most of the 2,000,000 are not Apple enthusiasts and they probably will never even know about any new ROM, nor would they know that the hell R.O.M. meant to begin with. So, a new ROM would only make the situation worse since it would bring up compatibility problems. As for Apple's "renewed" support... b*lls**t.

GS+
Pangea Software currently consists of three members. How did Dave Triplett (artist) and Gene Koh (musician) get involved?

BG
Well, a year and a half ago, when I was looking for a publisher for Xenocide, I put a post on America Online. Dave saw the post and got in touch with me and set me up with Micro Revelations. Gene came in around January (1990) I think. We knew about Gene, but had been unable to use his talents until I rewrote my music drivers so that a good musician could use them.

GS+
Thousands of GS users have enjoyed your shareware games. Is it true that several of the games only took you 24-hours to create?

BG
Yes. Grackel, the first freeware game was written in exactly 24 hours. I started it at 8:00 p.m. one night and finished it the following evening at 7:54 p.m. Copy Killers also took 24 hours, but it was spread over a three-day period. Quadronome took about 30 hours over a four-day period. Senseless Violence I was the most time consuming of all because of the graphics. It took six weeks.

GS+
Xenocide is an awesome display of what the GS can really do. Will we see another game of this quality and magnitude from you again in the near future?

BG
Well, Micro Revelations has verbally agreed to publish Cosmocade, but it isn't definite yet. Cosmocade has the best animation, graphics, and music of any of our games, but it isn't as good as Xenocide in game play. Xenocide was a masterpiece that I don't think I'll ever be able to match.

GS+
Your motto is "Programming is an Art Form." Why do you believe this? Other than your own games, what GS programs can be labeled as true "masterpieces"?

BG
Yes! Programming *is* an art form! I live it, I love it, I want more of it. I used to be a computer science major, but I quit since all they were teaching were the "rules" of programming. Programming is like painting. You cannot just know how to draw, but you must have it in you to draw. A programmer with no creativity is like a Burt Reynolds movie.

Well, besides Xenocide, my favorite game is—hmmm... most games for the GS suck pretty bad. My favorite Apple II game is Alien Typhoon—it was a programming marvel for its time and it was fun. I don't know what to say about my favorite GS game, though. Task Force is extremely well done, but it's kinda redundant. SuperStar Ice hockey has some of the best artificial intelligence I've ever seen. Battle Chess is a graphic masterpiece. Hmmm... dunno. To tell you the truth, I really don't play video games much—I just like to design them.

GS+
What will we see Brian Greenstone doing in the next five years? Will it include Pangea Software and the Apple IIGS?

BG
Well, right now I am directing a Public Service Announcement for the Runaway Hotline. Once I graduate I hope to produce and direct for TV. Unless Atari or Sega comes along and begs for me to work for them, I'm afraid my programmer days will be numbered.

The only way that I'll be programming in a year from now is if I get a good break or if shareware becomes worth the time.

GS+
Is there any advice you would like to give to the up and coming programmers out there?

BG
Yes, my advice to all the aspiring programmers out there is learn the 68000—it's the chip of the future. Don't let people tell you how to program. Remember that it's an art form, so develop your own styles. Most of all, take breaks and catch some rays. I've found that most women can tell the difference between a Cathode Ray Tube tan and a real tan.

GS+
One last question, was it your or Dave's wonderful (but deranged) idea to include all of the colorful "guts" present in a few of Pangea's games?

BG
Well, Xenocide was 90% finished before I got in touch with Dave, so pretty much all of the "blood and guts" in there was my idea. As for the gory stuff in Senseless Violence, both Dave and I came up with those ideas.

EPILOGUE
I have found Brian to be a very friendly, intelligent, and humorous person. He is a little down on the Apple IIGS at this time because of Apple's (lack of) support in the past and the low return on his shareware games. I hope GS users start to support his shareware and commercial efforts so he will continue to create GS "masterpieces".

In following issues of *GS+*, I hope to bring you more in depth interviews with the people that make the GS shine!

Brian M. Winn

Xenocide is distributed and sold by:

Micro Revelations Inc.
P.O. Box 70430
Reno, Nevada 89570.

We here at *GS+* give this game our highest recommendation. If you like arcade games and want the best the IIGS has to offer, buy Xenocide! In regard to Pangea's shareware offerings, you can send your five-dollar shareware fees or contact Brian Greenstone at:

Pangea Software
10918 Kirwick
Houston, TX 77024.

You can send comments or questions to Brian M. Winn by writing e-mail to "WTSBrian" on America Online or writing to:

WinnTECH Software
3279 45th St. W
Webster, MN 55088.

PING

By Daniel Webster

So, you want to write arcade-type video games for the GS? You picked a great computer to do it on because with the built-in sound and graphics capabilities of the GS, you can make even simple programs shine. As an example of this, I have written a GS-specific version of the classic game, Video Table Tennis. The premise is simple. In the two-player mode, each player uses their on screen "paddle" to return a "ball" which is bouncing between the players. If you miss the ball, your opponent gets a point. Generally the game is played to 21, and the first person to reach that score wins. Simple, right? Now all we have to do is code the program. This game is written in Assembly, using the *Merlin* assembler. I highly recommend assembly for all games because it is the fastest language available, and for arcade games, speed is a must.

There are 17 files in the **PING** folder on your *GS+* disk. Fourteen of these files are source files, 1 is a macro file, 1 is a command file and the last is the actual PING executable file that you run to play the game. These files are shown in Figure 1, along with a brief description.

[**Editor's note:** Merlin stores all text files with the hi-bit of each character set. If you attempt to read these source files with anything other than the Merlin editor, they will appear to be full of "garbage." To view these files with EGOed, open the file, press Command-A to select all the text in the file, and then select "Clear Hi-Bits" from the Edit menu.]

When writing a game, as in all GS applications, the first thing you need to do is to start the proper tools. I will assume you have some basic knowledge of assembly and the Toolbox, otherwise this article would overrun the magazine (for the next 6 issues)! For an arcade game, we only need to start a few basic tools—to manage memory and handle the sound. The source file **PINGSTART.S** does the following:

`SetRes` - This routine sets the control-Y vector to point to a ProDOS quit. Why do this? Well, if your program crashes, all you need to do is to press control-Y and hit return and a ProDOS quit is executed (saving you from having to reboot). I *highly* recommend you include this feature in your programs.

Next it starts the tools we need. Those tools are: The Tool Locator, the Memory Manager, the Miscellaneous Tools, QuickDraw II and the Sound Tools.

The subject of tools is a long and involved one. Suffice it to say that the tool startup routine used in the `PINGSTART` segment is generic enough that it can be used in any game that does not require the desktop. If you need more information on the tools and their uses, you really should pick up copies of the *Apple IIGS Toolbox Reference: Volumes 1, 2, and 3*. There is no substitute for these books! Once you have started up the tools you need, you don't have to worry about them and can get down to the business of writing your game.

The game itself is in the source file called **PINGGAME.S**. Once you get past the definition of the variables, the first routine you come to sets the border color to black. The border color info is stored in the first nibble (half a byte) of the byte at location $E0C034 (the "$" signifies a hexadecimal value.) First I store the old border color in the variable `BOARDER` then set the border color to 0 (black). When the user quits, the old border color is restored. This is more of a courtesy than a necessity, but people get really teed-off when you mess with their display! Then, after initializing a couple of variables (which I will explain later), I go to the `LOADTITLE` subroutine which loads the title screen. In order to understand how this works you must first get a feel for how the GS displays SHR graphics.

FIGURE 1 - THE FILES THAT MAKE UP PING

PINGSTART.S	Starts the tools.
PINGGAME.S	The actual game program, calls all the other routines.
HIGHLOW.S	Lookup table for the Super High Resolution (SHR) Screen.
INSTR.S	Contains the hex data for the instruction screen.
BLUEDRAW.S	Draws the Blue Paddle.
GREENDRAW.S	Draws the Green Paddle.
BALLDRAW.S	Draws the ball.
MAKEPING.S	Command file that assembles and links the source files.
PINGPIG.S	Hex data for the pictures of the paddles and the ball.
TITLE.S	Hex data for the title screen.
SCORE.S	Hex data for the score display at the bottom of the screen.
NUMBERS.S	Hex data for the pictures of the numbers.
NUMDRAW.S	Draws the numbers.
PAUSEDRAW.S	Draws the word "Pause."
PAUSEPIC.S	Hex data for the picture of the word pause.
PING.MACS.S	Macro file for Ping.
PING.SYS16	The executable file. Run from the Finder or other program launcher.

GS graphics are *bit-mapped* into the section of memory starting at $E12000. What this means is if the SHR screen is active and you write a value into bank $E1, anywhere between $E12000 and $E19CFF, each nibble (half a byte) shows up as a dot on the screen. Groups of bytes written to this section of memory are therefore displayed as a picture. Each possible value, from 0 to 15 ($0-$F), represents a different color. The colors represented by these numbers are determined by the *palettes* which start at location $E19E00. There are 20 such palettes, each consisting of 32 bytes. Each 2-byte word in a palette defines a different color. This gives us 16 colors per palette (32/2 = 16). Each pallete word consist of 3 numbers (one nibble of each palette word is unused), each of which occupies a nibble and represents a different value for one of the 3 color guns (Red, Green and Blue). These values can be anywhere from 0 to 15 for each color, giving a total of 4,096 possible colors (16 x 16 x 16 = 4096). The numbers you place in bank $E1 are used as references to the 16 color entries in a palette. You can have up to 16 different palettes, each of which can be assigned to any of the 200 horizontal lines of the SHR screen. The 200 bytes starting at location $E19D00 determine the resolution (320 or 640) and the palette (0-15) for each line to use. Example: Placing a zero at memory location $E19D00 tells the computer that the first line of the SHR screen uses the first color palette and is set for 320-mode.

Now that you have a working knowledge of the SHR screen you are ready to find out how I load and display my title screen. The first thing I do in the subroutine LOADSCREEN is to place a $0000 in memory location $E19E0A. That's the location in palette 0 that holds the information for screen color 5. Since I am loading a zero there, it sets all three color guns equal to zero for color number 5. Therefore, 5 is black for the moment. Since my title screen (the text that says "Daniel PING Webster", etc.) is all in one color, and that color is 5, nothing is visible when I load it to the screen. It is loaded by being read from the data segment TITLE in the following indirect indexing routine:

```
LDA TITLE,X
STAL $E15160,X
```

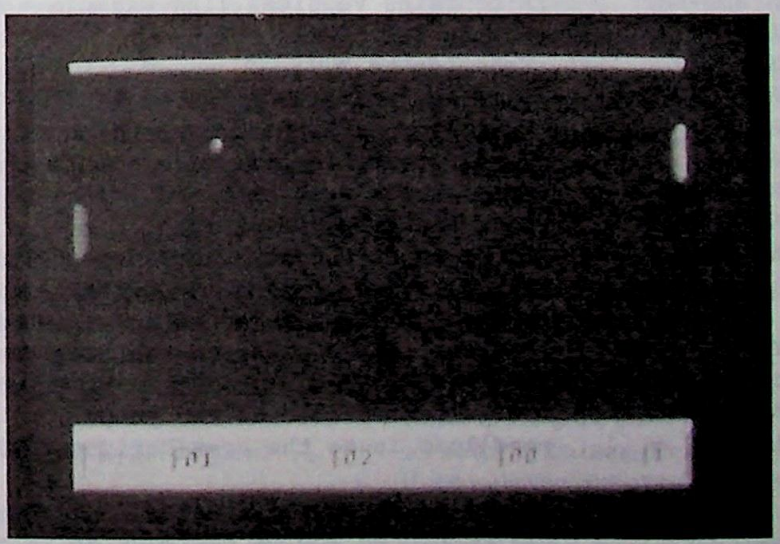
As you can see, the destination memory location is in the middle of the SHR screen, and each time you increment X to load the next byte it is stored in the next location of the screen. Once all 3984 bytes are read onto the screen, I jump to the lower half of the screen and load bytes 3984 through 5760. This contains the copyright information. Now we are ready to do the palette manipulation required to make the title appear to fade in.

The subroutine FADIN does the first part of the palette manipulation. What it does is to slowly increase the blue color component of palette 0, color 5, from $0 (off) to $F (maximum intensity). This gradually changes color 5 from black to blue, and the graphic seems to fade in. The vertical blanking routine is used to keep the image from flickering as it fades in. After that, I jump to the DELAY subroutine, fade the image out, and return. The fade out is accomplished in the routine FADOUT, which does the same thing as FADIN except it slowly decreases the blue color component of the fifth colour from $F (pure blue) to $0 (blue component off). The delay routine is a nested loop; i.e. one loop within another. What it does is decrement the Y register from $FFFF to zero, and then check to see if the X register is zero. If the X register is greater than zero it decrements it by one and runs through the Y decrement again. Before I jump to the DELAY subroutine, I load the X register with the number of times I want to run through the delay—in this case, 9.

Next we have to remove the title screen. This is done in the subroutine called CLEAR. Even though we can no longer see the title, it is still there. In order to get rid of it, CLEAR just writes a zero into every memory location from $E12000 to $E19D00, clearing the SHR display area. It does not affect the palettes.

The instructions are loaded in the same way as the title, except that the starting location to which each line is loaded in increased by $A00 (10 lines) to provide the spacing between each line. Each line is 960 bytes long, so after writing each 960 bytes to memory I just add $A00 to the location where I did the last write. Instead of jumping to a delay routine between FADIN and FADOUT, I jump to a routine which waits for a keypress, called ANYKEY. It reads the memory location $00C000, which will have a value greater than $80 when a key is pressed. When the value goes above $80 I just return and fade the image out. This allows the user to take as much time as required to read the instructions, then just press a key when done.

Again we need to clear the SHR screen, but this time we must also load the color palettes that the game will use. That is done in the subroutine CLEARPAL. First



it writes a zero into every location from $E12000 through $E19D00. Next it loads the color palettes we are going to use (6 of them, 0 thru 5) starting at location $E19E000. It also loads the numbers 1 thru 4 into location $E19D00 and $E19DAA, pointing the first and last four lines of the playing area to palettes 1 thru 4. Palettes 1 thru 4 are identical to palette 0 (the game's main palette) except that for color 0 they have shades of gray instead of black. The zeros loaded into the top and bottom now point to these gray colors, and lines appear at the top and lower section of the screen. These are to be the borders for the playing area of the game. The last thing we need to do to get the playing screen ready is to load the picture which holds the stats. This is done in the routine called LOADSCORE, and is also similar to the routine that loaded the title screen. The only difference is that this routine also loads the number 5 into the memory location starting at $E19DAE for 26 bytes, setting the area occupied by the bottom display to point to the sixth color palette, which is a gray scale palette.

The next thing I do is write the number zero to each player's score. This number is not really a number, it is a graphic image. The routine used to draw the numbers operates in the same way as the routine to draw the ball and paddles. My basic drawing routine uses an indirect indexing scheme similar to the one we used to load the previous graphics. The file **NUMDRAW.S** contains 3 basic subroutines. The first is called NINIT. What this does is takes the graphic's vertical location (the horizontal line it starts at) and adds 8 to that to get the depth (the line number at which we stop drawing), and stores it as BDEPTH. The picture is drawn horizontally, completely writing the bytes to each horizontal line before moving down to the next vertical one. It is, therefore, drawn top to bottom. In order for the routine to know when to stop drawing, it needs to know the line number which is the last line of the picture. This routine gives us that.

The second routine is called NUMBYTE. This takes the number of the graphic you are to display (in this case, the "picture" of the numbers 0 thru 9), looks up the starting address of that graphic, and loads the graphic data (bytes) into a buffer from which we can read it onto the screen. The variable which holds the pointer to the graphic is called NO. If NO is equal to zero, we get the starting location of the first graphic, if it's equal to 1 we take the second, etc. We load this starting location into the zero page variable LOW. The buffer that is to hold the graphic is called NTEMP, and it is filled in the following indirect indexing routine:

```
;Load from graphics start
   LDA (LOW),Y
   STA NTEMP,Y   ;to NTEMP
```

The Y register is increased until it is equal to the number of bytes in the graphic, in this case 36. At that point all 36 bytes of the number graphic have been copied into NTEMP.

Now that we have the starting and ending locations of the graphic, and have loaded the picture into our buffer, it is time to write it to the SHR screen. The routine NDRAW does this. The first thing it does is load the Y register with the horizontal location of the picture. The next thing it does is it takes the vertical location and performs a lookup to find the starting address of the vertical line. This is needed because line 0 starts at $2000 and line 1 at $20A0, etc. Instead of calculating the start of each line by "multiplying" the vertical line number by $A0 (160 bytes = 320 nibbles = the number of pixels per line) it is easier to create a look up table which holds the starting location of each line. Then we simply load the line number (say 10) then read the 10th byte in the table which has the starting location for line number 10! Since each line number is 2 bytes long, we actually have to read the 20th byte (10 x 2 = 20). There is an even easier way to do this. Apple has already thought of including this

FIGURE 2 - USING THE SHR LOOKUP TABLE

```
LDY NUMHORIZ      ;Y register contains Horizontal for example = 40
LDA NUMVERT       ;A contains Vertical line example = 10
ASL               ;Double it   (20)
TAX               ;Transfer to X
LDA HIGHLOW,X     ;Load the starting location of line 10 from lookup table
XBA               ;My lookup table is backwards (sorry!)
STA LOW           ;Store it to use in indirect index routine
LDX XCOUNT        ;Keeps track of byte number of shape
LDA NTEMP,X       ;Load the Xth byte of shape
STA [LOW],Y       ;Store it at the starting location plus the horizontal, since it's line
                  ;10 the start address is $E12640 ($E12000+(10x160)) plus the Horizontal
                  ;offset of 40 ($28) so it starts at $E12668.
INY               ;Increment Y to point to the next Horizontal location
INY
LDA NTEMP+2,X     ;Get the next word of the shape
STA [LOW],Y       ;and load it to the next horizontal location
.
.                 ;Since the shape is 2 words long the first Horizontal line
.                 ;is completed and we can move on to the next one
```

```
VAR
 JSR MOUSE       ;Reads the Vertical location of the mouse
 JSR JOYSTICK    ;Reads the Vertical location of the joystick
 JSR BLUEDRAW    ;Draws the blue paddle at the new Mouse location
 JSR GREENDRAW   ;Draws the green paddle at the new Joystick location
 JSR READKEY     ;Checks to see if any keys have been pressed
 JSR WAIT        ;Delay routine
 LDA BALLON      ;Check to see if the ball is in play
 BNE VAR1        ;If its not it skips the ball move routine
 JSR BALLMOVE    ;Checks to see if the ball has hit a wall or paddle and moves
                 ;it appropriately
 LDA BMISS       ;Has the blue paddle missed the ball (BMISS = 1)
 BNE BRESET      ;If it has look to see if the mouse button has been pressed
                 ;to send the ball back into play
 LDA GMISS       ;Has the green paddle missed the ball? (GMISS = 1)
 BNE GRESET      ;If it has look to see if the Joystick button has been
                 ;pressed to send the ball back into play
 JSR BALLDRAW    ;Draws the ball at it's new location
 JMP VAR         ;Do it again
VAR1
 JSR MOUSEWAIT   ;Wait for the mouse button to start the ball
 JSR JOYSTICK    ;Wait for the Joystick button to start the ball
 JMP VAR         ;Do it again
```

FIGURE 3 - THE MAIN LOOP

information in the GS, and in the QuickDraw II tool set is a function called _GetAddress which contains the address of a built-in look up table. Sometimes I believe that Apple has thought of everything, but then I remember that the GS still runs at 2.8 Mhz. Faced with this limitation, I decided to use my own look up table. Once the starting byte of the line is found, we simply read the first byte of our shape into that location *indexed* by the horizontal position, now stored in the Y register. This is illustrated in Figure 2 on page 10.

Using the lookup table saves us a lot of math. Just continue this routine for the number of lines in the graphic. You increment NUMVERT (the vertical line number) until it is equal to BDEPTH, then the graphic is complete. Then you set NUMVERT back to what it was when you started. That's all there is to drawing a bit-mapped shape on the SHR screen. Now, back to the game segment.

Once we have drawn the numbers representing the score, we draw the ones for the speed (SHOWSPEED). This uses the same drawing routine, just loading a different horizontal location into NUMHORIZ and a different number into the variable NO which points to the graphic. Finally, we display the mode (1 or 2 player) in the subroutine PLAY. Again, we use the same number drawing routine, we just change the horizontal location, and the pointer to the graphic (number shape) we want to load.

The next routine, CLAMP, is the one which clamps the mouse (sets the minimum and maximum values the mouse can return). This is done using the Toolbox call _ClampMouse. Since the onscreen paddles only move up and down, we set the minimum and maximum horizontal values to the same value: 4. We then set the minimum vertical value to 0 (the top of our playing area) and the maximum value to 150 (the bottom of the playing area). The playing area actually runs from lines 0 to 174, but the paddle is 24 lines high, and therefore the top of it can't go beyond 150.

Now we are ready to start the game!

The entire game is contained in the loop called VAR. This is shown in Figure 3. BRESET and GRESET both do the same thing. First, each jumps to a subroutine called CLEARBALL which erases the ball from the screen. It does this in the same way we draw any shape, except that it loads all zeros into the shape table and writes them to the last location of the ball. Then it resets the ball's horizontal location to one that is across the screen from the player to receive the ball (BALLHORIZ), and loads a vertical location equal to that of the opponents paddle (BALLVERT). You could actually use any horizontal or vertical locations to "serve" from. Then we jump from the reset routine back to VAR and wait for a button press to set the ball into play. Here's a run-down on how each of these routines works:

MOUSE - Uses the Toolbox call _ReadMouse to get the horizontal and vertical locations of the mouse.

JOYSTICK - Does 1 of 2 things. First it checks to see if the green paddle is to be controlled by the computer or the joystick. If the variable COMPUTER is set to zero then the green paddle is computer controlled, and it jumps to the subroutine

CONTROL (detailed below). Otherwise, it reads the joystick. To read the joystick, I use the Toolbox call _FWEntry, which allows you to use classic Apple II subroutines from the Toolbox environment. The subroutine I call is the standard paddle read subroutine located at $00FB1E. Because the joystick is an analog device, unlike the mouse, I read it *every other time* I go through the loop, to allow the counters time to reset. If this is not done, the second paddle is difficult to control since returned values fluctuate so much. Once I get the joystick's vertical location, I divide it by 1.7 to set the minimum and maximum equal to the size of the playing area. Division by fractions in assembly is a lot like jump-starting a car with "D" cell batteries. It requires extra work and a little patience. In the subroutine DIVISION, I first load the returned vertical component (GREENVERT) into my all-purpose variable HOLD. I then place that value into the X register. Next I multiply that value by 10 by adding 10 to the accumulator each time I decrement X. I then divide this by 17 by incrementing the X register (from 0) each time I subtract 17 from the accumulator (whew!). Finally, I get a value between 0 (0/1.7) and 150 (256/1.7). The value 150 is the farthest vertical point at which the paddle is drawn.

CONTROL - Called when in the one-player mode. This routine reads the ball's vertical position, and checks to see if it is above or below the center of the green paddle. If it's above, it decrements the green paddle's vertical location (moving the paddle up), and if it's below, it increments the paddles vertical location (moving the paddle down). I can, if there's interest, develop a routine to allow the player to change the computer's skill level. We'll let your letters determine that. Otherwise, you can write it yourself—it would make an interesting project.

BLUEDRAW - This subroutine is external to the main program. It does the following:

First, it checks to see if the paddle has been moved. If it has not, it skips the erase routine and just draws the paddle. If the paddle *has* moved, it loads zeros into the paddle's old location (erasing it), then draws it at its new location.

The draw and erase routines operate exactly like the routine we used for the numbers, except since the paddle is always the same shape, we do not have to use a lookup table, and can write the bytes directly from our source (**PING.PIC.S**).

GREENDRAW - Operates the same as BLUEDRAW, only with the data for the picture of the green paddle.

READKEY - Memory location $00C000 is the location in classic Apple II's that holds the "Has a key been pressed?" information. If the value at that location is greater than #$80, a key has been pressed. After that I just subtract #$80 from that value and compare it to the ASCII values of the keys I am checking for.

WAIT - This routine just counts down from the value of the variable PAUSE to zero. The larger the value, the greater the delay.

BALLMOVE - Which calls: CHECKBLUE and CHECKGREEN.

These first of these three routines first checks the variable BALLLR to see if the ball is moving left or right (0 = left, 1 = right). If it is moving left (towards the blue paddle) it checks to see if the horizontal position of the ball is equal to or less than that of the paddle. If it is not, it decrements the horizontal position "moving" the ball left. If it is, it checks to see if the vertical location of the ball falls between the top and bottom vertical locations of the blue paddle. If it does not, the blue paddle has "missed" the ball, and we load 1 into BMISS, BALLON and BSTART. If it does, we first call the routine that makes the bounce sound (DoNoise) then we check where on the paddle the ball has hit. This is to determine the angle with which the ball "leaves" the paddle. If the ball hits the topmost portion of the paddle, we load a 4 into BALLUD. When the ball move routine sees a 4 in BALLUD, it decrements the ball's vertical position twice for each single horizontal move. This means the ball has a greater angle. Since the IIGS defines (0,0) at the top left of the screen, you must *decrement* the vertical to go "up". If the ball "hits" the paddle more toward the center, we load a 3 into BALLUD. The 3 indicates a single decrement for each horizontal move. If it hits the center area of the paddle, we load a 2. This tells the ball move routine not to increment or decrement the ball for each horizontal move, and the ball travels straight. The same is done for the lower and lowest sections of the paddle, except I use a 0 to indicate a single vertical increment for each horizontal move, and 1 to indicate two.

All of the possible actions for this are summarized below in Figure 4.

The ball's horizontal location is always either decremented or incremented while it is in play. The only other thing we need to check is if the ball has hit the top or bottom of the playing area. First, load the variable BALLUD. If it is equal to 2 then we don't have to check for a hit, because it

FIGURE 4 - FIGURING THE RETURN ANGLE

IF THE BALL HITS THE PADDLE	MOVE THE BALL
0 to 5 pixels from the top	Up 2 positions vertically for each single horizontal move.
5 to 10 pixels from the top	Up 1 position vertically for each single horizontal move.
10 to 15 pixels from the top	Straight across the screen.
15 to 20 pixels from the top	Down 1 position vertically for each single horizontal move.
20 to 25 pixels from the top	Down 2 positions vertically for each single horizontal move.

is moving straight across and therefore, can't hit the top or bottom walls. If it is equal to either 4 or 3, then it is moving up, so we check to see if the BALLVERT is equal to 2 (top of the playing area). If it is, we jump to the sound routine, then we load the *opposite* of the BALLUD variable into BALLUD. That means if BALLUD is 4 (fast up), we change it to a 1 (fast down), and if it is 3 (slow up), we change it to 0 (slow down). The next time the routine is called to move the ball, it will move the ball in the new direction! If the ball is moving down (incrementing the vertical), we check to see if BALLVERT is equal to or greater than 163 (bottom of playing area minus the depth of the ball) then do basically the same thing as we did for the screen top.

BALLDRAW - BALLDRAW uses the same drawing routine we've seen before, except that it never erases the ball. Since the ball is never moved more than 2 pixels in any direction, we can just draw the ball with a "shadow" around it. Then when the ball is moved, it overdraws the old ball, leaving only the shadow visible. Since the shadow is the same color as the background, the old ball is completely covered. This is shown in Figure 5, at right.

As you can see, when the ball is moved and redrawn, the new ball has erased the old one!

There is really very little else to tell. The source code details out the variables and what they each represent. The basics that this game teaches can be used as building blocks to write more sophisticated arcade games. Remember, this program is not public domain! Steve paid good money for this! (OK, maybe not *that* good, but I'm easy.) The bottom line is that this game is not for uploading to your favorite BBS or distributing at a user group meeting. Even if you modify the code it is still the property of *GS+*. However, the techniques are all yours and I hope to see them put to good use. The future of the IIGS is in our hands, and as far as I'm concerned, games are the way to go. Well, maybe not, but they don't hurt. Until next time, good luck with your programming! Try not to flood the market with games... the competition is tough enough on me as it is.

FIGURE 5 - COVERING THE OLD BALL WITH THE NEW BALL

"o" and "o" represent the background color and "x" and "x" represent the ball color:

Ball drawn at location (X,Y).	Ball redrawn at location (X+2, Y+2)
oooooooooo	oooooooooo
oooooooooo	ooooooooooo
ooxxxxxxoo	ooooooooooo
ooxxxxxxoo	ooooxxxxxxoo
ooxxxxxxoo	ooooxxxxxxoo
oooooooooo	ooooxxxxxxoo
oooooooooo	ooooooooooo
	oooooooooo

SHUFFLE

By Josef W. Wankerl

It is Apple's windowing system that makes the Apple IIGS a powerful, but easy-to-use, computer. Sometimes, however, it can be a bit frustrating when one window is hidden behind another. Having to take your hands off of the keyboard to use the mouse to switch windows can be a royal pain, not to mention a waste of time. Shuffle takes care of this problem.

USING SHUFFLE

Shuffle is a *Permanent Initialization File* (PIF) which lets you cycle through the windows on the screen with the press of a few keys. To install Shuffle, copy the file **Shuffle** from inside the **Shuffle** folder on your *GS+* disk to the ***:System:System.Setup:** folder on your boot disk and then reboot your computer. Shuffle will then be installed. There is also a special Finder icon file to help the Shuffle file stand out. For more information on it, see "Icons" on page 21.

To see Shuffle in action, open up a couple of windows on the desktop. Then, *press* and *hold* the Command (also known as Open-Apple), option, and shift keys *at the same time*. It may take a moment for the computer to recognize the keys, so be sure to hold all of the keys down until you see the next window on the screen activate. If you continue to hold the keys down, Shuffle will begin to cycle through all of the windows that you have open. When the window you want to work with becomes active, release the keys. That's all there is to it.

WHAT'S A "PIF?"

Well, like I said before, PIF stands for *Permanent Initialization File*. As part of its startup process, the IIGS loads and runs all active PIFs that it finds in the ***:System:System.Setup:** folder. That's the "Initalization" part. After a PIF is loaded and executed, it is left in memory. That's the "Permanent" part. Since PIF files "hang around" in memory, they can be called again and again to perform their particular task.

IT ONLY TOOK 10 MINUTES

Shuffle is a really short program (most PIFs are) that is written in ORCA/M. There are two parts to Shuffle, the installation and the background task. The easiest part of Shuffle was the installation code (Figure 1). All it does is install the second part of Shuffle, the background task! And, as you can see, all it took was an `_AddToRunQ` Toolbox call.

The *RunQ* is much like the Heartbeat interrupt manager. However, unlike the Heartbeat manager, RunQ tasks are only called when the system is free to make Toolbox calls. Since the Toolbox may not be "available" when you first press the keys, this is why you have to press and hold the keys to activate Shuffle. A Heartbeat task, on the other hand, can be called at any time, even when the system is not able to make Toolbox calls. Since Shuffle uses the Toolbox, the RunQ was the way to go. The RunQ is fully documented in the *Apple IIGS Toolbox Reference: Volume 3*, on pages 29-3 through 29-5.

The Shuffle RunQ header (shown in Figure 2) defines a *period* of 0. This means that once the Shuffle task is installed, it should be called by the system as often as possible. Whenever it is called, Shuffle first checks the keyboard modifier register to see if the specified keys are held down and, if they are, it then tries to shuffle the windows. First, a check is made to see if there is actually a window open on the screen. If there is not, then no action is performed. Next, the front window's frame is checked to make sure it can be sent behind all the others. Alert windows and dialog boxes must *not* be shuffled! If the window can be moved, it is then sent behind all other windows with the `_SendBehind` Toolbox call. The remainder of the Shuffle source code is in the **Shuffle** folder on your *GS+* disk.

Although this program seems small and simple, it is one of the most useful utilities I've used. It makes the Finder and AppleWorks GS much more manageable! As usual, if you find a problem with this program, fill out the problem form supplied on your *GS+* disk and let us know about it.

FIGURE 1 - INSTALLING THE RUNQ TASK

```
ShuffleInit     START
                PushLong #RunQHeader     ;Point to RunQ item header
                _AddToRunQ               ;Add this routine to the RunQ
                rtl                      ;Return to caller
                END
```

FIGURE 2 - DEFINITION OF A RUNQ HEADER

```
RunQHeader      START
                ds 4            ;Space for link to next RunQ item.
Period          dc i2'$0000'    ;Call as often as possible. (Expressed in 1/60 seconds.)
                dc i2'$A55A'    ;RunQ integrity signature.  Always the same.
                ds 4            ;Space for system to save last execution time.
```

BATTERY BRAIN

By Josef W. Wankerl

WHAT'S A BATTERY BRAIN?

When the IIGS was introduced 4 years ago, having it remember configuration information in an area of battery-backed RAM (BRAM) was a new concept for Apple II computing. Since then, Apple has shoved more and more information into the BRAM area, and it has become quite a chore to reset everything after a battery goes dead. In an effort to make this loathesome task easier, we proudly present, Battery Brain!

Battery Brain is a Control Panel Device (CDev) that allows you to save to disk, and later restore, your BRAM configuration. Aside from being useful as insurance against a dead battery, these BRAM configuration files are also useful if you have a room full of inquisitive students or a IIGS full of expansion cards that require your system to be set up in a particular fashion. For example, I have a BRAM file that correctly configures my IIGS so that I may use my Audio Animator and I have another that I load when I want to use my modem. A system reboot is required to put all of the parameters in each BRAM file into effect, but it's much easier than going through the Control Panel, changing each of its options one-by-one, rebooting, and then realizing that you've forgotten to change something! Which, of course, means *yet another* reboot!

HOW DO I USE IT?

To install Battery Brain, use the Finder to copy the file **BatteryBrain** from the **BatteryBrain** folder of your *GS+* disk into the ***:System:CDevs:** folder of your boot disk. Battery Brain will then be available from the Control Panel NDA. You might also want to copy the **Brain.Icons** file into your ***:Icons:** folder so the Finder will recognize BRAM Configuration files and display a custom icon for them.

When you select Battery Brain from the Control Panel, you will see only two options: "Load..." and "Save...." The Load option lets you select a BRAM configuration file from disk which will replace your current BRAM settings. A system reboot is required for all of the changes to take effect. It should be noted that BRAM configurations on a ROM 01 IIGS are different than those for a ROM 03 IIGS. If you try to load a configuration file that was saved from a IIGS with a different ROM, a dialog box is presented informing you of the situation and the current BRAM configuration is left alone. The Save option lets you save your current BRAM configuration to a file.

HOW DOES IT WORK?

Battery Brain was written entirely in ORCA/C v1.1. Text Edit and its associated tool sets are started up and shut down only for the Help button dialog. Standard File is started up and shut down as needed. This program follows the basic CDev guidelines. If you need further reference as to how CDevs work in general, see "All About Control Panel Devices" on page 7 of *GS+* V1.N4, or get yourself a copy of Apple's CDev technical note: *FTN.C7.XXXX*.

ANY TRICKS?

The most interesting algorithm for this program is the saving of the pathnames from the Standard File tool calls. Basically what happens is that prefix 8 is first retrieved and squirreled away. Then the prefix saved in Battery Brain's resource fork is read in and prefix 8 is set to that prefix. Standard File is then called. When the user finishes with the Standard File dialog, prefix 8 is retrieved again and written out to the resource fork of the CDev. The old prefix 8 is then restored.

Why go to all this trouble? Well, when we were designing Battery Brain, we could not agree on whether or not to force the BRAM configuration files to be kept in the ***:System:System.Setup:** folder or to let the user decide where to keep them. ***:System:System.Setup:** is the most *logical* place to keep these files, but it might not be the most practical place for floppy disk users.

So, the best compromise seemed to be letting the user decide and having Battery Brain remember where the files are. This has the added advantage that, if you are working in another directory, using Battery Brain won't force you to crawl back through all those folders to get where you originally were. It is a little extra trouble for the programmer, but it really is worth it from the users' point of view.

There are two inline assembly language sections to Battery Brain. The first one takes the BRAM parameters from the BRAM file and changes the Control Panel settings accordingly. The routine used for this task (`ToBRamSetup`) is documented on page 273 of the *Apple IIGS Firmware Reference* by Addison-Wesley.

The second assembly language routine is a function that returns the ROM version of the IIGS that Battery Brain is running on. I had planned to convert it to C for the final release but decided that it would be a good demonstration of how to make tool calls and save the results from an inline assembly language section. The source code for both routines is commented completely.

During the final days of testing, I found what may be a nasty bug in ORCA/C 1.1. The "optimize" `#pragma` doesn't seem to work right. When I compiled Battery Brain with all optimizations off, things worked as they should. When I compiled with all optimizations on, Battery Brain would generate GS/OS errors when loading a BRAM configuration file. I have tracked it down to the `ReadGS()` call giving the error because the buffer area is not set right. It seems that ORCA/C somehow skipped the dereferencing of the buffer memory handle or it dereferenced it incorrectly.

We hope you like Battery Brain. It's already made life at our tiny office easier, and we hope it will do the same for you. If you have any questions about Battery Brain, or suggestions for improvements, please don't hesitate to contact us here at *GS+*.

RANDOM IIGS PROGRAMMING NOTES

By Josef W. Wankerl and
Steven W. Disbrow

This issue, we have updates to Transfusion (from *GS+* V1.N6) and EGOed (from *GS+* V1.N2). The updated programs (complete with source code) are on your *GS+* disk. (For more information on how to install these programs, see "How To Use The GS+ Disk" on page 20.) This article describes the changes to these programs and any programming tricks that were used in creating the changes. For those of you that are just joining us, let's take a second to explain exactly what EGOed and Transfusion are.

EGOed is a New Desk Accessory (NDA) text editor. To use it, you *must* use IIGS System Software v5.0.2 or later. EGOed allows you to edit and print plain text, Teach Text, and AppleWorks Classic Word Processor files from inside any desktop program that supports NDAs.

Transfusion is an NDA terminal program. It allows you to go online from within any desktop program that supports NDAs. To use it, you *must* use IIGS System Software v5.0.2 or later and an *external* modem. Transfusion does not yet support internal modems.

EGOed UPDATE

This EGOed update, v1.31, makes things easier for those of you without a hard disk, and adds a useful item to the File menu. First, lets look at the new File item.

CLOSE IT UP

The newest item added to the EGOed File menu is, *Close*. When you select this item (or press Command-W), EGOed will check to see if any changes have been made to the current edit file. If the file has not been changed since you last saved it, the EGOed window will close. If changes *were* made, EGOed will ask you if you want to save the file. If you click the "No" button, EGOed will close without saving the changes. If you click "Yes" (or press the return key), EGOed will save the changes and then close itself. If you click the "Cancel" button, you will be returned to the file you were editing.

Enacting a "Close" feature for NDAs that use resources is a little bit tricky. In a normal NDA, you would simply save anything that needs to be saved and pass a pointer to your NDA window to the Desk Manager's `CloseNDAbyWinPtr()` tool. With resources in the picture, there is another task that needs to be taken care of before you can actually close the NDA. Basically, after you get the user to confirm the fact that she wants to close the NDA (and you have saved anything that needs to be saved), you *must* ensure that current resource file is the resource file belonging to the application. This is accomplished using the Resource Manager call `SetCurResourceApp()`. You can then close your NDA by calling `CloseNDAbyWinPtr()`.

THE FLOPPY SHUFFLE

In case you haven't noticed, those of us here at *GS+* use hard drives and, we are constantly recommending that *everyone* should have a hard drive. However, for those of you that can't write it off as a business expense, it can be a bit difficult to justify the expense with either your wallet or, more importantly, your spouse.

Because of our use of hard drives, we sometimes don't realize the difficulties that users have with running some of our programs from floppy disks. For some of you, EGOed has been an extra special problem because it expects to be run from the ***:System:Desk.Accs:** folder, and it expects its filename to be EGOED. This meant that users with floppy-based systems could not use EGOed along with any of the desk accessory loaders that exist in the public domain. (These desk accessory loaders allow you to load and use desk accessories after you start up your computer. Thus, you can start up your computer with a minimum system disk and then load and use only the desk accessories that you need when you need them.) Well, thanks to the suggestions of a few brave readers, (Jeff Hartkopf and Grant Delaney were the first to send the solution along with their suggestions, so they get the official tip o' the hat. Thanks fellas!) this is no longer a problem. EGOed now uses the System Loader call, `LGetPathName2()` to obtain the pathname of the file that EGOed was loaded from. So, you can now use EGOed with your favorite desk accessory loader. I would love to tell you exactly how `LGetPathName2()` works, but we are running out of space and it's documented in the source code on your *GS+* disk, so take a look there. It's in the `DAInit()` procedure.

Along with this new flexibility for floppy-based systems comes a whole bunch of dangers. What if the disk you loaded EGOed from is not online when you open EGOed? What if the system disk containing all of your fonts is not online when you open a Teach Text file? What if... Yikes!

Fortunately, GS/OS provides an incredibly easy solution. GS/OS keeps track of a number of *global system preferences* that can easily be checked or set by using the GS/OS calls, `GetSysPrefs()` and `SetSysPrefs()`. These system preferences are kept in a single word (two bytes). Each preference is a "do this" or "don't do this" type affair, that is represented by a single bit in the preferences word. At this point in time, Apple has defined only one system preference that can be set, but it just happens to be exactly what we need!

This preference, which is defined in bit 15 of the system preferences word, tells the system whether or not to display a *volume mount dialog* when GS/OS can't find a disk that it is looking for. By setting this preference bit to 1 whenever EGOed takes control, we ensure that the system will ask for any disks that it needs when it needs them. When EGOed turns control of the system back over to the application, we reset the system preferences to what they were when

EGOed was called. (This is just common courtesy.) It's a very simple thing to do, as one look at the EGOed source code will prove.

One warning though, when you see that prompt to insert a disk... **do it!** There is a "Cancel" button in the systems volume mount dialog, and it is possible to click that button and have things continue as you would expect. But, on a few occasions, I have clicked "Cancel" and had the system lock up. As always, it's better to be safe than sorry.

For more information on how these changes were done, refer to the EGOed source code. To make it easier to find the changes in this EGOed update, I have placed a comment line containing the string "v1.31" before each change. Simply use the TML Pascal II editor to search for this string and you will be able to quickly find the changes. If you do not get the *GS+* disk, you can look in the *GS/OS Reference: Volumes 1 and 2* (Beta draft), or *Exploring Apple GS/OS and ProDOS 8* by Gary B. Little for more information.

AND FINALLY...

Several readers have reported problems with getting EGOed v1.3 (from last issue) to load Teach Text files. Basically, when they selected Open from the File menu, there were no Teach Text files shown in the directory listing of the V1.N6 *GS+* disk. After scratching my head for a couple of days, I received a letter from M. Masaki of Ojai, CA. Mr. Masaki reported the exact same problem, but he also supplied the missing piece of the puzzle: He was using System Software v5.0, *not* v5.0.2. I quickly scrounged up a copy of the old System Software and put it to the test... sure enough, Teach Text files simply would not show up in the Open dialog! Extremely strange. Another problem is that when you try to save a file as Teach Text under System Software v5.0, you would get an "Invalid Pathname Syntax" error and the file would not be saved.

The solution to this problem is simple enough, make sure you are using System Software v5.0.2. If you don't yet have a copy, contact your local Apple dealer. If you don't have a "local Apple dealer," it is also available from America Online, GEnie and several of the other popular online services. As an absolute last resort, contact us here at *GS+* and we will try to help you get a copy.

TRANSFUSION UPDATE

Version 1.0.1 of Transfusion is exactly the same as version 1.0 except that whenever the Choose Font dialog box is opened, no characters will be received from the buffer because that seemed to cause problems with characters being sent into oblivion and sometimes the machine crashed.

The changes from the programmer's view are just as subtle... basically, a new transferring state, *Idle*, was added. When the transfer state is Idle, no characters are received from the buffer. Whenever `ChooseFont()` is called, the transfer state is first put into Idle mode and then when the call is done the mode is restored. That's all there is to it. Hopefully for the next issue of *GS+* I'll have my 1K and 4K XModem routines fully debugged and ready for distribution.

ALL YOU NEED

The EGOed update, complete with TML Pascal II source code, is on your *GS+* disk in the folder, **EGOed.1.31**. The Transfusion update, complete with ORCA/C source code, is in the folder **Transfusion.101**. For more information on installing these updates on your system disk, refer to "How To Use The GS+ Disk" on page 20.

Remember, we want to know what you think! If you find a bug in one of our programs, be sure to fill out the problem form on your *GS+* disk and send it to us. And, if you have a suggestion for an enhancement to any of our programs, send that in too!

GS+ CLASSIFIEDS

1/2 PRICE SOFTWARE

The Hunt For Red October
(novel not included) - $ 20
Silent Service - $20
Sub Battle - $5
Where in the U.S.A.
is Carmen Sandiego? - $22.50
Questmaster - $20
Neuromancer - $20

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

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 3) of *GS+* is December 1, 1990. 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 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: _____

THE MOLEHILL

By Josef W. Wankerl

The Finder is a wonderful utility program and launcher, but it has its limits. The Finder can only launch files with type SYS or S16. I have often become frustrated because I couldn't launch BIN or EXE files from the Finder. Also, I got really fed up with people complaining, "the Finder can't launch my EXE file and this other super marvelous wonderful fantastic program launcher can!" so I set out to conquer this. The Finder is perhaps the best application yet written for the Apple IIGS, and it's being taken for granted.

There are good reasons why the Finder does not launch BIN and EXE files. Historically, BIN files were either programs or a collection of data. Since you can't tell if the file is a program or not except by launching it and seeing if the machine crashes, the Finder doesn't allow you to launch BIN files and inadvertently put your computer in a state where you may not be able to recover except with a reboot. However, EXE files are always programs. The Finder doesn't launch those type of programs because most of them require some support from the launching application which the Finder wasn't originally designed to give. The Finder is actually not capable of launching EXE files.

BIN

To force the Finder to launch BIN files, all that is needed is to set up a custom icon for the BIN filetype that sets the name of the application to launch to ***:BASIC.Launcher**. The Finder assumes that all files (other than SYS and S16 files) are data files and need a host application to use them. When you set the application name for an icon and open a file of that particular type, the Finder looks in the application name field and if it finds a name, it launches the host application and tells it which data file to use. The BASIC.Launcher program starts ProDOS 8 and loads in BASIC.System. Then it loads and runs the file that the Finder gives it. The file is typically a BASIC program, but BIN files are also accepted.

In the **Icons** folder of your *GS+* disk are two icon files that I have provided to demonstrate launching BIN files with the Finder. The file **BIN.Icons** contains the icon I have designed to launch any and all BIN files. You can also constrain the launching of BIN files to those that have a certain filename. The **BIN.Smpl.Icons** file is a demonstration of an icon designed to launch only a specific program, **BIN.Sample**. This will allow the launching of that one BIN file but it will not allow the launching of all BIN files that could possibly crash your computer. The only way to launch all BIN files is to have the **BIN.Icons** file in the Icons directory of your startup disk. I have provided two sample BIN files (actually they're the same file, they just have different names) in the **EXE.Launcher** folder, on your *GS+* disk. The **BIN.Sample** file can be launched from the Finder after the **BIN.Smpl.Icons** file has been placed in an **Icons** folder. Both the **Generic.BIN** and **BIN.Sample** files can be launched from the Finder if the **BIN.Icons** file has been placed in an **Icons** folder. Icons are a bit quirky in that the last icon file read in takes precedence over the previous ones. So, if you have a custom icon, make sure you put it in your **Icons** folder *after* the generic icon or the generic icon will supersede your custom icon and your beautiful artwork won't be displayed.

EXE

Launching EXE files is a bit more difficult. I had to write a short host application for EXE files as well as making an icon. The icon file, **EXE.Icons**, is in the **Icons** folder on your *GS+* disk and contains icons for both the EXE.Launcher program as well as a generic EXE file icon. The host application for the generic EXE icon is EXE.Launcher. EXE.Launcher is written in assembly language. All it does is check the message center for a launch message and if there is one, it loads the EXE file and executes it. Any program launcher that posts messages when it launches applications will be able to take advantage of EXE.Launcher. However, as far as I know, the only program launcher versatile enough to post messages is the trusty Finder. After EXE.Launcher has done all its work, it returns back to the program that launched it (most likely the Finder). To install EXE.Launcher, copy it from the **EXE.Launcher** folder on the *GS+* disk to the **System** folder on your boot disk. Then copy the file **EXE.Icons** from the **Icons** folder on the *GS+* disk to the **Icons** folder of your boot disk.

Some EXE files will not be able to be run by EXE.Launcher, though. They require a special shell that provides for additional support. EXE.Launcher does not provide *any* support at all, it just loads the program and tells it to try to do its work. Also, some EXE files don't follow all the rules, such as checking to see if the launching application supplies additional support, and they may crash your system. The first time you launch a particular EXE file from the Finder, you should be prepared for the worst. However, if the EXE file launches correctly the first time, the probability of it crashing due to EXE.Launcher launching it are so slim it's hardly worth mentioning.

MAKING IT WORK

To be able to launch EXE files, you first have to know the format of the message the Finder posts to the Message Center as well as the requirements for EXE files. When EXE.Launcher executes, it starts up the Tool Locator so it can use the Message Center and start other tools, then it starts the Memory Manager, and finally it starts up the Text Tool Set because EXE files require it to be started and initialized. Since this article isn't about the Text Tool Set, pick up the *Apple IIGS Toolbox Reference: Volume 2* for more information about starting and initializing the Text Tool Set.

continued on page 35...

RUMORS, WISHES & BLATANT LIES

By the late Professor G. S. Gumby

TSK, TSK!
We certainly did get lots of call about the items presented in this column last issue! Apparently people read everything on this page except the title! For shame!

THE OBITUARIES
Well, the Grim Reaper has been exceptionally busy in Apple IIGS land recently! At least three major players in the IIGS market recently gave up the corporate ghost. The ones we know about are:

Virtual Realities
From what we understand, a classic case of overspending on startup costs and overestimating the market. Nice office, nice secretary, nice furniture, nice salaries, etc., but none of the nice people in IIGS land bought their nice products.

The Apple IIGS Buyer's Guide
Gosh! What a surprise! They looked so healthy! They had such a lot of, um, pages? No. Ads? No. Reviews? No. Readers? We don't know. And can you guess who was nice enough to pick up all the subscriptions that the *Buyer's Guide* left behind? *MacWorld*? No! *Byte*? No! Well then, it must have been *inCider*! **[Editor's Note:** The person that wrote this has been sacked. The *Apple IIGS Buyer's Guide* always had more pages and readers than we did, and we always bought a copy whenever we could find one. It helped us "fill in the blanks" on product information when *certain people* forgot to include it in their reviews. It was a valuable tool for us here at *GS+* and we will miss it.]

Ingenuity
Apparently, Dave Westbrook just decided he had had enough and skipped out to the Orient. Too bad, these were the only guys that seemed to be able to keep Applied Engineering on their toes. Speaking of AE, how long do you think it will be before they start running ads that say, "We told you so! Nyah, Nyah!"

AN OPEN LETTER TO JOHN SCULLEY
What? Are you kidding? This is a **family** magazine! We can't say those things here!

THE NEW AD CAMPAIGN
Last issue, we reported that Apple was on the verge of coming up with a new ad campaign featuring the Apple II. Well, just this week, our publisher proclaimed that he had actually seen the first fruits of this campaign, and it was a TV commercial! Featuring an Apple IIGS! We asked him to tell us what he had seen: "Well, there was this kid, and he was banging on this IIGS. Then he went outside and played for a while. Then he banged on the IIGS some more. Nothing seemed to happen, so he went outside and played some more. At first I thought Nory had videotaped me at work and had sent it to *Americas Funniest Home Videos*. But then, the kids mom came and took him to McDonalds©®™. He ordered a Happy Meal©®™, I think. It didn't make me want to buy an Apple II, but it sure made me hungry!"

ZIPPITY DO DAH!
Well, whatta ya know! I just got off the phone with Dick Stivers at *ZIP Technology*. He called to tell us that ZIP was ready to release not one, but three new accelerator products for the IIGS!

The top of the line, the Model 1600 **ZIPGSX**, is the one that we reported on in *GS+* V1.N5. It's a card (like the TransWarp GS) that contains 16K of cache memory (upgradeable to 64K) and runs at 8MHz. It is fully DMA (Direct Memory Access) compatible and can be upgraded to faster speeds as faster chips become available. It also has a socket to put your old CPU in. A nice touch! It will retail for $350, with an introductory price of $279. It is supposed to be available now.

The two other products (available in late October) are *very* small circuit boards that plug into the socket that your CPU currently lives in! The Model 1525 **ZIPChipGS Plus** (not to worry, we won't sue!) has 16K of cache memory, runs at 8MHz and is fully DMA-compatible. It is not upgradeable in any way, and will sell for $300, with an introductory price of $249.

The last chip-based product is the Model 1500 **ZIPChipGS**. It has 8K of cache memory and runs at 8MHz. It is not DMA-compatible, however, it can be upgraded for DMA compatibility. It will carry a retail price of $250, with an introductory price of $199.

It certainly sounds like ZIP has covered all the bases with this announcement! Stay tuned, we have a review unit of the Model 1600GSX on the way! And you can be sure we'll review the Model 1500 and Model 1525 when they are available. If you can't wait, call ZIP technology at (213) 337-1313 for more information, or (800) 955-5520 to place an order.

YOU CAN DO BETTER?
Got a rumor? Got a wish? Got a blatant lie? Got a stretcher? I've fallen, and I can't get up! Send a pair o' medics to:

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

THE *GS+* USERS' GROUP CONNECTION

We want to compile a list of IIGS Users' Groups and/or IIGS Special Interest Groups (SIGs) that are a part of regular Apple II Users' Groups. If you are a member of such a group, have your president contact us. All we need for the list is the name and address of the group. However, if you give us a free subscription to your groups newsletter, we'll give your group a free magazine-only subscription to *GS+*! Send that information and/or newsletter subscription to:

GS+ Users' 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.

There are eleven 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. Use EGOed v1.31 to view it.

BatteryBrain
BatteryBrain - This is the Battery Brain CDev. For a discussion of its installation and use, see "Battery Brain" on page 15.
Brain.H - This is the custom `#include` file for the Battery Brain source code.
Brain.CC - This is the ORCA/C source code for the Battery Brain CDev.
Brain.REZ - This is the REZ language code for the Battery Brain CDev.
Make - This is an APW EXEC file that is used to compile and link the C and REZ source code.
MakeR - This is an APW EXEC file that is used to compile only the REZ source code and link it with the C object code.

BWG
This folder contains the seven pictures discussed in "Brush With Greatness" on page 4. These pictures are stored in Apple Preferred Format (APF) to save disk space, so you will need a program that can read APF graphics to view them.

EGOed.1.31
EGOed - This is the EGOed NDA.
EGOed.1.31.p - This is the TML Pascal II source code for EGOed v1.31.
EGOed.r - This is the TML Pascal II resource file for EGOed v1.31.
EGOed.rez - This is the REZ language file that is used to create the file EGOed.r.
Smaller - This is an APW EXEC file that is used to compact EGOed after it has been compiled with TML Pascal II.

EGOed v1.31 requires System Software v5.0.2 or later. This means that your system must have at least 512K of RAM. It will not work correctly with System Software v5.0 and it will not work at all with System Software v4.0! To install EGOed, use the Finder to copy the file **EGOed** from the **EGOed.1.31** folder on your *GS+* disk to the ***:System:Desk.Accs:** folder on your startup disk. Once you have the file copied, restart your IIGS to make EGOed available from the Apple menu.

EXE.Launcher
This folder contains thirteen items. For a discussion of their installation and use, see "The Molehill" on page 18.

Icons
This folder contains the Finder icons discussed in "Icons" on page 21.

OrangeCherryRev
This folder contains one file: **Orange.Cherry**. This file contains reviews of the 13 remaining Talking Schoolhouse programs which we could not fit in the magazine. These reviews are not complete in and of themselves. They should be read in conjunction with the "Orange Cherry Talking Schoolhouse" review on page 30. This is a Teach Text file. Use EGOed v1.31 to view this file.

PING
This folder contains 17 files. For a detailed description of each file, see "PING" on page 8. If you only want to play the game, the file you are interested in is **PING.SYS16**. Simply double-click on this file from the Finder and the game will run. To pause the game, press the esc key. To quit back to the Finder, press Command-Q or, press the 'Q' key while the game is paused.

Problem.Form
This is the *GS+* bug report form. If you find a bug in one of our programs, fill out this form and send it to us. This is a Teach Text file. Use EGOed v1.31 to view it.

Writers.Guide
This is a Teach Text file that tells you what you need to do to write reviews, articles, programs, etc. for *GS+*. Use EGOed v1.31 to view it.

Transfusion.101
Transfusion - This is version 1.0.1 of the Transfusion NDA. For more information about this latest version of Transfusion, see "Random IIGS Programming Notes" on page 16.
XFusion.H, XFusion.CC, XCom.CC, XWindow.CC, XTrans.CC - These are the ORCA/C source code files for the Transfusion NDA.
XFusionHistory - This is a text file detailing the changes that Transfusion has gone through.
Make - This is an APW EXEC file that is used to compile Transfusion.

Transfusion v1.0.1 works only with System Software v5.0.2 and later. This means that your system must have at least 512K of memory. It will not work System Software v4.0! To install Transfusion, copy the file **Transfusion** from the **Transfusion.101** folder on your *GS+* disk to the ***:System:Desk.Accs:** folder on your startup disk. Once you have the file copied, restart your IIGS to make Transfusion available from the Apple menu.

Shuffle
Shuffle - This is the Shuffle PIF. For a discussion of its installation and use, see "Shuffle" on page 15.
Shuffle.ASM - This is the ORCA/M source code for Shuffle.
Shuffle.Macros - This file contains all the macros used by the Shuffle.ASM file.
Make - This is an APW EXEC file that is used to assemble Shuffle.

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

ICONS

By Steven W. Disbrow

We have some really great icons on this issue's disk, so, as usual, let's hop right to the descriptions!

The following icons were artistically rendered by our own Michael Quinn. They should be placed in the Icons folder of your startup disk.

Brain.Icons - These icons are for the BRAM parameter files created by Battery Brain (see page 15). Copy this file into the **Icons** folder of your startup disk.

These next few icons were drawn by Michael to accompany "The Molehill" on page 18.

EXE.Icons - This file contains icons for the EXE.Launcher program and generic EXE files.

BIN.Icons - This is a generic icon for BIN files that allows them to be launched from the Finder like any other program.

BIN.Smpl.Icons - This is an icon for the BIN.Sample binary program which allows it to be launched from the Finder.

The following icons were drawn by myself to accompany this issue's programs:

Ping.Icons - A horrid little icon for Daniel Webster's PING (page 8) that I threw together at the last minute. Copy it into the **Icons** folder of the disk you run PING from.

Shuffle.Icon - This is an icon for the Shuffle Permanent Initialization File (page 14.) This icon should be copied into the **Icons** folder of your startup disk.

XFusion.Icon - A decent little icon for our Transfusion NDA (page 16.) Copy it into the **Icons** folder of your startup disk.

GS.Plus.Icons - This file contains icons for things that show up all the time on the *GS+* disk. There is a neat icon (by Joe Wankerl) for any disk whose name starts with "GSP.", an icon for Teach Text files, and a brand new icon for our EGOed NDA (page 16.) Copy this file to the **Icons** directory of your startup disk.

As always, we want to see your custom icons! If you have any that you have done, send them to us here at *GS+* so that we can share them with the rest of our readers.

REVIEWS

GS SAUCE MEMORY CARD

Retail Price: $150 (card only)
Typical mail order price: 1 MB - $179
2 MB - $269
4 MB - $459

Harris Laboratories, Inc.
1143 Riverwood Drive
Burnsville, MN 55337
(612) 890-4498

Reviewed by Brian M. Winn

THE NIGHTMARE

"Out with the old and in with the new" was the philosophy I used when I sold my aging GS-RAM board. I was hoping to purchase a RamFAST/SCSI disk controller card and an Enhanced VisionPlus card (marketed under various names) soon, and they both required DMA (Direct Memory Access) compatible memory. Applied Engineering offered an upgrade for my current card, but I wasn't going to spend a hundred bucks for it. I decided I could sell my GS-RAM for a hundred dollars and purchase a 4 MB, DMA compatible GS Juice Plus (made by Ingenuity) for just two hundred more. What a deal, right? Boy did I make a mistake!

I sold my GS-RAM and ordered the GS Juice board directly from Ingenuity. At this time I could not run any GS software or do any programming, because I only had the 256K RAM that was on the motherboard of my old ROM 01 machine. Two entire weeks went by. I was getting nervous so I called Ingenuity. They said that they normally have a two week delay on shipments, and my card would ship the next day. A week went by and the card did not show up! So, I called again. No answer. After investigating further, I found the company had gone out of business! Great—three and a half weeks wasted. Of course they already billed my VISA account, but that is another story.

I was suffering from severe computer withdrawal by this time. I needed a memory card, fast. A local Apple dealer recommended the GS Sauce memory card by Harris Labs. He told me if I ever "upgraded" to a Macintosh, I could pull the memory off the card and plug it into the Macintosh. I felt insulted by this statement, so I informed him that if someone ever "gave" me a Macintosh, I would probably pull the memory out of it and put it in my GS. He seemed fairly shocked by this devotion to the GS. Oh well, at least he had an Apple IIGS on display in the store.

I decided on the GS Sauce and ordered it through mail order. Needless to say, it was over a month from the time I sold my GS-RAM until I had my computer up and running GS software again. Looking back on the whole ordeal, I am glad that I ended up with this card. However, next time I'm thinking of doing something like this, I am going to use the philosophy, "If it's not broken, don't fix it."

OK, WHAT ABOUT THE CARD?

Harris Laboratories, maker of memory enhancements for both the Apple and IBM worlds, has filled a void that MDIdeas' OctoRAM card left when they went out of business. Like the OctoRAM, the GS Sauce card uses SIMM (Single Inline Memory Module) boards made popular by the Macintosh. Each SIMM contains 8 chips that are surface mounted on a small board which plugs into a memory expansion connector on the GS Sauce board. SIMMs are compact, easy to install, and have recently dropped in price. Careful shoppers can find 1 MB SIMM boards at around $70.

The GS Sauce is small compared to the GS-RAM, has few components on it, and looks well-designed. The majority of the card is the memory itself. The card will accept either four 256K SIMMs (for a total of 1 MB) or four 1 MB SIMMs (for a total of 4 MB). My card was loaded with four 1 MB SIMMs. The memory slants out at a slight angle making the card wider than most. But, it is not wide enough to interfere with a card in slot 7. When fully populated with 1 MB SIMMs, the GS Sauce card will give you 4.25 MB of RAM in a ROM 01 Apple IIGS, and over 5 MB of RAM in the newer ROM 03 model.

GS Sauce, like most newer GS memory cards, is fully DMA compatible. This feature allows certain peripherals to access memory directly, bypassing the central processing unit. This accelerates input and output immensely. There are currently four peripherals (that I know of) that require DMA compatibility: the Apple High-Speed DMA SCSI card, the Enhanced VisionPlus (marketed under various names), the MultiKache disk controller card, and the RamFAST/SCSI disk controller card. If you don't have a DMA compatible memory card now, you should consider getting one. More peripherals will require it in the future, and it will increase the likelihood that your memory card will be compatible with future GS computers. If you don't know if your memory card is DMA compatible, contact the manufacturer.

A two-page manual was in the plain white box that the GS Sauce came in. It was well-written and to the point. The manual briefly describes the GS Sauce and covers installation of the card in six easy steps. Except for setting the DIP switches on the card, you really don't need the manual at all. These switches tell the card how much memory it has and which SIMMs it is using (1 MB or 256K). The switches on my card were already set accordingly.

DOES MORE MEMORY HELP?

Surprising but true, having more memory does improve your computers performance. First of all, you can set aside extra memory for a disk "cache" (this option is found in the RAM CDev in the Control Panel NDA). This will result in rapid load times and instantaneous returns to the Finder. Secondly, if you are using memory-hungry programs, you may notice a very slight increase in speed. And last, but not least, you shouldn't have to worry about running out of memory every

time you open another document in AppleWorks GS.

WHAT'S MISSING?

A software package coupled with the card would have been pleasing. It wouldn't have had to have been anything remarkable, perhaps just a utility to make sure the SIMMs were functioning properly. A CDev or desk accessory that makes a printer buffer in memory would have also been well received.

I also wish Harris Labs would have included a "spec" sheet with the card. Most computer peripherals that I have purchased have included such a sheet. I enjoy examining these technical notes, and it makes writing a review just that much easier. I guess you can't have everything, though.

OTHER THINGS TO CONSIDER

The GS Sauce card is made in the USA, works flawlessly, and is one of the lowest priced memory cards available. It is also backed by a limited lifetime warranty! Hooray for Harris Laboratories! Hopefully the lifetime warranty will become a standard throughout the computer industry.

I talked to Harris Laboratories on the phone and actually went to their "lab" to see them in action. They were very friendly and willing to help out if there was a problem. They even use Apple IIs in their office. One unusual thing I found is that you cannot buy the card directly from them. You must purchase it from your local dealer or your favorite mail-order firm.

SIMMs, like all computer equipment, can go bad. The problem is that, when even a single chip on a SIMM goes bad, you have to replace the entire SIMM! Memory cards that use sets of 8 individual chips have the advantage that if a single chip goes bad, that's all you have to replace. SIMMs do have the advantage of being very popular in the Macintosh community. And, because of this, Macintosh mail order firms are constantly trying to offer them at the lowest price possible.

WHAT'S THE CONCLUSION?

If you want a high priced, well-known memory card, you can look elsewhere. I, however, am very content with my GS Sauce. With its low price, high quality, and lifetime warranty, this card is hard to overlook. After using the GS Sauce for several months, I have discovered that I simply could not live without it!

SALVATION: WINGS

Programmed by Joe Jaworski

Retail price - $79.95
Typical mail order price - $50
Not copy-protected
Requires 1 MB RAM

Vitesse, Inc.
13909 Amar Road, Suite 2A
La Puente, CA 91746
Information: (818) 813-1270
Technical Support: (818) 813-1274
FAX: (818) 813-1273
Orders: (800) 777-7344

Reviewed by Steven W. Disbrow

[Editor's note - The product reviewed here was a review copy provided by the manufacturer.]

WINGS!

No, no, you haven't picked up *Macworld* by mistake. There really is a new product for the IIGS called Wings (not Wingz). Wings is a program launcher/disk utility that is aimed at those IIGS users that feel the Finder just doesn't cut the mustard. As opposed to the icon-based interface of the Finder, Wings uses lists of programmable buttons (128 buttons in all) and a multitude of Command-key shortcuts to allow you to quickly access your programs.

Actually, the Salvation: Wings package contains not one, not two, but three, program launchers: Wings, MiniWings, and MicroWings. The star of the show is Wings, but, since this package does cost almost $80, we'll look briefly at the other two programs.

MINIWINGS

This is a slick, programmable launcher that allows you to very tightly control which programs get run on your IIGS. MiniWings allows you to build custom program lists which allow access only to the programs you specify. Using a text editor, you create a MiniWings Script file that defines the programs you want to be selectable, the colors and styles (round, drop shadow, etc.) of the buttons that the user must "press" to run each program and even a desktop pattern or graphic file (a company logo perhaps) to be used as a backdrop.

MiniWings is a little complex, but I can see a great many uses for it. Especially for folks that don't want their children (or roommates) running rampant with every program on the old hard drive. Just build a MiniWings disk that allows them to use only the programs you want them to use.

All in all, MiniWings is a very neat launcher that would have made a fine product all by itself.

MICROWINGS

If you are a IIGS "old timer," you probably remember the very first IIGS program launcher. Basically, all it consisted of was a Standard File dialog that let you choose a program to run. That's all MicroWings is. Not very impressive, but if you are going through life without a hard disk, a program launcher that takes up less than 2K can be a very handy thing to have.

THE BIG BIRD...

And now we get to the biggest bird in the flock, Wings. When I say "biggest," I am not kidding. Wings takes up over 240K! So, if you are thinking about using it from anything other than a hard disk, you had better think again.

Assuming that you will be using a hard drive, what can Wings do that the Finder can't? Ever wish that the Finder could display text files? Wings can. Ever wish that the Finder could play sound files? Wings can. Ever wish that the Finder

could display standard picture files? Wings can. Ever wish that the Finder could run BIN or EXE files? You guessed it, Wings can. Almost everything I ever wished the Finder could do, indeed, everything the Finder *should* do, Wings can.

Just about every utility you can think of (except, of course, the ones that are provided in the rest of the Salvation series) is included in Wings. Among some of the more useful utilities Wings includes are: Sort Directory, Change the Error Beep (only affects Wings), a screen blanker (only affects Wings), file finder, customized hard disk head parking, easy activation/deactivation of NDAs, CDAs, Drivers and Inits. Wings can even undelete files, provided that you undelete them immediately after they are deleted.

As an example of the power that Wings puts at your fingertips, let me tell you a story: Being too lazy to get up and walk the 6 feet to my pro-gsplus BBS computer, I opened up the Transfusion NDA and dialed up the BBS to check my mail. After reading my mail, I ventured into the message area to see what was new. One of the messages was from a local IIGS owner that had just obtained a 100 MB hard drive. He wanted to know exactly how to go about setting up the Apple Programmers Workshop (APW) on his hard drive. "Gadzooks!" I thought to myself. "Don't people read manuals anymore?" I had just about resigned myself to typing a 40 line explanation, when it hit me... "I can use Wings to generate a catalog of my APW setup and then I can send that to him in my reply! I'll answer his question by example!" I quit the program I was in and returned to Wings. I reopened Transfusion, and began a reply to his question. Then I selected "Batch Catalog" from the Wings "Goodies" menu. I then specified my APW directory as the directory to catalog, and several seconds later, Wings presented me with a Text Editor window containing the catalog information. Since the Wings Text Editor uses the standard Text Edit keystrokes, I simply selected all of the text in the window (by pressing option+shift+downArrow), and pressed Command-C to copy it to the clipboard. I then selected the Transfusion window and pressed Command-V. All of the text was pasted into Transfusion, which sent it over the modem to the BBS. In less than 1 minute, and using less than 10 keystrokes, I had completely answered his question by providing a very complete example.

"I TAUGHT YOU EVERYTHING *YOU* KNOW, BUT NOT EVERYTHING *I* KNOW..."

Like it or not, Wings is going to be compared to the Finder. I've already done so in this review, and now I have to do it again. As I said earlier, all of the things that you wish the Finder *could* do *are* here. Unfortunately, all of the things that the Finder *can* do are *not* here. It's mostly little things, things that you probably take for granted. But, like your shoes, when they disappear, you notice that they are gone very quickly!

Most of these "omissions" are a direct result of the fact that Wings works so much differently than the Finder. For the most part, they are neither good nor bad. Wings and the Finder are simply different. The differences I am about to point out, however, are, *in my opinion,* significant enough that potential buyers should be aware of them. In presenting these differences, I will briefly describe how the Finder handles a task (I'm assuming that everyone is familiar with the Finder... which could be a mistake) and will then describe in more detail how Wings handles that same task. (Let me make it very clear that these are not *bugs* in Wings! They are just important differences that Wings has with the Finder!)

MODES OF OPERATION

In the Finder, you can see all of your online devices at once. At almost any time, you can select any item (a file, disk, etc.) that you can see, and then apply an action to it (Open, Close, Print, etc.).

Wings works in "modes." Wings defines nine different modes in its File menu: Text Processor, Play Sounds, View Graphics, UnDelete, Delete, Rename, File Info, Launch and Auto. When you select one of these modes, Wings changes its directory display (you can only view one directory at a time in the main Wings window) to show only the files that can be activated in that mode. For example, in "Play Sound" mode, Wings will show only sound files that it knows how to play.

"Auto" mode is the closest thing to a "Finder-like" mode of operation that Wings has. In Auto mode, Wings displays every file in the current directory. When you double-click on a file, Wings figures out what kind of file it is, and takes the appropriate action. While this sounds like the best of both worlds, there is at least one very annoying problem with the way Wings implements these modes. For example, I am now using the EGOed NDA to write this review. I want to know how much room the review is taking up on disk. I switch to the Wings window and click on the file, Wings.Rev. Then I realize that I have Wings in "Auto" mode not "Info" mode. Since Wings.Rev is a Teach Text file, if I were to double-click on it, Wings would display it. So, I press Command-I. Instead of giving me file info on Wings.Rev, Wings switches into "File Info" *mode* and deselects the file I just clicked on! I must now double-click on Wings.Rev to get the info I wanted. And then, if I want to launch another program, I have to go back into either "Auto" or "Launch" mode. This, to me, is very annoying, and a waste of time and keystrokes.

But, this is not just *my* opinion. On page 12 of the book, *Human Interface Guidelines: The Apple Desktop Interface,* Apple Computer Inc. goes to great lengths to explain why applications should not force users into "modes." They also explain, in very broad terminology, the types of situations that could justify the use of modes (error conditions, alerting the user to danger, etc.). It does not seem to me that anything Wings does justifies its use of modes. A better solution would have been to implement mode-switching via the option key and/or a separate "Mode" menu. This way, double-clicking on an item in the main window could still invoke Wings' "Auto" Mode, but the Command keys could have then been used to *override* the current mode by performing *immediate actions* on any files that might be selected. Better still, implement the "UnDelete" feature in the File Utilities (discussed below) and drop the modes entirely. This would be more in line with the Human

Interface Guidelines, and with just about every other IIGS application.

FILE UTILITIES

In the Finder, you can have up to 10 windows open at once. Any or all of these windows can display the contents of a directory. You can freely copy, move, duplicate, launch, etc., the items in any of these windows.

The main window of Wings only displays one directory at a time. By entering the appropriate mode, you can perform various file maintenance functions on the files shown in that directory. In order to copy files, you must press Command-F to invoke the Wings File Utilities window. This window displays two different directories and so you can use it copy and/or move files between those directories. You can also use it to verify, rename, and delete files in either of those directories. This is also the only place in Wings that you can create new Folders. It's odd not having all of these options available from the menu bar, but once you get used to the way the File Utilities window works, it's very flexible and quite usable.

A similar window is used to handle volume utilities (copying, formatting, etc.). It also takes some getting used to, but once you are, it is easy to live with.

PROGRAM LAUNCHING

In the Finder, when you double-click on something, it "opens." If the something you double-click on happens to be a program (or a document whose icon tells the location of the program that created it), the Finder launches it (or the program that created the document). If the Finder does not know how to open what you click on, it gives you a message that tells you that, basically, it does not know how to open it.

As I said earlier, Wings uses lists of buttons to implement its program launcher. Each of these lists is called a page. There are eight pages of buttons with 16 buttons on each page. That's a total of 128 buttons that you can program to quickly launch your applications. Each page can be given a different name and you can quickly move between pages using the left and right arrow keys or a handy pop-up menu.

But how do you get those programs linked to those buttons? It's easy. To program the buttons yourself, you enter the Wings button programming window. The first thing you do is choose a button to program. Then you supply the name that Wings should display in the button. Then you click on the Find button and Wings brings up a Standard File dialog that you can use to easily specify the pathname to the file. The only way it could be easier is if it could program itself—and it can!

Wings has a very handy feature, "Auto Program." When you select this option, Wings will search one of your online disks (you tell it which one) for programs that it recognizes. When it finds one, it programs the next available button to run that program. Wings has lots of other button programming options, which I won't go into here, but suffice it to say that it is very easy to program and maintain the program launching buttons.

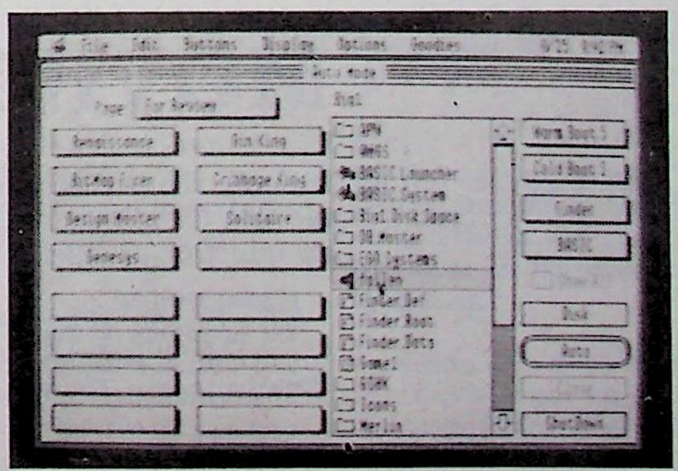
Once you have your buttons programmed, all you have to do is click on one and the appropriate program is run. Wings also has a keyboard equivalent for each button, so you don't even have to use the mouse to run your programs. What could be easier?

You can also launch programs from the directory list in the main Wings window, which brings us to the major program-launching difference between Wings and the Finder: Wings does not use Finder icons. But then, why should it? After all, Wings isn't the Finder!

Well, the directory list in the main Wings window is very similar to a Finder window with the view set to "by Name". Instead of using Finder icons (which I think we can safely assume everyone has), Wings keeps within its resource fork a great many custom icons that it uses to identify file types. This takes up lots of room and is one of the reasons Wings is so dang big. Also, since Wings does not use Finder icons in its display, you cannot launch an application by double clicking on one of its documents. This is one of the major strengths of the Finder and is, in my opinion, a major shortcoming in Wings.

Another problem with this is that a great many people have spent a great deal of time developing custom Finder icons and, quite frankly, they might not want to give them up. IIGS users get a certain pride from seeing their icons up there on that screen. "Sentimental value" may seem like a silly argument for Finder icon support, but just tell that to folks that wear four-year-old sneakers or still use their Apple II+.

The easiest solution would be to dump all of those custom Wings icons into a standard Finder icons file and begin supporting the use of Finder icons. Even if Wings only supported "small" Finder icons, this would greatly enhance the power of the Wings program launcher and protect the time that thousands of IIGS owners have invested in the Finder (and thus make it easier to sell those users copies of Wings).



So, having detailed some of the differences between Wings and the Finder, let me wind down by listing the few things I have found to be missing from Wings.

1) There needs to be an activate/deactivate CDevs option in the Activate System Files window.
2) Wings needs a "Next Window" command. (Actually, every program needs one—that's why we wrote Shuffle.)
3) Wings does not save its Page Setup. You must reset it every time Wings is run.
4) Nowhere in Wings can you find out if a disk is write-protected.
5) The only place you can determine the free space on a disk is in the Volume Utilities window. Neither the main directory display nor the File Utilities show the free space left on a disk. This would be very handy when copying files from one place to another.
6) The scroll bar in the Clipboard window does not work.

FINALLY, A REAL PROBLEM

When you shut down Wings, it tells you that "You may now switch off your Apple IIGS safely," just as you expect. What you don't expect is the button marked "Cancel" that appears in the same dialog! If you click on that button, you are immediately returned to Wings! A very neat trick! Or is it?

Again, I am, as I write this, in Wings. I am using our EGOed NDA to write this review. In the Finder, when I have EGOed (or any other NDA) open and I shut down the computer, the system tells EGOed that it is shutting down and it had better check to see if I need to save my work. When I have EGOed (and other NDAs) open in Wings, and I try to shut down the computer, this does not happen. In fact, when the "You may now switch off your Apple IIGS safely" dialog appears, I can still see the EGOed cursor flashing on the screen! Apparently, Wings is not doing a proper GS/OS shut down. This is *very* bad and, in a worst case, could lead to loss of a work in progress. So be careful!

Not using a standard GS/OS shut down call also seems to cause a problem when you click the "Restart" button. Sometimes it works, sometimes the computer just hangs up and you must restart with the old three finger salute.

TOO MUCH STUFF

Hopefully, by this point, you get the picture. Wings is an ultra-slick program launcher that has just about every utility you could want. In fact, there's too much stuff in Wings to fit it in this review. And while Wings has features you've only dreamed of using in the Finder, it may take you some time to get used to the way Wings does the things you now take for granted.

And, worst of all, you simply might not be able to afford Wings. Let's face it, the Finder was, for all intents and purposes, free. It came with your IIGS. Add a few choice NDAs (EGOed, ShowPic, Background Music, Find File, etc.) and you have most of the utilities that are in Wings. And, since they are NDAs, they will be available from every desktop program you run, not just the Finder! Although Wings is worth every single penny, $80 is simply an unrealistic price when faced with that sort of competition. Since MiniWings and MicroWings are products that will appeal to only a small number of IIGS owners, Vitesse should consider unbundling them from the Salvation: Wings package and selling them separately. This would allow them to sell Wings at a slightly more reasonable retail price and would knock the mail-order price down to a point where Wings could very quickly become the program launcher/disk utility of choice in the IIGS world.

In summary, Wings is great. I use it a lot and will probably buy additional copies for our other IIGSs. As usual, Vitesse's documentation is excellent—another very important factor to consider when purchasing a program this complex. In fact, Wings is probably the most comprehensive and powerful desktop utility program available for the IIGS. If you use it only for its utilities, it will make your life much easier. However, it does have some operational differences from the Finder that some will want to consider before putting down their money. But, if you crave the sheer power that Wings has to offer, and you have the money and a hard disk, Wings will be just the ticket!

WORLD GEOGRAPH

Minnesota Educational Computing
Corporation (MECC)
3490 Lexington Avenue North
St. Paul, MN 55126
(800) 228-3504

Retail Price: $139
Typical mail order price: $56
Lab packs and network version available
Copy-protected
Requires 768K RAM

Reviewed by Greg Zimmerman

DETAILS, DETAILS

World Geograph is a very serious educational tool. It combines a large factual database of information on 177 countries around the world broken down into 55 categories with maps that interact with the database.

The program is recommended by MECC for grade six through adult. It is compatible with System Software v5.0.2, will run on either a ROM 01 or ROM 03 machine, comes on two disks, and is copy-protected (backup disks are included). Two 3.5-inch drives are recommended, otherwise there will be some disk swapping.

The manual is a well-written 150 pages of instructions and appendices. It is impossible to fully benefit from the extensive features of this program without first reading the manual. A separate classroom guide is also available at the retail price of $19.

MECC also publishes a companion program, U.S.A. Geograph, which is a more in-depth study of the geography of the United States.

MECC has a lifetime disk replacement policy. If a disk ever fails to work, they will replace it. The replacement is generally free, but there may be a small postage charge depending on the circumstances.

MECC also has a toll-free number you can call for customer service. The people at MECC are very helpful, and if the customer service representatives are not positive of the answer to your question, they get someone on the phone that is knowledgeable about the program you are calling about. I ended up talking directly to the designer of World Geograph to get the answers to some of my more technical questions.

MECC does plan to update the data disk so that the program information is kept current in this rapidly changing world.

WHAT IT CAN DO

The main theme of World Geograph is its ability to allow the comparison and display of information about countries on various maps. For example, if you wanted to compare the GNP of all the countries in the world, you can bring up a map of the world with the countries broken down into color-coded categories, in which the colors represent the country's ranking in total GNP. If you wanted to see how France compares to the rest of the world in military expenditures, you can bring up a map that will show you. If you don't know where a country is located, the map will tell you that, too. And the maps are generated in successively detailed format, from maps of the world and maps of continents, down to maps of 21 different regions. All of the database information can also be brought up in virtually any comparative mixture for display in charts and graphs as well as maps.

The database information can be exported to other programs for the creation of custom reports. And the user may add up to three new database categories to the already existing 55 that are built into the program.

This program can do so much, so rather than write what would be a ten-page review reciting its features, the best way to describe its potential is to step through one of many realistic scenarios for its use.

Let's suppose that Iraq invaded Kuwait in August. With school starting at the beginning of September, a creative 10th grade social studies teacher decides to integrate current events into the geography curriculum to make the subject more interesting, and to get across the point that geographic facts, just for their own sake, are virtually useless. It's how those facts compare, how they interact, how they impact our world and our lives that make geography a necessary subject.

The first day of class, the teacher starts to describe a wonderful project about Iraq to the new class, and of course the first question that comes up is, "Where is Iraq?" I don't want to exaggerate my recollection of the statistics concerning geographical knowledge (or lack there of) in high schools across America, so lets just say that on average, at least half the class would not even know where Iraq could be found on a map.

The teacher takes a long hard breath, and decides to assign each member of the class the same project.

"Prepare a report about Iraq and the surrounding region, presenting geographical and other facts, that impact on the current crises. Prepare this report assuming the reader knows nothing about the geography of Iraq, and present facts that will lead to discussion and conclusions concerning the current military conflict. Bring this report to class with you tomorrow."

Everyone looks around the room, and knows that they are in for a very long night.

After school that day, the entire class descends on the library, all trying to find and check out the same books and encyclopedia volumes. Well, I can assure you, that with World Geograph at their disposal, every one of these kids headed to the wrong place. While World Geograph is not intended to be a stand-alone educational tool, it is more than up to the task of quickly and creatively enlightening even the most uninformed child on the assigned subject, and it can prepare and print out the information needed in a viable report format.

So, arriving at the library at the end of the line, with no books left on the subject, one student (I'll call him Jon) heads to the computer lab, and boots up World Geograph.

So, having detailed some of the differences between Wings and the Finder, let me wind down by listing the few things I have found to be missing from Wings.

1) There needs to be an activate/deactivate CDevs option in the Activate System Files window.
2) Wings needs a "Next Window" command. (Actually, every program needs one—that's why we wrote Shuffle.)
3) Wings does not save its Page Setup. You must reset it every time Wings is run.
4) Nowhere in Wings can you find out if a disk is write-protected.
5) The only place you can determine the free space on a disk is in the Volume Utilities window. Neither the main directory display nor the File Utilities show the free space left on a disk. This would be very handy when copying files from one place to another.
6) The scroll bar in the Clipboard window does not work.

FINALLY, A REAL PROBLEM

When you shut down Wings, it tells you that "You may now switch off your Apple IIGS safely," just as you expect. What you don't expect is the button marked "Cancel" that appears in the same dialog! If you click on that button, you are immediately returned to Wings! A very neat trick! Or is it?

Again, I am, as I write this, in Wings. I am using our EGOed NDA to write this review. In the Finder, when I have EGOed (or any other NDA) open and I shut down the computer, the system tells EGOed that it is shutting down and it had better check to see if I need to save my work. When I have EGOed (and other NDAs) open in Wings, and I try to shut down the computer, this does not happen. In fact, when the "You may now switch off your Apple IIGS safely" dialog appears, I can still see the EGOed cursor flashing on the screen! Apparently, Wings is not doing a proper GS/OS shut down. This is *very* bad and, in a worst case, could lead to loss of a work in progress. So be careful!

Not using a standard GS/OS shut down call also seems to cause a problem when you click the "Restart" button. Sometimes it works, sometimes the computer just hangs up and you must restart with the old three finger salute.

TOO MUCH STUFF

Hopefully, by this point, you get the picture. Wings is an ultra-slick program launcher that has just about every utility you could want. In fact, there's too much stuff in Wings to fit it in this review. And while Wings has features you've only dreamed of using in the Finder, it may take you some time to get used to the way Wings does the things you now take for granted.

And, worst of all, you simply might not be able to afford Wings. Let's face it, the Finder was, for all intents and purposes, free. It came with your IIGS. Add a few choice NDAs (EGOed, ShowPic, Background Music, Find File, etc.) and you have most of the utilities that are in Wings. And, since they are NDAs, they will be available from every desktop program you run, not just the Finder! Although Wings is worth every single penny, $80 is simply an unrealistic price when faced with that sort of competition. Since MiniWings and MicroWings are products that will appeal to only a small number of IIGS owners, Vitesse should consider unbundling them from the Salvation: Wings package and selling them separately. This would allow them to sell Wings at a slightly more reasonable retail price and would knock the mail-order price down to a point where Wings could very quickly become the program launcher/disk utility of choice in the IIGS world.

In summary, Wings is great. I use it a lot and will probably buy additional copies for our other IIGSs. As usual, Vitesse's documentation is excellent—another very important factor to consider when purchasing a program this complex. In fact, Wings is probably the most comprehensive and powerful desktop utility program available for the IIGS. If you use it only for its utilities, it will make your life much easier. However, it does have some operational differences from the Finder that some will want to consider before putting down their money. But, if you crave the sheer power that Wings has to offer, and you have the money and a hard disk, Wings will be just the ticket!

WORLD GEOGRAPH

Minnesota Educational Computing
Corporation (MECC)
3490 Lexington Avenue North
St. Paul, MN 55126
(800) 228-3504

Retail Price: $139
Typical mail order price: $56
Lab packs and network version available
Copy-protected
Requires 768K RAM

Reviewed by Greg Zimmerman

DETAILS, DETAILS

World Geograph is a very serious educational tool. It combines a large factual database of information on 177 countries around the world broken down into 55 categories with maps that interact with the database.

The program is recommended by MECC for grade six through adult. It is compatible with System Software v5.0.2, will run on either a ROM 01 or ROM 03 machine, comes on two disks, and is copy-protected (backup disks are included). Two 3.5-inch drives are recommended, otherwise there will be some disk swapping.

The manual is a well-written 150 pages of instructions and appendices. It is impossible to fully benefit from the extensive features of this program without first reading the manual. A separate classroom guide is also available at the retail price of $19.

MECC also publishes a companion program, U.S.A. Geograph, which is a more in-depth study of the geography of the United States.

MECC has a lifetime disk replacement policy. If a disk ever fails to work, they will replace it. The replacement is generally free, but there may be a small postage charge depending on the circumstances.

MECC also has a toll-free number you can call for customer service. The people at MECC are very helpful, and if the customer service representatives are not positive of the answer to your question, they get someone on the phone that is knowledgeable about the program you are calling about. I ended up talking directly to the designer of World Geograph to get the answers to some of my more technical questions.

MECC does plan to update the data disk so that the program information is kept current in this rapidly changing world.

WHAT IT CAN DO

The main theme of World Geograph is its ability to allow the comparison and display of information about countries on various maps. For example, if you wanted to compare the GNP of all the countries in the world, you can bring up a map of the world with the countries broken down into color-coded categories, in which the colors represent the country's ranking in total GNP. If you wanted to see how France compares to the rest of the world in military expenditures, you can bring up a map that will show you. If you don't know where a country is located, the map will tell you that, too. And the maps are generated in successively detailed format, from maps of the world and maps of continents, down to maps of 21 different regions. All of the database information can also be brought up in virtually any comparative mixture for display in charts and graphs as well as maps.

The database information can be exported to other programs for the creation of custom reports. And the user may add up to three new database categories to the already existing 55 that are built into the program.

This program can do so much, so rather than write what would be a ten-page review reciting its features, the best way to describe its potential is to step through one of many realistic scenarios for its use.

Let's suppose that Iraq invaded Kuwait in August. With school starting at the beginning of September, a creative 10th grade social studies teacher decides to integrate current events into the geography curriculum to make the subject more interesting, and to get across the point that geographic facts, just for their own sake, are virtually useless. It's how those facts compare, how they interact, how they impact our world and our lives that make geography a necessary subject.

The first day of class, the teacher starts to describe a wonderful project about Iraq to the new class, and of course the first question that comes up is, "Where is Iraq?" I don't want to exaggerate my recollection of the statistics concerning geographical knowledge (or lack there of) in high schools across America, so lets just say that on average, at least half the class would not even know where Iraq could be found on a map.

The teacher takes a long hard breath, and decides to assign each member of the class the same project.

"Prepare a report about Iraq and the surrounding region, presenting geographical and other facts, that impact on the current crises. Prepare this report assuming the reader knows nothing about the geography of Iraq, and present facts that will lead to discussion and conclusions concerning the current military conflict. Bring this report to class with you tomorrow."

Everyone looks around the room, and knows that they are in for a very long night.

After school that day, the entire class descends on the library, all trying to find and check out the same books and encyclopedia volumes. Well, I can assure you, that with World Geograph at their disposal, every one of these kids headed to the wrong place. While World Geograph is not intended to be a stand-alone educational tool, it is more than up to the task of quickly and creatively enlightening even the most uninformed child on the assigned subject, and it can prepare and print out the information needed in a viable report format.

So, arriving at the library at the end of the line, with no books left on the subject, one student (I'll call him Jon) heads to the computer lab, and boots up World Geograph.

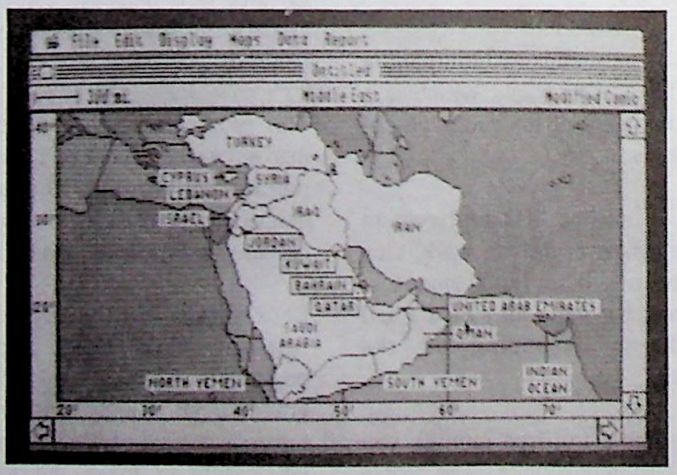
The first thing the student sees after the program launches, is a map of the entire world showing the boundaries between countries. So where is Iraq anyway? Using the pull-down menus, Jon selects "Locate Nation." A listing of all the countries appears in a window in alphabetical order, and Jon scrolls down and selects Iraq. A more detailed map of the Middle East appears, with one country flashing for a few seconds. Sure looks a lot bigger than Kuwait. Also, it looks smaller than Iran. Hey, this guy is smart—learned really quickly about attacking bigger countries. This is a good map of the region, so Jon decides to save it by pulling down the "Report" menu and saving it for his report.

Looking at the map of the Middle East, Jon can't decide if these countries are in Europe or Asia, so using the "Maps" pull-down menu, he selects Asia, and a map of that continent appears. Sure enough, the Middle East is in Asia. Jon decides that the beginning of his report will be a map of the world, then a map of Asia, then a map of the Middle East, so he saves the Asia map as a report section, and then goes back to the world map and saves it also.

Now, just how big are these countries, and what else can we find out about them? Jon returns to the map of the Middle East. Clicking once on Iraq, and then clicking on Kuwait while holding down the shift key, selects both countries and highlights them on the map. Using the "Display" menu, Jon selects "Data Cards" and a new window pops up, with the complete data base information on each of the two countries. Reading through each database one at a time is not what Jon had in mind, so he now selects "Data Tables" from the same menu, and a scrolling chart appears listing all the database categories, and all the info for those categories that apply to Iraq and Kuwait. This is good, because Jon can now see the information in each category for each selected country at the same time. This makes comparisons a lot easier. The database categories are presented in alphabetical order, and scrolling to just the fourth one, Area, Jon finds that Iraq has 169,000 square miles, and Kuwait has only 6,880 square miles.

Looks like a fair fight is really shaping up in the Middle East. Scrolling through the rest of the categories, Jon finds that the Kuwaitis eat better than the Iraqis (daily calorie intake of 3,344 versus 2,155), that while Kuwait is all desert, Iraq has some areas of Mediterranean and steppe, finds that the Iraqi death rate is over three times higher than that in Kuwait (it doesn't say if this is the natural causes death rate), and that Kuwait has a GNP that is 2/3 of Iraq's, even though it so much smaller. And Jon is only up to the letter "G" in the database. Moving along, it appears that Kuwaitis, live longer and—oops—spend only 5% of their GNP on the military. Looks like the Emir was a little too laid back in the heavy weapons procurement area. Iraq spends over 42% of its total GNP on the military. This sounds like a really big number, so before moving on, Jon decides to see how this compares to the rest of the world. Bringing back up the world map, Jon selects "Compare" from the pull-down menus, and two windows pop up. One to select the country that he wants to compare to everyone else, and the second, to select the database category with which to make the comparison. Selecting Iraq, and Military Expense, produces a new world map, with some unusual information. Iraq is orange, and the rest of the world is now brown (I'm a little color blind, so your colors may be different than my colors). Selecting "show key" from the pull-down menus, the color key for the map appears, and tells Jon that brown countries spend less on their military as a percentage of GNP than orange countries. Closing the "key" window, Jon notices that at the top of the world map, it says "Iraq ranks 1st in Military Expense (as a percentage of GNP)." Well, it's starting to look like the U.S. may need more than a frigate or two to put an end to this thing. Jon decides that this map is very informative, and saves it for use in his report.

These new facts perked Jon's interest, so using the pull-down menus Jon selects "World Quartiles," selects "Military Expense" from the pop-up window that appears, and now the World Map is broken down into four colors, designating which countries spend what percentage of their GNP on the military compared to everyone else. Using "show key," Jon finds that the color blue designates countries in the highest quartile—those that spend the most on the military. Disturbingly, the middle east looks like a suburban swimming pool on the map. Selecting region map, and choosing Middle East, Jon brings up a new map, with the same information, but which is a larger, more detailed map of only the Middle East. Jeez, the only countries that aren't blue are Cyprus, Turkey, and you guessed it, Kuwait. This guy in Iraq is pretty smart after all. And looking at the blue stain on the map, Jon realizes why there seem to be a lot of explosions in this area of the world. It looks from the map like they've got almost all the world's weapons in the Middle East. Jon saves these two maps, and decides it's time for something other than a map in



his report, so he brings back up the map of the Middle East, selects Saudi Arabia, Kuwait, and Iraq, and then selects "Display Graph" from the pull-down menus.

Up pops a window asking which category Jon wants on the graph, and this time he picks "average July temperature." (Jon remembered seeing something about how hot things were over there). Up comes a graph showing the three selected countries average July temperature. Each is over 90 degrees, and Jon saves this for his report.

Are there any mountains in the Middle East, or is it all hot desert? Jon first double-clicks on the Middle East map, to select the entire region for database purposes, then selects "Search" from the "Data" pull-down menu, and up comes the window that allows the user to search the database with custom criteria. Jon clicks on "add rule," scrolls down to "natural features," clicks on the "contains" choice, and types in the word mountain. Setting the word mountain as a rule is making it a word to be searched in the selected database, natural features, for the selected nations, which in this case are all the countries in the Middle East. Pushing return once to set the rule, and another time to do the search, brings up a map of the Middle East, with only Iran and Lebanon highlighted. This means that of all the countries in the region, only these two have mountains. Jon saves this map, and decides to see if all these Middle Eastern countries are oil exporters. He first re-selects the entire region, then using the search selection, he chooses "Exports," and types in the word oil. The program tells him that none of the nations have "oil" as exports, so he goes back to edit his search criteria, but before doing so, chooses "show list," which gives him a scrollable list of all the different kinds of entries in the selected database record, exports. Jon finds that the right word is "petroleum," and he edits his rule and does the search. Seems everybody except six countries in the region export oil. Jon saves the map, and moves on. He is able to add to his rules for the database search, by specifying information in many different ways. Do the oil exporters have higher per capita income? Do they spend more on the military? Do they live longer? Jon is able to extract all this information and more on just this one comparative subject from the database, and save maps, charts and graphs showing his findings.

Jon also prints out the entire data card for Iraq, in order to add more detailed information to this report. This page will list all the information in the Iraq database, including a list of bordering countries, a breakdown of the population by age groups, as well as economic information such as major imports, and per capita income.

Jon spends another half hour or so, selecting, comparing, and saving maps and information. In the process, he learns about countries in the area such as the United Arab Emirates, Oman, Bahrain, Israel, etc. With the power of colored comparative maps and charts, Jon is easily able to assemble a report that communicates the information quickly and easily. It takes Jon less time to prepare the report than it does to print the thing. Leaving the computer lab, with plenty of time to get home before dinner, Jon notices that the library is still packed with frantic students, trying to find, assemble, and write up the information they need for their reports.

This example (remember, the example I have given does not show you everything that the program can do) should give you a good idea of what World Geograph is all about. It brings up information in easy to use formats, be they maps, charts, or graphs. This information can be manipulated in almost any way the user desires, to compare, study, or present the information in an understandable format. All the information can be saved and printed on a LaserWriter, a GS-compatible Epson, or an ImageWriter II (MECC says that it will probably work on other GS-compatible printers, but that these are the only ones they have tested it with). It can be printed in a wide variety of formats and sizes, with special options for things such as titles and page numbers. It takes full advantage of the color capabilities of the ImageWriter II, and allows for great organization of the information that the user develops into a report format, arranged in any order the user desires.

WHAT MAKES THIS A GOOD PROGRAM?

The best thing about World Geograph is that through the maps, it makes the manipulation of the database information very easy. Most of the work of digging out information and assembling it, is done for the user. This allows the user to spend more time creatively and in exploration. It's so easy to follow up thoughts and ideas with investigation and comparison, that the user is encouraged to do more, and to learn more. It truly reinforces the concept that a good picture is worth a thousand words.

World Geograph is not meant to take the place of other resource materials and study methods. Obviously two disks do not contain all of the information that may be relevant to any particular geographical subject. But it is a motivator, an educator, and an idea generator. And the database is sufficiently detailed that it does give not just the big picture, but a lot of little ones as well.

A good geography teacher could build an unlimited number of lessons around this program. Interesting lessons, fun lessons, and most of all, lessons that will show the students about geography in a way that will make them want to know more, and in such a way, that they will remember what they learned.

ANYTHING WRONG?

So what's not to like? Well, it could load faster, it shouldn't be copy-protected, and there are a few things within the program that could be improved.

An example of program improvements that would be beneficial is in the Database Tables. It is not possible to set the titles (the names of the countries) in a manner similar to the way it can be done say in an AppleWorks GS spreadsheet. This means that the user must either remember all the country names as he scrolls through the information, or constantly refer back to them. When looking at comparisons of only a couple countries, this is not problem. But when comparing lots of them, it is a pain.

One other program improvement I'd vote for would be to expand both the number of databases that come with the program, and the number of databases that can be added by the user.

The program uses the "insert your original disk so we know you are not a thief" method of copy-protection. I got mine working on a hard drive (without the need to insert the original each time), and was able to make working copies, by changing byte 0F8 on block 69 from BO to 80 on the program disk, using the Copy II+ sector editor. **I DID THIS TO A COPY, NOT THE ORIGINAL, AND THEN TRANSFERRED THE COPY TO THE HARD DRIVE.** Also, Copy II+ has a parameter for making working backups, however it did not work on my disk. Of course it may work on yours, and I would advise trying it first, before attempting to deprotect the disk manually. A set of backup disks is included with the program, so the copy protection is not really annoying in the area of making a backup, but it is annoying because it does prevent use on a hard drive without inserting the original each time.

Considering the amount of graphics that the maps and charts entail, the screen change times are more than acceptable, even on a stock IIGS.

Loading time on a ROM 01 IIGS, with 1.25 MB, and two 3.5-inch drives (no disk swapping), was one minute for the program disk, and after clicking on the proper icon, another one minute to get to the world map. Launching from a 20 MB MacCrate Hard Drive, on a TransWarped ROM 01, the loading time from the System Software v5.0.2 Finder was only 16 seconds.

DO I RECOMMEND IT?

World Geograph is one of the few educational programs on the market that begins to harness the power of the IIGS. The interaction of the maps and the database make it an extremely powerful, interesting, and easy-to-use learning tool.

In the right circumstance, I would recommend World Geograph without reservation. Is it for five-year-old kids? No! Is it a game that your kids are likely to pull out and play with all the time? No! But, as an educational tool in schools, it could be invaluable. As a home reference and report generator for grades six through twelve, it would also be very helpful.

Simply put, this is an extremely high quality piece of software. Users in the proper age group will find this to be the finest geography program available for the IIGS.

ORANGE CHERRY TALKING SCHOOLHOUSE SERIES

ABC's, Addition and Subtraction, Alpha Chimp, Animals Activity Set, Clock, Colors and Shapes, Dinosaurs, First Reader, First Writer, Money, Numbers, Reading Railroad, Speller, School Bus, U.S.A. Map

Retail price: $49 (each)
Typical mail order price: $42 (each)
School edition: $59 (each),
includes backup disk
512K RAM required

Orange Cherry Software
P.O. Box 390
Pound Ridge, New York 10576
(800) 672-6002

Reviewed by Greg Zimmerman

Orange Cherry's Talking Schoolhouse Series is fifteen different programs that teach children basic skills utilizing the sound and graphics of the IIGS. While all of the programs have similarities, they are all stand alone products that can be used individually.

Because I never tried to review fifteen programs in one article prior to this one (for that matter I never tried to review two programs at the same time), I decided that a little up front "getting organized" would be helpful (for me and you).

To save my fingers, I am not going to use the word "Talking" in naming each program every time in the review. But rest assured, they all talk.

The review, due to the similarities of the programs, will consist of four sections. First an overview of the series with some of the things that all the programs have in common. Next, in a break with the normal format, my summary and conclusions about the whole series. The information in these first two sections must be considered when reading each review, because it is not all repeated in every program description. Third, a description and specific review comments about two of the programs. These two reviews are the first two programs taken in alphabetical order, and are representative of the good and bad points of all of the software in the series. The fourth section contains short reviews of the other thirteen programs in the series and may be found on this issue's disk in the folder **Orange.Cherry.Rev**. They are in Teach Text format, and require any text editor that can read Teach Text files (such as EGOed, which is included on this issue's disk) to read from your screen or to print.

Even my four little beta-monsters got tired of "doing" computer after each of them put these fifteen programs through their paces, so I hope your eyes are well-rested.

OVERVIEW

The Talking Schoolhouse Series is mainly intended for young children. The recommended age for users ranges from preschool to fourth grade, (ages three to nine) except for the U.S.A. map program, which is recommended for ages eight to adult.

The software is intended to be installable on a hard drive, but because the company sells protected and unprotected versions of the disks, you may buy a protected version which cannot be put on a hard drive. Depending on which program it is that you cannot install on your hard drive, the

company says they will either replace the disk with a deprotected version (for free), or will tell you over the phone how to do the installation.

The software is graphics and sound intense, with some very high quality graphics and animations. There is a lot of sound, and all the "Talking" is done in human voices that are very lifelike.

The software is all compatible with System Software v5.0.2. I also tested many of the programs on a ROM 03 IIGS, and had severe crashing problems with the Talking Animals Activity Set. It loaded when launched from the 3.5-inch disk (not from the Finder), but when I did one of the activities, it crashed repeatedly. The company has now fixed this problem, and will give a free replacement disk to anyone needing a ROM 03 compatible version. I did not go through all the activities on all the programs using a ROM 03, but Orange Cherry says they all will work properly, and will replace for free any disks that have compatibility problems.

The software is generally very slow to load, and slow to change screens among the different modules in each program. VERY SLOW! I'm not going to mention this in every program summary, but rest assured, they are all slow.

Each of these programs has some good stuff in it, but also each has its problems. Because all of the programs will run on a 512K machine in spite of the large amounts of sound and graphics, many IIGS owners who have not upgraded the memory in their ROM 01 machines will be able to use this software.

All of the programs allow the user to adjust the volume level using the mouse on the screen, as opposed to going into the control panel.

The software generally has what I consider to be problems with the way it has been programmed. This relates both to the user interfaces that are in the software, and bugs that have not been removed.

Each of the programs comes with a ten-page manual which explains how to operate the software, and offers suggestions for other activities related to the program.

Orange Cherry has eleven more GS-specific programs in development that the company says will all be released at different times throughout the next twelve months. Three of these future programs are being advertised in the September issue of *inCider*, but they are not yet available.

Orange Cherry also has a lifetime replacement policy for all their programs. They will replace free of charge, any disk that is accidentally damaged. This is great, and especially so for software intended for use by young children.

Last, this is a very cooperative, friendly, and helpful company. I called several times to inquire about specific items relating to the programs, and I was always transferred to someone that had the answers and was knowledgeable about the software. This company is trying very hard not just to market a great IIGS product, but also to follow it up with good service. As an added bonus, all this good service comes on a toll-free number.

SUMMARY/CONCLUSIONS

The Talking Schoolhouse Series is a major effort by Orange Cherry at serving the IIGS educational market. Many, many of the activities contained in the various programs are great learning concepts and creative ways to get the message across to the kids. The sound is great, the graphics are very good, sometimes outstanding, and the animations are very good as well.

I give Orange Cherry a lot of credit for so heavily supplying the IIGS market. And knowing that they have so many more programs in the works, I was looking forward to being able to give a glowing review.

Well, after I got done with all the individual program descriptions and reviews, I actually felt bad that upon close examination, I could not recommend the purchase of most of these programs. I really wanted to like these programs, because there is some unique stuff here, and I do have four kids to feed software to. But the software, notwithstanding its good points, has a number of problems that, when combined with the relatively high price, leaves me unable to recommend that parents go out and buy all of it.

The stuff is unbearably slow. You will be hard pressed to find any slower loading IIGS software. It is hard to enjoy anything about this software unless it is run from a hard drive.

Some of the two-disk sets require so much disk swapping if you only have one 3.5-inch drive, that my children quickly got turned off to those programs. They did enjoy some of the individual activities on those disks, but they didn't pull those programs out a second time because they are a pain to operate.

The user interfaces are not well thought out. Some of the programs literally seem to brag that they follow the Apple Human Interface Guidelines. They don't seem to, at least to my kids or to me. Some of the interfaces are all directions written on the screen. For young children, this is a waste of time. Even within the same programs, different activities have different interface features. Some of the modules are impossible to escape from without rebooting the machine, or completing the activity. Some of the programs do have voice instruction, but in many instances, the instruction is not complete enough for a young child to operate the program because other instructions are written on the screen, or require reading ability to make menu selections. I could easily imagine a teacher in a computer lab or a classroom setting with children that cannot already read, going nuts chasing around to each kid trying to explain how to get around the program that the kid is doing. This was the most disappointing feature of the software, and it gave me a better appreciation for those children software manufacturers that spend the time to really get it right.

Many of the programs were prone to random crashing on my hard drive. There was no apparent pattern to the crashes, and I do not have this problem generally. I recopied many of the programs that were crashing, in case I was at fault, but the problem still persisted. This is not a

problem when booting directly from the floppy.

Some of the programs have several different activities that are for completely different developmental levels of children. One module on the disk might be good for a two- or three-year-old, and another would require a six-year-old to appreciate.

Even with the faults, all of the software has its good points. Some of the learning modules, once you get into them, are outstanding. But if you can ignore the faults, then you have to look at the stack of disks on the table, and decide if they are worth an investment of $700 or $800. I could not arrive at the opinion that $700 or $800 was best spent on this software.

So, to wrap it up, these programs could be great, but they are not. A little smoother, more consistent interface, faster operation, more follow-through on the voice instruction method, and a more consistent grouping of activities within programs for particular developmental levels, and I would be recommending almost all of them to you. But, in their present state, at over $40 a crack, I am of the opinion that only a couple are worth the investment, and the effort.

So, you brave souls that want to hear some details should light the fire, put your feet up, and read on.

TALKING ABC'S

Talking ABC's is a two-disk set, with each disk bootable, and able to run separate from the other. The two disks are Learning Letters and Learning Words. This set is recommended by the publisher for preschool through second grade children. However, younger children will require parental help due the method of instructions that are explained below.

Each disk consists of three activities. They are Learning Letters, Recognizing Letters, and Finding Letters. The activities and the graphics on both disks are identical. Only what the voice says is different.

On the Letters disk, in Learning Letters, the child pushes any letter on the keyboard, and a graphic of an object whose name starts with that letter appears on the screen, as well as both the upper case and lower case of that letter. A human voice says the name of the letter at the same time that the screen appears. On the Word disk, the graphic and the letters are identical, except that the voice says the name of the graphic instead of the name of the letter.

In Recognizing Letters, the program puts a graphic of an object on the screen, and the child is to find and push the letter on the keyboard that is the first letter in the name of the object. Both disks go through the alphabet in order in this module, and the only difference between the modules, is that with the Letter disk, a voice says the name of the letter, and with the Word disk, a voice says the name of the object.

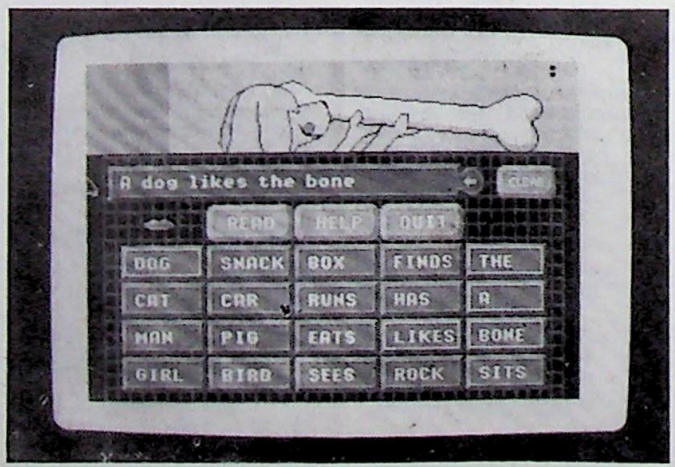
If the child pushes the wrong letter, the voice repeats the letter or the object, and the child may try again. I pushed the wrong letter over 20 times, and the voice just keeps coming back with the same thing. Apparently the child is to continue to hammer on the keyboard until he gets lucky, or accidentally hits the escape key and is rescued with a return to the main menu.

In Finding Letters, a keyboard is displayed on the screen, with one letter colored in. The voice says the letter (or a word that begins with that letter in the Word disk), and the child is to push the same letter on the keyboard that is colored in on the screen. This module proceeds sequentially through the alphabet, and again, the child may press the wrong answer as many times as patience allows. Once the right answer is selected, the program will move on to the next letter.

A fourth menu choice allows the child to adjust the volume from the screen using the mouse. This on screen volume adjustment is a common feature in the entire series so I won't mention it again.

This program has some good stuff in it, particularly for young children learning to recognize letters, and their location on the keyboard. However, the program is seriously flawed.

When you make a menu selection, a detailed instruction screen appears that must be read to know what you are supposed to do. The "voice" doesn't tell the child how to do the program—I guess they are supposed to read about it. The problem is, that if they could read, they wouldn't be doing this program. So, particularly for the young audience that this program is intended for, close parent or teacher supervision will be required until the child learns their way around the program. If this was the only program they used, they'd probably remember whether to press the space bar for menu choices, and what to do in each module (or how to get to a module to begin with), or how to get rid of the instruction



screen they can't read to begin with. A little cleanup in the user interface and the method of direction, would make this a very good program, instead of a program that is somewhat hard to get into for young children.

The module that teaches the location of the letters on the keyboard is a great idea. The Learning and Recognizing modules are similar (though not as good) to other early letter recognition programs already available for the IIGS.

To give you an idea of the loading times for all of these programs, the Letter disk, booted from a 3.5-inch drive on a ROM 01 IIGS, took over one minute just to get to the screen that says "Please wait while loading". Another forty seconds or so after that, you are presented with the main menu.

On a ROM 01 IIGS, with a TransWarp and booting from a 40 MB InnerDrive, the loading time was reduced to 20 seconds.

I would have a very hard time recommending that someone shell out $40+ dollars on this program. Better, smoother, easier to operate software is available for young children in the intended age group. The program does have some good features, but they do not overcome the handicaps mentioned above.

TALKING ADDITION AND SUBTRACTION

This program has four learning modules. They are Sets and Numbers, Picture Problems, Working with a Number Line, and Coloring through Math. It is recommended by the publisher for ages five to eight. Younger children (down to age three) can do some of the activities, however, parental help will be required even for some six- and seven-year-olds.

In Sets and Numbers, the child enters a number between 1 and 10 (if they can read the directions), pushes return (if they can read the directions), and then the number (both numeric and spelled out) appears on the screen, with that same number of objects. The voice says the number, and shows and talks through the counting of the objects, not as one, two, three, but as first, second, third, etc.

In Picture Problems, a math problem in addition or subtraction that uses the numbers 1 through 10, appears on the screen. Next to each number is a like number of objects. The child is to type in the correct numerical answer, which is then displayed with a like number of objects. The child is given another chance to get the right answer if he makes the wrong choice the first time. If the answer is still incorrect, the proper response is displayed on the screen and the program moves on to the next problem.

The concept of this module is very good, in that the child can better understand the math because of the objects that go along with each number.

In Working with a Number Line, a frog jumps along a line of numbers from 1 to 10. A problem is displayed at the same time, and the voice reads the problem out loud. So, the frog may stop on the number 3, and the voice (and display) says the problem is "3 + 1". The child must click using the mouse on the correct answer on the number line, and the frog will advance to that number if the correct response is given. After two wrong answers, the program gives the child the right answer, and the program continues. All the problems have two parts, so that in the above example, once the child selects the number four, the program then adds a number in the same problem such as "3 + 1 + 2 = ?". This is a good way for kids to better understand the interrelationship of numbers.

This module, is a very good learning concept. My kids had fun doing it, and stuck with it for quite a while.

In Coloring Through Math, a drawing of a hot air balloon scene appears on the screen. Different parts of the balloon have the numbers zero through ten written on them. The voice reads a problem as it is presented on screen, and the child is to click with the mouse on the correct word answer in the drawing. As each right answer is given, that part of the drawing is colored in, and a new problem appears.

When quitting this program, it turned my screen border black, and to change the border color, you must access the control panel (you don't have to do anything except go to the control panel, and exit it).

Loading time from a 3.5-inch disk was over two minutes, with over one minute just to get to the "please wait" screen.

I kind of like this program. It has some very good features, but as before, the instructions should be "talking," so that the child who can't read doesn't need constant supervision. In this case, the fact that the activities were high quality made the problems a little more bearable. I'd give this program a borderline purchase recommendation.

This concludes the short reviews of the first two programs in the Orange Cherry Talking Schoolhouse series. For short reviews of the other thirteen programs in the series, see the files in the folder **Orange.Cherry.Rev** on this issue's *GS+* disk.

COMING SOON

Features

Using a LaserWriter with your IIGS
Beginner's Guide To System Disks - Part III: All About The Installer

Reviews

Sinbad and The Throne of the Falcon
AE 3.5-inch drive
Rastan
Design Master vs Genesys
Captain Blood
Quickie Hand Scanner
Questmaster
Salvation: Renaissance
Pirates!

QIX
Programmed by Ryan Ridges
and John Lund

Retail price: $3 95
Copy-protected (Key disk)
Requires 1 MB RAM

Taito Software, Inc.
276 West Esplanade
North Vancouver, B.C. V7M1A5
(800) 663-8067 or (604) 984-3344

Reviewed by Dave Adams

Back in the early 1980's, when video arcades were at their peak, I used to spend my dollars getting thumped at every game available. I watched my cities burn in Missile Command, died horrible deaths at Star Wars, and all of my friends called r "Pac-wimp." Then one day a new game was introduced which spared me further ridicule. The game was QIX (pronounced "kicks"), and for some unknown reason I found that I had a special ability to play this game. I was spared social ostracism. Fast Forward to 1990. I admit that I am not the greatest Arkanoid player in the world, and my Mean 18 game has been known to cause serious players to get violently ill. I do OK at adventure games, but when it comes down to hand-eye coordination arcade style games, I cringe when forced to play. One day the phone rang and Diz was on the line. "I just got this game called QIX. Would y want to review it?" I jumped at the chance. Déjà vu.

THE BASIC STUFF

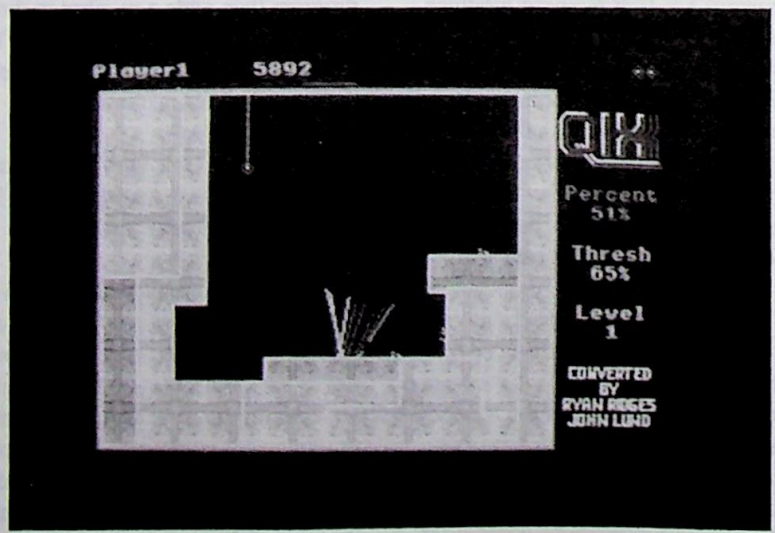
QIX comes on one 3.5-inch disk and is copy-protected. Like Arkanoid II (the previous game from the team of Lund and Ridges) you can make a backup copy but you will be required to insert your original game disk each time that you start to play. You can play the game with a joystick or you can opt to use the keyboard instead. The keyboard is adequate, but to really rack up the high scores, use a joystick. In QIX, you will literally live or die by the quality of your joystick.

The object of the game is the same as the arcade version. A computer virus (the QIX) has infested your system. For those of you that have never played QIX before, the QIX looks like those old brightly colored pickup sticks that you used to play with as a child. However these sticks are constantly moving, expanding, and contracting in various ways that constantly change the size and shape of the QIX and keep you guessing as to where it will move next. Your job is to isolate the QIX into as small a section of the screen by filling in sections of the screen (which represents the computer's memory) without touching the QIX and becoming infected. As if that wasn't enough, the QIX also spawns nasty little sub-viruses that can also kill you if they touch you. These sub-viruses are of three types: SPARX (which travel along your edges of the screen and any sections that you have claimed), SPRITZ (which float in the unclaimed memory), and the dreaded FUSE (which will follow you up your trail line and kill you unless you complete a section of memory). To finish a level you must reclaim (immunize) 65% of the memory. This is your base threshold to advance. Any extra percentage claimed earns you bonus points. As you increase in levels, the amount of memory that you must claim increases.

To immunize areas of memory, you must venture out from the sides of the screen and draw a line that will create a filled section. Your marker must start and end on either an immunized section or screen edge. You may draw this section at one of two speeds. A slow draw earns twice as many points as a fast draw. If you trap a SPRITZ in a filled section you will receive bonus points, and all fast draws will then receive the same point values as slow draws until you die. You must carefully balance the benefits of high points (slow draw) and the need to stay alive (fast draw) throughout the game. If any sub-virus or the QIX touches you, or you run across your own line, you are killed. On certain levels you are only required to split two QIX from each other to advance to the next level. Successful completion of that operation will multiply point values on the subsequent boards (i.e. points double, triple, quadruple, etc.). Extra lives are awarded every 50,000 points. All of these features are explained in the short, but excellent, instructions to the game.

THE GOOD STUFF

As soon as you boot QIX, you can't help but be impressed by the wonderful graphics and sound. The opening soundtrack and the following songs throughout the game are spectacular. If you have a stereo card in your GS you are really going to like this game. The graphic screens are colorful and of very good quality. The "unrolling" effect of the title screen in simply incredible. You can't help but be reminded of Arkanoid II throughout the game. If you liked those graphics then you will like QIX's graphics. The overall effect of the graphics and sounds are wonderful. The game itself



plays very smoothly and the screens are drawn very fast.

THINGS THAT COULD BE IMPROVED

The only major fault I have with QIX is the scarcity of earning extra lives, especially in the beginning portions of the game. As you increase in levels they are easier to obtain. In the initial stages it is not unusual to die often until you get the hang of the game. This can be frustrating for many people. If you have a modem there is a CDA (Classic Desk Accessory - available through the Control Panel) on various information networks (CompuServe, America Online, GEnie, etc.) that lets you cheat and award yourself extra lives. This is a most handy thing to have at times. Other than that there are very few features in the game that could be considered faults. QIX loads fairly quickly (much faster than Arkanoid II) and is a delight to play. Once again, the better the quality of your joystick, the better the scores that you will earn. Of course, no copy protection would be very nice....

SOME IDEAS TO PONDER

If you are just starting at QIX, the most important thing is to be patient. There is no time limit on any screen. The sparks are not a big threat in the early levels. You can slow them down by creating irregular outlines along the edges of the screen. The biggest thing is to come up with a way to deal with the QIX. Some people stay away from it and try to rack up points with slow draws. Others try to use fast draws to divide the screen up into sections (using very thin, long lines) getting as close to the edge as possible. Then they use a slow draw to complete the section to get a large amount of points and claim a large chunk of the screen. A mixture of both plans usually works well. Watch the QIX and SPRITZ at all times, and don't stay in one place too long. Even with all of this, a certain amount of luck and skill comes into play. This game requires strategy, patience, and quick action when necessary to win. Good luck.

SUMMARY

QIX is an excellent game. I highly recommend it. Ridges and Lund (and Taito) are doing wonderful things for the IIGS arcade game market. Please support them by not pirating. QIX is another of those "Easy to play, hard to master" type of game that gets you addicted, and keeps you playing for a long time.

THE MOLEHILL

...continued from page 18

After things have been started, the file information message needs to be retrieved from the Message Center. To do this, a memory handle must be created for the Message Center to use. After the handle is allocated, the _MessageCenter call is made with the getMessage parameter for message ID number 1, which is the assigned ID number for file information. On return, the handle has been resized so that the information will fit and the message is retrieved. Then the message is deleted so confusion doesn't result from old messages hanging around. The message format for the retrieved message is shown in Figure 1. The open/print flag is checked to make sure that the document is being opened and not printed. If it is being printed, the program just returns to the launching application (usually the Finder) without performing any action. If it is being opened, the EXE file is loaded into memory with the _InitialLoad tool call. EXE.Launcher will only launch one EXE file. If there is more than one pathname in the message, EXE.Launcher will ignore them and only launch the first EXE file in the message list.

Once the EXE file is in memory, it will be called with a jsl instruction. Before the jsl, however, the registers must be set properly. First, the accumulator should contain the UserID returned from the _InitialLoad call. Then the X and Y registers are set to zero telling the EXE file that there is no shell identifier or command line. Some shells have an eight-byte shell identifier followed by command line parameters that may be passed to the program. EXE.Launcher does not support either of these options, so it sends zeros. After the EXE file returns to EXE.Launcher with a rtl, EXE.Launcher performs a _UserShutDown tool call on the EXE file's UserID which removes the EXE file from memory. Then the tool sets are shut down and EXE.Launcher terminates with a standard GS/OS quit call.

Some EXE files will terminate with a GS/OS quit call instead of a rtl. EXE.Launcher does not trap the quit call as some shells do. But since the EXE program ends with a quit call, you should be able to change the file type of the program to S16, instead of EXE, and launch the program normally from the Finder.

Even with the limitations and side effects of launching additional file types from the Finder, the point is that it can be done, and it can be done rather simply.

If you find a problem with EXE.Launcher, be sure to fill out the problem form on your *GS+* disk and send it in. If you have any questions or comments, please be sure to send those in as well!

FIGURE 1 - FILE INFORMATION MESSAGE FORMAT

```
4 bytes - Handle to next message
2 bytes - message type ($0001 for file information.)
2 bytes - open/print flag ($0000 = open, $0001 = print.)
pascal string - pathname of file to open or print
pascal string - yet another pathname
(more pascal strings)
1 byte - always $00 to signify the end of the message
```

SOLITAIRE ROYALE

IIGS version: programmed by Jake Hoelter
graphics by Jody Sather
sounds by Ilan Zipkin

Retail price: $34.95
Typical mail order price: $22
Not copy-protected
Requires 512K RAM (768K recommended)

Spectrum HoloByte
A division of Sphere, Inc.
2061 Challenger Drive
Alameda, CA 94501
Orders: (415) 522-3584
Customer Support: (415) 522-1164

Reviewed by Noreen Ribaric

From the people that brought us Tetris, a new package of solitaire card games has been released for the IIGS. There are eleven different solitaire games in all—eight for adults: Pyramid, Golf, Klondike (the most commonly known solitaire game), Canfield, Corners, Calculation, Three Shuffles and a Draw, and Reno; and three for children: Concentration, Pairs, and The Wish.

Solitaire Royale supports the Apple Human Interface Guidelines. By this, I mean it is a desktop program that is mouse- and menu-driven, and supports desk accessories. The menu bar in Solitaire Royale contains 6 menus: Apple, File, Edit, Start a Game, Help, and Settings. The Apple menu is where you will find your desk accessories, plus an About Solitaire... item and a How to Play item (which is only selectable if you are actively playing a game). The File menu allows you to save a game in progress, open a saved game, close the current game, or quit. The Edit menu has no use in Solitaire Royale, but is included for desk accessory support. The Start a Game menu is where you select one of the 11 games to play. The Help menu gives you information on how to play Solitaire, and also ways to "cheat" while playing a game. The Settings menu contains options such as selecting from 5 different types of decks, each with a different colorful design (if you have more than 512K of RAM), and whether to show a left-handed pointer or not. You can also change the desktop color from this menu, along with a few other things.

The game is not copy-protected and can be copied to a hard drive. Although the manual says the program must be copied into a folder in the root directory of your hard drive, in fact, it can be copied into any directory.

The games are very easy to play. You pick your choice of the eleven games from the Select A Game menu. The cards are then dealt out and the game begins You select the card you want to move by clicking the mouse on it once, then you click again on the pile you want to move it to. This is different from the click-and-drag approach that I've found common in a lot of other card game programs. All the games are fun to play, as well. (Well, the children's games were a bit too easy for me, but I'm sure they're perfect for who they are meant for—and Steve liked them, too, because he could win those!) There is a wide variety in the play of all the games; they are not just slightly different from each other. Some of the adult games are harder than the others. Calculation and Three Shuffles... both require logic and planning to win (I finally won Three Shuffles..., but have yet to win Calculation—Arggh)! Some of the others are won mostly by luck. Solitaire Royale's authors did a good job of selecting the most appealing and entertaining solitaire card games for this program.

There are also multiple game options. One of them, Tour (Aunt Anne's Game) takes you through all 8 adult games and totals the points earned in each of them. A scoreboard keeps track of the top 5 scores earned during Tours. They can be viewed from a menu that replaces the Start a Game menu while playing a Tour. I found myself playing Tours over and over again, since the 8 different games make it hard to get bored, and the challenge to better your score (and win one of the games) gets you addicted. Another option is Tournament Play. This allows you to select one of the adult games, and will allow a number of players to play the same game (meaning the cards are laid out exactly the same for all the players) so they can compete against each other. The top 5 scores for each of the 8 card games are kept for Tournament Play, as well.

The documentation for this program is very good. Although the manual is for the Macintosh version, almost everything in it pertains to the IIGS version as well, and an addendum sheet is included to explain the few differences. The manual not only explains how to use the program, but it also gives very good instructions and rules for all of the card games. The rules can also be viewed in a window while you are playing!

This program is very well-designed. The colors and graphics are very appealing, and there are realistic sound effects in the appropriate places (shuffling, dealing, and cheering when you win) if you have more than 512K of RAM. You must have 1 MB of RAM to get all the sound effects, but if you have 768K, you get all but one. I think this concept of varying the amount of "extras" depending on how much memory you have should be incorporated into all programs. (Some programs require you to have the 1 MB of RAM to even run the program, instead of disabling some options.) I only found one small bug in the program. There is usually a small window in the bottom of the screen that displays the name of the game you are playing and your current score; however, it did not appear when playing Klondike.

This program does an excellent job of bringing the most popular solitaire card games to the IIGS. The wide variety of games can occupy your time for hours. I found myself getting addicted to it, especially since I wanted to win at least one of the games, and had to tear myself away to write this review. I have played many other computer card game programs for several computers and I found Solitaire Royale to be one of the best!

TRASH CAN AWARD

Inner*Express*
Retail price: $129
Typical mail order price: $89

New Concepts, Inc.
665 West Jackson Street, Suite #2
Woodstock, IL 60098
(815) 338-4227

Reviewed by Steven W. Disbrow

"Speeds up Falcons, InnerDrives and OverDrives by 200%-300%"
For the last few months, that is the ad copy that has been used to sell the Inner*Express* accelerator for hard drives produced by the now defunct Ingenuity, Inc. Originally, the Inner*Express* was to be sold by Virtual Realities, Inc. But, they went under as well, and the product was taken over by New Concepts, Inc. Being an InnerDrive owner myself, and being perpetually behind schedule, I was very interested in anything that could speed up operations here at the office. Our InnerDrive was already amazingly fast; speeding it up by 200% could only make my life easier! As soon as the product was available, I got myself one and prepared to plug it into our InnerDrive here at the office.

Reality
Reading the Inner*Express* manual, some things began to strike me as... odd. According to the manual (and the ads that I have seen), a "stock" InnerDrive, without the Inner*Express*, can only read/write 81 blocks per second. I was amazed that my InnerDrive was so slow! It booted to the Finder in 25 seconds, which (considering all the junk I keep on my hard drive), I thought was pretty good. So, I tried duplicating their test: All you do is select the drive in the Finder and select "Verify" from the Disk menu. You then take the number of blocks on the disk and divide it by the number of seconds that it takes the Finder to verify the disk. This gives you a result in blocks per second. When I did this, I was astonished to find that my InnerDrive was reading 213 blocks per second! Without the Inner*Express*! Surely there must be some mistake! The Inner*Express* manual said that with Inner*Express* I could expect a top speed of somewhere around 284 blocks per second (with a TransWarp GS). Even without a calculator, I could see that this was a far cry from the 200% to 300% speed increase claimed in the ads! Being the inquisitive type, I was determined to install the thing anyway. A couple of hours later, I had the Inner*Express* installed and prepared myself to be sped back to the Finder in record time. A mere 25 seconds later, I was back at the Finder!

Hoping that it was just a (very) bad dream, I reran the Verify Disk test. Indeed, my InnerDrive was faster! Installing the Inner*Express* had increased the speed of my InnerDrive from 213 blocks per second to 234 blocks per second.

Using the formula in the Inner*Express* manual (**(increased speed / original speed) x 100**) and the wording in the Inner*Express* ads, it could be stated that the Inner*Express* "sped up my InnerDrive by 110%." However, this would be an incorrect statement. This formula gives a percentage of the original speed, and not an increase in speed. The correct statement would be that my InnerDrive was now running at 110% of its original speed. If the Inner*Express* had not sped up my InnerDrive at all, it would still be running at 100% of its original speed! Therefore, the correct statement would be that my InnerDrive had been given a 10% speed increase by installing Inner*Express*. This can be arrived at by using the formula (**(increased speed - original speed) / original speed) x 100**. Confused? I was, too.

Needless To Say...
I was disappointed. I called New Concepts and asked them what the deal was. They had no idea. I then contacted the author of the Inner*Express* firmware. He said that the tests done for the manuals (and later included in the ads run by New Concepts) were performed on an older InnerDrive without any kind of driver.

So, the ads were true, but incredibly misleading! You see, Ingenuity included with each InnerDrive this nifty little GS/OS driver called, the InnerDriver. It sped the drive up immensely when used with GS/OS. That's why my InnerDrive was reading 213 blocks per second—I was using the InnerDriver that had been installed on my InnerDrive at the factory. However, if you run your InnerDrive without the InnerDriver, it's a real dog. One of our readers that had also bought an Inner*Express* ran some tests for me (before he installed his Inner*Express*) to verify this (thanks Norman!). Without the InnerDriver installed, his InnerDrive read a measly 45.8 blocks per second. With the InnerDriver installed, his InnerDrive read a whopping 218 blocks per second. Using the formula in the Inner*Express* manual this comes to a 470% speed increase when using the **InnerDriver!** These numbers look remarkably similar to the numbers New Concepts is running in their ads!

So, the comparisons that are being run in the Inner*Express* ads are, basically, an InnerDrive with no custom GS/OS driver versus an InnerDrive with a custom GS/OS driver. This is not a fair comparison, because the InnerDrive already comes with a custom GS/OS driver, the InnerDriver! What they need to do is run a comparison of the Inner*Express* versus the InnerDriver. Without this crucial comparison, and by misstating what their formula actually represents, the ads for Inner*Express* are very misleading!

However, I suppose it would be very difficult to sell this product for $129 using such a comparison in the ads!

Bottom Line
If the Inner*Express* was being sold for what it truly is (an alternative, custom GS/OS driver for the InnerDrive that will allow die-hard speed-freaks to get a bit more out of an already fast hard drive), and it were sold at a realistic price (no more than $50), it would not have gotten this Trash Can Award. In fact, I would probably recommend it! However, the current advertising for Inner*Express* is very misleading and the performance of the product simply does not justify such an outlandish price.

GS+ BACK ISSUE INFORMATION

September-October 1989 (Volume 1, Number 1)

$4 magazine only
$5 disk only
$6 magazine + disk

- 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

November-December 1989 (Volume 1, Number 2)

$4 magazine only
$6 disk only
$9 magazine + disk

- Beginner's Guide to the Finder - Part 1: The Basics
- EGOed - An NDA text editor (TML Pascal II source code on disk)
- Update Info - AWGS v1.1, HyperStudio v2.0, System 5.0.2
- Brush with Greatness - Tips on drawing faces (pictures on disk)
- Reviews of TML Pascal II, Source Code Library II, Cutting Edge Keyboard, Battle Chess, Dark Castle, Dungeon Master, Neuromancer, Laser Force

January-Febrary 1990 (Volume 1, Number 3)

$4 magazine only
$6 disk only
$9 magazine + disk

- Beginner's Guide to the Finder - Part 2: Mousing Around
- 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)
- Reviews of HyperStudio v2.0, Graphic Disk Labeler, Programmer's Online Companion, Xenocide, Keef the Thief, Life & Death, The Three Stooges

PLUS: Icons, Rumors, and more!

March-April 1990 (Volume 1, Number 4)

$4 magazine only (SOLD OUT!)
$6 disk only
$9 magazine + disk (SOLD OUT!)

- Beginner's Guide to the Finder - Part 3: All About Icons
- 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
- Reviews of InnerDrive, Vulcan, Salvation - Guardian, ORCA/Dissembler, Computer Eyes (with digitized pictures on disk), Jam Session (with songs on disk), Ancient Land of Ys, Tunnels of Armageddon, Where in the World is Carmen Sandiego?

May-June 1990 (Volume 1, Number 5)

$4 magazine only
$6 disk only
$9 magazine + disk

- AppleFest Report
- Beginner's Guide to System Disks - Part 1
- All about GS/OS prefixes - with 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, plus a sneak peek at the Zip GS

July-August 1990 (Volume 1, Number 6)

$4 magazine only
$6 disk only
$9 magazine + disk

- KansasFest Report
- Beginner's Guide to System Disks - Part 2
- Transfusion - An NDA terminal program that allows you to go online from within any desktop program (and perform XModem file transfers in the background)!
- 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?

All prices include $1 shipping and handling. For orders outside the U.S.A., Canada, or Mexico, add another $1 for surface mail, $5 for air mail (per order, not per issue). Mail back issue requests to: *GS+* Back Issues, c/o EGO Systems, P.O. Box 15366, Chattanooga, TN, 37415-0366, or call (615) 870-4960 to verify availability. Please include your **phone number** on all orders placed by mail (in case we are sold out of an issue)! For Master Card or VISA orders placed by mail, also include your card number, expiration date, and signature.