New! HyperCard IIgs
The Next Step In Multimedia

System Software For GS and Mac
All in the Apple Family

Hard-Disk Drives
The SCSI Side of Life

Reviews:
BannerMania
LogoExpress
Gold Rush!
EuroWorks 3.0
THE APPLE II CULTURE REBORN

By Paul Statt, Senior Editor

HyperCard — Apple’s popular multimedia programming product for the Mac — is reviving the do-it-yourself spirit of the Apple II.

The best thing about any Apple computer has always been its accessibility — that anybody with the urge could learn to program and create unique software. But as Apple’s computers have become more powerful and more sophisticated, they’ve also gotten harder to program. The Apple IIs and the Macintosh, for instance, don’t have a native computer programming language built in, as BASIC was in the original Apple II. Like all Apple IIs, the GS comes with Applesoft — but the machine’s native mode is the 16-bit 65816.

HyperCard — for the Mac and now for the Apple IIs — brings the hacker’s spirit back to Apple. HyperCard IIs is a nearly perfect clone of the Mac version — with all the advantages and disadvantages of Apple’s Macintosh HyperCard 1.25. It adds color — a feature not found even in HyperCard 2.0, the latest Macintosh version.

HyperCard IIs, like HyperCard for the Macintosh, is a way to organize information. But it’s much more than a database manager. At its most prosaic, HyperCard is a
programming language. At its most poetic, it’s a way of life. As a language it’s up-to-date, making use of what’s known as object-oriented programming.

Programming with an object-oriented language is like building a car on an assembly line: All the parts are there; your job is simply to put them together. The HyperCard lifestyle is as old-fashioned as the original idea of the Apple computer — that anybody should be able to write a program, that businesspeople should write business software, that teachers should write educational software, that kids can even create their own.

As you read about HyperCard IIcs, you may think you have to learn to write HyperCard programs — called scripts — to enjoy it. That’s not true. Thousands of programmers have already written HyperCard stacks on subjects ranging all the way from ecological awareness to pornography.

Teachers will appreciate HyperCard’s five levels of control, from browsing in the stacks (other people’s) to scripting your own, with typing, painting, and authoring in between. But just about everybody who uses HyperCard — a program designed to put you in control — will want to write his or her own stacks. Basically, with HyperCard you can do anything the GS can do. Scripting is where the fun begins.

**ON THE BUTTON**

The key to understanding HyperCard is understanding the role of cards, stacks, fields, buttons, and backgrounds — the objects the HyperTalk programming language gives you to play with.

A card is what you see on a single screen (as in Figure 1); a stack is a collection of cards. You could think of it as a stack of index cards, but these cards are live, and you control them. In HyperCard IIcs, a card must fill the entire screen, although it’s possible to hide the menu bar at the top. (HyperCard 2.0 on the Mac allows cards of different sizes.) The text and graphics you see on the card appear on two levels: background elements, which carry over from card to card, and the material unique to this card alone.

A background (Figure 2) can be as simple as a white screen, or as pretty a picture as you can scan, draw, or paint with HyperCard’s built-in graphics tools.

Every card needs a background; in general, as we’ve noted, you put things you want to see on more than one card into the background. If you don’t want to see anything repeated on other
cards, you’ll include an empty background.

In addition to graphics, most backgrounds contain one or more buttons, the switches or controllers that make something happen (display another card, play music, perform calculations, get help, print, use a modem), and fields, areas containing text or numbers.

Common background buttons include the home, next card, and previous card icons (small pictorial representations, Figure 3). One misconception is that a HyperCard stack must have these buttons, but they aren’t necessary. (See the accompanying sidebar, “Why Did You Do It That Way?” for a report on button programming.)

<table>
<thead>
<tr>
<th>Table. Elementary functions used in HyperCard IIgs, as specified by the Standard Apple Numerics Environment on the Apple IIgs’ 65816 microprocessor.</th>
</tr>
</thead>
<tbody>
<tr>
<td>log2(x) computes base 2 logarithm of x</td>
</tr>
<tr>
<td>ln(x) computes natural (base e) logarithm of x</td>
</tr>
<tr>
<td>ln10(x) computes natural logarithm of (x+1)</td>
</tr>
<tr>
<td>exp2(x) computes 2^x</td>
</tr>
<tr>
<td>exp(x) computes e^x</td>
</tr>
<tr>
<td>exp1(x) computes e^x - 1</td>
</tr>
<tr>
<td>cos(x) computes cosine of x</td>
</tr>
<tr>
<td>sin(x) computes sine of x</td>
</tr>
<tr>
<td>tan(x) computes tangent of x</td>
</tr>
<tr>
<td>atan(x) computes arctangent of x</td>
</tr>
<tr>
<td>random(x) computes pseudorandom number with x as its seed</td>
</tr>
<tr>
<td>compound(r,n) computes (1+r)^n</td>
</tr>
<tr>
<td>annuity(r,n) computes 1 - (1+r)^-n / r</td>
</tr>
</tbody>
</table>

Buttons, whether background or unique, can appear as almost any graphic (Figure 4). Some don’t appear at all — they can be invisible. Some buttons look like the electrical switches on which they’re modeled, some look like round rectangles, some look like pictures (icons). HyperCard includes a number of icons in a special file you can use in your stacks. An Icon Editor stack that lets you draw your own is included with HyperCard IIgs (Figure 5). Triad Ventures has also written a desk accessory, included in its HyperCard IIgs Utilities package, to help you create icons.

WORD AND IMAGE

Buttons are one way HyperCard gets graphics onto the computer screen, but hypermedia includes good old text, too — in objects called fields (Figure 6), as we noted above.

If you’ve had any experience with database
"Why Did You Do It That Way?"

“In Italy for 30 years under the Borgias they had warfare, terror, murder, and bloodshed, but they produced Michelangelo, Leonardo da Vinci, and the Renaissance. In Switzerland they had brotherly love, they had 500 years of democracy and peace, and what did they produce? The cuckoo clock.”

— Orson Welles as Harry Lime, "The Third Man," 1949

While nothing as dramatic as murder and bloodshed occurred during the creation of HyperCard IIs, we did have an earthquake, but that was probably just a coincidence, users are often justifiably curious about the process of designing and creating a program as complex as HyperCard IIs. Usually the question is posed while screaming for the head of the "idiot who wrote this thing," but nevertheless, what follows are some of the issues we dealt with while writing HyperCard IIs.

At the most basic level, our major design requirement was to make an Apple II version of HyperCard that was completely compatible with HyperCard 1.25 on the Macintosh computer. At the same time, we wanted to make improvements to the program that would benefit users and take full advantage of the capabilities of the Apple II — hence a color HyperCard with improved printing and reporting capabilities, as well as additional functionality in the HyperTalk scripting language.

A complete description of how we designed and wrote HyperCard IIs would be extremely long, so as one of the two engineers who worked on HyperTalk, I’m going to confine my discussion to just two features in HyperTalk that I believe represent some of the design decisions we made. The play command. We agonized over the design of the play command for a long time. In Mac HyperCard, Play lets you utilize the single-voice sound capabilities of that machine. The play command produces the requested sounds immediately. Multiple play requests are executed sequentially, not simultaneously. The GS, however, can generate more complex sounds than the Mac.

Naturally, we wanted HyperCard IIs to be able to take full advantage of the GS sound capabilities, but we were under several constraints. First and foremost, the GS version had to be completely compatible with the Mac play command. In addition, there were language constraints. HyperTalk, as a language, is supposed to be English-like and simple to read and understand. Whatever syntax we arrived at had to be comprehensible by the average user, not just acoustic engineers. And finally, whatever we came up with had to be useful to the majority of users — we weren’t going to add a super-powerful command only three people would need.

Initially we were extremely ambitious, and tried to create a syntax that would give you extensive control over the considerable sound capabilities of the GS. Regrettably, designing a syntax that was powerful, understandable, and backward compatible proved unrealistic, given the time constraints we were under. The language-design issues were difficult because of a rather odd situation — the GS’ sound capabilities were too powerful! We simply couldn’t figure out a way to unleash all that power and have the syntax understood by nonprogrammers.

We then tried a much simpler approach, a natural extension of the original syntax: one that would let you direct a particular sound to a specified channel, and optionally delay its execution. After setting up several sounds on separate channels, you could then tell the GS to start, and all sounds would play simultaneously. We had high hopes for this approach, but we were tripped up by the problem of synchronizing multiple sounds. Specifically, we had trouble resolving the language issue (and it’s a lot more involved than you might think): how to accurately make sound B start 10 seconds, or even harder, 10 beats, into sound A.

Deadlines were looming, so rather than...
ming.) You can attach an XCMD either to a single stack or to the HyperCard program itself. Triad Ventures, for instance, has already written some XCMDs to facilitate playing music (available in its MIDI Music package).

PLUS AND MINUS

One of HyperCard’s secret strengths is that it’s a pretty good database manager (Figure 8). Apple plays that down because it doesn’t want software companies to complain that Apple’s controlling the market.

Besides, "information engine" sounds more glamorous than "database manager." But teachers who need class records and individuals who don’t need relational structured query language (SQL) just to keep Christmas card lists will find that they can write or buy HyperCard stacks that do the job.

Actually, HyperCard IIs’ number-crunching power would make some database managers envious. Financial functions such as annuity calculation and compounding for computing interest, and mathematical functions such as natural logarithms and trigonometric functions (see the accompanying Table), are available — and you won’t find those in AppleWorks Classic. HyperCard IIs uses Standard Apple Numerics Environment (SANE) protocol, which gives you access to a wealth of functions, and can speed up calculations considerably if you add a math coprocessor to your GS.

The program’s most obvious strength is that your GS will be able to run stacks created for the Mac, and your Mac will be able to run GS stacks. At press time, the pair of stacks that perform the conversion, HyperMover GS and HyperMover Mac, weren’t available. But if you compare the look of a Mac stack and its Apple IIcs cousin, you can see that converting one to the other shouldn’t be too difficult.

We tried to copy the text of a HyperCard program, called a script, from a GS to a Mac LC via an AppleTalk network, but it wasn’t quite that simple. The text arrived intact, but the program didn’t quite work on the Mac. You’d need to...

---

"Mac or GS, HyperCard brings the hacker’s spirit back to Apple."

---

John Lawler, Apple Computer
change some commands, but you could do the translation in a mechanical way — doing a find-and-replace on the offending structures. Tedious for you, but a snap for HyperMover.

The HyperMover programs will be available to developers and user groups. But other challenges remain: Stack conversions will pose a tricky legal question, for one thing. If you buy, say, a company's GradeBookStack for Macintosh HyperCard, do you need to buy another copy or a site license to run it on HyperCard IIcs? If you can legally use any Mac stack, a lot of stacks — commercial as well as public-domain — will be available for HyperCard IIcs.

A number of stacks are already available for Roger Wagner's HyperStudio — you don't even need to own HyperStudio to use some of them if they were created with the runtime version of the program. (See "Shareware Solutions," What's New, p. 20 in this issue.) That won't be possible with HyperCard.

It remains to be seen how many stacks will be written for HyperCard IIcs. Mac HyperCard ignited interest initially among commercial developers, but that burned out rather quickly. At press time, we knew the Boston Computer Society was writing a disk of GS shareware stacks, and Triad Ventures had written a commercial MIDI Music stack using Apple's new GS MIDI Synth tool.

**AUDIO AND VIDEO**

Given the musical talent of the GS, it's a shame HyperCard IIcs can't sing on its own, or at least play the piano. The version we tested had a choice of two musical instruments: harpsichord and a "boing" sound, which simply doesn't sound very musical. A Scripter's Tools stack included with HyperCard IIcs, however, contains 11 other sound resources for instruments such as piano and guitar, which you can attach to a single script or to the program itself. (See the sidebar, "Why Did You Do It That Way?" for details.)

HyperCard doesn't have built-in sound digitization, however, as HyperStudio does. (See Figure 9. HyperStudio also includes a microphone, much like the Mac LC, making it easy to add sound to a stack.) Is Apple avoiding putting too much musical muscle into any of its products because it fears legal complications with Apple Corps, the Beatles' record company?

As we noted above, the current version of HyperCard IIcs also doesn't include commands for running a videodisc player, as HyperStudio does. HyperStudio has to offer lots of extras, because, lacking a HyperTalk-style program-

---

**HyperCard IIcs script**

```plaintext```
```on mousedown```
  ```set numberFormat to 0.00```
  ```get card field "Amount"```
  ```multiply it by card field "Rate"```
  ```divide it by 100```
  ```put it into card field "Interest"```
  ```add card field "Amount" to it```
  ```put it into card field "Total"```
  ```divide it by 12```
  ```put it into card field "Monthly"```
```end mousedown```
```HyperCard 1.25 script```
```on mousedown```
  ```set numberFormat to 0.00```
  ```get card field "Amount"```
  ```multiply it by card field "Rate"```
  ```divide it by 100```
  ```put it into card field "Interest"```
  ```add card field "Amount" to it```
  ```put it into card field "Total"```
  ```divide it by 12```
  ```put it into card field "monthly"```
```end mousedown```
```Figure 6.``` At the top, HyperCard IIcs; on the bottom, HyperCard 1.25 for the Mac. This card uses the mathematical muscle of HyperCard to compute, in a not very elegant way, the monthly payments on a year's loan. The HyperTalk script for the "Calculate" button is identical on both cards — it was copied from Apple's HyperTalk Beginner's Guide. This demonstrates the relative ease with which Mac and Apple IIc stacks can be converted to each other's format. With the exception of the background buttons and the "Calculate" button, all objects on these cards are fields.
```Figure 7.``` HyperTalk scripts for the HyperCard IIcs and HyperCard 1.25 versions of a button that calculates interest (Figure 6) are identical; except that in the Mac script, the lines "on mousedown" and "end mousedown" appear automatically, in HyperCard IIcs you need to write them out. HyperTalk sounds enough like English that reading a script is painless; writing in HyperTalk requires more discipline.
ming language as it does, adding features to the program is hard.

SPEAKING THE LANGUAGE

HyperCard is a general-purpose database manager, paint program, programming language, and even something of a word processor. The drawback? HyperCard — either version — is slow. HyperTalk simply adds another layer between you and the machine. HyperTalk is an interpreted language, which means that each line of code must be translated into something the GS understands before the machine starts to do anything. The advantage of an interpreted language is that the same code works on different machines. That's especially clear in the case of HyperCard: It's easy to write code that works on both the GS and the Mac.

HyperTalk is a plain-speaking, unaffected programming language, but a programming language nevertheless. You have to mind your Ps and Qs when you write a HyperCard stack — make sure you spell everything correctly and leave the right number of spaces.

HyperTalk is forgiving in some ways. It's unusual in that it lets you use two different words in certain cases to mean the same thing — slow and slowly, or go to and go, for instance. But it's about as much like spoken English as the commands players type into a text-adventure game. It requires some discipline to make yourself understood.

Pricing and Availability

At press time, final decisions about the price and distribution of HyperCard IIs have not been made. Jane Lee, product manager for HyperCard, told us in November, though, that the price "should be around $99." That's what HyperCard for the Macintosh costs. "It will initially be distributed by Apple," Lee said, in contrast to HyperCard Mac, which is now distributed, in part at least, by Claris, Apple's software subsidiary. "It will be available through dealers and APDA (the Apple Programmers and Developers Association, phone (408) 562-3810, (800) 282-3732), and we are pursuing licensing by Apple-authorized user groups."

Lee noted that because "you can't use [HyperCard IIs] out of the box with your new [Apple IIs] CPU — it needs 1.5 megs and a hard drive or network to run — HyperCard IIs will not ship with the CPU," as Macintosh HyperCard does.

In addition, Lee said, "HyperMover will be available to developers and through APDA."
HyperStudio is easier to use than HyperCard, if by that you mean you can write a program without writing a line of code. HyperStudio never gets to the scripting level. It’s indeed possible to build a HyperCard IIgs stack without HyperTalk by simply combining elements, but it’s like baking a cake from a mix — the result will be edible, but bland. In any case, HyperTalk isn’t exactly difficult. It strives to look like English, and although the impersonation isn’t always successful, the meaning of a line such as

|put the date into field “Today”| should be pretty clear. You’ll need to take some time, and perhaps read a book or two, before you can write a script in HyperCard.

Addison-Wesley publishes excellent HyperCard references, and at press time also had a HyperCard IIgs book in the works. You may need to spend a weekend learning about it; an evening might do it. But you won’t need to take a college-level course as you would to learn Pascal or C.

If you don’t want to type even one line of program code, HyperStudio can do much of what HyperCard can, and some things it can’t (such as videodisc control and sound digitization), simply by pointing and clicking a mouse. Some teachers will want their students to have the chance to create stacks without programming; others may want students to learn to write a computer program.

An interesting aspect of HyperCard. GS or Mac, is that you never need to save a stack. If you’ve experienced a crash or two, you’ll understand the wisdom of this feature in a programming language. HyperStudio, which lacks the language, lets you decide when and if you want to save. If you want to experiment with HyperCard, you have the option to Save a Copy of your work up to a certain point.

A BIG PROGRAM

To use HyperCard, you need a big GS: 1.5 megabytes of memory and a hard-disk drive or network. Those are the official system requirements. We used a 1.25-megabyte GS with a 40-megabyte hard drive to test a beta version of the program. It was usable with that setup, but delays and out-of-memory messages were frustrating. The system ran more smoothly with 4 megabytes of RAM and an accelerator.

Easter speed really isn’t crucial, but the hard drive is. After all, the whole point of HyperCard is that it’s a way to manage lots of information — it gets interesting only when you have lots of cards. Some of the best Mac stacks, for instance — such as The Visual Almanac, shown in Figure 2 — run from CD-ROM discs with half a million megabytes. An AppleTalk network would be a good way for a school to run HyperCard IIgs.

Not everyone has hard-disk drives, CD-ROM players, and local-area networks, though. That’s the crucial advantage HyperStudio has over HyperCard IIgs: HyperStudio runs on a GS with 512K of RAM and a 3.5-inch floppy-disk drive. You can even create stacks that run with less memory, and run without HyperStudio.

But let’s not take anything away from the supreme achievement of HyperCard IIgs — it’s a Macintosh program that runs on the Apple IIgs. The hundreds of Mac HyperCard programs that are available are now also available for the GS. HyperCard IIgs should actually help make HyperCard legitimate, and help maintain Apple’s dominance in the school computer market.

The irony is that HyperCard may eventually help make the Macintosh legitimate in schools. Schools that have GSes can now use HyperCard IIgs to run existing Mac stacks, many of which are designed for education. Schools, or teachers, who buy Macs can use the new machines to write educational software that will run on both Macs and Apple IIgses. The twain have met.