4. EXPERIMENTAL CONTROL AND SIMULATION

Chaired by Jonathan Vaughan, *Hamilton College*

Using HyperCard and Apple events in a network environment: Collecting data from simultaneous experimental sessions

RON HOFFMAN and JOHN MACDONALD
Dalhousie University, Halifax, Nova Scotia, Canada

HyperCard is a useful tool available to teachers and researchers who have access to Macintosh computers. Version 2.1 of HyperCard, in conjunction with the Macintosh Operating System 7, includes some significant new capabilities for users with multiple computers in a network, as is common in many teaching laboratories. This paper describes a HyperCard stack that allows users who wish to run HyperCard-based experiments on several computers simultaneously to conveniently retrieve experimental data from those remote computers onto one program, and to perform several other useful tasks in a computer network.

HyperCard has become a popular tool for researchers and teachers, who value its hypertext capabilities and the relative ease of programming in the HyperTalk language. This authoring software, supplied with every Macintosh computer since 1987, has been used to construct a variety of computer-based applications, such as a courseware stack on subjective contours and visual illusions (Dougherty, 1990); a hypermedia resource library for behavioral research (Ray & Mitchell, 1992); interactive tutorial stacks on person perception and attribution theory (Petty & Rosen, 1991); tools for producing CAL software (Levidow, Hunt, & McKee, 1991; Martin & Growney, 1992); experiments in cognition (Tversky, 1990); administration of a block-design task (Martin & Wilcox, 1989); preparation, administration, and analysis of tests and surveys (Geisler-Brenstein & Brenstein, 1989); administration of figural tests (Schulberg & Nichols, 1990); and research on topics such as problem solving (Blaye, Light, Joiner, & Sheldon, 1991; Carter & Walsh, 1992; Quinn, 1991), text processing (Nason & Zabrocky, 1988), response modality in a digit-span task (Martin & Allan, 1991), and tutoring systems (Lee, 1992). The range of tasks to which HyperCard has been applied is a testimonial to both the flexibility of the program itself and the ingenuity of its users.

Typically, such applications were designed to be used on a single computer at a time. Researchers wishing to run multiple subjects could design their programs to collate data from successive sessions on a single computer for later retrieval, but if they had access to several computers on which they could simultaneously run subjects, collating data was problematic. In teaching situations like that in our experimental psychology laboratory, where students run themselves through experimental procedures in a facility with one computer for each student in the class, gathering the data afterwards has been a manual process, only slightly improved with the option of having each computer print its data onto one of the network printers for later collation and analysis. What was missing was a mechanism for one computer to poll each of the other computers, and to retrieve electronically the appropriate data to a central program where the desired treatment of the data is performed.

A potential solution to this problem came in 1991 when Apple Computer released a new version of the Macintosh operating system, System 7, which implemented a form of interapplication communications utilizing what were termed Apple events (Apple Computer, 1991c). These Apple events are in essence signals adhering to a defined protocol which can be passed between and understood by different computer programs, either on the same computer, or on different computers connected via a network. And, although at the time of the formal public release of System 7 most of the major commercial software suppliers had not yet successfully updated their prod-

The authors wish to thank Fionnuala Murphy, Michael Antle, Imogen Fox, Janice McPhee, and Graham Schmidt for their assistance in testing this software. Correspondence should be addressed to R. Hoffman, Department of Psychology, Dalhousie University, Halifax, NS, Canada B3H 4J1 (e-mail: hoffman@ac.dal.ca).

Figure 1. The Main Menu screen. (Each button leads to a new screen to perform the desired function.)

ucts to take advantage of these Apple events, a new version of HyperCard (Version 2.1) was released which did have the ability to send and receive Apple events.

Fortuitously, in the fall of 1991, the new Psychology Computer Laboratory at Dalhousie University opened with a network of 22 new Macintosh computers.¹ All the computers in the laboratory run System 7 and HyperCard 2.1, which is one of several standard packages of software included with each computer. Since we often use HyperCard-based experiments in our laboratory classes, we decided to construct a utility stack that would use Apple events to simplify some of the tasks involved in running experiments on a network. The immediate goal was to be able to retrieve data quickly from each of the computers on which we were running an experiment in a class. However, in the process of implementing that feature, we also found that HyperCard could be utilized for several other useful tasks in a network environment.

The intent in this paper is not to describe a finished product, since this stack continues to evolve, but rather to indicate the sorts of functions that have been added to HyperCard with the ability to send and receive Apple events. No doubt, others will find more uses for these new features, but the few that we have implemented so far show that Version 2.1 of HyperCard is a considerably more powerful tool than previous versions.

First, a brief summary of the functions so far implemented can be seen in Figure 1 which shows the Main Menu screen of the program. There are two columns of buttons which the user selects by using the mouse to move the cursor to a button, and then clicking the button on the mouse to take the program to the screen (*card*, if one uses the metaphor of a stack of cards) related to the particular function.

There is some restriction in the order in which the user must progress through the functions, since most functions require that the HyperCard application be opened on the other computers in the network to which the user wishes to communicate. Therefore, the first action must be to select which remote computers should be included. This will be discussed in more detail below. Once that selection is made, the next step is to open the HyperCard application on each of the selected remote computers. This is simply accomplished, by sending the appropriate Apple event from HyperCard on the user's computer (the *local* computer) to the Finder (a program on the Macintosh which provides the interface to the Operating System) on each of the selected remote computers. In response, the remote Finder opens the HyperCard application on the remote computer.

Once the HyperCard application is open on each remote Macintosh, the user can elect to leave HyperCard visible on the remote computer screen, or to hide the application and leave it running in the background. (Under System 7, several programs may be opened simultaneously, depending on the amount of memory available.) Most of the functions of this stack can be performed while HyperCard is running in the background, since HyperCard will still receive and respond to Apple events even though it is not actually visible on the computer screen.

The next six buttons refer to functions that send information from HyperCard on the local computer to HyperCard on the remote computers, and then cause the remote HyperCard to take some action and, in some cases, to send information back to the HyperCard program on the local computer. The final four buttons also refer to functions that send Apple events to the remote HyperCard, but the action that results on the remote computers will either

close programs running on the remote computers (Quit Remote Application, Quit Remote HyperCard) or cause the remote computers to either restart or to shut down (Restart Remote Macs, ShutDown Remote Macs). In the following sections, each of these functions is described in more detail, but first a discussion of the hardware and software requirements is appropriate.

HARDWARE AND SOFTWARE REQUIREMENTS

Hardware

One of the distinct advantages of the Macintosh computer line has been that networking hardware and software are included with each computer. Setting up a network of two computers requires only two LocalTalk-compatible connectors and a length of appropriate cable. The connector is plugged into the printer port on the rear of each computer, and the cable joins the two connectors. Connecting a printer to the network in the same way allows both computers to share one printer. Additional computers can be added in the same way (up to a practical maximum of about 20 units, including printers), and the overall cable length can allow relatively long runs with little noticeable degradation, so that the computers need not be in the same room. Additional small networks can be linked together into a multiple-zone larger network.

The Macintosh Operating System Version 7.0 and later all require that the computer be equipped with a hard disk drive, since the system and its associated files require substantially more disk space than previous systems do. As well, even a minimal version of System 7 uses about 1 MB of RAM, so Macintosh computers running System 7 need a minimum of 2.5 MB of RAM, and 4 MB would be desirable. In the Psychology Computer Laboratory at Dalhousie, all of the Macintosh computers have a minimum 4 MB of RAM and an internal 40-MB hard disk drive.

Software

One of the highly touted advantages of Apple's System 7 is the ability to share files between several computers without the necessity of a dedicated file server (on a separate computer). This is accomplished by turning on a feature called File Sharing. The HyperCard stack described in this paper utilizes a different function of System 7 called Program Linking. Both of these functions use software modules that are included within the System 7 package, but users need to be sure to follow the full installation procedure to ensure that all modules are installed on each computer. The installer program that comes with the System 7 disks offers a choice of installing a minimal system, but this will not install all the necessary components of the network software.

Each computer in the network should also have the HyperCard program Version 2.1 installed on its hard drive, although the new HyperCard 2.1 Player (Claris, 1992), which Apple began supplying with Macintosh computers sold in October 1992 will also perform most of the functions described here.

The functions described below require additional software that is available for noncommercial users either free or for minimal cost. The first is the AutoGuest INIT (Apple Computer, 1991a), a system extension that is available on the Developer CD distributed by the Apple Programmers and Developers Association (APDA). Although this system extension is not necessary (and can create security problems, which will be discussed later), it greatly simplifies the use of the stack on a network by eliminating the dialog boxes that otherwise appear on each computer's screen when linking over the network. (Installing AutoGuest is very simple: copy the file into the System folder on each Macintosh hard drive, and then restart the computer.)

The network control stack described here also utilizes several other pieces of copyrighted software that extend the capabilities of HyperCard by adding external commands (XCMDs) and external functions (XFCNs).² Collections of such externals are available for downloading at no extra cost from FTP sites such as Info-Mac or Apple Computer, or through commercial networks such as CompuServe or America OnLine.

Setting Up the Network

Once physically connected as described earlier (Ethernet may also be used, but is not described here), each computer must be prepared for network communication. First, each computer must be running the System 7.0 (or later) version of the operating system. Then, on each computer, the Sharing Setup control panel in the System folder must have Program Linking turned on. A dialog box asks for the user to enter Owner Name as well as a Macintosh Name, which must be unique for that AppleTalk Zone (a zone is a grouping of devices such as computers and printers, and many such zones may be interconnected in a network). If the user does not enter a Macintosh name after entering the Owner Name, the computer will generate a name automatically.

Next, the user should open the Users & Groups control panel, and select the icon for Guest user. Double clicking this icon will open a dialog box, and the user should turn on the Allow Linking option and then save the changes. If the AutoGuest INIT has not yet been installed, it should be copied into the system folder, and the computer restarted.

Finally, the user should select the icon for the HyperCard 2.1 application, choose Sharing... from the File menu, and in the subsequent dialog box, select Allow remote program linking.

As mentioned earlier, in order to use all the features implemented by the control stack, the HyperCard application on each remote computer must be able to access several external functions and commands. We accomplish this by having two specific stacks: first, the stack containing the FinderEvents XCMD (Pugh, 1992), and second, a stack containing other XCMDs and XFCNs (which for convenience we put all in one stack). In order for the remote HyperCard to access these stacks, the stacks must be located on a disk (or network volume) whose icon ap-

pears mounted on the desktop of each remote computer, and the user must be able to specify the pathnames to the stacks from the remote computer. (In other words, the external stacks can be shared from a server.) Since the control stack is used primarily in a lab situation, in which all computers have similar configurations, the stack assumes an identical setup on each machine, but this is not an essential requirement.

APPLE EVENTS

There are many different types of Apple events and more are being developed. Some of these have been grouped into collections, or *suites*, of related functions (such as the Required Suite, the Core Suite, the Text Suite, or the Table Suite); others are simply described as Miscellaneous Standards. Technical information about these Apple events is updated periodically and published in the *Apple Event Registry* (Apple Computer, 1992), available on the Developer CD. Although some developers may choose not to publish their custom Apple events, the fact that many are doing so promises to make these small computers much more powerful tools for teachers and researchers.

HyperCard and Apple Events

HyperCard 2.1 recognizes the four Apple events that form the Required Suite. These are the Open Application (oapp), Open Document (odoc), Print Document (pdoc), and Quit (quit) Apple events (the four-letter abbreviation in parentheses is the ID of the event). Three additional Apple events from the Miscellaneous Standards are also recognized by HyperCard: Close Document (clos), Do Script (dosc), and Evaluate (eval). The last two are particularly important in sending information (which can include data or commands) between HyperCard applications running on separate computers on a network (Apple Computer, 1991b).³

Sending to HyperCard

With the advent of Apple events, the normal HyperTalk send command has been extended so that when one is sending to a second HyperCard application (e.g., on a remote computer), the Do Script Apple event is used. The receiving HyperCard would respond by performing the action specified in the sent expression. For example, a command such as

send "Go card 5" to program "Zone:Macintosh:HyperCard"

(where Zone:Macintosh specifies the pathname of the remote computer) would result in the HyperCard stack on the remote computer immediately displaying the fifth card in the stack.

As long as what is sent to the remote HyperCard application is a valid expression, the remote HyperCard will immediately perform the action specified in the expression.⁴ As will be demonstrated below, this is a powerful feature.

THE HYPERNETMASTER STACK

The following sections describe how the control stack can be used to retrieve data from HyperCard-based experiments running simultaneously on several networked Macintosh computers. Other features in the current version of the stack will be discussed afterward. In order to follow the scripting examples, readers should be somewhat familiar with the HyperTalk programming language used in HyperCard; anyone programming HyperCard-based experiments will have sufficient understanding to follow the examples.

Select Remote Macs

Before sending any commands to remote computers, the program allows the user to select which remote computers to communicate with, from all those that are available on the network. From the Main Menu, the user clicks Select Remote Macs, and the Configuration card screen (Figure 2) appears. There is actually a series of cards that define various network configurations, but the one last used is shown by default in order to facilitate repeated use of the same configuration in a laboratory. If the user wishes a configuration different from the default, there is a popup button (see Figure 3) in the upper right-hand area of the screen, which will produce a scrolling menu list of all existing configurations when the mouse is placed over the arrow and the mouse button is held down. The current configuration is indicated with a checkmark. If the user selects a different configuration, a different list of Macintosh computers will appear in the two lower fields on the screen.

Selecting a new configuration is straightforward, despite the complex appearance of this screen. The bottom choice on the popup menu is New Card. This creates a new card in which the two bottom fields have been cleared. (The fields at the top of the card are not part of the Configuration, as explained below.)

After selecting New Card and giving the card a name (the name is automatically highlighted after selecting New Card so that the next letters entered on the keyboard replace a card number that the program automatically generates), the user selects which zone contains the desired remote computers from the scrolling list labeled Zones. If the computers are in the same zone as the local computer is in, no choice is necessary, but it is possible to have a configuration with remote computers in several zones. Next, a click on the Update Lists button will produce a list of all Macintosh computers in the selected zone that are turned on and have Program Linking enabled. The list will appear in the PPCToolbox field and the Macintosh field. If a computer appears in the Macintosh field but not in the PPCToolbox field, then Program Linking has been turned off on that computer, and it must be reset before it can receive Apple events.

The user can then click on the name of any computers that are not desired in the configuration. Next, clicking

Figure 2. The Configuration screen to select the remote computers.

Figure 3. A pop-up field at the upper right allows the user to select other configurations, or start a new Configuration by selecting New Card.

Figure 4. The screen for Open Remote HyperCard. (Clicking the Open button will send the field contents to each remote computer selected in the Configuration card, after the field contents have been edited appropriately. The script in the field is one line.)

on the Add to List will add the pathnames of all the remaining computers to the lower left field. If there were any names there before (e.g., if the user is editing an existing configuration), the new list will be added at the end, so it is advisable to click on the Delete Duplicates button to eliminate any redundancies. If the user decides not to use a computer on the list, clicking once on its name will dim its name and deselect it. Clicking on the Delete Unselected button will eliminate the name from the list altogether.

The final item needed in the configuration is the pathname on each remote computer to the HyperCard 2.1 application. If all the computers selected in the network use the same pathname to the HyperCard application (as is likely in a laboratory situation), it is only necessary to enter the pathname once, and then click on the Replicate Paths button to have the information replicated the needed number of times.

Open Remote HyperCard

After the configuration is finalized, the user returns to the Main Menu, clicks the Open Remote HyperCard button and is presented a screen (Figure 4) with a single Open button and a field. Clicking the button causes the contents of the field to be sent to each remote computer in turn, after automatically editing the content to the field as appropriate for each remote computer (using information from the configuration card previously specified). In this case, the local HyperCard is sending an Open Selection Apple event (Pugh, 1992) to the Finder on each remote Macintosh causing the Finder to open the HyperCard 2.1 application on the remote computer.

Open Remote Stack

Once HyperCard is running on each remote computer, commands can be sent directly to each remote HyperCard application, which can perform a variety of tasks. If in a HyperCard-based experiment a group of subjects are to be run simultaneously, with each subject on one of the selected remote computers, the Open Remote Stack card (Figure 5) permits the user to enter the pathname to the experiment stack (if the same pathname is used on all selected computers). Clicking the Open button will then cause the local HyperCard to send the message

go stack stackname

to each remote computer in turn; *stackname* is the path to the specified experiment stack. Each remote HyperCard receives the message and treats it as a script to perform, thus opening the named experiment stack.

If the experiment stack is designed so that once opened it is ready for the subject, the subjects are ready to run. But if the stack requires additional action on the part of the experimenter to prepare it for the subject (for example, clicking a Setup button may cause old data to be cleared or an Instruction screen to appear, etc.), that action can be initiated remotely by using the Send a Message card (discussed later).

Request Data

Assuming that no further preliminary action was necessary, however, and that the subjects have performed the required experimental task, the user can proceed to the Request Data card (Figure 6). As shown, this card has three fields in which the user can specify (1) the ex-

Figure 5. The Open Remote Stack screen. (After entering the path to a target HyperCard stack on the remote computer, the user clicks the Open button to send the appropriate Apple event from HyperCard on the user's computer to HyperCard on the remote computer. The remote HyperCard will then open the stack named in the field.)

Figure 6. The Request Data screen. (Clicking the Request buttons will result in the user's HyperCard's sending a Request Apple event to HyperCard on the remote computer, which will respond by sending back the contents of field Data on card DataCard of the HyperCard stack identified in the top field.)

periment stack on the remote computer (this information is automatically entered from the Open Remote Stack card), (2) which card in the stack contains the data, and (3) which field on that card contains the data. When the Request button is clicked, the local Hypercard sends the message of the form

request card field *dataField* of card *dataCard*

to each remote HyperCard, where *dataField* is the specified field containing the experiment data, and *dataCard* is the specified card in the experiment stack. Note that it is not actually necessary to tell the remote HyperCard to go to the specified card; thus, it is possible to retrieve data from an experiment stack while the subjects are still using the stack (e.g., they may be reading briefing information included at the end of the experiment).

The requested information is returned to the local HyperCard (in the variable or container named *it*), and, along with information identifying the remote computer from which the data were requested, it is automatically written into a scrolling field on the next card in the control stack. After the data have been retrieved from all the selected remote computers, the card containing the retrieved data appears (Figure 7) along with two buttons allowing the user to export the data to a text file or print the data. In the example shown, three lines of text have been retrieved from a remote computer named Psychology 170 in zone PSYCH-LT1. If the data were numerical, simple scripting changes could permit whatever sort of desired statistical manipulation the user would like to have done in HyperCard, or the text file may be exported to some other application.

While the Request Data card is set for only one data field, it is a simple scripting task to retrieve data from multiple fields if the experiment stack is constructed to record data in more than one field. The data retrieval process is reasonably fast. In a recent class experiment in the laboratory at Dalhousie, data from two fields were collected from 21 remote computers (in the same AppleTalk zone), and means calculated, in about 40 sec.

To summarize the discussion thus far, after selecting a group of remote Macintosh computers on a network, one can send from the HyperCard 2.1 program on the user's computer three types of Apple events, which (1) cause the Finder on each remote computer to open the HyperCard application on that computer, (2) cause HyperCard on the remote computer to perform some action, and (3) retrieve information (such as the contents of a field) from each remote HyperCard. Figure 8 illustrates the three types of Apple events and gives a sample of the form the script would take to execute the appropriate action.

OTHER FUNCTIONS

Send a Message

Earlier, we indicated that it is possible to perform additional actions on stacks other than simply open them on remote computers and request data from them. With the Send a Message card, the user can send to each selected remote computer a short message typed into the Message Box window in HyperCard on the local computer (using the top button in Figure 9). The receiving computer will then display its own Message Box window containing

Figure 7. The Retrieved Data screen. (Requested data from remote computers is written to the field on this card.)

Figure 8. Three important types of Apple events that can be sent from HyperCard. (For each type, the box at the right gives an example of the syntax of script to send the Apple event.)

Figure 9. The Send a Message screen. (Clicking the top button will send the contents of the HyperCard Message Box (not shown) on the user's computer to the Message Box on the remote computer. The lower field allows users to type in longer scripts, which are then sent to the remote computer by using the bottom button at the left.)

whatever was in the sending Message Box. The way this happens is more complex than might seem necessary, but there is a reason. For example, it would be possible to type a command into the local HyperCard Message Box (such as go next card) and send the contents directly to the remote HyperCard, which would respond by displaying the next card in whatever stack is running on the remote machine. (In HyperCard 2.0 and later, it is possible to have multiple stacks open simultaneously, but the action is performed on the stack in the frontmost window.)

However, display of the contents of the local computer's Message Box on the remote Message Box involves a two-step process (conceptualized by the commands *send message Box to remote HyperCard*, and then *display what was sent in the Message Box on the local HyperCard*), which would take more than one line, and the Message Box is limited to one line. The solution, as shown in the upper field in this card, is to enter the appropriate message into the local computer's Message Box, and then send a command to the remote computer to Request Message Box from the local computer. Since the reply to a Request command is returned into a local variable *it* on the computer which initially sent the request, the next line of the script in the field tells the remote HyperCard to *put it into the Message Box*, which results in the message from the local computer appearing in the Message Box on the remote computer. In fact, the entire process is accomplished by writing the sequence of steps into a script in the top field of this card, and then sending the contents of the field to the remote computer. (The first and last lines of the script shown are not necessary, but serve to alert a user on a remote computer that a message is coming.)

Following this reasoning, then, it is possible to build elaborate scripts and send them to the remote HyperCard, which will perform the required action. The lower field on this card is provided so that the user may construct a script to be performed by the remote computer. Clicking on the Send Script button causes the contents of the lower field to be placed into a variable RemoteScript on the local computer, and a message is sent to the remote computer to *request* the variable RemoteScript from the first computer. The contents of the variable RemoteScript are received and placed into the variable *it* on the remote computer, and the next command asks the remote computer to put the first line of the variable *it* into the Message Box (so you can check that it arrived) and then send the contents of *it* to the remote stack itself. As a result, any executable statements in *it* will be performed by the HyperCard on the remote computer. This technique can be used to remotely control elaborate functions on the remote computers.

Copy a File

The control stack can perform several other useful tasks in the maintenance of a networked laboratory. In the Dalhousie Psychology Laboratory, all of the computers access a central server, which simplifies the distribution of updated software to each computer. After placing one copy of an experimental program on the server, the user can use the Copy a File function of the stack (Figure 10) to quickly copy the experimental program onto each of the computers that are to be used in an experiment. After entering the pathnames of the file to be copied into the top field on the card, and the destination location on the

HyperNetMaster Stack

Copy a file

Copy File

Copy Folder

Source path name: [Vol:folder:file]
Psychology 1:P2000 staff:Data Stack

Destination path name:[Vol:Disk:Folder:]
Macintosh HD:

Resource stack path:
Macintosh HD:Ron's:HyperCard Externals:Rinaldi Resources

```

Start using stack "Macintosh HD:Ron's:HyperCard Externals:Rinaldi Resources"
put "Macintosh HD:Ron's:Data Stack" into sourcePath
put "Macintosh HD:" into destPath
FileCopy sourcePath,destPath
Beep
Stop using stack "Macintosh HD:Ron's:HyperCard Externals:Rinaldi Resources"

```

← Main Menu →

Figure 10. The Copy a File screen. (Either single files or entire folders can be copied remotely by sending the script in the bottom field to each remote HyperCard.)

Figure 11. The Delete a File screen. (Single files or entire folders can be deleted from remote Macintoshes by sending the script in the bottom field to each remote HyperCard. The Get Directory button will produce a list of all files in the designated volume or folder on each remote Macintosh.)

remote computer in the second field, the user clicks on one of the two buttons on this card (Copy File or Copy Folder), depending on whether a single file or a folder containing one or more files is to be copied. A *do script* Apple event is then sent to the remote HyperCard, accompanying the script shown at the bottom of the screen. The remote HyperCard then executes the script, in which the copying action is carried out by external functions located in the stack whose pathname is listed in the third field.

In the example shown, a file named Data Stack is copied from a folder named P2000 staff on the server Psychology 1 to the disk Macintosh HD on the remote computer.

Delete a File

In a similar manner, the Delete a File card (Figure 11) permits the user to erase from the remote computer files or folders that are no longer needed. By entering the appropriate information identifying the file or folder to be erased from the remote computers and then clicking on the button, the user can tell HyperCard on the remote computer to perform the desired action.

This card also provides another useful function via the Get Directory button. Once the user specifies the volume name in the top right field, clicking this button will produce a list of all files and folders within that volume on each of the remote computers. The collated list then appears in a pop-up field, along with two buttons that give the user the choice of saving the list to a text file and printing the list on the selected printer. This allows the user to determine which files need to be deleted on each re-

mote computer without having to physically visit each machine.

Quit Remote Application

Under System 7 of the Macintosh operating system, it is possible to have more than one application program running at a time. Many users utilize screen-saver programs that automatically come on after a period of no input from the keyboard or mouse. If the screen saver is Apple event aware, it is possible to send a command to the remote HyperCard to either quit or close the application. On the Quit Remote Application card of the control stack, the user enters the name of the application on the remote computer, and clicking the Quit button then sends the message to the remote HyperCard. (Of course, it is possible to reopen the application later.) This will work with any Apple event-aware application running on the remote computers, not just screen savers, but it is clearly useful in a laboratory situation where you wish a specific screen to be displayed without interference from a screen saver.

Restart and Shutdown

These two functions are useful for either resetting all the remote computers or turning them off. In both cases, a message is sent from the local HyperCard to the remote HyperCard, which then sends a message to the remote Finder to either restart or shut down the remote computer. These functions utilize the FinderEvents XCMD (Pugh, 1992), so the remote HyperCard must have access to the stack containing the XCMD. The path to the stack must be entered in a field on the card.

SECURITY ISSUES

Once several computers are set up to communicate over a network using the procedures described in this paper, security becomes an issue. Allowing Guest access in the Users & Groups control panel means that any remote users can link to programs on the computer. To control access and increase security requires assigning user names and passwords, and then every time a link is established the password must be entered. If there is no danger of unwanted access to restricted or confidential information, Guest access may pose no particular risk, and users may wish to proceed as described earlier; but when there is such a risk, the user must disable program linking whenever it is not needed. The easiest place to do this is in the Users & Groups control panel, by selecting the icon for Guest, opening it, and then turning off the Program Linking option.

Obviously, the stack described in this paper is only used by those who need to perform the tasks that are simplified by the stack. In most cases, the users are instructors and teaching assistants in laboratory classes. Setting up the stack requires some knowledge of programming in the HyperTalk language of HyperCard, but in practice, once the appropriate information has been entered in the required fields (e.g., the name of the HyperCard stack which runs an experiment), it is relatively simple for users with little or no knowledge of HyperCard programming to use. To reduce the danger of inadvertently performing an undesirable action, the Userlevel property of HyperCard can be set to the minimum necessary after the information has been entered in the fields.

Obviously, the ability to remotely erase files and folders from a computer can create difficulties if the wrong file information is entered. As every computer user knows (and forgets at least once), it is essential to keep current backup copies of any important computer files in a safe location. The same lesson clearly applies here.

CONCLUSIONS AND IMPLICATIONS

The HyperCard stack described in this paper utilizes the ability of HyperCard 2.1 to send and receive Apple events to other computers in a network. One very useful capability makes gathering experimental data from multiple simultaneously run sessions very simple in HyperCard-based experiments. Other administrative functions are simplified, as well, with the result that the teacher and/or researcher can have improved control over the environment. The user can literally sit at one computer and control actions on all the other computers in the network configuration that normally would require a visit to each computer.

In the Psychology Computer Laboratory at Dalhousie, 20 Macintosh Classic and two Macintosh LC computers are networked with a separate server. To illustrate the savings in time and labor, once all the computers in the laboratory are turned on, and the control stack is running on

one of the computers, a user can launch HyperCard remotely on all of the other computers in approximately 1 min. Copying a file (40K) from the server onto each remote computer takes just under 4 min, but only requires minimal effort (one click of the mouse button). Opening that same file (a HyperCard-based experiment) on all the remote computers takes about 10 sec. As mentioned earlier, retrieving experimental data from that stack only took 40 sec, and that included a few seconds for the control stack to compute some descriptive statistics. Afterwards, the experiment stack was deleted from all the remote computers in about 50 sec, and in another 50 sec all the remote computers had been shut down. Only at this point did it become necessary to actually visit each computer on the network, to turn off the power switch.

The functions performed by the control stack to date are fairly obvious ones, but they are clearly useful nonetheless. In the future, more functions will probably be added to this stack, and the interface will be simplified. It is clear, however, that the arrival of Apple events, with the multiple-computer interaction that Apple events allow, offers new opportunities for teaching and research. HyperCard continues to permit the relatively rapid and simple development of applications, and this added functionality will no doubt spur even more imaginative uses.

REFERENCES

- APPLE COMPUTER (1991a). *AutoGuest INIT* [Computer program]. Cupertino, CA: Author.
- APPLE COMPUTER (1991b). *HyperCard new features guide*. Reading, MA: Addison-Wesley.
- APPLE COMPUTER (1991c). *Inside Macintosh: Vol. 6*. Reading, MA: Addison-Wesley.
- APPLE COMPUTER (1992). *Apple event registry: Standard suites* (Developer Technical Publications). Cupertino, CA: Author.
- BLAYE, A., LIGHT, P., JOINER, R., & SHELDON, S. (1991). Collaboration as a facilitator of planning and problem solving on a computer-based task. *British Journal of Developmental Psychology*, 9, 471-483.
- CARTER, S. R., & WALSH, D. A. (1992). A HyperCard-based tool for studying cognitive processes in complex problem solving. *Behavior Research Methods, Instruments, & Computers*, 24, 286-297.
- CLARIS CORPORATION (1991). *What's new in HyperCard Version 2.1* [Computer program]. Santa Clara, CA: Author.
- CLARIS CORPORATION (1992). *HyperCard player* [Computer program]. Santa Clara, CA: Author.
- DOUGHERTY, T. J. (1990). Contour: A hypermedia environment for teaching about subjective contours and other visual illusions. *Behavior Research Methods, Instruments, & Computers*, 22, 223-227.
- GEISLER-BRENSTEIN, E., & BRENSTEIN, R. J. (1989). The potential of HyperCard for psychological research and instruction: A general discussion and description of two research applications. *Behavior Research Methods, Instruments, & Computers*, 21, 307-311.
- LEE, A. Y. (1992). Using tutoring systems to study learning: An application of HyperCard. *Behavior Research Methods, Instruments, & Computers*, 24, 205-212.
- LEVIDOW, B. B., HUNT, E., & MCKEE, C. (1991). The DIAGNOSER: A HyperCard tool for building theoretically based tutorials. *Behavior Research Methods, Instruments, & Computers*, 23, 249-252.
- MARTIN, T. A., & ALLAN, W. E. (1991). An evaluation of touch-screen input for a HyperCard-based digit-span task. *Behavior Research Methods, Instruments, & Computers*, 23, 253-255.
- MARTIN, T. A., & GROWNEY, W. J. (1992). A HyperCard stack for

- the creation of graphically organized glossaries. *Behavior Research Methods, Instruments, & Computers*, **24**, 317-320.
- MARTIN, T. A., & WILCOX, K. L. (1989). HyperCard administration of a block-design task. *Behavior Research Methods, Instruments, & Computers*, **21**, 312-315.
- NASON, S., & ZABRUCKY, K. (1988). A program for comprehension monitoring of text using HyperCard for the Macintosh. *Behavior Research Methods, Instruments, & Computers*, **20**, 499-502.
- PETTY, L. C., & ROSEN, E. F. (1991). Using HyperCard to teach person perception and attribution concepts. *Behavior Research Methods, Instruments, & Computers*, **23**, 247-248.
- PUGH, J. (1992). *Bride of FinderEvents, Version 1.2.1* [Computer program]. Cupertino, CA: Apple Computer.
- QUINN, C. N. (1991). Computers for cognitive research: A HyperCard adventure game. *Behavior Research Methods, Instruments, & Computers*, **23**, 237-246.
- RAY, R. D., & MITCHELL, N. (1992). BehSys Anal: An artificially intelligent hypermedia resource library for descriptive behavioral research. *Behavior Research Methods, Instruments, & Computers*, **24**, 180-189.
- SCHULDBERG, D., & NICHOLS, W. G. (1990). Using HyperCard to administer a figural test on the Apple Macintosh. *Behavior Research Methods, Instruments, & Computers*, **22**, 417-420.
- TVERSKY, B. (1990). *Experiments in cognitive psychology series* [Computer program]. Stanford, CA: Stanford University, Office of Technology-Licensing.

NOTES

1. The Psychology Computer Laboratory was established in the Life Sciences Centre by the Academic Computing Services unit of Dalhousie University and is operated jointly by Academic Computing and the Department of Psychology.

2. Several copyrighted external commands and functions (XCMDs and XFCNs) are used in the control stack or by HyperCard on the remote computers, including the following ones available royalty-free for noncommercial use: ATalkZones 1.1, © Frederic Rinaldi 1991; DeviceList 1.3, © Frederic Rinaldi 1991; FullHPop 1.3, © Frederic Rinaldi 1991; PPCList 1.0, © Frederic Rinaldi 1991; GetDir 1.8, © Frederic Rinaldi 1990; FileCopy 1.5, © Frederic Rinaldi 1991; CopyFolder 1.5, © Frederic Rinaldi 1991; EraseFile 1.1, © Frederic Rinaldi 1991; RemoveFolder 1.1, © Frederic Rinaldi 1991.

3. A very concise and clear introduction to the use of Apple events in HyperCard is given in the *HyperCard New Features Guide* booklet, which comes in the HyperCard Developers Kit (Apple Computer, 1991b). The same kit also includes a HyperCard stack that provides much the same information (Claris, 1991) in a useful on-line source of information to developers of HyperCard stacks.

4. Errors that occur when a remote computer fails to perform the action required by an Apple event sent from HyperCard on the local computer are reported back to the local computer in a variable Result. Error codes are not discussed in this paper.