CONSTRUCTION AND MAINTENANCE OF WORLD-WIDE WEB SITES

Moderated by John H. Krantz, *Hanover College*

Use of HyperCard to create pages on the World-Wide Web

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

A variety of commercial and shareware programs are now available to assist in creating HTML pages for a World-Wide Web (WWW) site. However, users with some knowledge of HTML coding and a simple programming language, such as the HyperTalk language in HyperCard, can create custom tools that will simplify the task of creating and maintaining multiple-page WWW sites. This paper presents examples of two such tools and describes the basic steps they use to create HTML documents.

Psychologists are being urged to create sites of home pages on the World-Wide Web (WWW) for a variety of purposes. These include departmental information pages (Plaud, 1996), tutorials or teaching resources (Krantz & Eagley, 1996; Trapp, Hammond, & Bray, 1996), conference announcements (Mainwaring, 1996), or research data gathering (Schmidt, 1997). In all cases, material to go on such Web pages must be appropriately designed in order to be effective, and then must have the HTML codes inserted so that the material is properly displayed on Web browser programs. Instructions are readily available in print (Lemay, 1995) or on the Web itself (<http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>) on the mechanics of HTML coding. When the number of pages to be coded is small, it is possible to insert the codes by hand or to use one of the shareware (e.g., Best, 1996a) or commercial programs (Best, 1996b; PageMill, 1996) available for this purpose.

However, if your site will contain a large number of pages, hand coding becomes onerous, and commercial products to manage larger sites are expensive. Too often, Web-page designers resort to use of long pages, when it would be more appropriate to break the information into a number of separate pages that allow users more flexibility in choosing how much material to download to their browser.¹ It is possible to create custom tools to simplify this task.

Users of Macintosh computers often have the HyperCard program available. HyperCard has a variety of uses in psychology labs, including hypermedia teaching programs (Carreras, 1993; Dougherty, 1990) to demonstrate psychological phenomena; research tools (Gonzalez, 1993; Hoffman & MacDonald, 1993) for gathering data; and tools for statistical analysis (Meyer, 1995). Recent versions of HyperCard can also be used to simplify repetitive tasks involved in maintaining multiple-workstation laboratories (Hoffman & MacDonald, 1993).

HyperCard comes with its own programming language, HyperTalk. HyperTalk is a powerful yet surprisingly simple language, and most HyperCard users learn at least some of the language. One useful feature of HyperTalk, like other programming languages, is its ability to create text files. When combined with other characteristic features of HyperCard, namely the card metaphor with background fields shared among cards, it becomes possible to write simple program scripts to create a set of Web pages, identical in layout, but with differing text content on each page. With each page linked to appropriate index pages, this makes for very user friendly hypertext, where a click on a link takes a reader to the exact information desired without waiting to load and display extra information, such as occurs with long pages where all the information is on a single page. Once all pages are created on the local computer, the user only has to upload them to the appropriate Web server and ensure that proper privileges are set to allow the desired access via the Web.

This paper presents two examples using HyperCard stacks to create sets of HTML pages. In the first, an existing HyperCard stack containing photographs and textual information about students in a class was turned into

The authors thank Kim Passi for permission to reproduce her home page. Correspondence should be addressed to R. Hoffman, Department of Psychology, Dalhousie University, Halifax, NS, Canada B3H 4J1 (e-mail: ron.hoffman@dal.ca).

Figure 1. The first example HyperCard stack, with four text background fields—Name, E-mail, Supervisor, and Description.

a suite of WWW pages within the class Web site (not publicly accessible).² This was accomplished basically by the addition to the stack of a single button containing a HyperTalk script that took the information on each card of the HyperCard stack, formatted that information to appear on a Web page, and then created the appropriately labeled text file for uploading to a Web server. The key elements of the script are presented below.

The second example illustrates a more complex manipulation. In this case, a new HyperCard stack was created for the specific purpose of creating Web pages. An early version of this stack created pages to display abstracts for a conference of the Canadian Society for Brain, Behaviour and Cognitive Science in 1995 (<http://acs.dal.ca/csbbcs/Details95.html>), and other versions of the stack have been used to format research reports for the Dalhousie Department of Psychology (<http://acsweb.ucis.dal.ca/psych/ResRep.html>) and to produce HTML pages for the 1996 Conference of the Society for Computers in Psychology (<http://is.dal.ca/~hoffman/SCIP96/default.html>).

The scripting was basically similar in both example HyperCard stacks, but was less complex in the first. The first stack is described in some detail below, and a general description of the second stack is also given.

Creating Student Pages

In a 2nd-year honors independent research class at Dalhousie, we have for several years created a HyperCard stack for the students containing pictures of each student and brief introductions the students write about themselves. In a first class, they are taught to use e-mail, and their assignment is to e-mail this description to the course

instructor. The instructor then extracts the e-mail addresses for the students and their introductory statements, putting these in a HyperCard stack on a separate card for each student. Each card also contains a picture of the student. (The images were collected on videotape during class, and single frames were extracted using a frame grabber.)

In the past, the HyperCard stack was distributed to the students, who could run the stack on any of the computers in the laboratory or on their home computer (if they had some model of Macintosh). In 1996, the student information was incorporated into a class World-Wide Web site, along with other course-related material. Each student had a separate page containing all the information that had appeared previously on the HyperCard screen. Since the HyperCard stack already contained all the information elements, all that was necessary was to create a script in the language (HyperTalk) used by HyperCard to translate the information from the various fields on each card into the appropriately coded HTML document.

Figure 1 illustrates a screen (card) from the class stack. The card shows four text fields, containing the student's name, e-mail address, supervisor's name, and a scrolling field containing the introductory biographical information prepared by the student. In addition, a picture (a PICT image) of the student is displayed on the card. Each student in the class had a separate card.

Translating the HyperCard stack into a series of HTML pages was accomplished with a script attached to a single button added to the last card of the stack, which produced pages such as that shown in Figure 2. (First, in order to display the images through a Web browser, the PICT images were converted to GIF images, and the path to the

folder containing the GIFs was specified in the script for the button.) A single click on the button would result in creation of a separate HTML text file containing the information taken from each card of the stack, with appropriate HTML links to the GIF image, the class home page, and an index file (Figure 3), which was also created by the script. The e-mail address shown on each HTML page was given a MAILTO link to permit sending e-mail to the student from within the Web browser.

The script for the HTML button is listed in Figure 4. Readers familiar with the HyperTalk scripting language will be able to follow the script easily. The script extracted information from each card in turn, creating an HTML document using the name from the card and then writing in succession the appropriate HTML formatting codes, inserting text information from the card as appropriate. Thus, all HTML pages ended up with a similar format.

The script contained a loop that extracted the desired information from each card in turn, starting by putting

the student's last name into a variable `Who` and then creating a new text file with this name and the suffix `.html`, into which the HTML code was then written. The content of the Name field on the card was used for the title of the document, and a common header (the class title) was written to each HTML file. The name of the student was then written to the HTML file using the second largest heading style (H2). The student's e-mail address was extracted from the e-mail field on the card, placed in a variable `theMail`, which was then written below the student's name and followed by a `MAILTO:` tag, creating a link so that a user could click on the student's e-mail address and open an e-mail document addressed to that student (in the Netscape browser). The balance of the HTML document was created in a similar fashion.

The only unusual part of the script was a short section of six lines (lines 36–41) that determined how long the biographical text was (since students wrote differing amounts), and then added an appropriate number of line

Figure 2. The first HyperCard stack converted the information shown in Figure 1 to this home page.

Figure 3. In addition to individual home pages for each student, the script in the first example HyperCard stack produced an index page with links to each student's home page.

break commands to the HTML page. This prevented the text from wrapping around the displayed image.

Creating a Suite of Abstract Pages

The second example is more complex, primarily because it included several indexing methods. However, other than the indexes, the general procedure was similar to that in the first example: The information to go on each HTML page was present on a card of a HyperCard stack. Scripts within the stack then formatted the HTML text files, incorporated the information from the cards, added common elements such as headings, and produced indexes with all the appropriate links.

This stack was first used in creating the Web site for the 1995 meeting of the Canadian Society for Brain, Behaviour and Cognitive Science. A single text document containing the abstracts of all presentations was initially produced on a word processor, and that document was then imported into the HyperCard stack (by a script) so that each card held the abstract for a single paper (in one field), along with the title of the presentation (a second field), names of authors (a third field), and their institutional affiliations (a fourth field).

The key elements of the script that produced the final HTML documents of the abstracts were similar to those described already: Information was extracted from each card in the stack, a new HTML text file was created, and

the HTML formatting code was then written serially to the file, with the appropriate text items from the card placed in the desired locations. Thus, a separate HTML document was created for each abstract and displayed the title of the talk, names and affiliations of all authors, and the abstract of the talk. Each HTML document also contained links back to the home page for the site, as well as to two indexes.

This project differed from the one described previously in several ways, but primarily in the sophistication of the indexing. Good hypertext should provide the reader with easily understood navigation tools, and this principle was implemented in several ways in this project. First, at the top of each page displaying an abstract were two rows of links, one row linking to a list of article authors and the second row linking to a list of article titles. However, there were not just two index files, but a total of 54 such files. That is, for the authors there was an overall index (All Authors) that was a single document listing all authors (not just first authors) alphabetically by last name, showing next to the author's name the title of the paper, linked to the page for that abstract. Similarly, there was a single overall index (All Titles) listing the titles of the papers alphabetically, each linked to the abstract page for the article. In addition, for both authors and titles there were 26 separate index files, one for each letter of the alphabet. Thus, if a reader wanted to find

```

on mouseUp
--each card has 4 bg flds: Name, Email, Description, Supervisor
repeat with i = 1 to the number of cds in bg "Info"
    set cursor to busy
    put last word of bg fld "Name" of cd i of bg "Info" into Who
5    put bg fld "Name" of cd i of bg "Info" into Name

    -----
    --The next section creates the individual home pages for each student.
    --The complete path to the files must already exist; that is, each folder listed
10    --in the path must already be created, otherwise you get a "File Not Open" error.
    put "Macintosh HD:Home Pages:Class:" & Who & ".html" into theFile
    open file theFile

    --The next section writes the various bits of HTML code to each page
15    write "<html>" & Return to file theFile
    write "<title>" & Name & "</title>" & Return to file theFile
    write Return & Return to file theFile -- makes the file easier to read
    write "<CENTER><HEAD><B>Psychology 2500 Class 1996</B></HEAD></CENTER>" to file theFile
    write Return & Return to file theFile -- makes the file easier to read
20    write "<body>" & Return & Return to file theFile
    write "<H2>" & Name & "</H2>" & Return & Return to file theFile
    put bg fld "Email" of cd i of bg "Info" into theMail
    write "<a href=mailto:" & theMail & ">" & theMail & "</a>" & Return & Return to file theFile

25    --the gif files must be in a folder Images at the same level as the index file
    --each gif file must have the student's last name plus extension ".gif"
    put "../Images/" & Who & ".gif" into thePhoto
    write "<P><IMG Align=Left SRC=" & quote & thePhoto & quote & ">"
    & Return & Return to file theFile
    put bg fld "Description" of cd i of bg "Info" into theBio
30    put bg fld "Supervisor" of cd i of bg "Info" into theSupe
    write "<BLOCKQUOTE>" & theBio & "</BLOCKQUOTE>" to file theFile

    --add vertical space as appropriate to prevent wrap around image
35    put length of theBio into BioSize
    if BioSize <200 then write "<BR><BR><BR>" to file theFile
    if BioSize <300 then write "<BR><BR><BR>" to file theFile
    if BioSize <400 then write "<BR><BR><BR>" to file theFile
    if BioSize <500 then write "<BR>" to file theFile
40    if BioSize <600 then write "<BR><BR>" to file theFile
    if BioSize <699 then write "<BR>" & Return to file theFile
    write Return & Return to file theFile -- makes the file easier to read
    write "<P><P><P><HR>" & Return & Return to file theFile
    write "<P>Supervisor: " & theSupe & Return & Return to file theFile
45    write "<P>Return to <A HREF=" & quote & "../Students.html" & quote & ">Class List</A>" &
    Return & Return to file theFile
    --add the link to the Class Home Page
    write "<P>Return to <A HREF=" & quote & "../Default.html" & quote & ">P2500 Home Page</A>"
    & Return & Return to file theFile
50    close file theFile
end repeat

    -----
    --This next section creates an index file "Students.html" for all the files created above
55    --This assumes the cards are already sorted in alphabetical order
    put "Macintosh HD:Home Pages:Students.html" into theFile -- pathname to the local file
    open file theFile -- create the index file
    write "<html>" & Return to file theFile
    write "<title>" & "Students" & "</title>" & Return to file theFile
60    write Return & Return to file theFile -- makes the file easier to read
    write Return & Return to file theFile -- makes the file easier to read
    write "<body>" & Return & Return to file theFile
    write "<H2>" & "Psychology 2500 1996 Students" & "</H2>" & Return & Return to file theFile
    write "<UL>" & Return to file theFile
65    write "</body>" & Return to file theFile

    --Add the list of names and html reference
    repeat with i = 1 to the number of cds in bg "Info"
        set cursor to busy
70        put last word of bg fld "Name" of cd i of bg "Info" into Who
        put bg fld "Name" of cd i of bg "Info" into Name
        write "<LI> <A HREF=" & quote & "Class/" & Who & ".html" & quote & ">" & Name &
        "</A>" to file theFile
    end repeat
75    write "</UL>" & Return to file theFile
    write "<HR>" & Return to file theFile
    write "Return to <A HREF=" & quote & "default.html" & quote & ">P2500 Home Page</A>" to file theFile
    close file theFile
    set cursor to browse
80    end mouseUp

```

Figure 4. The listing of the script for the Make HTML button in the first stack.

an article by an author whose name begins with H, a click on the H link would jump to a page showing all such authors.

Access to these indexes was provided in the header at the top of each page displaying an abstract. These index links were also repeated at the top and bottom of each of the index pages themselves, so that the reader never had to look far for access to other information (since the All Authors and All Titles files were long, the reader had to scroll through the page). A link back to the home page for the site was placed at the bottom of each page.

It should be noted that these indexes do not replace a search engine, although they may obviate the need for use of an engine, depending on the nature of the information presented. It would be possible to replace the indexes with appropriate common gateway interfaces (CGIs) or server side includes (SSIs), but how to do so depends on what features are available on the Web server employed and requires coordination with the server administrator (Schmidt, Hoffman, & MacDonald, 1997). Search engines are typically under the control of the server administrator, as well. Indexes such as those employed in this example are independent of the type of server used and will retrieve pages faster than if CGIs or SSIs are used.

Due to space restrictions, we cannot fully describe the operation of this stack in this article. Although there is necessarily more scripting in this stack than in the first example, the scripts themselves are not much more complex than that shown in the listing in Figure 4.³

The entire process of creating all these linked files and indexes took very little time, even on a relatively slow computer. In the authors' experience, more time was needed to prepare the original text file (correcting spelling and typographical errors) than to create and upload the resulting HTML files.

Conclusions

The World-Wide Web is rapidly becoming one of the most important means of communication today, and psychologists are among many being urged to make information available via the Web. Although some commercial and shareware programs are available that can be used to create appropriately coded HTML files, so far little is available to help manage large numbers of linked files, and these products (such as SiteMill, 1996) are relatively expensive.

One result is that site creators are forced to use very long HTML documents, which must be loaded by the reader's browser before the information can be read.⁴ Coding HTML links to jump around such lengthy documents can be tedious, and the reader is often left with irrelevant information on the screen, which can be distracting. The two examples presented here illustrate a different approach, in which chunks of information can be presented on separate pages, with clear links back to the indexes.

These examples were created using fairly simple programming in HyperCard, a versatile program that has

been put to a variety of uses by psychologists in the past. The metaphor of a stack of cards, when paired with the HyperTalk programming language, makes it relatively simple to create customized applications, and psychologists who already use this program may wish to employ it further to construct their own tools for managing sites on the World-Wide Web.

Obviously, once tools have been created in HyperCard to assist in creating and managing a Web site, these tools can be used over and over to either update a site or create similar sites. Changes in the look of the Web pages are easy to implement by changing the appropriate scripting in the HyperCard stack. And, once the desired layout is obtained, the layout is automatically implemented on all the pages created with the stack. For sites with a large number of pages, this automation is clearly advantageous. We would like readers to conclude that good, effective Web-page design does not always require new, often expensive, specialized programs. Sometimes, the tools at hand can be conveniently adapted to the task.

REFERENCES

- BEST, R. C., III (1996a). *HTML Web Weaver Lite 3.0* [Computer software]. Potsdam, NY: Miracle Software.
- BEST, R. C., III (1996b). *World Wide Web Weaver* (Version 1.1.1) [Computer software]. Potsdam, NY: Miracle Software.
- CARRERAS, I. E. (1993). Perception: A HyperCard stack for demonstrating visual perceptual phenomena. *Behavior Research Methods, Instruments, & Computers*, **25**, 173-179.
- 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.
- GONZALEZ, G. M. (1993). Computerized speech recognition in psychological assessment: A Macintosh prototype for screening depressive symptoms. *Behavior Research Methods, Instruments, & Computers*, **25**, 301-303.
- HOFFMAN, R., & MACDONALD, J. (1993). Using HyperCard and Apple events in a network environment: Collecting data from simultaneous experimental sessions. *Behavior Research Methods, Instruments, & Computers*, **25**, 114-126.
- KRANTZ, J. H., & EAGLEY, B. M. (1996). Creating psychological tutorials on the World-Wide Web. *Behavior Research Methods, Instruments, & Computers*, **28**, 156-160.
- LEMAY, L. (1995). *Teach yourself Web publishing with HTML in a week*. Indianapolis, IN: SAMS.
- LIVECARD [Computer software]. (1996). Largo, FL: Royal Software, Inc.
- MAINWARING, S. D. (1996). Compilation and use of a World-Wide Web index of cognitive and psychological science resources. *Behavior Research Methods, Instruments, & Computers*, **28**, 152-155.
- MEYER, G. E. (1995). Power and effect: A statistical utility for Macintosh and Windows systems. *Behavior Research Methods, Instruments, & Computers*, **27**, 134-138.
- PAGEMILL 2.0 [Computer software]. (1996). Mountain View, CA: Adobe Systems, Inc.
- PLAUD, J. J. (1996). Resources relevant to the creation of a psychology department home page. *Behavior Research Methods, Instruments, & Computers*, **28**, 183-185.
- SCHMIDT, W. C. (1997). World-Wide Web survey research: Benefits, potential problems, and solutions. *Behavior Research Methods, Instruments, & Computers*, **29**, 274-279.
- SCHMIDT, W. C., HOFFMAN, R., & MACDONALD, J. (1997). Operate your own World-Wide Web server. *Behavior Research Methods, Instruments, & Computers*, **29**, 189-193.

SITEMILL 1.0.2 [Computer software]. (1996). Mountain View, CA: Adobe Systems, Inc.

TRAPP, A., HAMMOND, N., & BRAY, D. (1996). Internet and the support of psychology education. *Behavior Research Methods, Instruments, & Computers*, **28**, 174-176.

NOTES

1. This issue of whether to place information on WWW sites in long documents versus linked smaller documents is discussed in online HTML style guides, such as the Yale Web Style Manual (http://info.med.yale.edu/caim/StyleManual_Top.HTML) and WWW founder Tim Berners-Lee's Style Guide for Online HyperText (<http://www.w3.org/pub/WWW/Provider/Style/Overview.html>). In addition to nav-

igational problems, a large page loads and reloads more slowly and may download more information than needed. Suites of small pages allow the user to choose what to download.

2. Recently, a new product (LiveCard, 1996) has appeared that allows HyperCard stacks to run over the Web, but that was not the intent of the projects described here.

3. Interested readers may request a copy of the second example stack from the authors.

4. One advantage of a single long page is that it permits use of a Web browser's built-in Find command. This can be accommodated by creating a "long page" version and allowing the user to load as required.

(Manuscript received October 1, 1996;
revision accepted for publication January 16, 1997.)