Creating Documentation with Word for Windows

Last year, during Tech-Ed '94, Scott Boggan presented an interesting and comprehensive paper on using Word for Windows to create Windows Help and Documentation files. In the next few issues of *WindoWatch*, we will be taking a closer look at the latest versions of two of the common Windows authoring tools: *DocToHelp* version 1.6 and *RoboHelp* version 3.0.

The following article is published with permission.

Presented by: ***Scott Boggan***

Introduction

In the beginning, Windows™ Help development was a tedious, error-prone task. Working with few, if any, automated tools and word processing applications that seem undernourished by today's standards, Help authors laboriously added footnotes and control codes to build topic (.RTF) files that the Help compiler would recognize.

It is no coincidence that the advances in Windows Help authoring over the last four years parallel the evolution of Microsoft® Word for Windows itself. Word for Windows 6.0's powerful WordBasic macro language and flexible customization options give Word for Windows the horsepower to accommodate even the most complex Help assignments.

Today's Help authors also have access to Word for Windows add-ons that are designed for producing online Help, the most notable being RoboHelp and Doc-To-Help. RoboHelp is optimized for creating a Help system from scratch and Doc-To-Help focuses on converting printed documentation, but both share one thing in common: they integrate with Word for Windows to automate much of the drudgery once associated with creating Help.

Session Objectives

This session will demonstrate how Word for Windows can simplify Help authoring. We'll look at a simple WordBasic macro that makes authoring Help more efficient and examine the most popular automated Help tools commercially available. Finally, we'll look at a few tips and techniques used to customize Help and discuss a few issues on upgrading from Word for Windows 2.0x to Word for Windows 6.0.

This session assumes that you have a basic understanding of manually authoring Windows Help (that is, adding topic footnotes and control codes by hand). While automated Help tools reduce the need to slavishly edit topic and project (.HPJ) files, possessing these skills will help you better assess your needs and evaluate the technologies that are available.

How Word for Windows Can Help

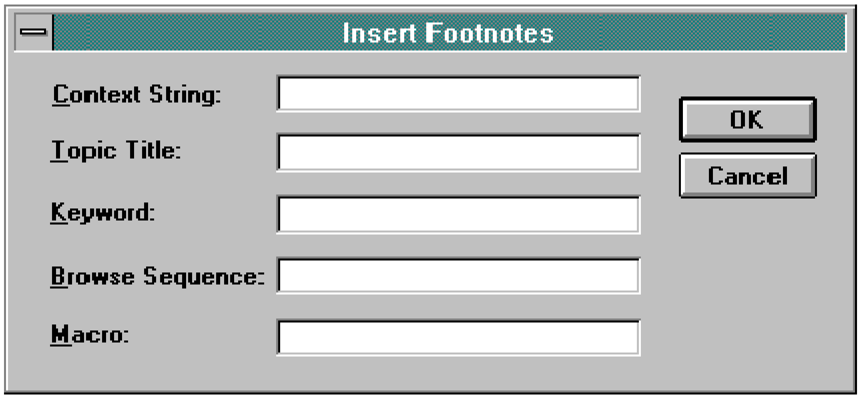
This section examines how WordBasic can automate a common Help development task. We'll perform the task manually and compare it to using WordBasic to complete the task.

Manual Authoring

Creating topic footnotes is one of the most laborious Help authoring tasks. Consider the steps: choose Footnote from Word for Windows' Insert... command, type the custom footnote mark, then type the appropriate text (context string, topic title, keyword, browse sequence, or macro) in the footnote pane. Since most Help topics have four footnotes, this becomes a large job over the course of even a medium-sized Help project.

A Simple WordBasic Macro

Using Word's Dialog Editor and the WordBasic macro language, it's easy to create a dialog box that prompts you for topic footnotes and inserts them into the topic file using the correct custom footnote marks. By letting you enter all of the footnote codes at once (instead of using the Insert Footnote command over and over again) this simple macro substantially decreases development time.



Using Word's dialog editor and WordBasic, you can automate routine Help authoring tasks

We won't go into the WordBasic code used to create this macro (see "Development Resources" at the end of this paper for more information), but without a great deal of effort you can create WordBasic macros that automate routine Help development tasks.

Automated Tools

Although WordBasic is relatively simple to use, planning, coding, and debugging a comprehensive set of Help macros is a considerable task (and probably tough to justify under the deadlines of your “real” job).

Fortunately, third-party vendors have done the work for you. This section focuses on two of the most popular Help development systems: Blue Sky Software’s RoboHelp and WexTech System’s Doc-To-Help. Both combine sophisticated WordBasic macros with code contained in external Dynamic Link Libraries (DLLs) to provide an authoring environment that automates the process of creating Help.

The similarity between the two products ends there, as each reflects very different assumptions about Help development. RoboHelp is aimed primarily at the author who is creating Help from scratch, while Doc-To-Help excels at converting existing print documentation into Help.

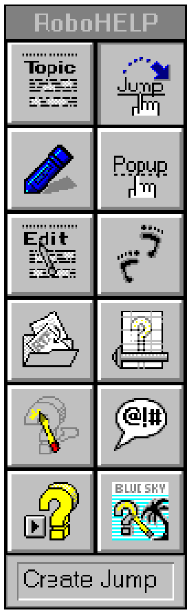
As inevitably happens with competing products, the features in Doc-To-Help and RoboHelp are beginning to converge. Both products now feature point-and-click access to Help macros and Help window definitions. The latest offering from Doc-To-Help (version 1.6) includes several extensions to Windows Help that provide valuable services (such as a Help navigation tool, a Help installer, and support for 256-color bitmaps) that have nothing to do with document conversion. Similarly, version 2.0 of RoboHelp introduced several features that convert print documents to online Help.

RoboHelp

RoboHelp is aimed primarily at the author who is creating Help from scratch, creating and debugging hypertext jumps, pop-up definitions, keywords, and graphics as he writes. With its floating toolbar and features such as its icon-based Help project file editor, RoboHelp provides a rich graphical interface to the various Help authoring components. This section walks through the main steps involved in creating Help using RoboHelp.

Creating a Help project. You first create a new Help project by opening RoboHelp’s Word for Windows template. RoboHelp displays a dialog box for you to name the project, choose the Help compiler you want to use (HC30, HC31, or HCP), and decide where to store the Help project. Of course, if you’ve already got an existing Help project, you can easily convert it into a RoboHelp project.

RoboHelp's floating toolbar. A floating toolbar appears in Word for Windows when you activate a document based on the RoboHelp template. The toolbar contains several options that let you quickly perform the most common Help tasks.

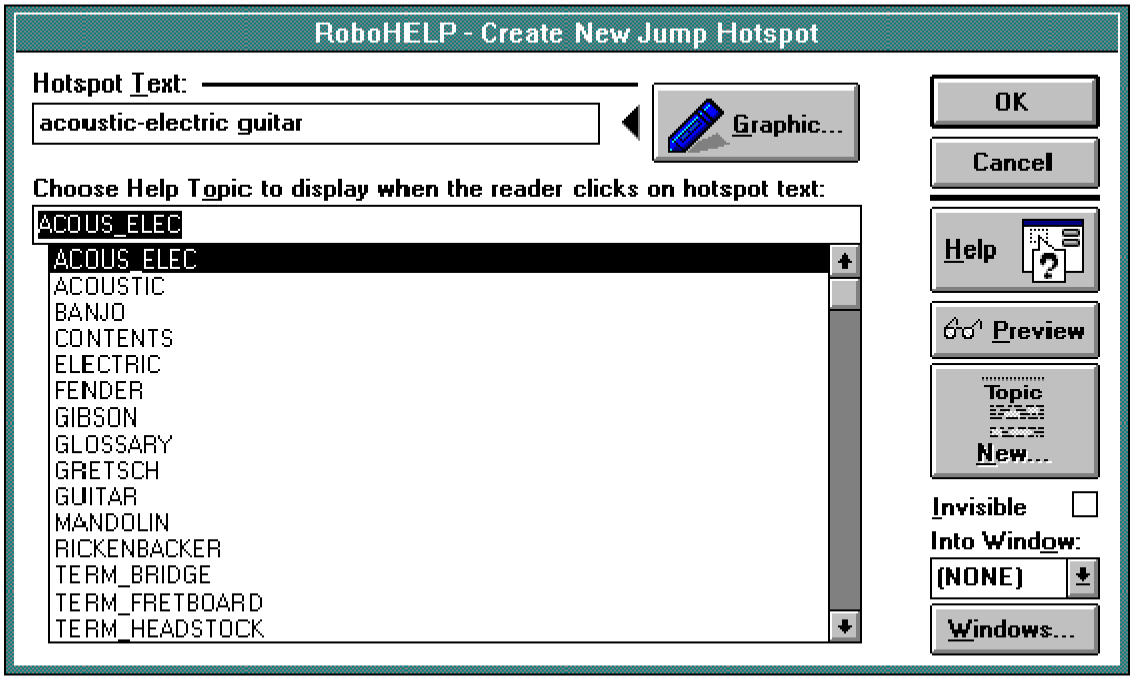


RoboHelp's floating tool palette

Left column	Right column
Create a topic	Create a jump
Add a graphic	Create a pop-up
Edit a topic	Locate a topic
Generate source (.RTF) files	Setup the project (.HPJ) file
Compile Help file	Run the Error Wizard (shows and explains compiler errors)
Run compiled Help	Display RoboHelp's Help

Creating topics, jumps and pop-ups. The toolbar's point- and click-access makes it easy to create new topics: it adds the proper footnotes for context strings, browse sequences, keywords, and macros. When you're creating a jump or pop-up, RoboHelp displays a list of all the context strings in the project. This makes it nearly impossible to create an unresolved jump or pop-up byreferencing an invalid context string (easy to do when entering the strings manually.

.



Hotspot Text: acoustic-electric guitar

Choose Help Topic to display when the reader clicks on hotspot text:

ACOUS_ELEC

ACOUS_ELEC
ACOUSTIC
BANJO
CONTENTS
ELECTRIC
FENDER
GIBSON
GLOSSARY
GRETSCH
GUITAR
MANDOLIN
RICKENBACKER
TERM_BRIDGE
TERM_FRETBOARD
TERM_HEADSTOCK

Invisible ☐

Into Window: [NONE]

Creating a hotspot is greatly simplified by a list containing all of the topics in the Help project.

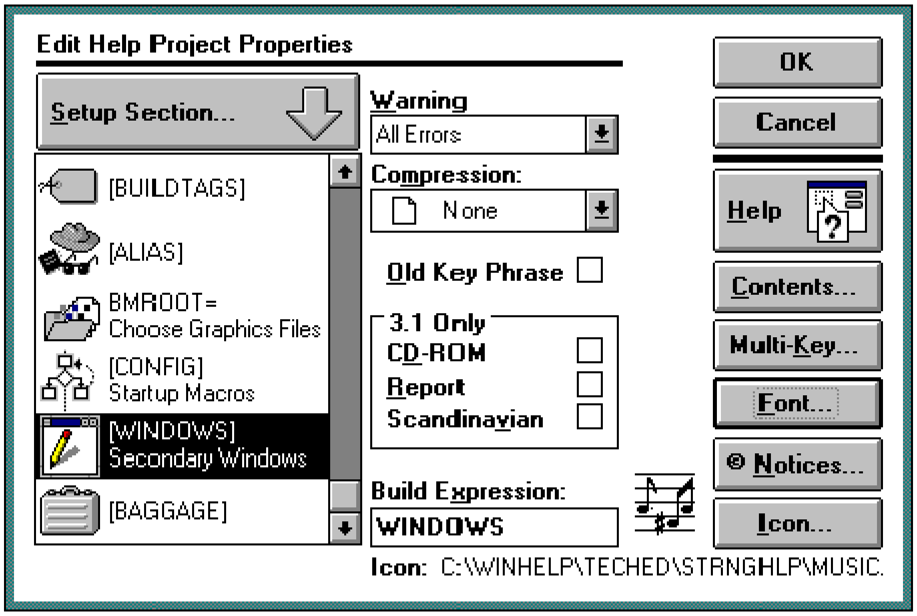
RoboHelp includes another toolbar called the Link Tester that further reduces errors: double-click a hotspot and it displays the topic with which it is associated.

Adding graphics. **Choosing the Insert Graphics button displays a dialog box from which you can view bitmaps and edit them using a bitmap editor (similar to the Mac's FatBits editor) included with RoboHelp.**

The Insert Graphics button also provides quick access to the Hotspot Editor (SHED.EXE) for you to create "multiple hotspot" graphics, where different areas of the graphic jump to different topics. RoboHelp has enhanced the Hotspot Editor so that it displays a list of the topics in the project and the syntax for the 50-odd Help macros.

Creating browse sequences. **RoboHelp automatically creates browse sequence strings in your topic files. Reordering browse sequences (another mundane Help development task) has been simplified through Word for Windows' outline feature: to reorder a topic in the browse sequence you simply drag it and drop it to the intended location. Reordering numbered browse sequences is more difficult, but numbers aren't usually required unless you want to create branched sequences (or "multiple-level browse sequences").**

Editing the Help project file. **Help's project file controls how the Help compiler builds the topic files. RoboHelp's graphical point-and-click interface frees the Help author from memorizing the syntax used for the nine project file sections and their various keywords.**



Warning: All Errors

Compression: None

Old Key Phrase ☐

CD-ROM ☐

Report ☐

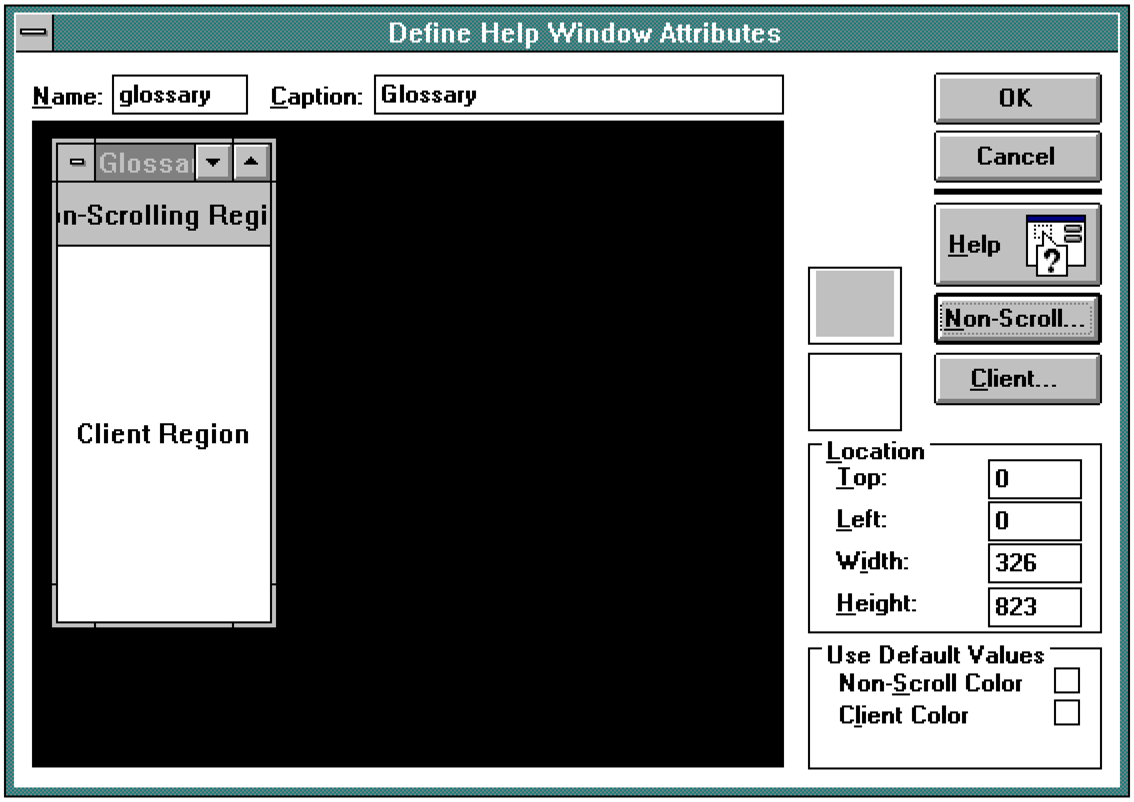
Scandinavian ☐

Build Expression: WINDOWS

Icon: C:\WINHELP\TECHED\STRNGHLP\MUSIC.

Point-and-click access to all of the various options makes it easy to setup Help project files.

Customizing the Help file using secondary windows and Help macros. **Secondary windows are one way to customize the presentation of your Help material, and RoboHelp's visual window designer makes it easy to define window size, color, and position for secondary (and main) Help windows.**



Name: glossary

Caption: Glossary

Location

Top: 0

Left: 0

Width: 326

Height: 823

Use Default Values

Non-Scroll Color ☐

Client Color ☐

RoboHelp lets you visually define Help window attributes.

Windows Help includes over 50 commands (or "macros") that you can use to customize your Help file. To help you remember the macro syntax, RoboHelp includes a built-in editor that gets you started with placeholders for each of the macros. On the down side, however, RoboHelp doesn't perform any macro syntax checking until you actually compile the Help file.

Debugging the Help file. **Before you compile the Help file, run RoboHelp's Wizard to test for broken hypertext links, duplicate context strings, and other errors. The Wizard helps analyze common errors and navigates through the topic files so you can fix any bugs before actually compiling the file into a binary .HLP file. Such precompilation saves valuable time, especially on a large Help project.**

Building the Help file. RoboHelp includes all three Help compilers: HC30 for Windows 3.0; HC31 for Windows 3.1; and HCP (a protected-mode version of the Windows 3.1 compiler for large Help files). RoboHelp runs the Help compiler you specify from Word for Windows (either as a background task or in an MS_DOS™ window), then displays a viewer that lets you jump directly to any errors in the topic or project files.

Summary. RoboHelp is an outstanding product that provides point-and-click access to nearly every aspect of Windows Help development. Its seamless integration with Word for Windows gives advanced and novice Help authors alike an elegant and easy-to-use development environment.

Blue Sky Software (800) 677-4946

Doc-To-Help

Reflecting a "print-centric" view of documentation development, Doc-To-Help excels at creating printed documentation and then converting it into a Windows Help file.

Using the structure of a print manual and its various cross references, index entries, and glossary definitions, Doc-To-Help uses sophisticated WordBasic macros to convert the formatting into the control codes recognized by the Help compiler. The following table summarizes Doc-To-Help's conversion logic.

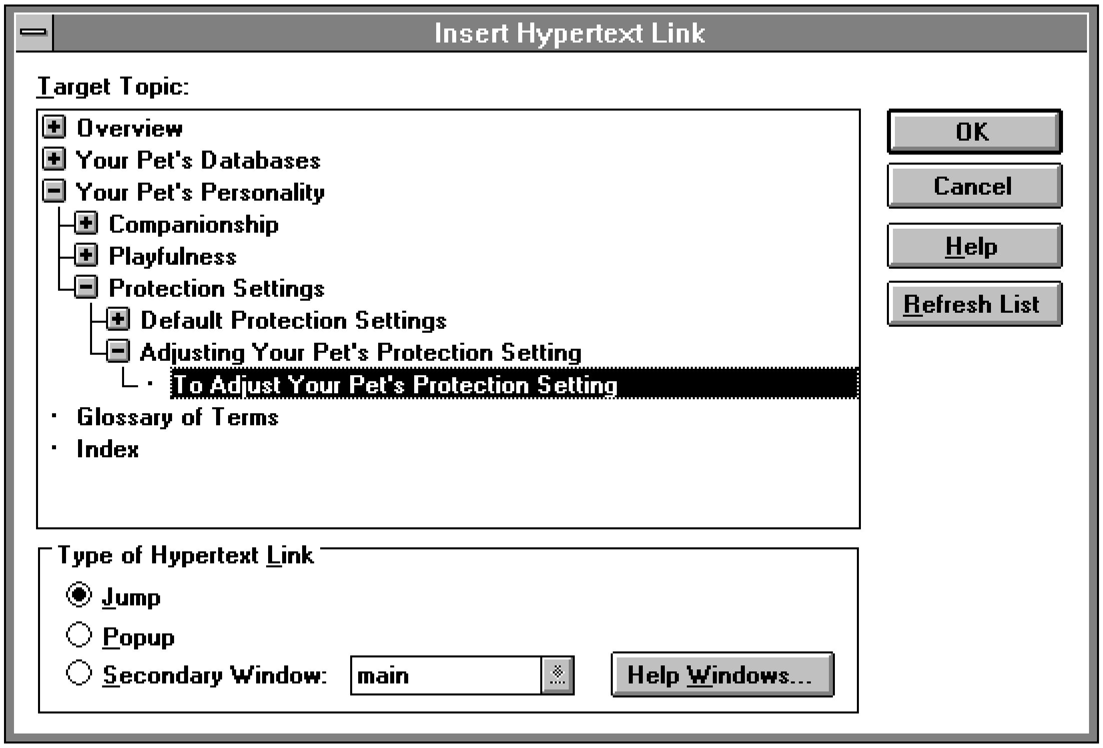
Print component	Help component
Table of contents	Main Contents screen
Heading style and associated text	Individual topics
Glossary entries	Pop-up definitions
Index entries	Search keywords
Cross references	Jumps
Topic order	Browse sequence

This section briefly highlights how to use Doc-To-Help to convert a print document into online Help and provides an overview of the new features in Doc-To-Help version 1.6.

Create a document based on D2H template. You must first create a new document based on one of Doc-To-Help's three professionally designed templates (8.5" x 11", sideheads, and 7"x 9"). If you've got an existing document, you can import it into this "shell" document.

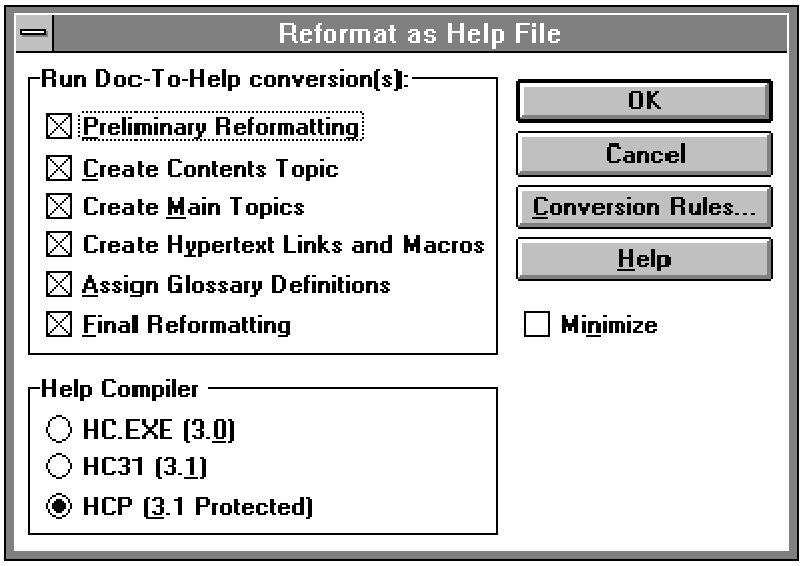
Apply styles to the text. Next, you either import or write your text. Since most of Doc-To-Help's conversion logic is based on style formatting, you must carefully apply Doc-To-Help's styles to your text. You can redefine the style attributes, but you must use Doc-To-Help's style names.

Insert additional hypertext links. To take advantage of the online medium you'll certainly want to create additional jumps and pop-ups. Doc-To-Help simplifies this by assembling a list of topics based on the headings in the document.



Doc-To-Help displays a list of topics for you choose from when creating a hotspot.

Reformat the print document as a Help file. Now Doc-To-Help is ready to convert the various components of the print document into a format that the Help compiler can recognize. This dialog box tells Doc-To-Help to work through the document, automatically coding topics, titles, and browse sequences based on the styles you used to format the text. It also creates jumps, pop-ups, and keywords. This step can take awhile (200 topics can take 30 to 60 minutes on a 386 computer), but you'll end up with an RTF (Rich Text Format) topic file that is ready to be compiled.


Reformat as Help File

Run Doc-To-Help conversion(s):
- ☒ Preliminary Reformatting
- ☒ Create Contents Topic
- ☒ Create Main Topics
- ☒ Create Hypertext Links and Macros
- ☒ Assign Glossary Definitions
- ☒ Final Reformatting

Help Compiler
- ○ HC.EXE (3.0)
- ○ HC31 (3.1)
- ● HCP (3.1 Protected)

Doc-To-Help prepares to convert the print manual to a topic file that is ready to be compiled into Help.

Create a project file and compile. **The final step is instructing Doc-To-Help to create a Help project file and run the Help compiler to build the .HLP file. Like RoboHelp, Doc-To-Help comes with all three Windows Help compilers.**

New in Doc-To-Help 1.6

The newest upgrade to Doc-To-Help — version 1.6 — includes several new utilities in addition to Word for Windows 6.0 compatibility. In a departure from its focus on converting print documents, these utilities extend the power of Windows Help. Each is backward compatible with Doc-To-Help 1.5, and most of the utilities (including each mentioned here) work with any Help system regardless of whether it was developed using Doc-To-Help.

As of this writing, version 1.6 is scheduled for release during the first quarter of 1994.

Doc-To-Help Navigator. **One of the greatest challenges for Help authors is designing their system such that the user doesn't get "lost in hyperspace." Many times, the only difference between a well-designed Help system and a merely passable one are the navigational aids provided by the Help author.**

The Doc-To-Help Navigator presents Help users with a window displaying a collapsible outline view of the current Help file. Unlike the common Help workaround — a dummy outline that only provides one level of expansion at once — the Navigator features multiple levels of expansion, allowing the user to quickly jump to any topic. By anchoring the Navigator window to a corner of the Help window, the user can always see the location of the current topic in relation

to the rest of the Help file. The Navigator also solves Help's printing deficiencies by allowing the user to select multiple topics — in sequence or out of sequence — and send them to a printer.

The Navigator is also valuable for the Help author. It can help locate the appropriate topic for a given context in the application and easily plug the context string or map number into the application programming code. This allows Visual Basic®, Microsoft Access®, and Visual C++™ programmers to develop context sensitive Help without writing additional code or distributing a runtime custom control (like a VBX file).

And finally, since the navigator lives in a brand new window — not in a secondary Help window — Help authors now have a total of three windows at their disposal.

256-color bitmap support. **Whether you need to illustrate a complex task or just want to improve the appearance of your Help file, you can use Doc-To-Help's support for embedded windows to overcome Help's 16-color barrier. Unlike other third-party DLLs, Doc-To-Help does not require the Help author to insert embedded window references in the topic file or edit the [BAGGAGE] section of the project file.**

Help system installer. **As Help systems become more sophisticated, they often require the Help developer to distribute several additional files (including DLLs, multiple .HLP files, and multimedia files) for the user to install.**

Doc-To-Help 1.6 includes a comprehensive setup program for installing the various files used by your Help system and creating a Program Manager group on the user's system. The program lets authors compress the files and build the disks without doing any programming. The installer is especially valuable for distributing standalone Help systems (that is, without an application).

Summary. **If you're using Word for Windows to create a print manual and using the manual to build online Help, Doc-To-Help is indispensable. Transforming an entire printed document to Help in one step sounds impossible, but Doc-To-Help's ingenious collection of macros do just that.**

On the other hand, if you're not interested in a print manual, Doc-To-Help's metaphor may seem indirect and confusing, like driving from Seattle to New York by way of Los Angeles. In particular, the initial steps seem awkward (it doesn't feel productive to create print-style index entries, table of contents, and so on, just to watch them get converted into their online counterparts), but in the end Doc-To-Help certainly automates much of the tedium and gets the job done.

ww

The new utilities included in version 1.6 add functionality to Windows Help and will be of interest to all Help authors, regardless of whether they use Doc-To-Help to create their topic files.

WexTech Systems, Inc. (212) 949-9595

Tips and Techniques

Automated tools will produce a professional-looking Help file, but for a truly custom look you may be looking for a few other tricks. Here's a few tips and techniques for adding more polish to your Help system, plus some notes on upgrading from Word for Windows 2.0x to Word for Windows 6.

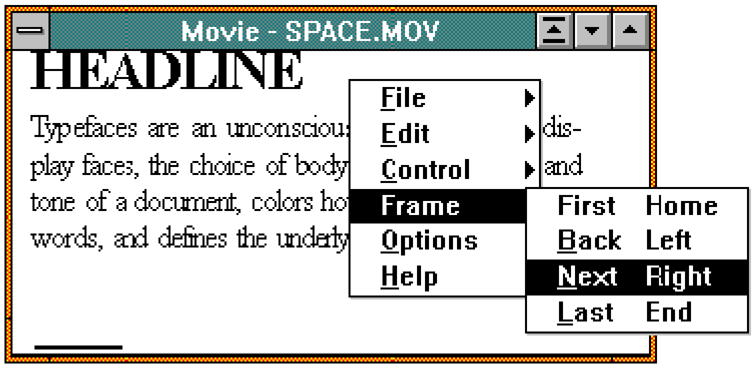
Extending Help with DLLs

Windows 3.1 Help's ability to call functions stored in external DLLs presents the Help author with many options for customizing Help. For example, Help authors can use DLLs to create embedded windows, allowing Help to display 256-color bitmaps (instead of the usual 16 colors) and animation sequences.

Lantern Corporation's Movie Development Kit includes a DLL that provides support for embedded Help windows. Movie presents Help authors with an easy way to extend their Help file without writing their own embedded window DLL, and you can ship Lantern's DLLs with your Help system free of charge.
This section briefly looks at creating an animation with Movie and creating an embedded Help window.

- Create the graphics. **The first step is creating the animation file with a paint or draw program. The example uses a series of five animations illustrating the effects of various letter spacing options on text legibility.**

- Create the Movie file. **Bring up the Movie editor and capture the animations, saving them as a .MOV file.**

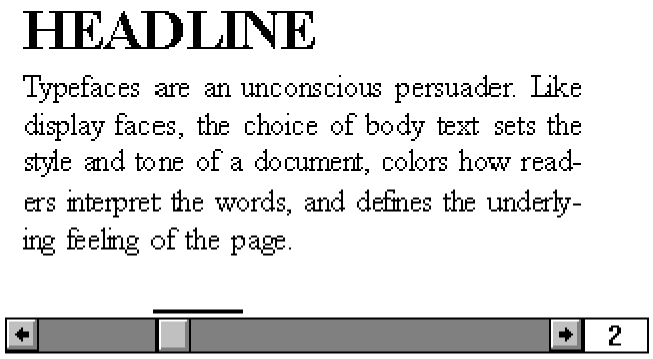

Movie - SPACE.MOV
HEADLINE
Typefaces are an unconsciou
dis-
play faces, the choice of body
and
tone of a document, colors ho
words, and defines the underly
File
Edit
Control
Frame
Options
Help
First
Home
Back
Left
Next
Right
Last
End

The Movie editor is used to capture and manipulate the animation sequence from a graphics program.

- **Embed the Movie file.** **Placing the following embedded window reference in the topic file creates the window when the user displays the topic. The command inserts SPACE.MOV into Help with an ID of 2, adds a slider bar and number at the bottom of the window, and removes the window border.**

  {ewc VoyEwh.DLL, VoyEwhMovie, ;ID=2 ;File=space.mov; Border=False; Toolbar=Bottom; Slide; Num}

- **Compile the Help file.** **Although most DLLs must be registered in the [CONFIG] section of the Help project file using the RegisterRoutine macro, Movie's DLLs don't require you to do this unless you want to use one of Movie's built-in macros. When compiled, the embedded Help window looks like this:**



HEADLINE

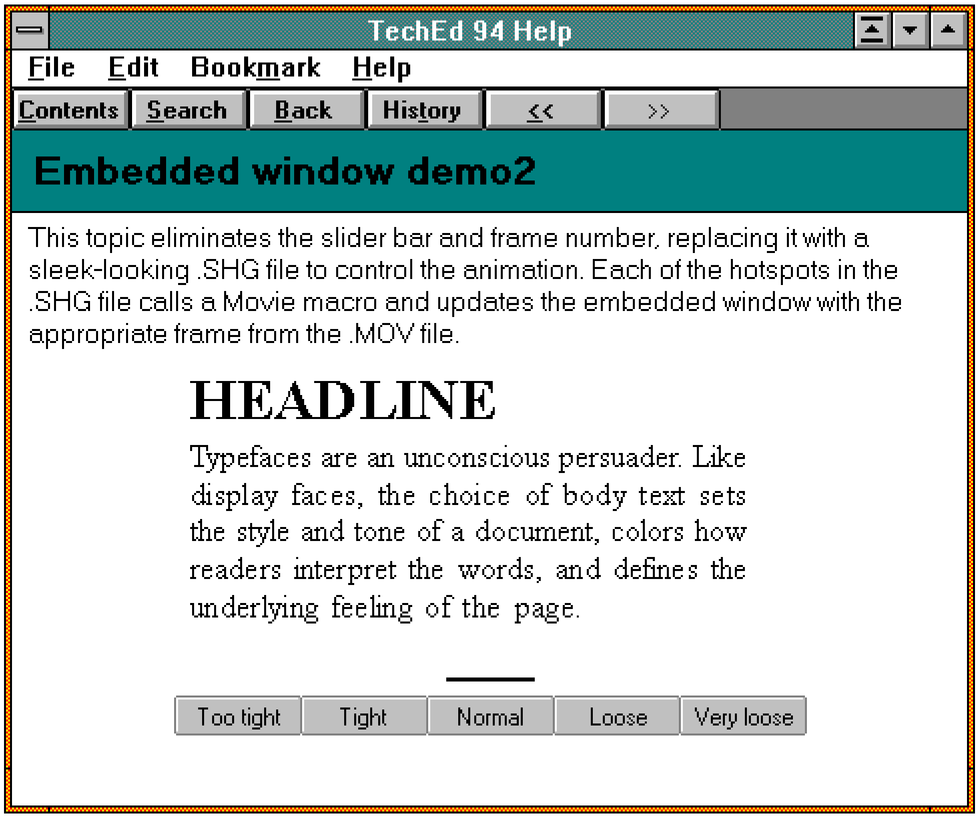
Typefaces are an unconscious persuader. Like display faces, the choice of body text sets the style and tone of a document, colors how readers interpret the words, and defines the underlying feeling of the page.

An embedded Movie window as it appears in a Help file.

You can distribute your .MOV files along with your Help file, or store them inside the binary .HLP file using the BAGGAGE command.

- **Customizing the embedded window.** **You can customize the embedded window by disabling the slider bar and replacing it with a multiple hotspot graphic created with SHED.EXE. Each hotspot runs a Movie macro that**

jumps to the appropriate frame. (Note: This requires you to register the appropriate Movie macros in the Help project file.)



Embedded window demo2

This topic eliminates the slider bar and frame number, replacing it with a sleek-looking .SHG file to control the animation. Each of the hotspots in the .SHG file calls a Movie macro and updates the embedded window with the appropriate frame from the .MOV file.

HEADLINE

Typefaces are an unconscious persuader. Like display faces, the choice of body text sets the style and tone of a document, colors how readers interpret the words, and defines the underlying feeling of the page.

An embedded window with the slider bar removed and a .SHG file substituted in its place.

Summary. **Movie makes it easy for Help authors to include animations and 256-color bitmaps in their Help system without writing their own DLLs or choosing another hypertext authoring system (such as Microsoft Multimedia Viewer).**

Lantern Corporation (314) 725-6125
Movie Development Kit 4.0

Handling Special Characters

One component of style in documentation is special characters such as smart quotes (‘ ‘ “ “), em dashes (—), en dashes (–), and bullets (•). Unfortunately, the RTF tokens Word for Windows uses for these characters are not usually recognized by the Help compiler, so when you compile the Help file these characters are missing and appear as blank spaces.

There is a workaround: Open the RTF file as Text Only and use Word’s Search and Replace to replace the special .RTF tokens with hexadecimal ANSI codes.

The table below lists the RTF and hexadecimal tokens for each character. Each of the RTF tokens includes a trailing space, so be sure to include a space in the Search string (but not the Replace string).

Character	RTF token	Hexadecimal token
Left single quotation mark	\lquote	\’91
Right single quotation mark	\rquote	\’92
Left double quotation mark	\ldblquote	\’93
Right double quotation mark	\rdblquote	\’94
Bullet	\bullet	\’95
En dash	\endash	\’96
Em dash	\emdash	\’97

This procedure is automated using the WordBasic macro called FixSpecialChars. To convert the RTF tokens to their hexadecimal equivalents using the macro, follow these steps:

1. Save the topic file in .RTF format and close the file.

2. Make sure Word for Windows is set up to confirm any file conversions. For Word for Windows 6, this is done by selecting the Confirm Conversion check box in the File Open dialog box. Word for Windows 2.0x users should select the Confirm File Conversion check box in the General section of the Options dialog box.

3. Choose the Open command from the File menu, and specify the name of the RTF file in the File Open dialog box.

4. Word for Windows displays the Convert From dialog box with Rich Text Format highlighted. Select Text Only and choose OK.

5. Choose Macro from the Tools menu, and select the FixSpecialChars macro from the list. Choose OK to run the macro.

6. Choose Close from the File menu to close the document. When prompted to save modifications, choose the Yes button.

   The topic file is now ready to compile.

Keep in mind that not all fonts support the entire ANSI character set, particularly bitmap fonts like MS Sans Serif. If the character is still missing after you compile the Help file, you should reformat the character using a TrueType font like Arial or create bitmaps and use the bmc statement to place them in the topic file.

Upgrading to Word for Windows 6.0

Help authors may experience a few slight problems when upgrading from Word for Windows 2.0x to Word for Windows 6.0. Each of these are explained below.

Displaying bitmap fonts. **Although Windows 3.1 includes several TrueType fonts, there is no guarantee that all users will have TrueType enabled. For this reason, many authors choose to format their online Help using bitmap fonts like MS Sans Serif and MS Serif.**

If your default printer is a laser printer (which is likely since most Windows systems use either a LaserJet® or PostScript® printer) Word for Windows 6.0 won’t display bitmap fonts onscreen. Instead, it substitutes TrueType fonts in their place, which makes it very difficult to format your topic file since the line endings will change after you compile it into Help.

To display bitmap fonts in Word for Windows 6.0, you must use a dot matrix printer such as the Epson FX-80. If you’ve got a dot matrix printer installed, choose Print from the File menu, then choose the Printer button and select the dot matrix printer. For information on installing a printer driver file, see the *Windows User’s Guide*.

Avoiding Help compiler errors. **Due to minor changes in the format of RTF files written by Word for Windows 6.0, you should get copies of the latest Help compilers. Although most topic files will compile without problems, some types of formatting may cause errors with older versions of the Help compilers.**

As of this writing the most recent version of the HC31 compiler is 3.10.445, and the latest HCP compiler is version 3.10.505. To find out which version of the compiler you’ve got, type the executable name (for example HCP) at the MS_DOS prompt.

You can download the latest Help compilers from Section 16 of the WinSDK forum on CompuServe. You can reach the SDK forum by typing go winsdk at any prompt.

Converting WordBasic 2.x macros. **Word for Windows 6.0 will convert the WordBasic macros in your Word for Windows 2.x templates automatically. Although most of your 2.x macros will work fine in Word for Windows 6, you may need to modify parts of them by hand to complete the conversion.**

Microsoft has published a white paper called *Converting Word for Windows Version 2.x Macros* that explains some of the gotchas in converting WordBasic

macros. If you’re having problems converting your macros, you should get a copy of the document.

Help Development Resources

The following publications are valuable resources for Help developers.

- ***Microsoft Word for Windows Developer's Kit* (Microsoft Press)**
- ***Converting Word for Windows 2.x Macros* (Microsoft white paper)**
- ***Developing Online Help for Windows* (SAMS Publishing) Available direct by calling (800) 228-5331**
- ***Microsoft Windows Help Authoring Guide* (Microsoft Developer Network CD-ROM)**

Scott Boggan is the co-author of Developing Online Help for Windows (SAMS Publishing). He has been developing Help since the introduction of Windows 3.0. Scott develops print and online documentation for ElseWare Corporation, a Seattle company specializing in portable font technologies.

Scott Boggan
ElseWare Corporation
101 Stewart Street Suite 101
Seattle, WA 98101
(206) 448-9600
scott@elseware.com
73160,2412