Lecture Notes:

Creating Electronic Documentation

Freda Salatino

Principles of Electronic Doc

Everything we've talked about thus far this term gets pulled in when it's time to create electronic documentation:

- Modularity
- Navigation
- Layout
- Usability

Modularity

Online "real estate" constraints urge the author toward greater modularity -- regardless of whether you're creating from scratch or "shlepping the book online."

Cut to the chase, presenting information at the lowest possible level of granularity.

Navigation

Online navigation should be as flexible as possible; a mix of linear and non-linear.

Don't be afraid to use all of the following:

- Cross-referencing (jumps, pop-ups)
- Indexing (K links and A links)
- Browse sequences
- Full Text Search options

Layout

Design your online page for maximum readability.

Online help should always be designed to share the screen with the application itself.

Online doc should be optimized for onscreen reading. Consider font size, page length, reduced emphasis on scholar's margins, and the 5" line length.

HTML doc can be similarly optimized by use of a CSS (Cascading Style Sheet).

Usability

Electronic documentation is not portable, so make your point quickly and insightfully. Tell users:

- What they can do with the software
- When they should do it
- Why they should do it
- Where features are located
- How they can do it

Types of information best put online

- Context-sensitive

    Help on dialogs

    What's this?

- Tool tips
- Step-by-step (procedural)
- Conceptual
- Reference

What you need to get started: Online help

- The software you're documenting
- Online help compiler
- Help Project File
- Source text in RTF format
- Optional:
- Contents file
- Header file (context-sensitive help only)

The Windows Help Compiler

Anything you use to compile client-side, Windows-Style help is going to have, at its core, the Microsoft Windows Help Compiler:

- HCP or HC31, for Windows 3.x (16-bit)
- HCW and HCRTF, for Windows 95/NT (32-bit)

Most Mac- and UNIX-based online help compilers duplicate the functionality of HCP/HC31.

The Help Project File

Text file that specifies:

- Default topic
- Fonts/character set
- Build tags to be used
- Where to look for .BMPs
- Other files to include in the build
- Macros or DLLs
- Copyright information
- Compression

- Sorting language
- Files to be compiled
- FTS options

Source Text in Rich Text Format

The WinHelp compiler is still optimized to prefer RTF created in Microsoft Word

Use hard page breaks to separate your topics

Navigation is created by use of footnotes

Jumps and pop-ups are created by use of underlining and hidden text

Artwork is imported by reference

Footnotes For Navigation

Unique identifier for this topic.

Provides the “hook” that identifies this topic in a jump or pop-up. In context-sensitive online help, provides the explicit link that forces this topic to display when the user requests Help (presses F1).

Can be any combination of ASCII characters; may include the underscore or dash character.

For example:

```
#    how2make_this
```

$ Topic title.

Identifies this topic during an Index search. Should be duplicated in the online Table of Contents. Should be English.

For example:

```
$    Holiday Crafts In Malaysia

$    Database Creation
```

K Index marker (alphabetical).

Provides the ability to locate topics through either Index or Full Text Search.

You can create individual “K notes” for each item, or create a single footnote that indexes the entire topic.

For example:

```
K    Alphabetical index markers;index
     markers,alphabetical; footnotes,K
```

\+ Browse sequence marker.

Defines a sequence of topics, to be browsed in the order in which they appear in the source RTF file.

Note: Must be used in conjunction with the BrowseButtons option in the Help Project File.

There can be more than one browse sequence set up for a help system. Be careful when naming the sequences and setting the browse order.

For example:

```
+    GRAPHICS 006

+    DATABASE 002
```

If the topics to be browsed are contiguous within the RTF file, all topics can be keyed to “000.” The compiler will set them up to be browsed in the order in which they appear.

For example:

```
+    000
```

Footnotes For Navigation *(Win95 Only)*

A Index marker (by key word).

Provides the ability to group topics by key word, then display all related topics to the user in the context of a particular online help topic.

You assign one or two “A-link” key words to a topic; the topic can then be retrieved by programming a “Related Topics” button.

For example, to program a button that displays all topics coded with the key word “windows,” you might type:

```
{button, Alink(windows)} Related topics
```

To code the “windows” A-link in each topic, you would type:

```
A    windows;uniquetopicID
```

Jumps And Pop-ups

Pop-ups are links to topics that appear above the place you just clicked, without replacing the original help topic.

Code a pop-up with a single underline in your RTF file. Code the topic ID (# footnote) in hidden text:

text to be linked[topic id]

It will appear in the compiled help file as a green link, with a dotted underline.

Jumps are links to topics that either replace the current topic, or appear in a secondary window.

Code a jump with a double underline in your RTF file; as with a pop-up, the topic ID must be formatted as hidden text:

text to be linked[topic ID]

It will appear in the compiled help file as a green link, with a single underline.

Importing Bitmaps

Artwork in compiled electronic documentation, is always in bitmap format. It is always imported by reference.

Always begin the reference in the approximate location where you want the bitmap to appear in the compiled help.

Use the format:

```
{bxy bitmapname.bmp}
```

Where…

b denotes that the referenced file is a bitmap

x denotes the desired vertical alignment of the bitmap within the allotted space

y denotes the desired horizontal alignment

The most commonly used reference is **bml**.

What you need to get started: HTML help

- The software you're documenting
- Source text in text format
- Browser
- Optional:

  Cascading Style Sheet

  WYSIWYG text editor

HTML Help Is Easier

There's nothing to compile

It's incredibly easy to hook it into the software you're documenting

You can use frames to provide navigation!

Since HTML is a WWW standard, electronic doc created in HTML is inherently portable. You can single-source your HTML help for all platforms that support a browser

You can create HTML Help without ever using a Microsoft product

HTML Help Has Its Drawbacks

For now, even if you use Microsoft's HTML Help Toolkit, the functionality of the online help you create is equivalent to Windows 3.1-style client-side help.

Secondary windows, built-in indexes, support for printing multiple topics per page, "what's this?" mouse-over help, all needs to be programmed in Javascript

Millions and millions of teeny tiny files…….