WordFormat

Introduction

WordFormat is capable of reading Microsoft Word documents, in version 1, 3 or 4 format, or text files, and making various kinds of systematic changes to them. It can at present output the result as a Word document in version 1 or 3 format. In some ways it can make up for the absence of a macro facility in Word, although it cannot do everything that a proper macro facility could do. On the other hand, it can do some things much more quickly and conveniently than a macro facility ever could.

Among the changes that WordFormat can make to files or documents, is to allow character formatting (bold, underline, etc., as well as font and size changes) to be specified along with changes to the text itself. Also, style sheets in Word version 3 are supported. One particular job that WordFormat is well suited to, is the situation where you need to take files (perhaps from other computers) that have embedded codes for formatting in the form of special character sequences, and convert them to a

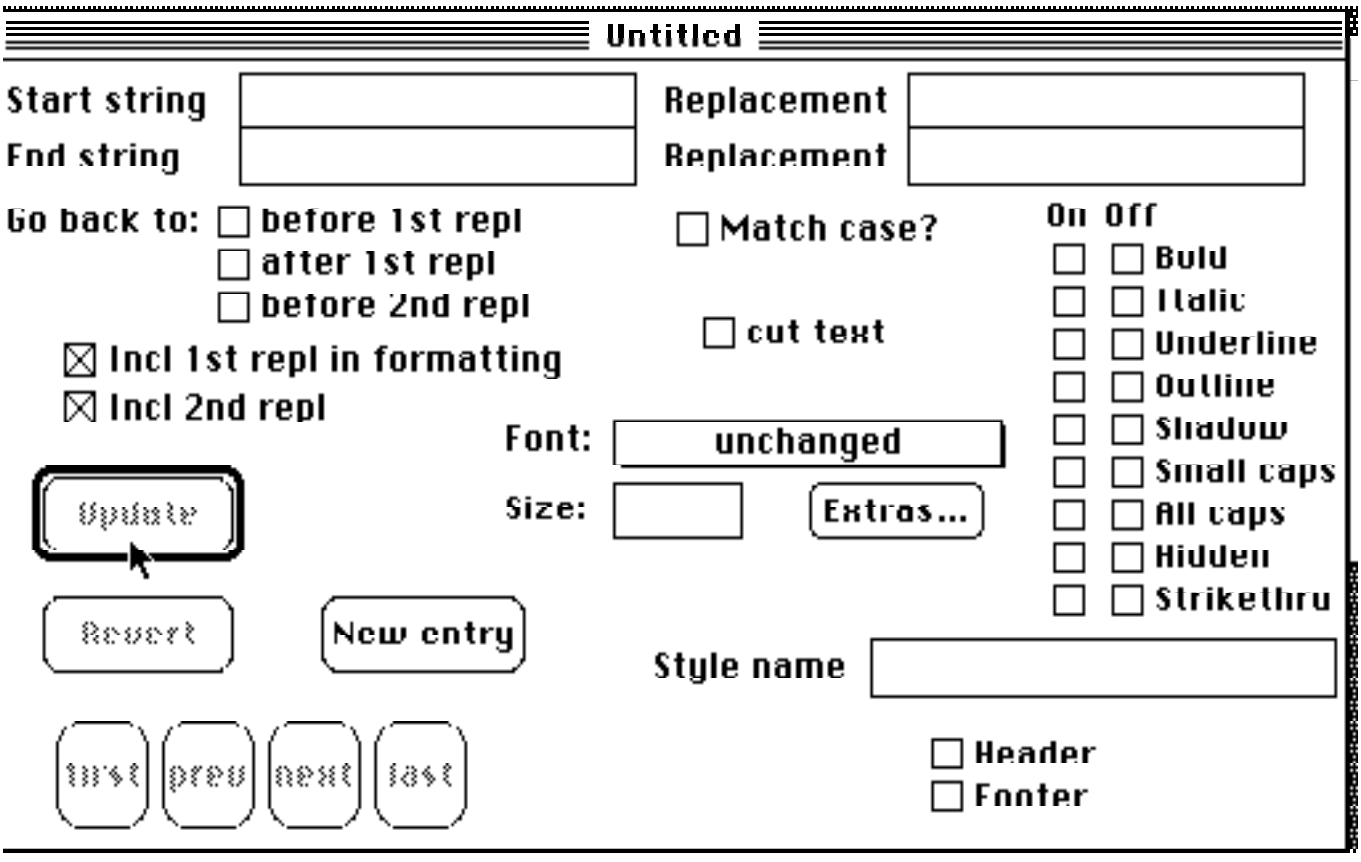
form more suitable for the WYSIWYG style of the Mac. This could also be the first step in a desktop publishing operation. The idea is not to try to do all the final formatting at this stage, but to do the bulk of the routine "hack work". For this reason we don't attempt to handle absolutely everything. Some things are best done in Word itself, where you can really see what is happening. We also don't attempt paragraph formatting, since style sheets are the most convenient way of doing this. We do allow a style name to be attached to any paragraph, which is the "hack work" part of using style sheets. The actual style definitions are left for you to do in Word - this is easy, especially as you can import your style definitions from another document in one operation.

Format tables

WordFormat's operation is controlled by a "format table", which specifies the changes to be made to any document that is processed. There is always a "current" format table. The current table can be changed at any time by opening a new table. Also, editing is always available on the current table.

When a document is processed by WordFormat, the current table is always the one that is used for the changes.

Each format table consists of a series of table entries, each one specifying a single change to be made. The current table always has a "current entry" which is the one displayed on the screen in a dialog box:



This display should be fairly self-explanatory.

Many formatting operations you can perform in Microsoft Word are available. Paragraph formatting is not, as we said above, as this is best done using style sheets.

As you can see, a table entry contains four strings, the "start string", the "end string", and two "replacement strings". Here's what these do. When a file is being processed, the program tries to find a match between a string of characters in the text of the file, and the start string. When a match is found, the matching string in the text is replaced by the first replacement string. Then the text is searched for a match with the end string. Assuming a match is found, the matching string is replaced by the second replacement string, and all text between the two match positions will appear in the format you specify in that table entry. We will give more detail on this operation below.

The dialog box contains buttons to let you move to the next table entry, the previous entry, or directly to the first or last entry. Once you start editing in the dialog box, you won't be able to move from the

current entry until you click either Update or Revert, or type RETURN which is the same as clicking Update. The "extras" button displays another dialog box which allows you to specify the point size and also to superscript or subscript, expand or condense the formatted text. These items used to be in the main dialog, but have been moved to this subsidiary dialog in version 4.6. This subsidiary dialog remains open, and really functions like a logical extension to the main dialog. If you have a large enough screen, the dialogs can be put side by side.

Formatting a document

To use the current table to format a document, choose "Format a document" from the Convert menu, or type Command-F. A standard file dialog box will appear to allow you to choose the input document. When you have done this in the usual way, the document will be read into memory, and another dialog box will appear to allow you to specify the output document name. At this stage you can eject the input document disk if you wish -

the document will already be completely in memory and its disk won't be required again (unless you are sending the output document to the same disk, of course). The formatting operation then proceeds. If you want to retain the existing formatting of a document, and add some new formats, styles or whatever, choose "Merge formats" instead.

You have a choice of output formats - Word version 3 or version 1 (the document formats are different). You can select the output format by choosing "Options" under the Convert menu. The default is Word version 3. WordFormat will read documents from all versions of Word—versions 1, 3 or 4, or straight text documents (without formatting). WordFormat will recognize these different document types and handle them automatically. If you want your final document to be in Word 4 format, use Word 3 output format from WordFormat. Word 4 will read this in just as quickly as its native format, and automatically convert it to its own format the first time you make a change. Because this operation is so quick and transparent, there is no real need for WordFormat to

be able to output Word 4 format directly. This could be added, but would make the program even bigger than it is already.

After running this program you will probably want to look at the output - this will require Word to be run. If you are debugging a format table, it may be easiest to set up WordFormat and Word to run under MultiFinder so that you can have a fast debugging cycle, assuming of course that you have enough memory. Remember to allow WordFormat enough memory to hold the document you are working on. See "memory requirements" below.

Details of operation

The way in which a formatting operation proceeds is as follows: Starting with the first character of the file being processed, the program tries to find a match with the start string of an entry in the current table. This search starts with the first entry and goes from "left to right" - that is from the beginning to the end of the table. The search may be case sensitive or insensitive - you may specify which by choosing "Options..." under the Convert menu.

If no match is found, the program moves to the second character of the file and starts from the beginning of the table again. Whenever a match is found, the changes specified by the table entry are carried out. Then if there is no end string to be matched, the program continues with the character in the file following where the first replacement string was put in. The first replacement string appears with the formatting specified. Following this string, any formats that were turned on by this table entry are turned off, and vice versa. Any other formats are left alone. If there was an end string, the program continues with the character following where the **second** replacement string was put in. Formatting specified by this table entry is applied to both replacement strings and all the intervening text.

Styles are handled slightly differently, since styles can only apply to whole paragraphs. If a style is specified, the program scans backwards for the first carriage return from the end of the first replacement string. This is where the style will come into effect. Normally there will be a carriage return (specified

by ^p) within the first replacement string - if not, you will get a warning when you edit the table entry, but you can ignore this if you know what you're doing. If there is no end string, the style will remain in effect until another style is specified. If there is an end string, and it contains a carriage return, "Normal" style will be put into effect there.

More sophisticated changes can be made using the "Go back" boxes. These allow an exception to be made to the normal forward scan through the file being formatted. If a table entry matches, and has one of the "Go back" boxes checked, then after the changes and formats have been applied, the program will continue processing from the character before or after the first replacement, or before the second replacement (depending on which box is checked).

Possible pitfalls

The "go back" feature is powerful, but dangerous! Careless use could put the program into a loop! The program checks for obvious error conditions such as the start string appearing as a substring within either

replacement string, when the "Go back" option that has been checked would cause that replacement string to be re-scanned. But there are many more subtle loop possibilities that we can't check for, so care is needed.

Another problem that can easily occur if you are making many complicated transformations in a table, is that you may end up with overlapping format ranges. This will lead to output that may be interesting, but certainly won't be what you want. The best way to avoid this is by judicious use of the two check boxes (new in version 3.1) "Incl 1st repl in formatting" and "Incl 2nd repl". These boxes give you a bit more control over the exact range of text to which the formatting will apply, that you specify in that table entry. The default (i.e. what you get in a new empty table entry) is to include both replacement strings in the formatting, which is what earlier versions of WordFormat always did.

Setting up a format table

If you start WordFormat by double-clicking on its icon, or if you choose "New format table" under the

File menu, you will be presented with a new untitled table, with one empty entry displayed. You can then set up this entry, and go on to set up further entries by clicking on the button "New entry" which creates a new empty entry just after the one you were looking at. This button has the same function as "Insert entry after" (Command-A) under the Edit menu. Alternatively you can create a new empty entry **before** the one you are looking at, by choosing "Insert entry before" (Command-B).

You can move between entries in a table in several ways. The main dialog has buttons to let you go to the first, previous, next or last entries. These have command-key equivalents, command-1 to command-4 (the same as Hypercard). Also the Goto menu will let you go straight to a particular entry - the start string for each entry appears in this menu with a tick beside the current one.

You can move entries within a table by using Cut and Paste. The operation of Cut, Copy and Paste may be slightly non-intuitive at first glance, but everything should fall into place once you realize

that the current entry is thought of as "selected" in the normal Mac sense. Cut deletes the current entry, and leaves you looking at the preceding entry (or the following entry if you were at the start of the table). The entry is saved on the clipboard. Copy does what you would expect, and copies the current entry to the clipboard without deleting it. Paste **replaces** the current entry with the entry in the clipboard, if any. So, to move an entry to another place in a table, choose Cut, then go to the place in the table where you want the entry to go, insert a blank entry there (in one of the ways given in the last paragraph), then choose Paste.

Special characters

Special characters such as tab can be specified in search or replacement strings, using almost the same scheme as Word uses in its "Find" dialog:

^t tab
^p carriage return (new paragraph)
^n new line (shift-return)
^b blank (space)
^s non-breaking space

^h non-breaking hyphen
^d section mark or forced page break
^g graphic
^v paste from clipboard
^^ ^

Case is not significant, so for example ^t and ^T are equivalent.

Graphics

We don't attempt to do anything terribly clever with graphics. If a graphic is read in an input document, the actual picture is not retained; all that happens is that its place is marked with a dummy box of default size - the same as if you had chosen "Insert Graphics" in Word. Likewise, if you use ^g in a replacement string, the resulting output will be a default box. This feature is only available for Word 3 output format.

Headers and footers

If you check either the "header" or "footer" box for a particular format table entry, then whenever that table entry applies, the matching text in the

document is moved into the header or footer. It will no longer appear in the body of the document. At the moment this is the only way of specifying headers or footers. Any headers or footers that are already present in an input document will be ignored. This omission should be corrected in the next version.

In the "Options" dialog (choose "Options..." under the Convert menu) there is a "Facing pages?" box. If you check this box, the output document will have the facing pages option. You will also see that the Header and Footer check boxes in the main dialog are replaced by Even header, odd header, even footer and odd footer, as required for a document that has the facing pages option.

This headers and footers support is only available for Word 3/4 output format.

Styles for Ready,Set,Go!

RSG supports style sheets, but unlike PageMaker these are incompatible with Word. If you have a Word document with styles, and you import it into RSG, the styles will be lost. But don't despair, help

is at hand. RSG does allow you to pre-specify styles in an ASCII text document, by placing the style name in angle brackets thus: <someStyle>this text will appear in style "someStyle". This is called "tagged text". WordFormat will output tagged text if you specify this option in the "options" dialog. If you specify this option, the tagging will be the **last** operation performed by WordFormat before it outputs the text. Thus if you have angle brackets in your input document but have converted them to something else in the WordFormat run, there will be no confusion. Also if you have styles in your input document and have added more styles in the WordFormat run, it will be the final result that will be tagged. This should be what you want.

Unfortunately tagged text has to be just that—text. Any formatting not done through styles will be lost. This is a "feature" of RSG, which we can't do anything about. Sorry—send your complaints to Letraset.

Cutting and pasting

The main dialog now has a check box marked "cut

text". If you check this box, then whenever this table entry matches, all the text which would normally be formatted is cut to a clipboard. This may be used as a way of deleting text, or of moving it. Any table entry with ^v in either replacement string, will, whenever it matches, bring in whatever is on the clipboard at the ^v position. (Command-V means paste, get it?)

Performance

The performance of WordFormat should not cause any complaint. Whereas certain very well-known desktop publishing programs that can read Word 3 documents caution you that "fast saved" documents may be read in very slowly, WordFormat manages a whole lot better. There shouldn't be any need to worry about how the document was saved from Word.

For the formatting operation, WordFormat uses a number of speedup techniques. In particular, before processing a file, it compiles a list of the initial characters in all the start strings of the current format table. It then uses a fast scan (coded in

assembly) through the file for a match with any of these characters, then checks if any of these are genuine matches.

As you can infer from this, WordFormat's performance will be poorest when the format table has a large number of start strings starting with the same character, especially if this character also occurs frequently in the file being processed. We now have a way of alleviating this problem. If you choose Options under the Convert menu, you will see a new item labelled "character for special performance enhancement". Put the offending character there. The result will be that WordFormat, when formatting, will build a subsidiary table of the **second** characters in all the start strings that start with the character you specified. This subsidiary table will then be used to speed up the search every time the specified character is encountered in the document. I expect the speedup will be quite significant in many cases. But of course, if performance hasn't been a problem for you, you don't need to do anything. "If it ain't broke, don't fix it."

Memory requirements

The WordFormat program requires about 100K of code to be resident in memory. Sorry about that, but it is partly the result of having to cope with the complexities of Word documents. To process a straight ASCII text document or a Word 1 document, the additional memory needed is little more than the size of the document itself. To process a Word 3 document, the additional memory needed is roughly double the size of the document, because of the extra work needed to handle the "Fast Save" format. If the document (Word 1 or 3) has a large number of font etc. changes, and "Merge Formatting" is chosen, the memory needed will increase somewhat.

WordFormat itself makes some estimates about the memory it will need for a particular job. Instead of just crashing if there is insufficient memory, it will give you a nice friendly helpful and informative message, and then crash. Hey wait a minute, I meant to say it won't crash. So this will give you the opportunity for some experimentation.

Recovering corrupted documents with WordFormat

This program may come in handy in a few ways that were not originally planned. Probably the most useful is that WordFormat may be able to read some corrupted Word documents that cause Word itself to bomb. Various hardware or software glitches may mangle some of the information in a document, and Word is rather sensitive to everything being in the right place. If it is a "Fast saved" document, then you may be able to recover the text with various file utility programs, but the text may appear in a strange shuffled order within the document. Here is where you could try WordFormat. Use an empty format table, check the "text only" box in the Options dialog, and choose "Format a document". So long as the fast save reordering information is not corrupted, you'll be in business. WordFormat will not even read the formatting information in the document, so if that is where the problem lies, you will recover all your text intact, and in the correct order. You will have lost the formatting, but that's all.

Shortcomings

This is the section you don't find in the documentation of most programs. The most obvious shortcomings are:

1. Incoming graphics aren't retained.

2. Forced page breaks and section breaks are treated as equivalent, so that any forced page breaks in the incoming document will be turned into section breaks.

3. There is no simple way of obtaining a hard copy of a format table. This would probably simplify debugging a complex table.

4. The method of handling special characters with ^p etc. is not particularly Mac-like. It does, however, correspond with what Word does.

5. We can't handle footnotes yet.

For many uses these shortcomings won't matter at all. They don't matter particularly to me. If I receive enough tear-stained letters I might be able to justify the time to do something about them. If it

turns out that nobody cares, then I won't either.

Copyright

 If you like it, you don't have to send any money (although I won't object if you do). I wrote it for myself and some colleagues because we needed something like it, in getting material originally prepared on a variety of machines ready for desktop publishing. It is useful to us, and I hope it may be useful to others as well.

Acknowledgement

WordFormat was originally written in Neon, surely one of the most underrated development systems for the Mac, now discontinued. It was an object-oriented development of Forth—a sort of amalgam of Forth and Smalltalk. Maybe it was ahead of its time. Anyway, as from v. 4.5, WordFormat is implemented in Mops, a home-grown system whose ideas are taken from Neon but which is implemented along different lines. (For those who insist on acronyms having to stand for something,

Mops could mean Michael's Object-oriented Programming System. Well, you did insist.) I do want to acknowledge my debt to Neon for many of the ideas incorporated into Mops.

Bug reports, etc.

The Word 3/4 document format is much more complicated than was the v.1 format. In particular, the "Fast Save" format bristles with interesting oddities. As far as I am aware, I have discovered most of these, and WordFormat will cope with them. Occasionally, however, something may arise that I haven't seen before, and you will get a message saying that the document contains some control information that we couldn't handle. The output may be correct anyway, depending on exactly what the unrecognized information meant. But in any case, saving the document again from Word with "Fast Save" checked off will usually remove the problem. This is a good strategy if you run into any kind of strange happenings with a Word v.3 or 4 "Fast Saved" document. If you get the above message, or anything else peculiar

happens, I would be grateful if you could let me know (with a copy of the "Fast Saved" problem document on disk, if possible), and I will be able to fix my program (and learn something new about the "fast save" format in the process, no doubt).

Finally, please send any other bug reports or requests for new features to me (Michael Hore, Numbulwar, NT 0852, Australia). All suggestions will be carefully considered, and maybe even acted on, eventually.

Finally, sorry for the lack of electronic communication, but this is a very remote community where I'm working on the language of the Aboriginal people who live here. We don't even have telephones yet. Most of Australia isn't like this, would you believe.

History of recent changes

Version 5.1 (November 90)

Goto menu added, also the dumping of a format table to a file as ASCII text, also the "character for performance enhancement" option described under

Performance.

Version 4.6.2 (July 90)

Fixes a few small bugs. The "cut text" option should now work as expected with different combinations of "include 1st replacement in formatting" and "include 2nd replacement". Another bug causing strange "percentage done" numbers to appear while processing files greater than 64K long has been fixed. I know, for most of us, it would be nice to have a percentage done that is a 10 digit number, but unfortunately WordFormat was kidding us.

Version 4.6 (March 90)

Added cut and paste feature. Also moved the point size and displacement sections of the main dialog to a subsidiary dialog, to reduce screen clutter (and make room for more features??). Improved error handling (if an illegal number is typed where a number is expected, we don't crash any more).

Version 4.5 (Oct 89)

Removed some bugs in the handling of Word

documents that made their unwelcome appearance in v. 4.0. Otherwise functionally identical to v 4.0, but switched implementation from Neon to Mops (see above). This has resulted in a considerable performance improvement, and reduced the size of the program as well.

Version 4.0 (July 89)

Can read Word 4 documents. Can read footnotes, and pass them through to the output document.

Version 3.2 (June 89)

Conversion of style names to Ready,Set,Go! "tagged text" format is now available, via a check box in the "options" dialog.

Version 3.1 (Dec 88)

Two check boxes have been added to the main dialog, "Incl 1st repl in formatting" and "Incl 2nd repl". See above under "Possible pitfalls".

A new button "new entry" has been added (with the same function as "insert entry after" under the Edit menu). This makes the process of setting up a new

format table a bit more intuitive.

A "text only" check box has been added to the Options dialog box (under the Convert menu). This box allows you to tell WordFormat to write a document out as an unformatted text file.