WHY I AM NOT A WORDIST:
PHILOSOPHICAL CONSIDERATIONS ON GRAPHICS-ORIENTED WORD PROCESSORS

by Lorenzo Peña

····· **http://www.jurid.net/dos/word.htm** ·····

Table of Contents

---

§1.— Two Different Views, to Start With

Back in the late 80's the word-processor MicroSoft-Word (henceforth **MSW** or Word) was starting to compete with WordPerfect.

The latter program was overwhelmingly prevalent in almost all circles, academic or otherwise, but even at that stage there was no shortage of supporters of *MSW — wordists* as I shall be calling them.

Their reasons were the following ones:

1.— Intuitiveness; which I construe as follows: whereas MSW appealed to the senses, by showing you (in an incisively visible way) what was going on and what you were expected to do in order to have your writing entered, stored and formatted, WordPerfect demanded an effort of abstraction, thought and planning on the part of the writer.

2.— Facility; whereas WordPerfect required you to memorize a number of function-key usages, MSW could be operated with almost no previous learning process.

3.— Smoothness; you could handle MSW-tools with less effort, in a more comfortable manner.

On the other hand WordPerfect was acknowledged to excel as regards results, productivity, potency. One of its advantages was the array of charsets it developed since WordPerfect 5.0 (May 1988). Moreover the WordPerfect macro language was strong and powerful, allowing you to do countless things hardly possible with MSW.

I could never verify any of the purported good qualities of MSW. In 1989 I tried a pirated version of MSW (everybody around me was using WordPerfect only). I failed to accomplish anything. Its intuitiveness was lacking as far as I could ascertain. Ever since my relationship with MSW has always been an unhappy one. I found the program slow, unwieldy, cumbersome, clumsy, awkward, unpredictable and baffling. Admittedly I had not studied it, but the almost-zero learning time seemed to me a fable.

There are of course several kinds of perceptual capacity and personality. Other people are surely more clever, more alert, less visually impaired than I am.

As regards WordPerfect 5 for DOS, the need to store commands in memory was not as arduous as that. Furthermore, WordPerfect 5.1 came in 1991 with a pull-down menu bar which reduced the need to memorize key mappings.

All in all I chose WordPerfect for the following reasons:

(1) Most people were then using it.

(2) OCR's were better attuned to save their output in WordPerfect format.

(3) The array of charsets allowed me to produce my papers using mathematical-logic and Greek symbols.

(4) The macro language was uniquely helpful for processing documents in a number of ways (and has remained to this day a decisive motivation for me to cleave to WordPerfect 5.1).

(5) The reveal-codes pane allowed me to control my work and to rationalize the use of formatting codes in order to achieve best results.

(6) I could enter, edit and shape my papers quickly and conveniently to my heart's content and eventually print it with diverse printers always with an elegant presentation.

---

§2.— The Graphic Environment

Enters the graphic environment. Windows was little by little bewitching people. IBM produced the operative system OS/2 which by default was only useable as a graphic environment.

It goes beyond the scope of this paper to delve into the good or bad features of graphic environments in general. At the moment what alone interests me is what such environments brought to word-processing.

(1) One of the most immediate advantages of graphic word-processors was getting over the constraint of charsets. When you write by hand you are free to scrawl whatever you like, whether it is a Latin character or not. For ages that freedom allowed people to insert foreign letters, devise new symbols and mingle text with drawings. Type-writers and text-oriented electronic programmes curtailed such possibilities by restricting you to a choice of letters (and exceptionally a number of previously listed graphic entities). The new graphic word-processors leave such limitations behind, by somehow allowing you to handle your text in a way more similar to hand-writing, which blurs or abolishes the boundary between text and graphics.

(2) Friends of the new word processors claim they are more friendly in so far as they show text and graphics alike black-on-white which is how we view text on paper.

(3) Most of all the *Wysiwyg* concept implemented in those programmes has been looked upon as one of their enhancing merits. Wysiwyg means: *what you see is what you get*. Wysiwyg brings the computer era texts back to the type-writer. When you typed a paper with your type-writer, what you were seeing on you paper sheet was the end result you delivered to your readers. Of course your paper could be then re-typed and printed under a different format altogether; its content was not identical to its form. But you could not use the content without the form unless you retyped it. With character-oriented word processors, on the other hand, content and form parted company. You wrote your paper, edited it, merged it with other pieces if necessary, stored it, shuffled it, cut and pasted blocks, all with little fuss and bustle. In the end, you formatted it as you liked it in order to print it (often with small characters, narrow margins and large sheet-sizes in order to circumvent editorial limitations, if the editors were naive enough to let you play that game).

Now all that is over. Reformatting remains possible but has become much harder. You are supposed to enter your text as if what you were writing was, at the same time, the final outcome you intend to deliver.

(4) Resorting to the mouse has even turned text-handling three-dimensional, since you can jump over line-, paragraph- and page-breaks.

I think those four so-called advantages are liabilities. The main culprit is Wysiwyg.

1.— Wysiwyg brings us back to the days when form and content could not be detached, which gives rise to a muddle or an amalgam. A content can of course exist only under some form or other. However the same content undergoes a number of form variations. There is some kind of isomorphism between those diverse forms, which is why all those presentations or formattings display the same content. As Wittgenstein argued in his *Tractatus logico-philosophicus*, a played melody, the gramaphonic disk, the written score — and indeed nowadays other methods of representation — share the same content and are (somehow or other) isomorphic to each other.

The harder the task becomes of separating content out of a particular form, the further we recede to a backward lack of freedom. A non-Wysiwyg way of handling a text does not try to show it under any particular display (such as it is destined to come out when printed), but, on the contrary, allows the writer to enter the text as an abstract entity, so to speak in a formless way (or under a form

especially adapted to screen viewing and keyboard handling, which is entirely different from such forms as are suitable for printing).

2.— Wysiwyg cannot be true to its purpose, since the screen is a kind of support altogether unlike paper. Physically that is obvious (I am not going to try to explain it). Blind, mechanic imitation of the paper print-out on screen only brings about a display which is very hard to read for visually impaired people. Promised solutions such as larger monitors have proved to be deceptive.

3.— An enlargement of available charsets is welcome. In fact projects such as unicode try to embrace a huge range of established or even artificial scripts and symbols of many languages, even of fictional languages. The larger the range, the harder it becomes to get a clear screen display and a quick processing. Of course we should like to have both, but, electronic resources being scarce, a balance is needed; all in all I prefer to keep clear readability and fast operation rather than to be able to write Chinese ideograms, Sanskrit, Amharic, cuneiform Sumerian, and so on.

Anyway, the spirit of Wysiwyg — at least if carried to the extreme — may imply giving up any finite list of characters, since it tries to blend graphics with text (as people can do in handwriting). But then the encoding of information becomes intractable. (When you read a manuscript usually there is a finite set of characters you assume each minimal part of the script belongs to; but of course your expectations can turn out to be wrong and you can reach the conclusion the writer is just inventing a new script or just drawing lines which mean nothing). From the information-oriented view-point that course of things seems to me more harmful than useful.

4.— Consequently, Wysiwyg means that a character is not taken to be a token of an abstract universal type (chose among a finite list of such types) but is regarded as the drawing you were supposed to have in mind when you typed it.

Therefore, it is automatically equated with its graphical representation within a certain environment; and then all other environments are geared to mirror that chosen or ascribed representation as closely as possible. Take any logic symbol.

In character-oriented word processors a symbol is encoded as a certain item of the list (e.g. [9;99] is the 99th character of the 9th charset); accordingly both the screen-display and the print-out are planned to correspond to the code. Graphics oriented word processors, instead, reduce the symbol to one of its representations, namely the one that presumably was meant by the author, such as the nth character of a certain font (Verdana xxx or whatever), since when

typing you are supposed to see what you both type and get. Any conversion is then aimed at preserving the same graphical result.

5.— Wysiwyg echoes Wittgenstein's maxim that what can be shown cannot be said and conversely. Our modern printing art has produced a variety of styles and fonts. We have lower-case and upper-case, roman and cursive (or italic), bold, and sometimes other appearance-attributes (old-English, Gothic, manuscript, exchequer, underlined, etc). Graphics-oriented word processors show those attributes the same way printed text does.

At the extreme opposite purely character-oriented languages (such as HTML and TeX) handle such attributes as linear segments within the syntagmatic chain. Thus in HTML the italized word 'venue' becomes '<I>venue</I>'. WordPerfect 5.1 chose a middle course: certain attributes could be shown but in a different way (e.g. as colours), but the underlying working approach was linear, as you could always view activating the reveal-codes pane. With reveal-codes on, the italized word 'venue' appeared as '[ITALC]venue[italc]'. This representation allows you a better control.

---

§3.— One Dimension or Two

The main difference between the two approaches is the divide between one-dimensionality and two-dimensionality.

One of the advantages of human language, as against other symbolic systems of communication, is its linearity, which enables us to enjoy a purely oral exchange along one dimension only, namely time. Thus humans can exchange messages while looking at other objects. Linguistic linearity is not absolute, since there are prosodic elements which are non-segmental and thus bring in a certain degree of two-dimensionality (stress, intonation, etc). However all in all our languages can safely be taken to be essentially lineal.

Linearity has also made it possible for our languages to be represented by writing. The main characteristic of our writing systems is that they reproduce linearity on a two-dimensional support by dint of a convention (word-wrapping and line-breaking as well as direction, be it left to right or the other way, or top to bottom or conversely).

A two-dimensional symbolic system such as bee-dance could hardly be represented in a linear way.

Not that such a linear representation is impossible. If the axiom of choice is true and a strong logic and set-theory are accepted (not necessarily classical logic and standard set-theory), then every set can be well ordered; hence any information about an n-dimensional space can be conveyed in a

language all whose messages are linear. From a practical view-point, though, the task would be extremely arduous or practically unfeasible.

Linearity, our main communicational advantage, has given us, humans, the opportunity to translate our languages into writing, printing, coding, and the other way round (decoding, reading).

Wysiwyg waives linearity and handles a written message like a drawing, independently of whether or not it can convey any message. It brings us back to a pre-human system of communication.

What is thereby achieved is the richness of visual information, not necessarily linguistically coded. What is partly sacrificed is the possibility to code and uncode, convert, translate and store the messages under different formats.

How is two-dimensionality obtained within graphics-oriented word processors? Look at a MSW document with a text editor. The document proper lies as plain (extended-ASCII) text in the middle of the file, interspersed with a few control characters, whereas a huge mass of gibberish, or bizarre binary codes, is heaped on top and on bottom of the file. Those lumps of codes contain pointers to items within the document proper. We can imagine something like that: if the 35th and the 82d occurrences of 'house' within a document are italized, there is a coded information to that effect — be it at the top or at the bottom of the file.

This is why, as MSW people have explained, there is nothing to reveal, and so no reveal-codes option is available under MSW. You, the user, cannot reveal codes; codes cannot be displayed as strings of text before your eyes, since they are attributes of text which can be shown, not worded. Words are words, and attributes are attributes.

Such a two-dimensional approach (which amounts to nothing but Wysiwyg) can be defended; but in the end it seems to me wrong, for four reasons:

(1) The pointer approach can only apply to a very narrow range of characters. In such a way you cannot combine Greek, Cyrillic, Western Latin, Eastern Latin, phonetic-notation, mathematical logic, algebra, etc. Wordists are likely to believe that each document is coded as belonging to a certain language and hence to be expressed with a particular charset. Such an assumption is unfounded. There have always been lots of linguistic mixtures.

(2) The pointer approach makes reformatting and conversions difficult. Rather it seems suitable for keeping the document as an unaltered intact block.

(3) The pointer approach hampers the task of preserving a useful distinction between hard and soft codes. Suppose you have a Verdana 12.0 points

font on; you write an italized word: '*virtue*'. A Wysiwyug program is prone to ignore the difference between coding it in a way which we can rendered as '<I>virtue</I>' and in a way which would be rendered as '<Font:Verdana12.0:italic>virtue</Font:Verdana12.0:italic>'. Convert your paper to a different type of document (Open Office or whatever) and choose a printer lacking the Verdana font — using Times Roman instead; the disheartening outcome will be a change of font (and size) at word-crossing.

(A soft code is one which can easily be adapted to reformatting and conversion, one which does not depend on any particular setup you have activated.)

(4) The pointer approach is contrary to the principle of economy. A character-oriented word processor allows you to choose '<I>virtue</I>' (the italized word *virtue*) rather than '<I>v</I><I>i</I><I>r</I><I>t</I> <I>u</I><I>e</I>'. You are free to use whatever you want, but most users prefer the economic choice (rather than the proliferation of lurking codes) for the sake of efficiency and speed.

Such a distinction is unavailable under pointer-oriented word processors, or at least outside the user's control. This is why graphic-oriented programmes produce those monstrously bloated documents: a few pages take up more than one megabyte!

Thus rather than Wysiwyg I prefer **Wysiwym**: *what you see is what you mean*. (Or something in-between which was the two-way approach chosen by WordPerfect 5.1.). I prefer to work with programmes such as TeX or HTML.

---

§4.— Conclusion

Wysiwyg is a two-dimensional procedure by which the screen behaves like a two-dimensional drawing mirroring a likewise two-dimensional sheet of paper, with images, graphics, display-enhancement devices and whatever the author fancies to incorporate into their text.

Information-oriented programs treat a document as a string of words (or monemes) interspersed with attributes (which are thus handled as interpolated syncategorematic monemes).

Information-oriented programs (or hybrid programs, such as WordPerfect 5.1) lend themselves easily to conversion, translation, encoding, editing, reformatting; whereas graphics-oriented programmes are better suited for the task of producing a complex output, half text, half graphics.

It is ironic that wordists are such people as prefer a program best fitted to producing results which are not strings of words. Word fans are not people of the word.

Economists have developed the Hotelling model: imagine a road along which there are two gas stations, near the extremes, and so at a long distance from each other. In order to compete, they entice their competitor's customers; little by little, they move closer to each other; they end up both in the middle of the road.

Such a dynamic may explain why a number of competitors of MSW, such as WordPerfect (and Linux Offices: Open Office, KWord, Abiword etc), tend to become more Wysiwyg, more like MSW.

Yet, the converse tendency has not materialized; as a truly Wysiwyg program MSW can hardly evolve in the opposite direction.

At the time of this writing (2005), the triumph of Wysiwyg-oriented MSW is obvious for everybody to see.

Still, information-oriented programmes and languages are a better solution, for seven reasons, namely:

— Connectedness (or two-way convertibility), thanks to the abstraction of characters and codes.

— Reformattability (same reason).

— Flexibility (each user can tame and customize the program as they fancy).

— Economy and proportionality: time and machine resource-expense ought to be commensurate with the results achieved.

— Control (you see what you mean).

— Universality (you can combine a number of languages and scripts within your text).

— Lightness: tasks are carried out quickly, promptly, with a few key strokes (once you have learned how to enter them).

I hope the information-oriented approach will prevail sooner or later.