"What You See is What You Have Got" — a user engineering principle for manipulative display?

Harold Thimbleby, Department of Computer Science
University of York, YORK, YO1 5DD. UK

Abstract

"What you see is what you get" is generally treated as a technically superficial phrase, but it is an outstanding maxim because both the user and designer understand it in the same way; any reasonable interpretation of it could have been implemented. A stronger phrasing (or rather, the same phrasing more strictly applied) has wider implications for system design and retains the generative potential for users. By providing facilities for "manipulative display", the form of a dialogue may further direct and enhance the users´ interpretation of its content.

Prologue

There are several major obstacles to designing good user interfaces:

(a) The standard ergonomics issue: designers tend to design for themselves unless they are intentionally humble and consciously aware of actual user needs.

In computing, the problem is possibly worse than in other technologies because there is no intermediary (e.g. a craftsman) between formalism and user interface (Thimbleby, 1982c). There is consequently considerable confusion between _user_ and _designer_ interfaces.

(b) User interface design appears to be simple.

Indeed, it _is_ if the interface design takes advantage of users' almost limitless capacity of adaption (and of acceptance).

(c) People who construct interfaces have considerably more control and understanding of them than any other class of user.

In particular, a computer scientist's training (and delight?) is specifically to handle complexity. This results in system designs with an excess of technical options. Users are not necessarily computer experts nor want to become ones by using a system which apparently expects them to be. Users may react emotionally to computer interfaces which require abstract computing skills and experience.

(d) What knowledge there is is multi-disciplinary and not systematised, and thereby may be ignored in preference for necessary but more obvious design constraints (e.g. formal properties of correctness).

Crucially, ergonomics cannot be used inductively – it is notoriously hard to reason about a design except by hindsight. Consequently there is a temptation to define human requirements in terms of abstract metrics. Yet computers are very small (in mind and etiquette) and it is nonsense to define `rational´ behaviour vis-à-vis the interface purely in their terms.

(e) Ergonomics is essentially experimental and case-study driven; and the technology continues to outstrip any experimental basis.

Computer scientists are neither trained in experimental procedures nor have the motivation. Thus computer scientists tend to over-generalise from specific experimental results. Furthermore, the innovative areas where ergonomists might have widest influence are generally too specialised.

(f) Social, psychological and other issues which are outside the immediate man-machine domain are readily ignored.

As an illustration, Carroll (1982) presents the `Adventure´ metaphor: why does a person get addicted to a game interface (where the user takes risks and faces unknown and under-specified tasks) but is unmotivated and feels lost with a structurally similar interface in a `work´ context (e.g. the straight computer interface)?

(g) There is considerable confusion between legitimate design principles and mere slogans.

Particular principles may become no more than slogans without the backing of applied reasoning from other ergonomic knowledge — and designer motivation. A particularly distracting activity consists of promoting prejudice as principle.

(h) Finally, `the human factor´ has become a euphemism for poor design.

The human user is perceived as a source of trouble, and many of his troubles (if noted) are classified as irrelevant to the functional specification and are dismissed as his own responsibility.

This paper is about **user** interface design and concentrates on the popular maxim:

What you see is what you have got

It is a principle that has the distinguishing advantage that it can lead both the designer and the user to have a considerable overlap in their concept of an interactive system — if used with care.

The purpose of this paper is to

- reestablish "what you see..." as a constructive rather than trivially descriptive principle
- demonstrate the utility of principles which can be shared with the user: `true´ **user**-engineering principles.

1.1 Historical aside

I discussed an attribute of interactive system interfaces (Thimbleby, 1979) which has significant similarity with the present topic, but I termed it "passivity" — a phrase which is neither catchy nor, crucially, of any intrinsic meaning to a user. [Note: A passive interface does not imply a passive system.] Now I suggest that a **user**-engineering principle must be expressed in user-referenced terms in order to enable users to verify its effective application for themselves — and to benefit from its consequential generative potential.

1.2 Terminological aside

Some apparent ambiguity in "principle" versus "user model" terminology is resolved: "what you see is what you have got" may be a construct in the user model repertory, and the same phrase may also be taken as a design principle to be enforced. The corresponding principle, in all cases, is ensure the users´ model (Gaines and Facey, 1975).

(A literature survey showed that no one agrees on what a user model is anyway.)

1.3 The popular view

The principle is often more readily (and more weakly) expressed, not in the present perfect tense, but in the present with future implications as in "What you see is what you get". This interpretation, at its weakest, states that hardcopy and display capabilities are sufficiently similar (and reliable) to ensure that what a user can see on a display can later be \`got´ on hardcopy (cf Lipkie, Evans, Newlin, Weissman, 1982; Rutkowski, 1982 etc.). Designers often assert that something, such as a text buffer, can be displayed "at any time": conversely, this is a case of [hopefully] "what you get is what you see".

Another, apparently stronger formulation, (but again with future implications) is "what you see is **all** you get". In fact, this has derogotary connotations: maybe the system is not powerful enough to have anything else to it than what you can see — no programming, poor resolution hardcopy etc. Of course, if you told a user that (truthfully), he would know considerably more about the system than most do!

Note that both these forms assume that what the user wants to \`get´ is not a model of interaction but a tangible result (cf Spence and Apperley, 1982, p53).

Expressed in the present perfect tense as "what you see is what you **have** got" however, the principle further requires the designer to ensure that what the user has <u>now</u> is exactly what he thinks he has from what he can see. This not only requires display integrity, and transparency to backup procedures (and so on), but some means of encouraging the user to acquire a valid interpretive model of what he can see. There are several ways to facilitate him, as will be seen.

Nevertheless, "what you see is what you have got" is certainly not such an appealing phrase expressed so pedantically: but just with such pedantry will interfaces be acceptable (whatever their banners). A referee of an earlier (rejected) version of this paper wrote: "Finally, the author´s use of the present perfect tense for \`What you see is what you have got´ only emphasises the tautology of the logic — <u>of course</u>, you do have what you see" (my emphasis). Another referee wrote: "But the existence of literally dozens of screen-based word processors for the Apple II computer, some bad and some good, is proof that the principle by itself does not necessarily lead to a good interface". Views such as these illustrate the difficulty in reasoning about the interface clearly.

2 Background review

Rarely do the designer and user share common ground, so the designer tends to design for himself or for a small actual group of users with similar professional skills to himself. This situation further distances the designer from the users outside his laboratory, and generally makes his parochially extolled system features irrelevant. [But when the user designs a system, it is even less likely to meet reasonable standards of quality.]

A well defined "user model" is required which simultaneously is user-referenced and has direct operational implications, so it can be used as a basis for so-called user-engineering design principles. The user could have a handle on the system and the designer could match the system to the users´ directed expectations. If ever verbalised, few user models are expressed in terms which relate adequately to both designers´ and users´ needs.

The partial model "What you see is what you have got" has much to commend it in this respect, and it is discussed in this paper. For many graphics and screen based tasks, the principle is highly relevant. Less obviously, the principle is applicable to robot control — indeed to any area of interactive state display (and not exclusively to VDU display). For more abstract applications it may be hard to define what it really is which has to be `got´ by the user. (For text editing it may simply be a fragment of the text being manipulated; but cf Fraser 1980.) One of the major issues attacked by "what you see is what you´ve got" is that it is very easy to provide a user with facilities which enable him to manipulate data (as if he had `got´ it), but to provide no uniform denotation for manipulable objects... thus, the user may have no cues, or even conflicting cues, as to what he may `get´. It is clearly a separate question whether getting something concrete in itself improves the user interface, except for naive or casual users.

2.1 Robots...

Consider a robot arm which may be manoevred by keyboard commands or programmed into a sequence of movements. A user programs a sequence of movements which terminate with the arm at some useful location, but during programming he accidentally issues a <u>non</u>-programmed move (which the user interface permits) which he subsequently compensates for by a <u>programmed</u> move. On replaying the program, the arm fails to reach the correct location, because of unnecessarily programmed compensation.

A the end of programming, what the user can see (namely, the correct arm position) does not signify what the user has (namely the incorrect program).

2.2 <u>Menus</u>...

Consider a five item menu, for which the selection procedure is to type a unique prefix [plus a terminator]. To permit the user to select from larger menus, the last item may be "Others"; so if the user selects "Others", upto five further items are displayed. In order to handle expert users, the prefix has to be unique over <u>all</u> menu items, not just those displayed.

Thus a user may be unable to type a satisfactory prefix because what he can see (namely five items) is not what he has got (namely a larger menu plus an item "Others"). The expert user, on the other hand, can `see´ the other menu items in his imagination.

2.3 <u>Display editors 1</u>...

Consider a user trying to insert an `e´ at the beginning of a line using a typical display text editor. Assume the user has positioned the cursor in the first column of a line, and has correctly set the editor in `insert mode´. He types `e´, and to his surprise finds that the `e´ overwrites the first character on the line.

In fact, the user had positioned the cursor between the carriage-return/linefeed pair at the <u>end</u> of that line (and so inserts the `e´ between them) – it is not possible to deduce that he has got the cursor in this awkward position from what he can see.

2.4 <u>Display editors 2</u>...

An editor has a search-for-string command, and the user searches for a literal string, "Mr. Jones". The editor fails to find "Mr. Jones", but finds "Mrs JONES" for the user instead. The editor tried to be helpful (or, rather, its designer did) and treats "." as a meta-character representing

<any character> and ignores case. Many editors, again with the best of intentions, allow the user to compose commands; so consider a user composing search and delete commands to delete Mr. Jones´ address! What the user can see, namely a string-to-search-for, is not what he has `got´, namely an expression. Hence meta-characters should be displayed in such a way that the user cannot confuse them with literals.

Even this is not a satisfactory solution as in many, or all, cases it would really be better if <space> matched any blank text. For example, would searching for "Mr. Jones" succeed if "Mr." and "Jones" were on consecutive lines? The problem, of course, is what does the user `see´ — what is the user´s model and how best can the implementation ensure the user´s model is compatible?

More issues are covered by Thimbleby, 1982ab, 1983.

3 Positive comments

If we speculate that a user does not have the technical expertise to understand how-it-works knowledge, then it is exceedingly difficult to enable him to construct a coherent model of a system. One way around this problem is to design an interface to exercise perceptual-motor skills rather than abstract symbol processing skills. In fact, speculating naivety is not essential, for by the car-mechanic-who-is-a-bad-driver analogy (or the computer-scientist-who-cannot-type version) we could deduce that a cognitive grasp of an implementation does not imply that it can be used effectively (at least for routine tasks).

Great power comes from visualisation and, as Shneiderman (1982) points out, intuition and discovery are often promoted by visual representation of formal systems. "What you see is what you have got" is an immediately appealing phrase for describing interactive systems because it asserts a simple property from which the user can construe realistic models of what the interface actually provides. The user is inspired to confidence: the system has no hidden states, and the user manipulates what he sees (not some underlying data for which the display is a non-trivial representation). In particular, the user can see the consequences of errors, and does not need to `debug´.

The user is encouraged to exercise perceptual skills with the interface, hence manipulative skills are naturally relevant (cf Brooks, 1977). Perceptual and manipulative skills can be acquired gradually, and as the user develops skill with the interface he may speed up at a self-determined pace. Conversely, acquiring cognitive skills can be a significantly more stressful task because decisions the interface forces on the user may appear to require immediate response, and because of complex social determinants.

It is notoriously difficult to document interactive systems adequately; "interactive systems should be experienced and not talked about" (Gaines and Facey, 1975). If it is obvious what has happened merely from the appearance of a display, nuances of interaction may be glossed over in the documentation (to some extent). A principle such as "what you see..." enhances intrinsic predictability (if implemented properly) and reduces the training load of new users significantly.

In each of the examples (2.1-2.4) above, the system could not maintain an image of the system suggested by what it displayed (at least to the ordinary user). This particular problem arose because the system did too much, by trying to provide too many options, functions or features, or providing too many levels of abstraction. By enforcing "what you see..." at a design stage, the system and its documentation is constrained to (literally) sensible proportions which encourages an orthogonal design rather than a collection of ad hoc features. [The arguments against simple interfaces are reminiscent of the arguments against the `restrictions´ of structured programming; despite frequent exhortations to simplify user interfaces (e.g. Palme 1978), the computing community encourages complexity and repeatedly confuses power with ease of use.]

4 Negative comments

"What you see is what you have got" is actually not enough for the designer, the interface requires additional organisation. For example, it is clearly inappropriate to show the user the computer console and its flashing lights and say, "Well, what you see is what you´ve got!"

Otherwise, the positive points above assume adequate representation and response rate (e.g. acquiring manipulative skills presupposes reliable and rapid feedback). And adequate hardware capability to support this is costly. Also, there is not much spare capability in contemporary display technology, so displays tend to be specialised. Effective interaction consequently implies device dependence: sometimes considered inelegant for technical reasons.

"What you see..." is more often used as a purely rhetorical statement about a system rather than an ideal which could have been achieved (see also the comments in section 1.3). The idea is often partially implemented and the user, not knowing what is easy to implement and what is not, will be led into unexpected traps. Thus he may be worse off than had the principle never been mentioned.

The user may not be looking at the display anyway — for example, if he is copy typing or interviewing — in which case the principle is inapplicable. Actually, there is really no problem if the user cannot pay undivided attention to the display: if display changes are incompatible (e.g. if the display changes without user request or cannot be changed as the user required) the user may be notified audibly or in other sensory modalities* which do not compete with vision.

Users may actually perform better if they use cut-down displays or even command (keyboarded) interaction rather than pointing. The ability to review displayed work reduces productivity (Gould, 1981); Rosinski, Chiesi and Debons (1980) show that copy-typing performance (speed and error rates) are unaffected by varying (1 line) window size, irrespective of typist skill. So, merely getting more information does not of itself improve interaction.

Users cannot cope with excess screen clutter (although how much clutter becomes unacceptable is user- and task-referenced), so if what is displayed is what the user "has" then the system must be simple (especially given contemporary VDU hardware) — for example, the user interface model should be passive (section 1.1). Yet a passive interface may seem too constrained for some task domains or designer environments. As mentioned earlier, designers often have a different view of a system and consequently design more for themselves than for actual users (cf. Shackel, 1979). In Michies's terms (1980) their required "human windows" are at different

* Terminal sounds have social overtones. Some thought must be given to problems such as whether <u>only</u> errors are notified by a specific sound, which everyone else can notice too.

implementation levels: the programmer-designer would find a good system for non computer experts "shallow" (but generally fast).

Finally, the application system may structure what the user can see in a special way (for specific task purposes) which restricts the user from manipulating the data in an `obvious´ fashion. For example, a program editor which is designed to make editing program text easier (or certainly less error prone), may enforce syntactical constraints during interactive editing which will almost certainly prohibit certain `obviously trivial´ (e.g. textual) changes to the program.

5 Discussion

Principles like "what you see is what you have got" (properly interpreted) are operational, unlike the vogue terms "user friendly", "fluid dialogue", "ease of use", "natural" etc. They can be used to form design guidelines AND as user-oriented explanations. Thus the designer is encouraged to think in user-referenced terms. Since the user is not necessarily a computer expert and therefore cannot explain (or understand, or accept, or accommodate to) system behaviour in terms of implementation, it is crucial to follow user-engineering principles to extreme limits and to pay nit-picking attention to detail. The designer´s extreme concern, like common sense, will often appear utterly trivial later (which is another reason why `simple´ interfaces are never made).

In the absence of suitable guidance from the designer how a system should be conceptualised (via the system, colleagues or documentation), users will invent their own models (e.g. Nickerson, 1981), and without assurance that these models are appropriate. Experiments by Gaines (1976) show that this sort of intuited model is over-complex and difficult to generalise (the user therefore expends considerable effort in frequently revising their model). The designers´ intended model and the users´ actual models may be at such variance that there is little mutual understanding, and little possibility of system improvement. There is therefore considerable advantage to be had by presenting the user model (or axioms) to the user before any other aspects of the system (cf Mayer, 1981).

Because we require essentially immediate feedback, display updates may be best achieved locally, without resort to backing store, distributed resources or (often time-shared) application processes (Baeker, 1980, p138); if feedback is achieved locally, the feedback will be simpler and more consistent (as it must be independent of detailed task semantics).

Morse (1979) outlines a number of designer principles for the effective use of displays.

6 Conclusion

"What you see is what you have got" is easily understood by both user and designer, and is expressed in such terms that

- The model can be understood and is readily memorised by the user without prior system experience.
- The user may generalise his knowledge. The user is confident as to what has happened (e.g. after an error), and does not need debugging skills.
- The user is encouraged to employ perceptual-motor skills fully. This is especially motivating.
- The designer can use the principle to meet clearly defined user expectations with specific techniques.

...the principle is simple, general and natural.

Having once suggested a specific user model, the designer is under an obligation to ensure its effective implementation through careful system design, which should maintain the user model as understood by the user — this approach also entails evaluation.

6.1 Generative principles...

Any principle which can be shared with the user so that the designer can `implement expectations´ (derived by the user from the principle), could be classified as "generative". We must search for similar generative user-engineering principles for other areas of man-machine dialogue, especially in tasks where there is a large assumed context with respect to the man-machine interface bandwidth (e.g. in expert systems).

Obvious relatives of "what you see is what you have got" include "what you do is what you get", and — to emphasise predictablity — "you can use it with your eyes shut" (which implies non-visual diagnostics).

Finally, we need more understanding of why and when guidelines are valid.

References

R. Baeker (1980), Towards an effective characterisation of graphical interaction, IFIP Workshop on Methodology of Interaction, Seillac II, 127-147, North Holland, Amsterdam.

F. P. Brooks, Jr. (1977), The computer "scientist" as toolsmith - studies in interactive computer graphics, Proceedings IFIP Congress, Information Processing 77, (ed.) B. Gilchrist, 625-634, Toronto.

J. M. Carroll (1982), The adventure of getting to know a computer, IEEE Computer, **15**, no. 11, 49-58.

C. W. Fraser, A generalised text editor (1980), Communications ACM, **23**, 154-158.

B. R. Gaines (1976), On the complexity of causal models, IEEE Transactions on Systems, Man & Cybernetics, **SMC-6**, 56-59.

B. R. Gaines & P. V. Facey (1975), Some experience in interactive system development and application, Proceedings IEEE, **63**, 894-911.

J. D. Gould (1981), Composing letters with computer-based text editors, Human Factors, **23**, 593-606.

D. E. Lipkie, S. R. Evans, J. K. Newlin, R. L. Weissman (1982), Star graphics: An object-oriented implementation, ACM Computer Graphics, **16**, no. 3, 115-124.

R. E. Mayer (1981), The psychology of how novices learn computer programming, ACM Computing Surveys, **13**, 121-141.

D. Michie (1980). Problems of the conceptual interface between machine and human problem-solvers, Experimental Programming Report, **36**, Machine Intelligence Research Unit, University of Edinburgh.

A. Morse (1979), Some principles for the effective display of data, ACM Computer Graphics, **13**, 94-101.

R. S. Nickerson (1981), Why interactive computer systems are sometimes not used by people who might benefit from them, International Journal of Man-Machine Studies, **15**, 469-483.

J. Palme (1978), How I fought with hardware and software and succeeded, Software — Practice and Experience, **8**, 77-83.

R. R. Rosinski, H. Chiesi & A. Debons (1980), Effects of amount of visual feedback on typing performance, Proceedings of the Human Factors Society, **24**, 195-199, Los Angeles.

C. Rutkowski (1982), An introduction to the Human Applications Standard Computer Interface, Part 1: Theory and principles, BYTE, 7, 291-310.

B. Shackel (1979). The ergonomics of the man-computer interface, Infotech State of the Art Report on Man/Computer Communication, 2, 299-324.

B. Shneiderman (1982). The future of interactive systems and the emergence of direct manipulation, TR-1156, Department of Computer Science, University of Maryland, MD 20742.

R. Spence & M. Apperley (1982), Data base navigation: an office environment for the professional, Behaviour and Information Technology, 1, 43-54.

H. W. Thimbleby (1979), Interactive technology: the rôle of passivity, Proceedings of the Human Factors Society, 23, 80-84, Boston.

H. W. Thimbleby (1982a), Character level ambiguity: consequences for user interface design, International Journal of Man-Machine Studies, 16, 211-225.

H. W. Thimbleby (1982b), Basic user engineering principles for display editors, Proceedings ICCC´82, London, 537-542.

H. W. Thimbleby (1982c), Interactive systems design: a personal view, Proceedings IEE International Conference on Man|Machine Systems, Manchester, 118-122.

H. W. Thimbleby (1983), Guidelines for `manipulative´ text editing, To appear, Behaviour and Information Technology.