Title
PAPER AND GLASS: GRAPHIC DESIGN ISSUES FOR SOFTWARE DOCUMENTATION

Author
Marcus, A.

Publication Date
1982

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Physics, Computer Science & Mathematics Division

To be presented at the National Bureau of Standards, Federal Information Processing Standards Software Documentation Workshop, Washington, DC, March 3, 1982; and to be published in the proceedings

PAPER AND GLASS: GRAPHIC DESIGN ISSUES FOR SOFTWARE DOCUMENTATION

Aaron Marcus

January 1982

Prepared for the U.S. Department of Energy under Contract W-7405-ENG-48

DISCLAIMER

Paper and Glass: Graphic Design Issues for Software Documentation

Aaron Marcus, Staff Scientist

Computer Science and Mathematics Department
Lawrence Berkeley Laboratory, 50B-3238
University of California
Berkeley, California 94720

Graphic design principles have been utilized in redesigning the interface for an information management system and for prototypes of typographicly enhanced textual programs. These principles are explained and examples of typical formats are shown to indicate the nature of improvements.

1. Introduction

Most programs and their supporting documentation pass through many stages of development, use, and maintenance. These software documents may appear offline on paper or video. They may also appear online displayed on a paper or glass-faced terminal. These documents communicate their contents to the reader primarily through alphanumeric symbols. These pages or screens of information must effectively communicate intentions, states, structures, and processes. While good conceptual organization and verbal editing are crucial to effective communication, a third component, the graphic design of these documents, has been neglected.

Graphic design is the discipline concerned with the communication of informational, emotional, and aesthetic content through the manipulation of typography, symbolism, illustration, color, spatial organization, and temporal sequencing. [1] Certain professionals in this disipline are concerned primarily with the communication of complex information through the design of charts, maps, diagrams, and other technical documents. Knowledge from these professionals and their literature can be applied to the task of designing the graphic presentation of software documentation which now faces builders, users, and managers of computer systems. Graphic designers usually are not involved in setting up conventions, standards, and specifications for producing software documentation. In order to educate the information specialists and computer scientists who normally rely upon their own limited expertise, this article focuses on the typographic principles of information oriented book or document design drawn from the professional literature and from the author's own experience as a graphic designer of computer-based documents [1;2;4].

The software documentation interface between the human being and machine is in the context of the person using a computer system and in the person building or maintaining a computer system. Elsewhere the author has termed these the inter-faces and the inner-faces of computer systems. [2] Basic principles of selecting visual signs and their attributes (such as their location, size, and boldness) for presentation on both paper and glass can enhance the legibility of software documentation as well as its readability, i.e., its appeal or friendliness.

2. Typographic Aspects of Graphic Design

The design task concerns determining a relatively high degree of fit among the different requirements of the components of every communication interface:

the sender (the machine or user)
the medium (the display device)
the receiver (the user or machine)
the message (the information content).

By means of the position, color, size, grouping, and temporal sequence of visual signs such as alphanumerics and symbols, the graphic designer must convey the usual facets of a software documentation system: continuous prose (e.g., help messages and lengthy explanations), interrupted prose (e.g., error messages, system status reports, examples), and tables or lists (e.g., source code, menus, data dictionaries).

Typographic design begins with a concern for the design of individual symbols. In many current display systems there is relatively little control over symbol design. A limited hardware set of characters is often used to display alphanumerics and other symbols. Because many terminals and printers currently operate with fixed-width characters, many of the principles given below are oriented toward this situation.

In online display, there is often little control over symbol design; it is likely that the standard medium for interaction may be a display showing 24 lines of 80 alphanumeric characters each. The use of reverse video, italic, or levels of brightness can not always be assumed. Even if these means of visual emphasis are not used, other approaches are available. For example, there can be a strong reliance on a horizontal line of hyphens to highlight certain titles or to separate divisions of the frame.

Even within severe limitations, attention to graphic design principles can improving the effectivenes of software documentation. Consider the use of all upper case words, a typographic approach which much documentation utilizes. The fixed width of the letters are often created by a 7x9 or similar dot matrix. In such conditions lower case letters with occasional capitals are more legible. Research shows [3, 35] that not being able to perceive word shapes (as is true for words set in upper case characters only) may slow reading speed by as much as 13%. Because line printers and terminals often have little space between lines in comparison to normal textbook typography, lower case letters are particularly important in providing visual space between lines of type and thereby improve legibility. In interactive situations, lowercase typography for machine messages and for the echoes of user input should be used whenever possible. When all capital settings are used, they should be used to highlight a restricted set of primary content elements, e.g., the main title of a frame or the module in which a prompt occurs.

3. The Grid
As for the design of a traditional printed book page, the graphic designer of software documentation must consider the visual field, the terminal screen or the printed page, as an entity whose proportion, size, and distance from the viewer are important to the design of information. Information is presented in conceptual frames of pages or screens. To assist the overall organization of elements within the frame and consistency from frame to frame, a reference grid of a few horizontal and vertical lines should be determined to locate certain standard positions for elements such as titles, prompts, etc. One of the most important functions of the grid is to establish certain basic divisions of the frame. The grid should establish one or more major columns of text of approximately 60 characters in width.

For fixed-width character printers or terminals, one simple approach to frame design is to use two primary locations: a single major column lying between character positions 21 to 80 and a special position at character position 1 for all secondary matter, such as subtitles for explanatory text or user input for textually oriented command and control interfaces. For subtitling, the reader can easily scan the overall structure of the document; for interfaces, the user's input and the machine's responses are visually distinct. Primary tab settings of 10 characters each and a secondary set every 5 characters can help divide the entire visual field into regular, modular units. Selection of upper or lower case alphanumeric characters and a grid influence other aspects of the typographic design, viz., character spacing, word spacing, line length, justification, line spacing, and the overall spatial structure of the frame.

4. Words, Lines, and Paragraphs
In situations in which character width is constant and letterform design is quite simple, word spaces are relatively large and lines of text tend to fall apart into a loose collection of alphanumerics. Wherever possible the typographic design approach stresses the need to keep words that belong together close to each other in word, line, and paragraph groupings. For example, only one word space is sufficient after a period in continuous prose to separate the end of one sentence and the beginning of the next. The graphic design approach also seeks to emphasize clear spatial groupings over the entire visual field in order to make distinctions of content. At the same time these spatial groupings are limited in their variation so that there is an overall visual consistency or rhythm within and between frames.

A typical oversight in most textual displays is using text lines of too great a width. Normally there should be approximately 40-60 characters per text line (about 10-12 words) [3, 29]. Research has shown [3, 33] that unjustified (unequal length) text lines are just as legible as justified text. In the case of fixed width characters, justification usually means that large gaps of empty space appear between words in order to achieve equal width text lines. These large spaces interrupt eye movement and impede reading. Especially for interrupted text, typographic design calls for unjustified paragraphs. This design feature has the added effect of making character position 21 visually the most important in the frame. An implied vertical line of the beginnings of text lines appears at this position. This becomes the location for many key words, text line beginnings, etc. The reader quickly develops the habit of scanning this location for most beginnings of information.

In fixed character width, fixed interline spacing situations, the space between groups of lines has limited variation. Whenever possible one should avoid any spacing larger that a single line skip. This may be used between paragraphs, line clusters, individual sets of menu prompts, user responses, etc. In this way a maximum number of text lines per frame can be utilized. Note that the horizontal line made of hyphens can replace a skipped text line and does not add another line to the already limited number of lines in a frame.

5. Tables and Lists
A major design principle is to limit the amount of variation wherever possible. This applies especially to tabular settings for tables and lists. In the case of fixed character-width situations, the most important words or word groupings are placed at or near (i.e., before or after) the tab setting at the 21st character position. All tables and lists require headings to describe the contents in general and to identify the parts if there are many. These titles should not scroll off the screen or disappear from continued pages; they should be regenerated as needed so that each frame includes sufficient titles to be comprehendable. All horizontal positioning of tables and lists is governed by the desire to keep codes, page numbers or other symbol groups close to the items to which they refer and to allow easy scanning down and across items.

6. Examples
The principles outlined above are embodied in two sets of accompanying examples. One set involves redesigned formats for the low resolution online interface [4] to an information management network which accesses very large geographic databases [5]. The other set arises from prototype redesigns of textual programs for display on high resolution terminals or printers. A comparison between old and new versions will clarify how earlier designs for frames were faulty and inconsistent. Improvements in the newer versions should be obvious. The examples appear in the accompanying Figures.

7. Conclusions
Most of the changes in the documentation formats have been relatively easy to implement within the software. These redesign features are more than a 'cosmetic' facelift to the system. By carefully considering not only what to show, but also when, how, and why to show it, a better understanding of the functionality of the system emerges in the minds of the builders and ultimately in the minds of the users of the computer system.

Many of the changes in design constitute working conventions rather than carefully proven standards. However, in the case of the first set of examples, many of the changes corresponded to recommendations of an independent critique of the system [6, 54-55]. In the second set it is also clear from informal discussions with users and implementors of computer systems that changes brought about by consideration of typographic design principles have made clear improvements that programmers as well as users can readily perceive. As these design principles and specifications for new documentation standards are more completely determined, they can be embodied in a graphic design manual [7]. This manual could assist future builders of documentation modules to maintain a consistent, high quality inter-face or inner-face for the computer system.

Acknowledgements

This work was supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy, under contract W-7405-ENG-48. The author also wishes to acknowledge Dr. Ronald Baecker, President, Human Computing Resources, Toronto, with whom the author developed the textual program visualization and Mr. Richard Sniderman of Human Computing Resources, who helped write some of the typesetting macros for that visualization.

References

1. Marcus, Aaron, "Computer-Assisted Chart Making From the Graphic Designer's Perspective," *Computer Graphics*, 14:2, 1980, 247-253.

2. Marcus, Aaron, "Graphic Design and Computer Graphics", *Industrial Design*, March/April 1982, in press.

3. Rehe, Rolf F., *Typography: How to Make It Most Legible*, Design Research International, Carmel, Indiana, 1974.

4. Marcus, Aaron, "Designing the Face of An Interface," *IEEE Computer Graphics and Applications*, 2:1, January 1982, 23-26ff.

5. McCarthy, John, et. al., "The Seedis Project: A Summary Overview", Publication 424, Lawrence Berkeley Laboratory, September 1981.

6. Bleser, Terry, Peggy Chan, and Mei Chu, "A Critique of the SEEDIS User Interface," Report GWU-IIST-81-04, Department of Electrical Engineering and Computer Science, The George Washington University, Washington, D.C., March 1981.

7. Marcus, Aaron, "A Graphic Design Manual for Seedis", in preparation.

Figure 1a: Undesigned Command Menu Descriptions Frame

Within the Computer Science and Mathematics Department of Lawrence Berkeley Laboratory, the author (who has a professional background in graphic design) has begun to apply the principles of information-oriented typographic design to the redesign of the interface for a large experimental geographic information management system called Seedis [5]. The interface for Seedis has gone through several stages since its genesis as a series of stand-alone batch programs in 1972, particularly as it expanded its functional capabilities. The current version of Seedis operates in an interactive VAX/VMS environement with a textual (i.e., essentially alphanumeric) interface. Seedis permits a relatively computer-naive person to examine data dictionaries, extract data from databases, to aggregate or disaggregate data between different levels of detail, and to display the selected data as a labeled table, dot matrix chart, pie chart, line chart, bar chart, or area/symbol choropleth map. In the Figure, note the illegibility of all capitals in comparison to upper and lower case and the interrupted list of command definitions.

Figure 1b: Designed Command Menu Descriptions Frame

The command menu description frame appears when the user types a question mark at any decision point, i.e., if there is some confusion about the proper response to the immediately preceding prompt. Note the organized appearance of text groups, the order of text elements, the use of rules, lower case, and specific tab settings. The full screen width is equivalent to 80 typewritten characters in width. Information on global commands is introduced in the very first information to the user. The standard form of the menu-prompt identifies the module (all capital letters) in which the user is currently working and the appropriate commands at this point. Note the use of the standard tab settings at position 1 and 21 and the consistent use of standardized verbs to describe the input commands. Global commands are separated from local commands appropriate to the particular decision point. The list is labeled to aid identification of its component parts.

```
?
 TYPE ONE OF THE FOLLOWING COMMANDS...
 ?            FOR THIS LIST OF COMMANDS
 HELP         FOR HOW TO GET HELP
 MORE         TO SEE NEXT SCREENFULL
 TABLE        FOR THE TABLE OF CONTENTS
 <N>          FOR PAGE <N>
 * <COMMENT>  TO ENTER A COMMENT IN THE LOG
 DATA <SEQUENCE LETTERS>       SELECT DATA CODES
 CANCEL <SEQUENCE LETTERS>     CANCEL DATA CODES
 FOR X <C>    SUBSTITUTE C FOR X IN DATA CODES -
              ALSO XX XXX XXXX Y YY YYY YYYY
 REVIEW       LIST DATA SELECTIONS MADE SO FAR
 SAVE         SAVE DATA SELECTIONS AND RETURN
 QUIT         CANCEL DATA SELECTIONS AND RETURN
 READY
```

```
                    DATA: <line letter(s)>, table, <page number>, CR
: ?

Input               Description
------------------- -------------------------------------------------------------
<line letter(s)>    select one or more data elements by line letter
table               display table of contents for this database code
<page number>       display a particular page
CR                  (carriage return) display the next page

?                   list avalable commands in this menu
help                describe data element selection
show                display table of contents for this database
review              list current data element selections and history
cancel              delete current data element selections for this database
quit                return to database selection menu

                    DATA: <line letter(s)>, table, <page number>, CR
:
```

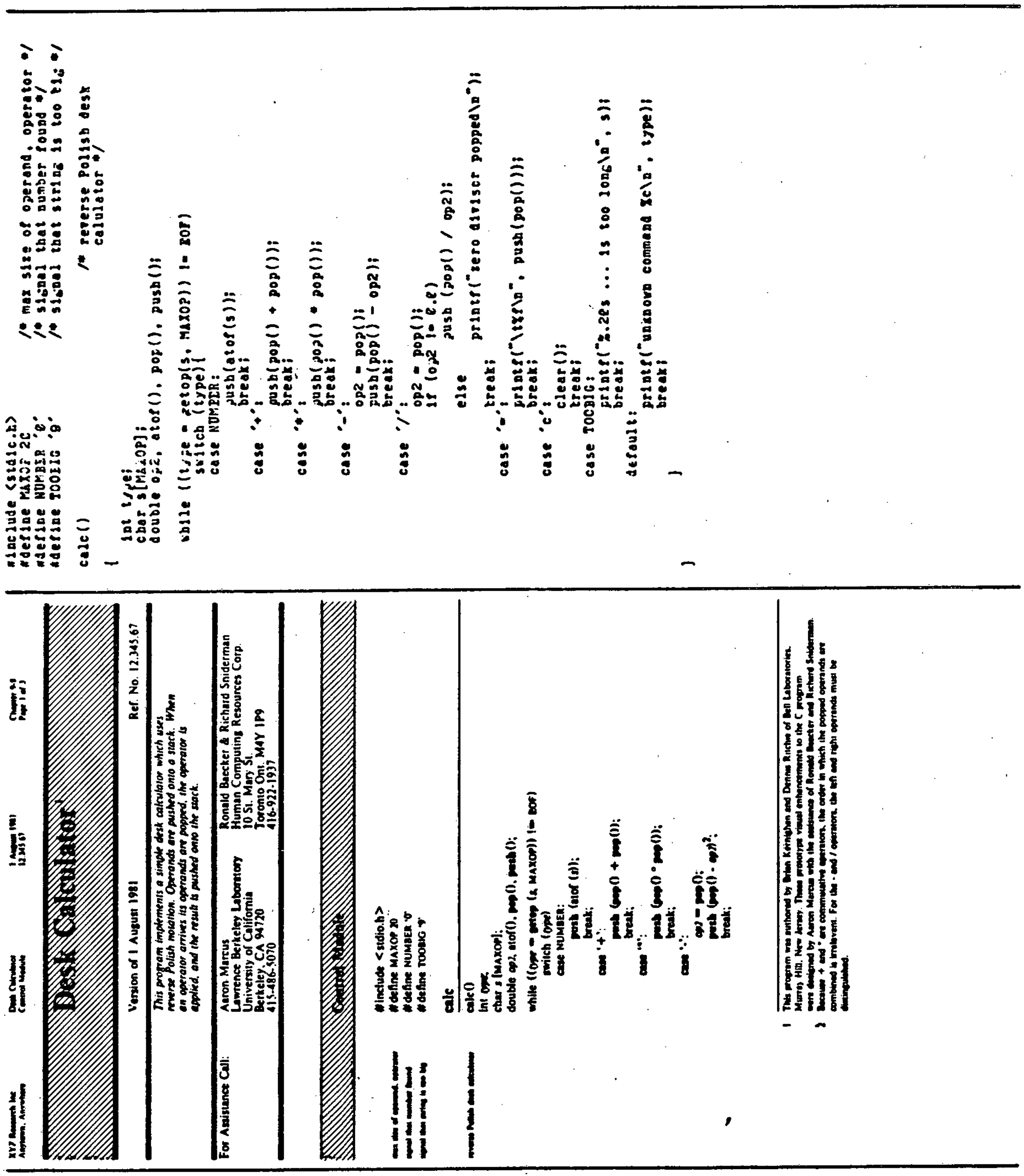
```
READY
  MONITOR.SEEDIS.HELP.

                         INTRODUCTION TO SEEDIS

 The three major processes in SEEDIS are:

 AREA: define a geographic study area (composed of states,
     counties, or census tracts)

 DATA: select data appropriate to the geographic study area
     chosen.  For  example, for a study area consisting of a group
     of states, only state level data, and  not  county  or  tract
     level data, are appropriate.

 DISPLAY: manipulate and display the data in table, chart, graph,
     and/or map form.

 Normally  AREA,  DATA,  and  DISPLAY  are  performed in the order
 given. However, once the geographic study area is defined (AREA),
 one  may  alternate  between  DISPLAY  and  the   selection   and
 extraction of additional items in DATA.
TYPE MORE TO SEE NEXT SCREENFULL
TYPE ? FOR A LIST OF COMMANDS
```

```
                    SEEDIS: area, data, display, profile
: help
                    USING SEEDIS
                    -----------------------------------------------------
                    LBL's Seedis is an experimental information system that
                    includes integrated program modules for retrieving, analy-
                    zing, and displaying selected portions of geographically
                    linked databases. Program modules in Seedis include:

          area      select geographic area (level and scope of analysis)
          data      select, extract, enter, or transform data
          display   manipulate and display data in tables, maps, and charts
          profile   produce standard socio-economic reports for selected areas

                    Normally Area, Data, and Display are used in the order
                    given. However, once the geographic study area is defined
                    in Area, you may alternate between Display and
                    the selection, extraction, or entering of additional items
                    in Data.

                    SEEDIS: area, data, display, profile
:
```

Figure 2a: Undesigned Help Messages Frame
Note the long lines of text, the clutter in the last paragraph caused by clumps of all capital words, the gaps in word spacing caused by justification, and the mixture of small indentations with centered headlines.

Figure 2b: Designed Help Messages Frame
Help messages are a standard one frame page description. Note the use of standard tab settings, unjustified text, the use of all capital headline together with hyphen line, removal of all capital keywords (replaced by exdented words, i.e., positional emphasis), and the use of second person in English language style. Further frames of information are available on the four key words listed.

Figure 3a: Undesigned Textual Program
This figure presents a typical C program in an elementary typographic form using fixed-width characters of a single size and typeface with limited horizontal spacing variation. There is little typographic hierachy. The program is more readable than those presentations that use all-capital typography and multiple commands per line, but there are still ways in which it can be made more readable.



XYZ Research Inc
Anytown, Anywhere

Desk Calculator
Control Module

1 August 1981
12.345.67

Chapter 9-1
Page 1 of 3

Desk Calculator[1]

Version of 1 August 1981 — Ref. No. 12.345.67

This program implements a simple desk calculator which uses reverse Polish notation. Operands are pushed onto a stack. When an operator arrives its operands are popped, the operator is applied, and the result is pushed onto the stack.

For Assistance Call:

Aaron Marcus
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720
415-486-5070

Ronald Baecker & Richard Sniderman
Human Computing Resources Corp.
10 St. Mary St.
Toronto Ont. M4Y 1P9
416-922-1937

Control Module

```
#include <stdio.h>
#define MAXOP 20
#define NUMBER '0'
#define TOOBIG '9'
```

calc

```
calc()
int type;
char s [MAXOP];
double op2, atof(), pop(), push();

while ((type = getop (s, MAXOP)) != EOF)
        switch (type)
        case NUMBER:
                push (atof (s));
                break;
        case '+':
                push (pop() + pop());
                break;
        case '*':
                push (pop() * pop());
                break;
        case '-':
                op2 = pop();
                push (pop() - op2)²;
                break;
```

1 This program was authored by Brian Kernighan and Dennis Ritchie of Bell Laboratories, Murray Hill, New Jersey. These prototype visual enhancements to the C program were designed by Aaron Marcus with the assistance of Ronald Baecker and Richard Sniderman.

2 Because + and * are commutative operators, the order in which the popped operands are combined is irrelevant. For the - and / operators, the left and right operands must be distinguished.

```
#include <stdio.h>
#define MAXOP 20            /* max size of operand, operator */
#define NUMBER '0'          /* signal that number found */
#define TOOBIG '9'          /* signal that string is too big */

calc()                             /* reverse Polish desk
                                      calulator */
{
   int type;
   char s[MAXOP];
   double op2, atof(), pop(), push();

   while ((type = getop(s, MAXOP)) != EOF)
           switch (type){
           case NUMBER:
                   push(atof(s));
                   break;
           case '+':
                   push(pop() + pop());
                   break;
           case '*':
                   push(pop() * pop());
                   break;
           case '-':
                   op2 = pop();
                   push(pop() - op2);
                   break;
           case '/':
                   op2 = pop();
                   if (op2 != 0.0)
                           push (pop() / op2);
                   else
                           printf("zero divisor popped\n");
                   break;
           case '=':
                   printf("\t%f\n", push(pop()));
                   break;
           case 'c':
                   clear();
                   break;
           case TOOBIG:
                   printf("%.20s ... is too long\n", s);
                   break;
           default:
                   printf("unknown command %c\n", type);
                   break;
           }
}
```

Figure 3b: Designed Textual Program
This figure shows a prototypical black-and-white visualization that would require a high resolution bit map display terminal or a very high resolution hardcopy device. The actual image was generated in Times Roman type using a computer-controlled phototypesetter, a rare but not unheard of hardcopy device. This image is one of a series of experimental prototype frames for offline or online documentation that illustrates the full potential of a graphic design aproach to textual program visualization. The image was designed by the author and Dr. Ronald Baecker with Mr. Richard Sniderman of Human Computing Resources Corporation. Spatial location, typographic symbol hierarchies, figure-field enhancements, indexes, abstracts, etc., are combined to create a clear, consistent, explicitly structured frame that is legible and appealing to the reader, based on a limited number of disussions with programmers who have viewed but not used this presentation.