May the God who watches over the right use of mathematical symbols in manuscript, print, and on the blackboard, forgive me this and my many other sins!

Hermann Weyl
The classical groups,
Princeton University Press, 1946,
p. 298n

TUGBOAT

THE TEX USERS GROUP NEWSLETTER
EDITOR BARBARA BEETON

VOLUME 5, NUMBER 2 • NOVEMBER, 1984
PROVIDENCE • RHODE ISLAND • U.S.A.

Addresses of Officers, Authors and Others

AZZARELLO, Arlene
I P Sharp Associates
220 California Ave
Suite 201
Palo Alto, CA 94306
415-327-1700

BECK, Lawrence A.
Grumman Data Systems
1111 Stewart Ave
MS B14-111
Bethpage, NY 11714
516-575-9838

BEETON, Barbara
American Mathematical Society
P.O. Box 6248
Providence, RI 02940
401-272-9500
bb@SU-AI

BERTELSEN, Erik
Regional EDP Center, Univ of Aarhus (RECAU)
Ny Munkegade
Bygning 540
DK-8000 Aarhus C, Denmark
45 6 128355; Telex: 64 754 recau dk

BLOCK, Neil
Hughes Aircraft Co.
Bldg. A1, M/S 3C923
P.O. Box 9339
Long Beach, CA 90810
213-513-4891

CANZII, G.
Università degli Studi di Milano
Istituto di Cibernetica
Via Viotti 5
20133 Milano, Italy
23.52.93

CARNES, Lance
163 Linden Lane
Mill Valley, CA 94941
415-388-8853

CHILDS, S. Bart
Dept of Computer Science
Texas A & M University
College Station, TX 77843
409-845-5470

CLARK, Malcolm W.
Imperial College Computer Centre
Exhibition Road
London SW7 2BX, England
01-589-5111, x1172

CODE, Maria
Data Processing Services
1371 Sydney Dr
Sunnyvale, CA 94087

CRAWFORD, John M.
Computing Services Center
College of Administrative Science
Ohio State University
Columbus, OH 43210
614-422-1741
CSNet: Crawford-J@Ohio-State
BITNet: TS0135@OHSTVMA

DÉSARMÉNIEN, Jacques
Département de mathématique
7, rue Rene-Descartes
67084 Strasbourg Cedex, France
33-88-614820

DUPREE, Chuck
Datapoint, Inc.
9725 Datapoint Drive, MS N-22
San Antonio, TX 78284
512-699-7200

EPPSTEIN, Maureen
Administrative Publication Manager
Stanford University
Encina Hall, Room 200
Stanford, CA 94305
415-497-9254
MVEppstein@SU-Score

FELIPPA, Carlos A.
Lockheed Palo Alto Res Lab
Org 52-33 B 255
3251 Hanover St
Palo Alto, CA 94304
415-858-4029

FUCHS, David
Department of Computer Science
Stanford University
Stanford, CA 94305
415-497-1646
DRF@SU-Score

FURUTA, Richard
Univ of Washington
Computer Science, FR-35
Seattle, WA 98195
206-543-7798
Furuta@Washington

GOUCHER, Raymond E.
TEX Users Group
P.O. Box 9506
Providence, RI 02940
401-272-9500 x232

GROPP, William
Dept of Computer Science
Yale University
Box 2158 Yale Station
New Haven, CT 06520
203-436-3761
Arpanet: Gropp@Yale

GUOAN, Gu
Computer Science Department
Stanford University
Stanford, CA 94305

HOBBY, John
Computer Science Department
Stanford University
Stanford, CA 94305

ION, Patrick D.
Mathematical Reviews
611 Church Street
P.O. Box 8604
Ann Arbor, MI 48107
313-763-6829

JÜRGENSEN, Helmut
Dept of Computer Science
Univ of Western Ontario
London N6A 5B7, Ontario, Canada
519-679-3039

KELLER, Arthur
Computer Science Dept
Stanford University
408C Margaret Jacks Hall
Stanford, CA 94305
415-497-3227
ARK@SU-AI

KELLY, Bill
Academic Computing Center
University of Wisconsin, Madison
1210 W. Dayton Street
Madison, WI 53706
608-262-9501

KNUTH, Donald E.
Department of Computer Science
Stanford University
Stanford, CA 94305
DEK@SU-AI

LUCARELLA, Dario
Istituto di Cibernetica
Università di Milano
Via Viotti 3/5
20133 Milano, Italy
23.52.93

MacKAY, Pierre A.
University of Washington
Department of Computer Science, FR-35
Seattle, WA 98195
206-543-2386
MacKay@Washington

MALLETT, Rick
Computing Services
Room 1208 Arts Tower
Carleton University
Ottawa (K1S 5B6), Ontario Can
613-231-7145

NICHOLS, Monte C.
Exploratory Chemistry Division
Sandia National Laboratories 8313
Livermore, CA 94550
415-422-2906

PALAIS, Richard S.
Department of Mathematics
Brandeis University
Waltham, MA 02154
617-647-2667

PIZER, Arnold
Department of Mathematics
University of Rochester
Rochester, NY 14627
716-275-4428

PLASS, Susan
Information Technology Services
Cypress Hall, Jordan Quadrangle
Stanford University
Stanford, CA 94305
415-497-3302

RODGERS, David
Textset, Inc
P O Box 7993
Ann Arbor, MI 48107
313-996-3566

ROKICKI, Tomas
Electrical Engineering
Texas A&M University
College Station, TX 77843
Rokicki@TAMU.CSNET

SCHULZE, Bernd
Inst für Angewandte Mathematik
University of Bonn
Wegelerstr 6
D-5300 Bonn, Fed Rep Germany
0228-733427

SMITH, Barry
Kellerman & Smith
2343 SE 45th Ave
Portland, OR 97215
503-232-4799

SOUTHALL, Richard
83 Eastern Avenue
Reading RG1 5SQ, UK
(0734)67267

SPIVAK, Michael
1660 West Alabama, #7
Houston, TX 77006

STERKEN, Jim
Textset, Inc
P O Box 7993
Ann Arbor, MI 48107
313-971-3628

STROMQUIST, Ralph
MACC
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706
608-262-8821

THEDFORD, Rilla
Intergraph Corporation
One Madison Industrial Park
Huntsville, AL 35807
205-772-2000

TOBIN, Georgia K. M.
Office of Research
OCLC Online Computer Library Center, Inc
6565 Frantz Rd
Dublin, OH 43017
614-764-6000

TUTTLE, Joey K.
I P Sharp Associates
220 California Avenue
Suite 201
Palo Alto, CA 94306
415-327-1700

WELLAND, Robert
Dept of Mathematics
Northwestern University
Lunt Hall
Evanston, IL 60201
312-492-3298

WHIDDEN, Samuel B.
American Mathematical Society
P.O. Box 6248
Providence, RI 02940
401-272-9500

ZABALA, Ignacio
Centro de Cálcolo
Facultades de Ciencias
Universidad de Valencia
Cametera de Ademuz
Valencia, Spain
011-34-6-357-4065

ZAPF, Hermann
Seitersweg 35
D-6100 Darmstadt, Fed Rep Germany

TUGboat, the newsletter of the TEX Users Group (TUG), is published irregularly by TUG, P. O. Box 9506, Providence, RI 02940. Annual dues for individual members of TUG include one subscription to TUGboat; fees are detailed on the membership form bound into the back of this issue. Applications for membership in TUG should be addressed to the TEX Users Group, P. O. Box 594, Providence, RI 02901; applications must be accompanied by payment.

Manuscripts should be submitted to a member of the TUGboat Editorial Committee, whose names and addresses are listed inside the front cover. Articles of general interest, or not covered by any of the topics listed, as well as all items submitted on magnetic tape, should be addressed to Barbara Beeton, American Mathematical Society, P. O. Box 6248, Providence, RI 02940.

Submissions to TUGboat are for the most part reproduced with minimal editing. Any questions regarding the content or accuracy of particular items should be directed to the authors.

TUG Membership Dues and Privileges

Memberships and Subscriptions

1985 dues for individual members are as follows:

North America:
- New (first-time) members or subscribers: $20.
- Membership and subscription renewals: $30, reduced rate of $20 for renewals received before January 31, 1985.

Outside North America (includes air mail postage):
- New (first-time) members or subscribers: $25.
- Membership and subscription renewals: $35, reduced rate of $20 for renewals received before January 31, 1985.

Membership privileges include all issues of TUGboat published during the membership (calendar) year. Anyone inquiring about TUG will be sent a complimentary copy of TUGboat Vol. 1 (1980), No. 1, along with a current copy of the membership list and forms for acquiring TEX82, joining TUG and ordering publications available from TUG.

Issues to domestic addresses are mailed third class bulk, which may take up to six weeks to reach their destinations. If you have not received an issue to which you are entitled, write to TUG at the address given below.

Institutional Membership

1985 Institutional Membership dues for educational organizations are $200; for non-educational, $300. Membership privileges include: designating up to 5 persons as individual members and special reduced rates for participation at TUG meetings and TEX-related courses and for purchase or lease of videotapes. In addition, institutional members are listed in each issue of TUGboat. For further information, call Ray Goucher at (401) 272-9500, ext. 232.

Submitting Items for Publication in TUGboat

The deadline for submitting items for Vol. 6 (1985), No. 1, *will be February* 1, 1985; *the mailing date will be March 14.* Contributions on magnetic tape or in camera copy form are encouraged; see "Submitting items to TUGboat", page 78, this issue. Editorial addresses are given on the inside front cover. For instructions on preparing magnetic tapes or for transferring items directly to the AMS computer, write or call Barbara Beeton at the address given, (401) 272-9500, ext. 299.

TUGboat Advertising and Mailing Lists

For information about advertising rates or the purchase of TUG mailing lists, write or call the TEX Users Group, Attention: Ray Goucher, P. O. Box 9506, Providence, RI 02940, (401) 272-9500, ext. 232.

General Delivery

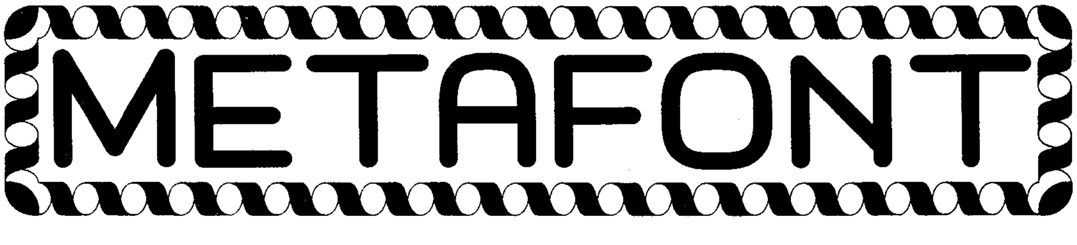
From the President

Pierre MacKay

This will be a short message. A great many questions arose during the meeting this summer and we will have to chew them over rather thoroughly before we can digest them. (The imagery of this remark is familiar to any reader of the TeXbook.) What made this summer's meeting particularly significant was the growing awareness that now that TeX has come of age, the TeX Users Group must grow with it. At earlier meetings, the question of when and even whether the various flavors of TeX could be adapted to various machines occupied a good part of our interest. The effort of implementation was the hottest topic on the program, and legitimately so. But in the ten months since the first distribution of TeX 1.0, the WEB system has so thoroughly proved itself that, in the case of larger systems, at least, here is little to discuss except that it works. Those who, like Lance Carnes, are trying to squeeze TeX onto ever smaller machines are far from finished with implementation problems, but the learning curve is going up steeply. METAFONT will require another burst of effort once it too comes of age, but most of the lessons learned from TeX will be directly relevant to the various implementations of METAFONT.

With that understanding, the TUG steering committee is looking very carefully at the restructuring of the annual meeting. We must now study how we can best serve a community which is able to assume that TeX is widely available and stable. In short, we must become even more a *Users* group than before. There is much to do. As Richard Southall forcefully pointed out this summer, layout and formatting are very demanding work, and require their own specific expertise. The LaTeX generic document formatter is almost complete, and offers one example of what we can look for in the future. AMS-TeX will provide a good deal of the special polish needed for complex mathematical typesetting. But we also need to take the kernel of TeX and PLAIN.TEX, and build designer's interfaces around it, packages which work the way layout designers work, and which allow the designer's mock-up to be interpreted directly and easily into TeX commands. We need to offer much more of this sort of thing at the annual meetings, and we probably need to offer it at more than one level.

If we are going to get it right, we need help. We learned a number of useful things from the members who attended the summer meeting, and we are going to put our knowledge to work in the coming year. But we would also like to learn more from those who were not there this summer. What can we do to make the summer meetings more attractive? What should we do more of, and what should we do less of, both at the meetings and in TUGboat? Where should the meetings be held? (Yes, we are considering this very seriously, although we feel that one more meeting at Stanford is appropriate in honor of the expected release of METAFONT.) How shall we set the balance between the needs of the relatively new user and the problems of the emerging macro wizard? Let us know, and we will do our best to make the TeX Users Group serve the Users even more effectively.



Ann Lasko-Harvill

Report of the Publications Committee

Robert Welland

A Publications Committee was formed at the summer TUG meeting. The temporary chairman of the committee is Robert Welland from the Mathematics Department at Northwestern University, and the remaining members are

Malcolm Clark	Imperial College, England
Maureen Eppstein	Stanford University, USA
Helmut Jürgensen	University of Western Ontario, Canada
Roberto Minio	Carnegie-Mellon, USA
Michael Urban	TRW Los Angeles, USA

The committee was asked to oversee publications which support the TUG community. It is to identify needs and encourage writers to fill them; and, when it is suitable, to arrange with Ray Goucher to have these works published by the AMS or some other suitable publisher.

Also, Ray was asked by the Steering Committee to set up a facility for distributing such materials. This will lead eventually to a central TUG distribution center.

As a start on our publication endeavours, Ray was asked to negotiate with Stanford University for the rights to publish the Stanford reports which pertain to TEX. He was also asked to lease from Stanford the Metafont course video tapes and make these tapes available to the TUG community at reasonable rates.

Michael Urban has agreed to update his fine TEX reference card and make it available to TUG.

Maureen Eppstein is going to act as a conduit for articles about TEX applications for publication in TUGboat. Her main goal will be the promotion of good design. She will collect material on format and layout and present to TUGboat some exemplary layouts together with the TEX code for their generation.

Helmut Jürgensen is going to edit a column in TUGboat devoted to TEX and Metafont software with emphasis also on WEBware.

Roberto Minio has agreed to build an extensive bibliography of TEX-related materials and of format and design materials.

Malcolm Clark suggested that we collect a list of books (and papers) which have been published using TEX so that we will have a ready set of TEX-produced examples to show other people. Please send any information on this matter to Malcolm so that he can compile it for TUGboat.

It was suggested that a TEX command structure dictionary be built (a task more easily conceived of than executed). Alan Spragens at Stanford's SLAC Computing Services Center has made a step in this direction and has offered to let us have the material he has developed. If someone in TUG feels that he or she has the language skills necessary to carry out this extremely challenging job and would like to do it, please get in touch with Bob Welland.

It was decided that we should increase to three or four the number of issues of TUGboat produced annually. The final decision on this matter is to be left to Barbara Beeton, the TUGboat editor.

It is easy to form a committee and start some projects, but to produce results this committee needs the help of the beneficiaries. If you have any suggestions or can help in any way, please give a hand.

Report of the Special Projects Committee

Arthur Keller

The Special Projects Committee was formed at the TEX Users Group meeting in August with the charter of organizing courses (at the TUG meeting and otherwise), and evaluating other activities for TUG. Arthur Keller, Monte Nichols, and Craig Platt have agreed to serve on this committee. The 1985 TUG meeting will be a week-long course followed by a two-day short course and the three-day meeting during the weeks of August 5–16, 1985. We have decided to offer the week-long Intensive Introduction to TEX again next year, and we are making arrangements to obtain the same facilities, as they worked well this year. The Annual Meeting Program Committee will handle the schedule for the three-day meeting. Suggestions should be sent to Arlene Azzarello or Joey Tuttle.

We are considering several alternatives for a two-day short course for the 85 meeting. We would appreciate feedback indicating your preferences or additional suggestions on courses and people to teach them. Alternatives include: (1) writing LATEX document styles/LATEX internals; (2) issues in document design; (3) a course on publishing by a publisher; (4) how to edit your own copy; (5) an introduction to the new Metafont; and (6) a course on TEX macro wizardry.

We are also considering methods of cooperating with the conference on TEX for Scientific Documentation in Varenna, Italy (see page 147 for more information on this conference).

Comments may be sent to Arthur Keller. (See the inside front cover for the US mail address, telephone number, and computer mail address.)

Submitting Items to TUGboat

Barbara Beeton

TUGboat, along with TUG, has been in existence now for five years, and shows distinct signs of continued survival. The editorial board is growing, with volunteers in charge of editing specialized columns on various subjects; their names and addresses are listed on the inside front cover.

Articles on all subjects related to TEX and TUG are welcome. If an article falls into an area covered by one of the editors, it should be submitted to the appropriate editor. Articles of general interest, or in areas not listed, should be sent to me. If you feel that some important topic has been neglected, and would like to become a columnist, please get in touch with me. TUGboat will be only as healthy as you make it.

Present plans call for three issues in 1985; this assumes that all 1985 issues will be put together in Providence. The deadline for receipt of material in Providence for the first issue is February 1; articles which are submitted through a subject editor should be sent earlier — Helmut Jürgensen (page 91) has asked that submissions reach him at least two weeks earlier, and the other editors would surely appreciate the same consideration.

It was suggested at the August meeting that one or more issues might be guest-edited, with only the printing and mailing done by TUG. Such an arrangement might result in an extra issue in 1985, or might limit the content of a particular issue to a particular subject. More concrete information will be published when known. Suggestions for areas of special interest for "topical" issues, and volunteers to guest-edit such issues, are particularly welcome.

Items are solicited in camera copy form, whenever possible. If copy has been prepared by TEX and is legible, it will ordinarily be used as submitted, reduced photographically if necessary (advisable for copy prepared on an output device with 200 dot/inch or lower resolution), with running heads applied. The following specs were followed in preparing this issue of TUGboat:

text style	amr10 on 12pt
two-column pages:	
`\hsize`, `\vsize`	18.75pc, 52pc
intercolumn gap	1.5pc
page width	39pc
one-column pages:	
announcements	page 75
`\hsize`	34pc
Southall article	page 79
`\hsize`, `\vsize`	24.5pc, 54pc
text style	amss10 on 12pt

Items are also welcome on magnetic tape (see below), or as net mail to me on the ARPAnet (`bb@SU-AI`), or deposited directly onto the Math. Society computer. (Call me for details of how to log in.) We are able to handle 800- and 1600-bpi tapes in either ASCII or EBCDIC. Some manual translation may be necessary with EBCDIC, so include a list of the special characters, with each character followed by its name, to assist in deciphering. Tapes in the following formats have been handled successfully; please specify clearly the format and density:

(a) labeled ANSI-standard format;
(b) blocked, fixed-length records (80 characters is convenient)
(c) the generic distribution format generated by `TPREAD`.

Significant spaces and `\␣` at the ends of lines are discouraged; such spaces tend to get lost and must be re-inserted by hand, especially if a file is transmitted on tape with fixed-length records.

A set of TEX82 macros for preparing TUGboat material is now under construction. I am attempting to follow the philosophies of LATEX and AMS-TEX, but neither macro package will be required to use the TUGboat macros, only `PLAIN`. As soon as it is ready for general use (almost certainly before the end of the year), a copy will be put on the distribution tape, and a message will go out over the TEXhax network; a suitable announcement will also appear in TUGboat.

First principles of typographic design for document production

Richard Southall

1 Introduction

Leslie Lamport and I taught a two-day course on 'First principles of typographic design for document production' as a preliminary to the TUG meeting at Stanford University, August 13–14, 1984.

What follows is an expansion, somewhat revised and restructured, of my lecture notes.

Objectives

I did not feel that it was possible in two days to teach anything useful about typographic design on a 'how to' basis (even if the problems of interpreting conventional typographic design practice in terms of TEX macros had been solved). Nor would it have been helpful to give preceptive solutions to a limited set of design problems. On the old analogy, I did not wish to give away fish to the course members, and I did not have time to teach them how to catch fish for themselves. In such circumstances, the best thing to do is to try to outline some productive ways of thinking about water and fishing-tackle, so that basic errors of approach to the problems of fish-catching can be avoided in the future. Thus I tried to work out, and justify from first principles, a useful way of thinking about the problems of document design; and to illustrate this way of thinking by looking at the ways in which typographic designers working in earlier technologies had tackled similar kinds of problem.

Subject area

The subject matter of my part of the course was written language.

Written language has two levels of structure, which may be called microstructure and macrostructure (Waller, 1980a). The microstructure of written language has to do with details of the sequence and arrangement of the characters that make up a written text. It is governed by the rules contained in a 'system of writing': the set of rules for writing a particular language with a particular script in a particular technological environment. (I owe this useful concept to John Mountford.) For most systems of writing, and certainly for those in which the written text is produced by a mechanical writing-system, these rules are reasonably well defined, and descriptions of them are readily accessible (Chaundy et al., 1957; 'Chicago manual of style', 1982; Dowding, 1966; Swanson, 1979; Walker, 1979); though they differ to a surprising extent between systems of writing, even in the same technological environment (Desarmenien, 1984; Walker, 1983).

The macrostructure of written language has to do with its division into semantic objects (which we can usually recognize, even if we can't make very exact definitions of them — things like chapters, subsections, paragraphs and list items) and their graphic embodiment in a document. The rules governing the macrostructure of written language are much less well defined than those governing its microstructure. Many typographic designers would deny that there were any such rules at all (though Twyman, 1981, formulates 'what many typographers would consider one of the few fundamental "rules" of typography'), because there are so many equally effective ways of designing a document. But if there are no rules (or almost none) that designers would agree on, there are some principles for effective design that they (or most of them) would agree with; and it was these principles that were the subject area of my part of the course.

Our discussions in the course were based on two postulates:

> Documents have a conceptual structure
> Graphic structures can be made that reflect conceptual structures

one axiom:

> The graphic structure of a document should reflect its conceptual structure

and one desideratum:

> The graphic structure of a document should be such that the document is as easy as possible to use.

The problem of document design for computer-based systems

In the past, the design of documents was done *a posteriori*. Books were written before they were designed: the conceptual structure of the author's thoughts was already in place, embodied in some graphic form or other, and the designer's task was to render or re-render that embodiment in a semantically effective and technically practicable way. (See Hewson, 1983, for a detailed study of the evolution of the graphic embodiment of Wittgenstein's 'Tractatus Logico-Philosophicus' through successive renderings — in manuscript, typescript and printed form — of the work.)

In the design of documents for computer-based document production systems, the problem is the other way round. The document has to be designed *a priori*, before the author's thoughts are present at all. The document designer's task is to devise efficient graphic embodiments for conceptual structures that are suitable to fit any thoughts that any author using the system might have.

The document designer cannot tell (let alone dictate) what content an author will put into each of the conceptual structures that the document design provides for, or in what order the structures will be used. It is not too hard to provide a graphic embodiment for each structure, that will behave reasonably if it is not used in what its designer would consider to be an unreasonable way; but what is unreasonable to a document designer may not be at all so to a mathematician or a philosopher.

2 Typographic structures

Models of text and models of documents

There is a simple 'bottom-up' model of written text, that considers it as a sequence of characters and spaces.

Sequences of characters make up	words
words	sentences
sentences	paragraphs
paragraphs	sections
sections	chapters
chapters	books (articles, reports, documents)

This model does not work very well, either as a way of representing real text or real documents. At the lowest level, it ignores things like changes in character style. At a slightly higher level, it doesn't deal at all tidily with quite important questions, like 'What is a sentence?' (or, indeed, 'What is a word?'). At a higher level still, it ignores many of the important things that aren't chapters (in particular, lists and tables) that go to make up documents.

As a means of understanding the nature of written text, this model is good for analysing the functions of non-alphabetic characters in words and straightforward sentences; but otherwise it is too simple to be useful with present-day technologies of mechanical writing. It was well suited to the train-printer and daisy-wheel era of document production.

A 'top-down' approach to the modelling of documents begins by looking for recognizable graphic objects, and the semantic objects of which they are embodiments, in a document.

This approach has been pioneered in the British Library research project on the graphic translatability of text, with work by P.E.Norrish in the Department of Typography & Graphic Communication at the University of Reading. This work has centred on the study of public information documents, which are usually much more complex in both graphic and semantic terms than the technical reports that designers for computer-based systems are most often concerned with. Norrish and her collaborators have made very detailed analyses of such documents, looking at the 'access structures' of headings and paragraph markers that allow users to find their way around a document, as well as the objects that occur in it.

This work is relevant to the problems of designing computer-produced documents in two ways. First, it points up the considerable complexity, in semantic as well as graphic terms, of real documents. It is all too easy to think that chapters, sections and subsections are the only kinds of object that need to be considered in designing the text of a document. Second, it draws attention to the large number of alternative graphic forms in which a particular semantic object can be embodied. Norrish has so far identified no fewer than forty-seven different types of 'graphic list' (semantic list structures in which the list items are distinguished by graphic means).

Section structures

However inadequate the chapter/section/subsection model may be as a way of representing the structure of real documents, it is very often used to construct the body matter of technical reports. If we look at some of the semantic properties of constructions of this kind, we can get an idea of the properties their graphic embodiments ought to have.

Three kinds of relationship exist between the elements of a chapter/section/subsection construction. There are relationships of hierarchy: chapters are higher-level elements than sections, which are higher-level elements than subsections, and so on. There are relationships of containment: higher-level elements contain lower-level elements. And there are relationships of sequence: objects at each level follow each other in the construction.

At all the levels of such a construction, the logical structure of the elements is very similar. Each chapter, section, subsection and so on contains a heading and a body: the heading relates to the body that follows it. The body of the element may contain paragraphs of text, or lower-level elements, or text followed by lower-level elements, down to the lowest-level element, whose body (by definition) is all text. Thus there may be 'descending' sequences of headings at the beginning of an element.

By our axiom, the graphic embodiment of a chapter/section/subsection construction should reflect its semantic structure. Since the visual appearance of the paragraphs of text within an element does not usually alter with the position of the element in the hierarchy (except perhaps at the very lowest levels), it is the task of the heading of each element to show whereabouts in the structure its element is located.

Thus, the hierarchical relationships between elements should be clearly expressed in terms of the graphic relationships of their headings; as should their relationships of containment and sequence. The means that are available to designers for realizing the graphic expression of such relationships are discussed next.

3 The typographer's tools

Typeface terminology

The terms 'typeface' and 'font', which in earlier technologies had separate and clearly-defined meanings, are now used more or less interchangeably in discussions of computer-based document production systems, to refer to fundamentally different entities. Great confusion results. The following definitions are proposed, as an aid to clear thinking:

Typeface: a set of distinctive, visually related shapes for some or all of the characters of a script, intended for mechanical reproduction

Style: a distinguishing visual characteristic of a typeface

Family of typefaces: a set of visually related typefaces with differing styles

Font: a set of renderings of some or all of the character shapes of a typeface, intended for use in a restricted range of output image sizes in a particular reproducing system

What happens in text?

Written language contains elements which are not alphabetic or numeric characters: punctuation signs, and space. It also has features which seem to operate at a higher level than the words of the text: capitalization, changes in type style and size, and the presence of vertical and horizontal space in varying amounts. These elements and features evidently have some part to play in written language, and some (but surprisingly few) attempts have been made to identify their functions (Mountford, 1980; Walker, 1979). The following list of 'roles fulfilled by graphic and spatial features in the articulation of verbal graphic language' (whose incompleteness is acknowledged by its author) is taken from Walker's paper:

1 Differentiation

  1.1 Emphasis
  1.2 Distinction/particularization
  1.3 Quotation
  1.4 Interpolation

2 Abbreviation

3 Introduction

4 Omission

5 Separation and connection

6 Presentation of numbers

The most significant of these functions seems to be that of differentiation/emphasis.

Graphic conventions and graphic capability

Any mechanical writing-system makes available to its users a certain graphic capability. This can be expressed in terms of the number of characters, typefaces and type sizes, and the facilities for defining amounts of horizontal and vertical space, that the system offers (Southall, 1982). Until very recently, the graphic capability of the writing-systems that were available to most computer users was extremely limited: a single 96-character font of one single-width typeface, with a single increment of horizontal space the same as the characters' width, and a single larger increment of vertical space.

Except for the lack of a 'half-line' increment of vertical space, this capability is roughly the same as that of an ordinary typewriter. Thus when computer-based document production systems came to be designed, there were a number of ready-made graphic conventions for expressing semantic function already available in the rules for the layout of typewritten documents. Almost all of these conventions could be taken over unchanged into the layout of computer-produced documents, and there was no need for system designers to think explicitly about the semantic functions that were being expressed by particular graphic configurations.

Laser page-printers, and the first versions of TEX and Metafont, made writing-systems with enormously increased graphic capability available to part of the computing community. The designers of document production systems did not always understand that graphic conventions derived from typewriting practice were not necessarily appropriate for systems with a graphic capability as good as, or better than, that of conventional typesetting.

The typographer's task

Looking at written language as being made up of semantically functional graphic elements as well as the words and phrases of the text, and bearing our axiom in mind, it becomes easy to define the typographic designer's task. This is to devise effective graphic means of expressing the different semantic functions that are required to embody in a document the conceptual structure of its author's work.

The way we are accustomed to reading continuous text leads us to recognize a paragraph as an area of text within which the space between successive rows of characters does not vary. Because of this, the graphic means that are available to the designer for expressing the functions of emphasis and differentiation within a paragraph are limited to those that do not change the space between rows of characters: capitalization, and changes in type style without changes in type size.

Outside the limitations of the paragraph, additional graphic means of expressing semantic function are available. The most powerful and flexible of these, and the least well understood by untrained designers, is the use of space. Others are changes of type style, and changes of type size.

4 Making text readable

Legibility research is an old subject (Tinker, 1963) and — as far as investigations of the typographic requirements for readable text are concerned — more or less a dead one (Spencer, 1969; Zachrisson, 1965). Current research in reading focusses much more on the perceptual and mental processes involved (Pirozzolo and Wittrock, 1981; Tzeng and Singer, 1981). Perhaps it is what computer scientists see as the excessive antiquity of the research on the layout of readable text that has led it to be so consistently ignored by the designers of computer-based document production systems. Its findings, though, are none the less valid for being old, or for being ignored, and they are as follows:

Lines of text should not be longer than 10–12 words (60–72 characters)

The appearing space between words in a line of text should be substantially less than the appearing space between successive lines of the text

The appearing space between words should not vary appreciably from line to line of the text.

These have been the canons of good typographic design for text for a very long time (Morison, 1951).

5 Designing headings

The functions of headings were discussed in Section 2. Headings mark the elements they belong to, and they show where those elements belong in the logical structure of the document as a whole. The latter function almost always means showing two things about an element: its place in a hierarchy and its place in a sequence.

The graphic hierarchy of the headings in a document should express the conceptual hierarchy of the elements the headings belong to. Thus the headings of higher-order elements should be graphically more prominent than the headings of lower-order elements. The connection between a heading and its element should always be explicit, so that in a sequence of 'descending' headings it is clear which element a particular heading belongs to.

In principle, the designer has three means for expressing hierarchy and connection in graphic terms: changes in type size, changes in type style, and the use of space. Where sequence is expressed explicitly, it is always by some sort of numbering system (Waller, 1980b).

In practice, the availability of graphic means to the designer of a document depends on the graphic capability of the writing-system the design is being made for. In a system with limited capability, the graphic means for expressing hierarchy may be exhausted before the bottom of the hierarchy has been reached. In such a case, the designer may choose to use a more elaborate numbering system for the headings in the document, so that the numbering expresses hierarchy as well as sequence.

This sort of circumstance, in which the content (in terms of characters and spaces) of a text is determined by the graphic capability of the system with which the text is rendered, is an instance of a major class of problems that have to be faced and solved by a realistic methodology of generalized document design.

6 Laying out the page

Margins

While our mechanisms of reading have not changed in the last few years, so that the rules for the layout of readable text can be carried over unaltered from traditional practice, the same is not necessarily true of the way we use documents.

The classical canons for the size of margins that are summarized by Tschichold (1965) apply to reading situations which were usually very different from the ones in which technical reports are used. Classically, books are codices: they open to a pair of facing pages. Books intended for continuous reading at normal reading distances are of such a size that this pair of pages forms a visual unit for the reader. In these circumstances, a symmetrical layout for the pages makes sense.

The letter-sized page which is by far the most common in technical reports is much the same size and format as the quarto page of traditional book design. Quarto formats were largely used in four kinds of book: reference books; bibles, in which the main text was broken up into verses, and the page often crowded with cross-references and notes; service books, for use in bad light by people who were very familiar with the text; and editions de luxe, where the width of the margins was often an index of the cost of the book (and hence the wealth of its owner).

Reference books and quarto bibles are meant for consultation rather than continuous reading. Service books are intended to be read at distances that are longer than normal. Editions de luxe are for admiration rather than use. There is no particular justification for taking over the convention of symmetrically laid-out pairs of pages from such books into the design of technical reports, whose letter-size pages are seen at normal reading distance (so that the visual unit is a single page) and are as often as not photocopied and put in a ring binder (which it is stretching traditional terminology rather too far to describe as a codex).

Line length

The use of the letter-sized sheet in laser printers derives from its use in the office, where conventional margins give a line length of six and a half inches or so: with the character pitch and line spacing of ordinary typewriters, a line of 65–75 characters and an easily readable page. The same margins, with the 10 point type that is the default size in many document preparation systems, give a line of 95–115 characters: far too many to be read effectively, even if the space between the lines of text is increased (Tinker, 1963). It is true that making a very long line means that many more characters fit on the page, and fewer pages are needed for a document of a given length. The consequent reductions in cost (which, as Leslie Lamport pointed out during the course, may be more apparent than real) have to be traded off against the reduced effectiveness of the document as a means of communication.

In fact, the most efficient way of filling a page with characters is to use double-column setting. Much narrower margins are tolerable in double-column than in single-column setting, so that more of the page area can be covered with type. However, the problems of designing headings, and of integrating the non-text elements of the document into the design of the page, tend to become more difficult.

Other components of the page

Text is not the only thing that appears on the pages of technical reports. There are footnotes, tables, figures and figure captions, as well as page numbers and running headlines or footlines. All these elements need to be distinguishable in visual terms: figure captions should not disguise themselves as footnotes, or headlines as part of the text. The designer needs to pay attention to unusual circumstances — What happens if a figure falls at the foot of a page? if a table whose column headings use the same style and size of type as the headline comes at the head of a page? — rather than laying out the page as if it was only to contain continuous text.

7 Conclusions

There is nothing in the foregoing that is not perfectly obvious as soon as it is pointed out (except perhaps the requirements for readable text, and they are in easily accessible research literature). Equally, I had no difficulty, with little more than a couple of numbers of TUGboat at hand, in finding many pages whose layout made them virtually impossible to read, and many instances in which the conceptual structures of documents were being concealed, rather than revealed, by their graphic embodiment.

Why is this? Why is the average computer-produced document so hard to use (and such a miserable object, in conventional graphic-design terms)? The reasons lie in a failure of communication between computer scientists and graphic designers.

This failure has two main causes. The first is that the designers of computer-based document production systems most often have no conceptual apparatus with which to think about graphic design problems. The thinking of traditionally-educated graphic designers tends to be unarticulated, visual, and concrete, and their vocabulary oriented towards particular graphic technologies; computer scientists' thinking tends to be algorithmic, symbolic, and abstract, and their vocabulary is that of mathematics. Ways of thinking about document design that computer scientists might find congenial are only beginning to be developed in the graphic design community, as a response to the problems posed by new information-handling technologies.

The second cause of failure is at the interface between graphic designers and computer-based document production systems, and has to do with the nature of the interface itself. Because graphic designers think visually, they need visual objects to think with; if the first task a designer has is to analyse the conceptual structure of a document, the second is to sketch out possible graphic embodiments for it. A skilled designer can get a long way by making drawings, but sooner or later a design has to be tested by being produced: and it is at this point, where the specifications on the drawing have to be translated into instructions to the document production system, that communication fails. The primitive concepts that are handled by computer text formatting languages, the terms the languages use to describe them, and the commands with which they are manipulated, are all completely alien to the experience of traditionally-educated designers.

The result, in the present state of affairs, is that traditionally-educated graphic designers *cannot use* computer-based document production systems, however much they would like to do so. The most urgent priority, for TEX and systems like it, is for interfaces to be developed so that designers can bring their skills to bear in an area where — to say the least — they are badly needed.

References

Chaundy, T.W., Barrett, P.R. and Batey, C.
The printing of mathematics
London: Oxford University Press, 1957

The Chicago manual of style (13th ed)
Chicago: University of Chicago Press, 1982

Desarmenien, J.R.
How to run TEX in a French environment: hyphenation, fonts, typography (Computer science report STAN-CS-1013)
Stanford, California: Department of Computer Science, Stanford University (1984)

Dowding, G.
Finer points in the spacing and arrangement of type
London: Wace, 1966

Hewson, R.L.
The visual presentation of a philosophical text: Wittgenstein's Tractatus Logico-Philosophicus
Unpublished dissertation: Department of Typography & Graphic Communication, University of Reading, England (1983)

Morison, S.
First principles of typography
Cambridge: Cambridge University Press, 1951

Mountford, J.
Writing-system as a concept in linguistics
Information design journal, vol.1 no.4, pp.223–231 (1980)

Pirozzolo, F.J. and Wittrock, M.C. (eds)
Neuropsychological and cognitive processes in reading
New York: Academic Press, 1981

Southall, R.
The problems of forms artwork production
London: Management & Personnel Office, HM Government, 1982

Spencer, H.
The visible word
New York: Hastings House, 1969 (2nd ed)

Swanson, E.
Mathematics into type
Providence, R.I.: American Mathematical Society, 1979

Tinker, M.A.
Legibility of print
Ames, Iowa: Iowa State University Press, 1963

Tschichold, J.
'Non-arbitrary proportions of page and type area'
in Osley, A.S. (ed)
Calligraphy and palaeography: essays presented to Alfred Fairbank
London: Faber & Faber, 1965

Twyman, M.L.
Typography without words
Visible language, vol.15 no.1, pp.5–12 (1981)

Tzeng, O.J.L. and Singer, H. (eds)
Perception of print: reading research in experimental psychology
Hillsdale, N.J.: Lawrence Erlbaum, 1981

Walker, S.F.
The presentation of information about house style: a reconsideration (IET Text-processing paper 1)
Institute of Educational Technology, The Open University, Milton Keynes, England (1979)

———
Unpublished thesis: Department of Typography & Graphic Communication, University of Reading, England (1983)

Waller, R.H.W.
'Graphic aspects of complex texts: typography as macro-punctuation'
in Kolers, P.A., Wrolstad, M.E. and Bouma, H. (eds)
Processing of visible language, vol.2
New York: Plenum, 1980

———
'Notes on transforming no.4: numbering systems in text'
in Hartley, J. (ed)
The psychology of written communication: selected readings
New York: Nichols, 1980

Zachrisson, B.
Studies in the legibility of printed text
Stockholm: Almqvist & Wiksell, 1965

Bibliography

The publications cited here are chosen, as far as possible, to be good pointers into the design research literature as well as being interesting in themselves.

Green, T.R.G. and Payne, S.J.
The woolly jumper: typographic problems of concurrency in information display
Visible language, vol.16 no.4, pp.391–403 (1982)

Hartley, J.
Designing instructional text
New York: Nichols, 1978

——— (ed)
The psychology of written communication: selected readings
New York: Nichols, 1980

———
Eighty ways of improving instructional text
IEEE transactions on professional communication, vol.PC-24 no.1, pp.17–27 (1981)

Jones, J.C.
Design methods: seeds of human futures
London: Wiley-Interscience, 1970

Macdonald-Ross, M. and Smith, E.B.
Graphics in text: a bibliography
Institute of Educational Technology, The Open University, Milton Keynes, England (1977)

Reynolds, L.
Legibility studies
Journal of documentation, vol.35 no.4, pp.307–340 (1979)

Spencer, H.
Pioneers of modern typography
London: Lund Humphries, 1969

———, Reynolds, L. and Coe, B.
Spatial and typographic coding in printed bibliographical materials
Journal of documentation, vol.31 no.1, pp.59–70 (1975)

Tschichold, J.
Asymmetric typography
New York: Reinhold, 1967

Twyman, M.L.
'A schema for the study of graphic language'
in Kolers, P.A., Wrolstad, M.E. and Bouma, H. (eds)
Processing of visible language, vol.1
New York: Plenum, 1979

Waller, R.H.W.
Journal typography in transition
Journal of research communication studies, vol.3 no.4, pp.335–349 (1982)

Wright, P.
'Usability: the criterion for written information'
in Kolers, P.A., Wrolstad, M.E. and Bouma, H. (eds)
Processing of visible language, vol.2
New York: Plenum, 1980

———
"The instructions clearly state . . . . " Can't people read?
Applied ergonomics, vol.12 no.3, pp.131–141 (1981)

Software

Editor's Introduction

Helmut Jürgensen

The Software Column of this TUGboat issue is a call for help—your contributions or suggestions!

Some of the things we would like to receive are outlined below. If you consider submitting material which you feel would fit into the Software Column but which is not covered by the list, please submit it anyway—or contact me.

TEX has come of age now—everybody said so at the TUG meeting. The main interest of TUG members seems to shift away from "mere" implementation questions gradually, and towards "real" typesetting problems. Nevertheless, it also seems that there are still quite a few difficulties arising when embedding the TEX and METAFONT systems into different environments in a user-friendly fashion. A lot of software has been and is being written to handle the problems. In fact, judging from some discussions I had at the TUG '84 meeting, it is written again and again and again. Therefore, I think the TUGboat and this column could help in distributing information and discussing ideas.

Please send descriptions of TEX and METAFONT-related software: They could be short, giving only name, purpose, availability, and—enough—important technical data. They could be long as well, if of sufficiently general interest. In some cases novel software methods may require a more thorough treatment. WEB-related problems would form another area of interest.

In general, information about software used in the implementation of the TEX, METAFONT, and WEB systems, in the adaptation to your environment or your research work, about enhancements to these systems, or software engineering comments related to these systems are welcome.

I should also appreciate receiving suggestions about the column in general, comments on articles, hints about related publications elsewhere, etc.

Please send your contributions and ideas to
Helmut Jurgensen
Department of Computer Science
The University of Western Ontario
London, Ontario
Canada N6A 5B7
You can also reach me by phone (519-679-3039) or via usenet (deepthot!uwo!uwo-hobbit!jurgensen).

Please remember, I need your contributions—otherwise the column will starve—and I must have them no later than two weeks before the general TUGboat deadline.

TEXtensions

How to Run TEX in a French Environment: Hyphenation, Fonts, Typography

Jacques Désarménien

1. Introduction

One of the greatest achievements of text-processing programs—especially TEX—is that they allow anybody to produce beautiful, typeset-like documents by merely typing a correctly prepared manuscript on a terminal, using a keyboard very similar to that of an ordinary typewriter.

As a consequence, before trying to train professional typists in TEX, one must ensure that a replica of the keyboard they are used to will appear on the terminal. In our situation, this means that some extra characters—é, è, ç among others—have to be included where they usually appear on a French keyboard. Some others must be displaced: for example, a and q have to be permuted and the digits retain their positions but must be typed by using the SHIFT key. The solution to these specific problems, and a certain number of improvements to the input of TEX, have been given by D. Foata and Y. Roy in Strasbourg [fr].

The main concern in processing French text is hyphenation. The set of patterns that TEX uses to hyphenate English works very poorly in French. The procedures are fundamentally different, not only because the etymology has much more importance in English, but also because the syllabifications do not obey the same rules in both languages. The same word might not be hyphenated the same way in French and in English: `ma-gni-fi-cence` *vs.* `mag-nif-i-cence`. We explain in another article

[d] how we have produced a set of patterns suitable for French. In section 3 below, we give a short account of this process, and we discuss the technical aspects of loading these patterns into TEX.

As we will see, in order to realize an acceptable hyphenation of French using TEX, the fonts have to be modified. This gives the opportunity to improve the appearance of some of the accented characters. Unfortunately, due to some restrictions of both **METAFONT** and TEX, it was impossible to fully achieve this last goal. We explain why, and propose some remedies.

Finally, some typographical conventions are different: Spacings are different in French and the quotation marks are replaced by so-called 'guillemets'. These characteristics have been included in our 'French' fonts and our 'French `PLAIN`' format.

Even if the ideas we develop were specifically oriented towards the purpose of using TEX to typeset French, we are certain that most of them are still valid for other languages.

2. French fonts

The necessity of having special fonts comes primarily from the way TEX hyphenates [k]. Let us just say that TEX attempts no hyphenation after the third letter preceding a non-letter or 'explicit kern', where TEX considers a letter to be a character with category code 11 or 12, having a non-zero lower-case code (`\lccode`).

`PLAIN` TEX produces the accented characters by superposing an accent over a letter. This is achieved by means of explicit kerns. It follows that no hyphenation is possible in a word containing accents, after the third letter preceding the first accent. Unfortunately many French words contain accented characters, the most frequent being the é: out of the 49,962 entries of our on-line French vocabulary [g], 15,970 contain at least one accented letter and among them 13,477 contain at least one é.

Another problem arises when typesetting accented characters with TEX. Many times, the placement of the accent is deficient. On low-resolution printers, it can even be inconsistent. Even if the rounding and conversion of the DVI units to the raster units is improved—that eliminates the worst defects and inconsistencies—some accents still have to be adjusted. Over some lower-case letters, they have to be slightly shifted, and they are too high and too steep over the capitals. Of course, some of these defects can be corrected by inserting extra kern, or even by rewriting the macros for the accents. Nevertheless, the best solution is to include all the accented characters as entities in the font: It gives the best positioning of the accents and enables TEX to hyphenate.

French uses thirteen accented characters: â, ê, î, ô, û, é, à, è, ù, ë, ï, ü and ç, plus the ligature œ. Even if the corresponding upper-case characters are missing from 'French' fonts as distributed by some type foundries, at least the accented upper-case `E`'s and the `C` cedilla cannot be dispensed with. In fact, good typography requires all fourteen upper-case special characters [frey].

TEX can handle 'extended' fonts of up to 256 characters, but only the first 128 are defined in the fonts currently used, leaving a lot of room available for extra characters. The solution seems obvious: Place the extra characters in the second half, with character codes from 128 to 257. Unfortunately, this is unfeasible at the moment. The main reason is that only characters with codes from 0 to 127 are allowed to have a nonzero `\lccode`. One further restriction comes from the fact that the 'old' **METAFONT** cannot generate fonts of more than 128 characters. This last difficulty will be eliminated when the 'new' **METAFONT** is fully operational.

Consequently, we tried to devise a temporary solution, allowing us, at least, to build and try the hyphenation patterns. This made it necessary to include a minimal number of accented characters.

By comparing the Computer Modern font tables and the standard ASCII character codes, it appears that only the codes from 0 to 32 and 127 are fully available. (Some others could be used, except in the 'typewriter' fonts.) As it was impossible to move all these characters to positions over 127, we had to get rid of a certain number of them. The best—if not the only—candidates were the unslanted Greek capitals, occupying positions 0–10. Not even the accents can be removed: The acute is needed on any vowel if a quotation from Spanish is to be typeset, and the same is true with the grave accent in Italian, the umlaut accent in German, and the circumflex in Esperanto.

Therefore, we had to abandon some of the accented characters. The ü is never used in modern French, except in a few proper names. The ù is used in one word—où (where)—and à in a few words, all of which are too short to be hyphenated. Each of the others may occur in the middle of words (though very seldom will ë and ï). We decided to include them. They are the ten accented letters: â, ê, î, ô, û, é, è, ë, ï and ç. The ligatures œ and Œ already appear in the 'Computer Modern' fonts.

As it is safer for a lower-case letter to be equal to its `\lccode`—this last parameter is an essential element of the `\lowercase` primitive—the code 0 cannot be used for a lower-case letter. It could be possible to assign this code to the É, which appears quite often at the beginning of a word. But the lower-case é could not be converted to the upper-case É via `\uppercase`: a character with 0 as its `\uccode` remains unchanged. So we did not retain this possibility, hoping that, very soon, new fonts and/or a change to the restrictions about the `\lccode` will allow more flexible capabilities of adding extra letters.

We also removed the Spanish open question and exclamation marks and replaced them by the opening and closing guillemets, for which we wrote some lines of (old) **METAFONT** code. Fortunately, these guillemets now have the same character codes as the ASCII characters '<' and '>'. Moreover, the guillemets usually are not available in typewriter style, and so do not conflict with the characters '<' and '>' in the typewriter fonts.

The accented characters have been produced by adding the **METAFONT** codes for the accent to the code for the letter, and slightly adjusting the horizontal positioning of the former.

Finally, we modified and increased the number of kerning instructions in the **METAFONT** code so that the spacing is consistent: the same implicit kern is inserted between x and é as between x and e, for example. We added the same kern before and after œ as before o and after e, and did the same for Œ. (In our opinion, this could be included in the ordinary fonts.) We noticed that the spacing was wrong between f and î in roman style, and inserted the italic correction. For the same reason some kern had to be specified between the apostrophe and the accented î. The result is acceptable. Unfortunately, no satisfying result could be devised in the case of the sans-serif fonts, especially in such a word as fît (it exists!), where the rhythm of the strokes is important. This confirms the point that the design of the accented characters has to be part of the work of the designer at a relatively early stage. Let us end by mentioning that, for reasons to be explained later, the 'extra space' parameter, known by TEX as `\fontdimen7`, has been set to a very small negative dimension.

A sample of one of our 'French Computer Modern' fonts can be found in Appendix I.

3. The hyphenation patterns

Let us just give a short account of the way we constructed a set of patterns to be used when typesetting French. This work is more extensively described in another article [d].

Liang produced the set of patterns currently used by TEX for hyphenating English by applying his 'PATGEN' program to a slightly edited version of the *Webster's Pocket Dictionary* (see [l]). The same approach was impossible in French: no French dictionary indicates the feasible break positions. Actually, the rules are quite simple, mostly based on the succession of vowels and consonants, with some unbreakable groups (bl, br, ch, chr, gn, ...). These rules were translated without major difficulties into a first set of 371 patterns, called 'phonetic patterns', from the fact that they were obtained from the phonetic syllabification rules. Unfortunately, in some instances the etymological hyphenation is preferred: for compound words, neologisms and scientific terms. Such a division is needed for about 3.5% of the words, and it appears that most of them are obtained by adding a prefix to an already existing term. The suitable 'etymological patterns' were found by scanning our on-line French vocabulary [g], containing almost 50,000 entries, and seaching for the occurrences of possible patterns. The resulting second set of 403 patterns ensures the right etymological hyphenations where they are needed, and in most nonexistent but possible cases (scientific terms, neologisms). The resulting set of 774 patterns was finally tested by checking once more the French vocabulary.

The conclusion is that 100% of the French words are correctly hyphenated—in our opinion, since no precise rule establishes when the etymological hyphenation has to be preferred to the phonetic one. No permissible hyphenations are missed, and no impermissible ones are introduced. Some words of foreign origin are specified in the `\hyphenation` list. One of the advantages of the method used to produce the patterns is that the meaning of each of them is clear: They can be removed, modified, and some more can be added. If enough care is taken, the effect of such manipulations will be perfectly predictible. Hence it is easy to adapt the set of patterns to anyone's needs, and to correct the possible errors.

In order to load the patterns, and to ensure a proper hyphenation of the words containing extra characters, we must assign these characters lower-case codes. These are obviously equal to the character codes. To allow `\uppercase` to work in an acceptable way, we assign to the accented characters the upper-case codes of the corresponding unaccented capitals. This does not apply to the ligatures œ and Œ: they just exchange their lower-case and upper-case codes like ordinary letters. Finally, the apostrophe is given a `\lccode` equal to its character code, 39.

The only admissible characters in the patterns are letters and digits, the latter, in our case, serving as coefficients (see [k, l]). In particular, no control sequences are allowed in the list of patterns. Consequently, the extra characters must be specified through the 'double hat' notation, used for specifying the ordinarily 'inaccessible' ASCII characters. For example, the é is denoted by `^^F`, corresponding to the code 6. (This kind of obscure encoding is used only when loading the patterns.) In our case, all of the special characters, except ë and ï, but including œ and the apostrophe actually appear in the patterns (`^^A` to `^^G`, `^^I`, `^^[` and `'`).

Some of the codes used for accented characters have been assigned special category codes in the `PLAIN` file: `^^A`, the ASCII code 1 (on SAIL the 'downarrow'), can be used for subscripts (category 8), and the ASCII tab, `^^I`, is synonymous with the space (category 10). Just before loading the patterns, these category codes have to be changed to 12 ('other character'). The initial values are restored just after loading. Obviously, no 'tab' (temporarily equivalent to è!) may appear in between. For the same reasons, in case some other ASCII characters are given a special meaning (due to a peculiarity of the local system for example), care must be taken to avoid any confusion with special characters.

4. Modifications to the `PLAIN` format

In order to test our set of hyphenation patterns, and to devise some simple ways of adapting TEX to French typography, we had to make a new format file. What we produced is only a modified version of `PLAIN` as it is described in [k, Appendix B]. Obviously, a more specific use of TEX in French—as is the case in English—will require its own format. In this case, it would be preferable to make it by incorporating the various French typographical conventions from the beginning instead of adapting an existing format.

First of all, as the unslanted Greek capitals are missing from the 'French' fonts, we decided to keep all the original ones and add some 'French' fonts. Hence, we defined two more families of fonts, the French roman and the French text italic, to begin with, and we contemplate adding new ones if needed (possibly French roman bold). Each of these families consists of only the ten point font, but in the near future, the same sizes will be available for them as for the (ordinary) roman and text italic. Corresponding to the macros `\rm` and `\it` we introduced `\frm` and `\fit`. We could have redefined `\rm` and `\it`, but, because of the lack of Greek capitals, we would have had to modify some code in the 'math macros' section. Furthermore, as long as some sizes are missing in 'French' fonts, it would have been necessary to let `PLAIN` switch, inside the same family, between 'ordinary' and 'French' fonts. It seemed preferable to let the typist know without ambiguity what kind of font he is currently using. Of course, it is not difficult to include such possibilities in `PLAIN`, and that will probably be the case when more 'French' fonts are included. After the French `PLAIN` format has been input, the default font is French roman 10 pt.

A happy consequence of our decision to retain all the fonts used in `PLAIN` is that no modification is needed to the definition of the math macros. The only modifications, apart from the font loading and some minor change at the very end, appear in the fifth section ('macros for text') of `PLAIN`. To minimize the changes, we decided to keep all the control sequences for the accents. To avoid the insertion of spaces inside a word, we added only the variants `\5` for the cedilla and `\e/` and `\E/` for the ligatures œ and Œ, ordinarily obtained by `\c`, `\oe` and `\OE` respectively. For the same reason, we decided that `\^i` and `\"i` could be used instead of `\^\i` and `\"\i`. (We are almost certain that nobody will ever need to accent a dotted `i`; in case we are wrong, it is still possible to make one by `\^{\char'i}`.)

To make the same control sequence behave in two different ways according to the nature of the font—'French' or 'standard'—a parameter has to be introduced to test it. The 'extra space' (`\fontdimen7`) has been set to a very small negative value for the 'French' fonts. This parameter is used when `\nonfrenchspacing` provides extra space after some punctuation marks. This is not the case in a French context, and in case someone uses it, its value is small enough to make sure that no spacing will be wrong. It is to be presumed that no font ordinarily needs such a negative extra space.

In math mode, the accent macros will produce the right accented characters, provided the 'French' fonts are used. But the same macros will result in error messages when used for à, ù or any of the accented capitals, or when the font is not a 'French' one. In any case, until new fonts containing all the accented characters are available, the 'French' fonts are intended for typesetting text only.

When fonts other than 'French' fonts are used, or when accented capitals are typed, the French hyphenation is still on, with the restriction we mentioned earlier: no hyphenation will be attempted after the third letter preceding an accent character. This implies that many legitimate breaking points will be missed. This is not a major restriction, since the fonts that do not have French versions are mostly display fonts. And since when using the ordinary `\hsize` with ten-point fonts, the relatively small number of times that TEX has to hyphenate makes it acceptable.

As we already said, the default font is French roman: the command to set this default appears in the last section ('Hyphenation and everything else') of `PLAIN`. Thus, any addition to the explicit `\hyphenation` list can be made at the very beginning of the user's file. Unlike the main pattern list, it may include control sequences, but they must expand into letters. Consequently, the current font must be a French one to allow accented characters to appear in the words for which an explicit hyphenation is given. (The use of the 'double hat' notations would work in any case, but is not recommended because some of the characters have category codes other than 12.) For the same reason, we had to modify the definition of `\showhyphens`. Instead of testing the hyphenation of a word in `\tenrm`, it is done in the current font, either French or standard.

Another problem arises from the apostrophe. For TEX, it is a letter. But the same character happens to be used as a single close quote. When it has the latter meaning, it might provoke some bad hyphenations towards the end of the preceding word (the worst examples being something like *co-qs'* or even *co-q"*). Such a hyphenation is very unlikely, both for statistical reasons and because the English-style quotation marks are usually not used in French typesetting since they are replaced by the guillemets.

The last set of macros we added deals with the spacing around some punctuation marks. The semicolons, exclamation points, and question marks are usually preceded by a thin space, which must not shrink nor stretch. The colons should be preceded by the inter-word space of the current line, which may vary from one line to another. The ordinary inter-word space follows any punctuation. Obviously, no line break is allowed before a punctuation mark. Moreover, these four punctuation marks usually are preceded and followed by a space when typed on a keyboard by French typists. The solution we have adopted is to make these characters active (category code 13). When they occur, they first test the current mode. If in horizontal mode, they remove the last glob of glue, if any (*i.e.*, the space before the punctuation is significant) and in this case replace it with an explicit kern (for the first three marks) or with unbreakable inter-word glue (for the colon). The explicit kern is such that its value is .4 times the inter-word space except in typewriter style, in which case it is exactly the inter-word space.

Finally, there exists a French typesetting convention for which we were unable to devise the correct macros: When a quotation is made *inside* another quotation, it begins with the same mark (*i.e.* opening guillemets). But, until the end of the inner quotation, the opening guillemets are repeated at the beginning of each line and are aligned with the left margin. In a sense, the lengths of the lines are shorter by the width of the opening guillemets during the inner quotation. The difficulty comes from the fact that it is impossible to modify the lengths of the lines inside a paragraph, unless the number of lines to be modified is known in advance.

The French quotes-in-quotes convention appears in the following paragraphs. They could appear in a review of our article [d].

Les problèmes posés par la division étymologique sont ainsi résumés par J. Désarménien (*cf.* [d]) :

« Tout le monde s'accorde à trouver légitime la division : `extra-ordinaire`. Le cas des mots composés est le seul à rencontrer cette unanimité. Dans les autres, les opinions sont très partagées. Frey est le plus catégorique : il n'admet de division selon la formation que pour des mots de composition complètement française, et la rejette sinon ... Le *Code typographique*, tout en adoptant ce même point de vue « ... [reconnaît] néanmoins que certains au-
« teurs de travaux scientifiques préfèrent la division

« étymologique qui fait ressortir la racine grecque « ou latine ». Quant à Gouriou, il écrit : « On pré- « férera cependant garder la coupure étymologique « chaque fois que les composants sont *aisément* re- « connaissables. » (L'italique est de Gouriou.) Telle est aussi l'opinion de Girodet. »

The solution we give essentially builds a first list of lines, until reaching the inner quotation; then it removes the last line from the list and appends what remains to the page. Then, this last line is used as the beginning of a second vertical box with different indentation parameters. Once more, the last line is removed and the rest is appended to the page with opening guillemets at the beginning of each line. Finally, this new last line is appended to the page as a horizontal list, and it is followed by the end of the current paragraph. A number of 'dirty tricks' were used to avoid the use of arguments inside the macros, to allow the right spacings and penalties between lines, and to avoid having two opening guillemets at the beginning of the first line of the second section when the last line of the first section is almost `\hsize` long.

5. Conclusions

Our main goal, when starting this work, was to build a good set of patterns for French hyphenation. We think that this has been achieved satisfactorily. The other problems we had to face—except those related to the spacing around punctuation marks—were consequences of this one.

The main limitation arose from the difficulty of adding characters to the standard fonts. There are essentially two options for the future. The first one is to modify the code of TEX itself to allow characters with codes over 127 to be assigned lower- and upper-case codes. We did not want to take the responsibility of a change that could become more extensive than expected.

The other possibility is to free the first 32 positions of the fonts. The characters currently appearing in these positions could be moved to codes over 128: They include Greek capitals, ligatures, accents and special characters, none of them needing an `\lccode`. The opportunity can be taken to replace the Spanish open exclamation and question marks by guillemets, which are used in a number of languages, and it seems essential that they be accessed directly. The special Spanish marks could be in the second half of the font. It could also be possible to design special accents for capitals. In this manner, a set of 'basic' fonts would be available. They could be used for typesetting English, even if some quotations from foreign languages had to be made.

Then 'local' fonts could be obtained by assigning the first 32 codes to what is needed: in French, the 13 accented letters plus the ligature œ, in lower and upper cases, in Spanish the accented vowels, ñ and ü, and so on. In some cases, characters normally in the second half of the font would need to be transferred to the first half, or they could simply be duplicated: the ligatures œ and Œ in French, the crossed ø and Ø in Danish and the dotless ı in Turkish are examples.

One of these choices would make it possible to limit the changes to the `PLAIN` format to a few, well determined areas, such as accents and spacing. There would be no need to add new fonts. It would be sufficient to substitute 'local' fonts to the 'basic' ones. If some care is taken when writing a format, it would even be possible to process *any* TEX source with *any* local set of fonts, the best results being obtained with the right set.

It seems to us that such an approach would simultaneously facilitate the adaptation of TEX to a language other than English, and keep its universality.

There is one last problem one is faced with when using TEX in a foreign language. It has been mentioned in the introduction: The input must fit the local typing habits. Unfortunately the number of ASCII characters is too small to allow a direct access to the extra characters. A solution must involve modified or programmable keyboards. Obviously, no rule can be established in this domain. Nevertheless, it seems preferable that such local systems be used to produce, in a handy way, a universally understandable TEX file. Such a system is currently working in Strasbourg.

Acknowledgements

I want to express my gratitude to all the 'TEX wizards' from the Stanford Department of Computer Science for their help. In particular, I am greatly indebted to Don Knuth for inviting me to Stanford, placing me in the best possible situation to start this work, and correcting some errors in the manuscript. I also owe a great lot to John Hobby: Most of the TEXnical subtleties are his and Don Knuth's; however, I am fully responsible for the mistakes.

In addition, I thank Oren Patashnik for his careful reading of the manuscript.

References

[d] J. DÉSARMÉNIEN, La division par ordinateur des mots français avec le logiciel TEX, preprint.

[fr] D. FOATA et Y. ROY, A Rationalized French Keyboard for Inputting TEX, Laboratoire de typographie informatique de l'Université de Strasbourg.

[frey] A. FREY, *Manuel nouveau de typographie*, à la librairie encyclopédique de Roret, Paris, 1835.

[g] M. GROSS, DLAS, vocabulaire français informatisé, Laboratoire d'automatique documentaire et linguistique, Université Paris VII.

[k] D. KNUTH, *The TEXbook*, Addison-Wesley, Reading, Mass., 1984.

[l] F. M. LIANG, Word Hy-phen-a-tion by Com-put-er, Ph.D. Thesis, Department of Computer Science, Stanford University, Report No. STAN-CS-83-977, 1983.

Appendix I

Text font 'French roman'

	0	1	2	3	4	5	6	7
'000		â	ê	î	ô	û	é	ç
'010	ë	è	ï	ff	fi	fl	ffi	ffl
'020	ı	ȷ	`	´	ˇ	˘	¯	˚
'030	¸	ß	æ	œ	ø	Æ	Œ	Ø
'040	‐	!	”	#	$	%	&	’
'050	(	)	*	+	,	-	.	/
'060	0	1	2	3	4	5	6	7
'070	8	9	:	;	«	=	»	?
'100	@	A	B	C	D	E	F	G
'110	H	I	J	K	L	M	N	O
'120	P	Q	R	S	T	U	V	W
'130	X	Y	Z	[	“	]	ˆ	˙
'140	‘	a	b	c	d	e	f	g
'150	h	i	j	k	l	m	n	o
'160	p	q	r	s	t	u	v	w
'170	x	y	z	–	—	˝	˜	¨

Appendix II

Modifications to the PLAIN format

Following are the modifications to PLAIN as described in Appendix B of *The TEXbook* [k]. The modified PLAIN format and the patterns given in [d] have been used to typeset reference [d] itself.

In section 4, Font information:

Two new fonts are loaded, and two new families are defined, for French roman and text italic. More fonts in more sizes will complete both families.

```
\font\tenfrm=fmr10
\font\tenfit=fmti10
\newfam\frfam \def\frm{\fam\frfam\tenfrm} \textfont\frfam=\tenfrm
\newfam\fifam \def\fit{\fam\fifam\tenfit} \textfont\fifam=\tenfit
```

In section 5, Macros for text:

The circumflex, acute, grave and dieresis ('trema' or umlaut) accents are entirely redefined. The original macros are removed and replaced by the following.
First, the accented characters are given names.

```
\chardef\@acircon=1
\chardef\@ecircon=2
\chardef\@icircon=3
\chardef\@ocircon=4
\chardef\@ucircon=5
\chardef\@eacute=6
\chardef\@ccedilla=7
\chardef\@etrema=8
\chardef\@egrave=9
\chardef\@itrema=10
```

Then, the superposed versions of these accents are defined. Note the tests for the dotted i for the circumflex and dieresis.

```
\def\@GR#1{{\accent18 #1}}
\def\@AC#1{{\accent19 #1}}
\def\@CIR#1{{\if i#1\accent94 \i \else\accent94 #1\fi}}
\def\@TR#1{{\if i#1\accent127 \i \else\accent127 #1\fi}}
```

Now, the 'built-in' versions of the same accents. Note the tests for the *un*dotted ı in the last two.

```
\def\FRENCH@AC#1{\if e#1\@eacute\else
  \@AC{#1}\fi}
\def\FRENCH@GR#1{\if e#1\@egrave\else
  \@GR{#1}\fi}
\def\@dotlessi{\i}
\def\FRENCH@CIR#1{\def\@param{#1}\if e#1\@ecircon\else
  \if a#1\@acircon\else
  \if i#1\@icircon\else
  \if o#1\@ocircon\else
  \if u#1\@ucircon\else
  \ifx \@dotlessi\@param\@icircon\else
  \@CIR\@param\fi\fi\fi\fi\fi\fi}
\def\FRENCH@TR#1{\def\@param{#1}\if e#1\@etrema\else
  \if i#1\@itrema\else
  \ifx \@dotlessi\@param\@itrema\else
  \@TR\@param\fi\fi\fi}
```

The next one is for the cedilla. Unlike the other French accents, the original macro \c is retained.

```
\def\FRENCH@CED#1{\if c#1\@ccedilla\else
  \c{#1}\fi}
```

Then the final macros are defined, testing the positivity of the 'extra space' parameter of the current font. The control sequence \5 produces the cedilla. The \c still works, but it produces ç by superposing instead of by using the character c cedilla, thus preventing hyphenation.

```
\def\^#1{\relax\ifdim\fontdimen7\font<\z@
  \FRENCH@CIR{#1}\else\@CIR{#1}\fi}
\let\^^D=\^
\def\'#1{\relax\ifdim\fontdimen7\font<\z@
  \FRENCH@AC{#1}\else\@AC{#1}\fi}
\def\`#1{\relax\ifdim\fontdimen7\font<\z@
  \FRENCH@GR{#1}\else\@GR{#1}\fi}
\def\"#1{\relax\ifdim\fontdimen7\font<\z@
  \FRENCH@TR{#1}\else\@TR{#1}\fi}
\def\:{\relax\ifdim\fontdimen7\font<\z@
  \@ccedilla\else \c c\fi}
\def\5#1{\relax\ifdim\fontdimen7\font<z@
  \FRENCH@CED{#1}\else\c{#1}\fi}
\def\e/{\oe}
\def\E/{\OE}
```

The following macros are added to the format. The first series is for the spacing before some punctuations. The punctuations behave in a special way only if they are used in horizontal mode and preceded by a nonnegative amount of glue (a space, for example). Otherwise they append themselves, with category code 12, as usual.

```
\frenchspacing
\catcode`\;=\active
\catcode`\:=\active
\catcode`\!=\active
\catcode`\?=\active
\def;{\relax\ifhmode\ifdim\lastskip>\z@
  \unskip\kern\fontdimen2\font \kern-1.2\fontdimen3\font
  \fi\fi\string;}
\def:{\relax\ifhmode\ifdim\lastskip>\z@
  \unskip\nobreak\ \fi\fi\string:}
\def!{\relax\ifhmode\ifdim\lastskip>\z@
  \unskip\kern\fontdimen2\font
  \kern-1.2\fontdimen3\font\fi\fi\string!}
\def?{\relax\ifhmode\ifdim\lastskip>\z@
  \unskip\kern\fontdimen2\font
  \kern-1.2\fontdimen3\font\fi\fi\string?}
```

The second series deals with inner quotations. It is quite tricky. The macros are used in the following way: The paragraph containing the inner quotation begins with `\qquotes`; the guillemets opening the inner quotation are typed '*<', the guillemets closing it are typed '>*', then the paragraph ends normally. If there are more than one inner quotation, the final '*' of all but the last one must be replaced by `\finniveau2`. The example on page 95has been typed:

```
\qquotes <Tout le monde s'accorde ...
... ce m\^eme point de vue *<\ldots [reconna\^it] n\'eanmoins ...
... grecque ou latine>\finniveau2. Quant \`a Gouriou, il \'ecrit :
*<On pr\'ef\'erera cependant ...
... {\fit ais\'ement} reconnaissables.>* (L'italique est ...
... l'opinion de Girodet>\par
```

The opening '<' is mandatory. The closing '>' may be omitted. In that case, no closing guillemets appear at the end of the double quotation. The text preceding '*<' or following the second '*' may be empty. This allows the double quotation to run for more than one paragraph. The paragraphs in between begin with opening guillemets, but end with no special mark, as is customary in French. More precisely,

```
\qquotes (text 1) *<(text 2)*\par
\qquotes *<(text 3)>* (text 4)\par
```

produces a paragraph break in the middle of the double quotation consisting of `(text 2)(text 3)`. Because many groupings are involved in the following macros, any change of parameters must be declared *before* `\qquotes` in order to be effective inside. Finally the '*' is given a special meaning between `\qquotes` and the end of the double quotation. But its previous meaning is restored from this point, even if it had been made active and given a special definition before `\qquotes`.

```
\def\@setpartozero{\widowpenalty=\z@ \clubpenalty=\z@
  \interlinepenalty=\z@ \brokenpenalty=\z@ \displaywidowpenalty=\z@}
\newbox\@tempbi\newbox\@tempbii\newbox\@tempbiii
\newbox\@tempbiiii\newbox\@tempbvi\newbox\@tempbvii
\newbox\@openquotes
\newdimen\@hminusem
\newif\ifnonvoid
\catcode`\*=\active
\def\@transfervbox#1#2{\nonvoidtrue
  \loop
  \setbox\@tempbi=\vbox{\unvbox#1\global\setbox\@tempbiii=\lastbox
    \unskip}%
  \ifvoid\@tempbiii\nonvoidfalse\fi
  \ifnonvoid
  \setbox\@tempbii=\vbox{\unvbox#2\box\@tempbiii}%
  \setbox#1=\box\@tempbi\setbox#2=\box\@tempbii
  \repeat}
\def\@transferaddvbox#1#2{\nonvoidtrue
  \setbox\@tempbi=\vbox{\unvbox#1\global\setbox\@tempbiii=\lastbox
    \unskip}%
  \setbox#2=\vbox{\box\@tempbiii}%
  \setbox#1=\box\@tempbi
  \loop
  \setbox\@tempbi=\vbox{\unvbox#1\global\setbox\@tempbiii=\lastbox
    \unskip}%
  \ifvoid\@tempbiii\nonvoidfalse\setbox#1=\box\@tempbi\fi
  \ifnonvoid
  \setbox\@tempbii=\vbox{\unvbox#2%
    \hbox to\hsize{\copy\@openquotes\unhbox\@tempbiii}}%
  \setbox#1=\box\@tempbi\setbox#2=\box\@tempbii
  \repeat}
\def\@sendtopage#1{\nonvoidtrue
  \loop
  \setbox\@tempbi=\vbox{\unvbox#1\global\setbox\@tempbiii=\lastbox
    \unskip}%
  \ifvoid\@tempbiii\nonvoidfalse\setbox#1=\box\@tempbi\fi
  \ifnonvoid
  \unhbox\@tempbiii\unskip\break
  \setbox#1=\box\@tempbi
  \repeat}
\def\@star{\egroup
  \@transfervbox\@tempbvi\@tempbvii
  \@transferaddvbox\@tempbvii\@tempbvi
  \setbox\@tempbvii=\vbox{\unvbox\@tempbvi
    \global\setbox\@tempbiiii=\lastbox\unskip}%
  \@transfervbox\@tempbvii\@tempbvi
  \noindent \@sendtopage\@tempbvi
  \unhbox\@tempbiiii\unskip\unskip\unpenalty}
```

```
\def\finniveau2{\egroup
  \@transfervbox\@tempbvi\@tempbvii
  \@transferaddvbox\@tempbvii\@tempbvi
  \setbox\@tempbvii=\vbox{\unvbox\@tempbvi
    \global\setbox\@tempbiiii=\lastbox\unskip}%
  \@transfervbox\@tempbvii\@tempbvi
  \noindent \@sendtopage\@tempbvi
  \setbox\@tempbvii=\vbox\bgroup\@setpartozero
  \catcode`\*=\active \let*=\@staropen
  \noindent \unhbox\@tempbiiii\unskip\unskip\unpenalty}
\def\@staropen<{\global\setbox\@openquotes=\hbox{<}\egroup
  \setbox\@tempbvi=\vbox{\unvbox\@tempbvii
    \global\setbox\@tempbiiii=\lastbox\unskip}%
  \@transfervbox\@tempbvi\@tempbvii
  \noindent\@sendtopage\@tempbvii
  \setbox\@tempbvi=\vbox\bgroup\@setpartozero
  \catcode`\*=\active
  \let*=\@star
  \hangindent=\wd\@openquotes\hangafter=1
  \setbox\@tempbvii=\hbox{\unhcopy\@tempbiiii\unskip\unskip
    \unpenalty}%
  \@hminusem=\hsize \advance\@hminusem by -2em%
  \ifvoid\@tempbiiii\indent\copy\@openquotes
  \else
  \parindent=\z@
  \ifdim \wd\@tempbvii>\@hminusem \unhbox\@tempbvii\break
  \else \unhbox\@tempbvii\nobreak\ \copy\@openquotes\fi\fi}
\def\qquotes{\setbox\@tempbvii=\vbox\bgroup\@setpartozero
  \catcode`\*=\active \let*=\@staropen}
\catcode`\*=12
```

In section 8, Hyphenation and everything else:

The `\showhyphens` macro is modified, and the default font is French roman 10 pt.

```
\def\showhyphens#1{\setbox0\vbox{\parfillskip\z@skip\hsize\maxdimen
  \pretolerance\m@ne\tolerance\m@ne\hbadness0\showboxdepth0\ #1}}
\normalbaselines\frm % select french roman font
```

The set of patterns and exceptions is different, and appears *after* the font is set to French roman to allow accents to appear in the hyphenation list.

```
\input frhyph
```

German Hyphenation and *Umlauts* in TEX

Bernd Schulze
University of Bonn
Institute for Applied Mathematics
Wegeler Str. 6, D-5300 Bonn, W. Germany

Since we first heard of Donald Knuth's plans for writing a typesetting system for mathematical texts, we have observed the development of TEX with great interest because our Institute regularly publishes a great number of mathematical preprints. As soon as we could get hold of the first Pascal version we installed it and started gaining experience. Since then we have been able to help several German institutions with their TEX installation. While doing this, we noticed that there were two main obstacles to the applicability of TEX to German publications, namely hyphenation and coding of characters particular to the German language.

So there was the need to make the hyphenation capabilities of TEX available for the German language. Our solution to the problem also simplifies typing of the special characters. We took a different approach than the University of Milan (TUGboat Volume 5, No. 1, pages 14–15) where the TEX program itself was modified to incorporate Italian hyphenation rules. In order to maintain compatibility we rejected that idea.

In the German language, there are some hyphenations that can only be handled by their explicit specification in an exception dictionary. The provisions made by the discretionary hyphen are sufficient (cf. The TEXbook, p. 96). But these cases are rare; satisfactory hyphenation is indeed attainable by making use of nothing but the possibilities Donald Knuth has already built into his program. To explain our approach we have to make a short excursion into the hyphenation algorithm.

When a word has to be hyphenated, TEX searches a given set of character sequences (hyphenation pattern set) for all the patterns that occur in this word. Positive numbers are associated with each of the character pairs of each pattern. For all character pairs of the given word, we use the highest number of all the patterns we have found. A simple rule now allows us to decide where hyphenation is possible: if this number is odd, we may hyphenate here, if it is even, hyphenation is forbidden at this position.

It is clear that the algorithm itself is language-independent. In order to apply it to a specific language you have to generate a pattern set for this language. In his thesis at Stanford (1983), Frank M. Liang took a probabilistic approach to convert a hyphenated dictionary of any language into such a pattern set. Based on Webster's Pocket Dictionary and an additional list of more than 1000 words, he created the file `HYPHEN.TEX` that is now part of the TEX distribution tape. With these patterns TEX is able to reproduce about 90% of the hyphens of the original input dictionary, and it will generate no wrong ones. As a consequence, with high probability TEX will also hyphenate correctly English words that are not in the dictionary. But although English and German are quite similar languages, faulty hyphenations will occur frequently in German words, e.g. `ko-r-re-la-tion-s-anal-yse` vs. the correct `kor-re-la-ti-ons-ana-ly-se`.

We had access to a hyphenated German dictionary with more than 127,000 entries, and, as Liang's program `PATGEN.WEB` is on the TEX distribution tape, we were able to generate a hyphenation pattern set for the German language (`GHYPHENU.TEX`). This file is larger than Liang's (6082 vs. 4447 entries), so it is necessary to enlarge the variable *trie_size* in TEX. These patterns give satisfactory results, e.g. TEX hyphenates `kor-re-la-ti-ons-ana-lyse`.

Using a little trick in our hyphenation patterns, we were able to integrate the German *Umlauts* *ä*, *ö*, *ü*, *Ä*, *Ö*, *Ü* and the letter *ß* into the TEX system. At the same time, we made typing and reading easier.

In plain TEX, *Umlauts* have to be entered in a cumbersome way. `\"` is a macro that sets a trema atop the following letter, so `\"a` gives *ä*. This is acceptable if needed only rarely, but hard to type in longer German texts. And a word with an *Umlaut* will not be hyphenated because TEX views it as a special character. By catcoding `"` as an active character (which calls the same macro as `\"`) the input sequence for *ä* is shortened to `"a`. But ergonomically it would be favourable to interchange both characters to `a"`. We can do this if we follow the advice of the TEXbook, p. 46, and specify in the font metric files that `a` and `"` should be ligatured, giving *ä*.

All that remains is to prevent TEX from hyphenating a word between `a` and a following `"`. Liang, using only the numbers 0 to 5 as hyphenation values, got an acceptable number of breakpoints and, on the other hand, limited the pattern set to a reasonable size. Starting with the same range of numbers, we coded *Umlauts* in the hyphenation patterns as **a"** (as in the input file) and inserted a hyphenation value of 6 between the two characters. So TEX will never hyphenate at this position.

Of course we now need different pixel files with bitmaps for the *Umlauts* at positions where the ligature information points to. Currently, we can use this method only for output devices with built-in character sets and manually-generated font metric files. We have a version of our pattern set where all entries containing an *Umlaut* are deleted (`GHYPHEN.TEX`). As a temporary solution, we load this for use with the standard TEX fonts until Metafont gives us the ability to create the necessary pixel files.

Although the German character ß (pronounced as sharp s) is not an *Umlaut* it can be handled in the same way. It is coded in the hyphenation pattern set as **s6"**, it can be entered as **s"**, and a ligature specification in the font metric files results in ß. With `\def\3{s"}`, we defined a macro that is easier to type and looks almost like the real ß. This circumvents also the problem of the blank character after control sequences because a ß may appear in the middle of a word and at its end.

Both `GHYPHENU.TEX` and `GHYPHEN.TEX` are now used by many German TEX installations. If you have questions or propositions concerning German hyphenation and *Umlauts*, feel free to contact me.

METAFONT

A Course on METAFONT Programming

Donald E. Knuth
Stanford University

During the spring of 1984, four dozen brave students attended an unusual class at Stanford University, taught by two brave professors and by another reckless one. The subject of these lectures was type design in general and the use of the new METAFONT in particular. The course was necessarily improvisational, because METAFONT was still taking shape during the entire time; but I think it's fair to say that the lectures hung together quite well and that the experience was rewarding for all. Videotapes of these 27 class hours are available for rental to anybody who wants to share in the adventures we had. I believe that anyone who is interested in the subject and has the time and opportunity to see these tapes will learn lots of important things.

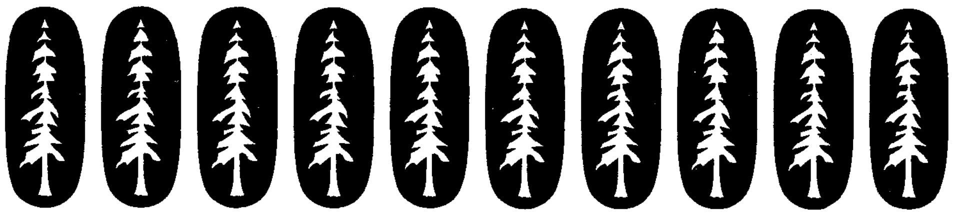
The main reason I can make this claim is that the two brave professors referred to above were Richard Southall and Charles Bigelow, who gave outstanding lectures in alternation with my own contributions. Southall's lectures covered the general subject of "Designing Typefaces," and he broke this down into five subtopics:

(1) Definitions — What is the difference between fonts and typefaces, between type design and calligraphy?
(2) Quality criteria — How can we objectively judge the success of text typefaces?
(3) Facets of the job — What does a type designer have to do?
(4) Methodology — How does traditional knowledge and practice teach us to tackle the problem of type design?
(5) 'Ideal' designs — Can anyone tell us what shapes the characters ought to be?

Bigelow lectured on the history of letterforms, from ancient times to the present. It was instructive to see how character shapes have changed as the technology has changed: Alphabet designs were originally created by a "ductal" process, i.e., by the movements of a writing tool; then printing types were produced by a "glyptal" process, i.e., by carving in metal; and nowadays most letterforms are produced by a digital or "pictal" process, i.e., by specifying patterns on a discrete raster. The work of master type designers of all eras was presented and critically evaluated, and Bigelow concluded by discussing the current state of the art in commercial digital typefaces and in designs for CRT displays. All of the lectures by Southall and Bigelow were lavishly illustrated, in most cases by unique slides from their personal collections.

My job was to relate this all to the new METAFONT. My luck held good throughout the quarter, as new pieces of the language would begin to work just about two days before I needed to discuss them in class. That gave me one day to get some programming experience before I was supposed to teach everybody else how to write good programs themselves. I lectured about (1) coordinates, (2) curves, (3) equations, (4) digitizing, (5) pens, (6) transformations, and (7) the syntax of METAFONT.

The students did several instructive homework problems. First there were two assignments done with cut paper, to illustrate the important differences between "what we see" and "what's there." Then came Homework #3, the first computer assignment, which was to write METAFONT code for Stanford's symbol, El Palo Alto (the tall tree); each student did just two of the branches, since it would have been too tedious to do all twelve of them, and I combined their solutions to obtain the following results:



(Each of these trees is different, although many of the individual branches appear in several different trees because some of the branches were worked on more often than others.) The purpose of this exercise was to help the students get used to the ideas of coordinates and simple curves, as they became familiar with the computer system and its text editor. An organic shape like a tree is very forgiving.

The fourth homework assignment was much more interesting, and we called it "Font 1." The class created a new typeface with a sans-serif, calligraphic flavor; we had just enough people who had completed Homework #3 so that everybody could be assigned the task of creating one uppercase letter and one lowercase letter. I presented an uppercase 'U' and lowercase 'l' as examples that would help to set the style; but of course each student had a personal style that was reflected in the results, and there wasn't much unity in our final font. This fact was instructive in itself.

I had prepared two METAFONT macros to draw penlike strokes and arcs, and the students were required to draw everything with those two subroutines. This was a signification limitation, but it helped to focus everyone's attention by narrowing the possibilities. The students were also learning the concepts of meta-design at this time, because their programs were supposed to be written in terms of parameters so that three different fonts would be produced: normal, bold, and bold extended. This gave everyone a taste of METAFONT's algebraic capabilities, in which the computer plays a crucial rôle in the development of one's design.

The best way to describe the outcome of Homework #4 is to present the font that we made:

ABCDEFGHIJKKLM
NOPQRSTTUVWXYZ
abcdefghijklm
nopqrstuvwxyz

This font of type, the first
to be produced by the new
METAFONT system, was
designed by Neenie Billawala,
Jean-Luc Bonnetain,
Jim Bratnober,
Malcolm Brown,
William Burley, Renata Byl,

Pavel Curtis, Bruce Fleischer,
Kanchi Gopinath,
John Hershberger,
Dikran Karagueuzian,
Don Knuth, Ann Lasko-Harvill,
Bruce Leban, Dan Mills,
Arnie Olds, Stan Osborne,
Kwang-Chun Park, Tuan Pham,
Theresa-Marie Rhyne,
Lynn Ruggles, Arthur Samuel,
New Wave Dave,
Alan Spragens, Nori Tokusue,
Joey Tuttle, and Ed Williams.

(See also the examples of bold and bold extended at the close of this article.) As I said, we didn't expect Font 1 to have any unity, but I was pleased that many of the individual characters turned out to be quite beautiful even when the parameters were changed to values that the students had not tried.

The fifth and final homework assignment was more interesting yet. Everybody was to design a set of eight characters that could be used to typeset border designs. These characters were called NW, NM, NE, ME, SE, SM, SW, and MW in clockwise order starting at the upper left; here 'N' means North, 'E' means East, 'S' means South, 'W' means West, and 'M' means Middle. The height of each character was determined by the first component of its name, and the width was determined by the other component. Thus, for example, NW and NM were required to have the same height; SE and ME were required to have the same width. As a consequence, the four characters with M's in their names could be used as repeatable extension modules to make arbitrarily large rectangles together with the four corner characters. But there were no other ground rules besides these mild restrictions on height and width, and the students were urged to let their creative minds dream up the greatest borders that they could program in METAFONT.

It was especially exciting for me to see the completed border projects, because I was impressed by the originality of the designs and (especially) because I was glad to see that the new version of METAFONT was working even better than I had dared to hope. We still need to wait awhile before we'll know how adequate **METAFONT** will be as a tool for letterform design, but already we can be sure that it's a super tool for borders! The results of this experiment appear below and elsewhere in the pages of this issue of *TUGboat*.

I should mention some of our experiences related to the "high tech" nature of a class like this. None of the computers accessible for classes at Stanford had a high-resolution screen with graphic capabilities, so we had ordered SUN workstations to fill the void. When those machines finally arrived—a week before the class was scheduled to begin—they were a new model for which new software needed to be written in order to put them into the campus network and connect them to various peripheral devices. The manufacturer balked at letting us see the source code of their software, but we needed it in order to get going. We also found that we couldn't use their version of UNIX anyway, because it allocated each file on our main disk to a specific workstation; that would have forced each student to log in at the same workstation each time! Furthermore their PASCAL compiler was unusable on a program as large as **METAFONT**.

So we decided to use a locally developed operating system called the V-System or V-Kernel, due to Profs. David Cheriton and Keith Lantz and their students. Fortunately one of those students, Per Bothner, was a member of the TEX project, and he had also written his own PASCAL compiler. Unfortunately, however, we couldn't use the V-System without connecting all of our SUNs to a more powerful machine like a VAX, and we didn't own one. To make matters worse, the building in which we had planned to put our SUNs was being renovated; we were originally scheduled to occupy it in January, but each month another problem had delayed the construction, and it was clear by the end of March that we wouldn't have any place to put the SUNs until May at the earliest!

Here again Prof. Cheriton saved the situation for us, because he had independently been making plans to set up a teaching lab with graphic workstations in another building. His workstations hadn't arrived yet, so we were able to loan part-time use of our SUNs in return for the use of his lab. Furthermore he had a new VAX that we could install next door.

The actual timetable went something like this: On March 24 I had finished coding a subset of **METAFONT** that I hoped would be enough to use in the class, but I hadn't started to debug it yet. On April 1, I obtained the first successful output of that program on a small test case, and **METAFONT** also displayed a character correctly for the first time on my screen. (This was on the SAIL computer, a one-of-a-kind 36-bit machine on which I have done all of the development of TEX and **METAFONT**.) The next day, April 2, I learned about the possibility of using Cheriton's lab for our course; the room was still without furniture, computers, and air-conditioning, but David Fuchs and other people pitched in to help get things moving there. On April 3, Per Bothner successfully transported **METAFONT** from SAIL to a SUN workstation using the V-System. And April 4 was the first day of class.

Pavel Curtis

The V-Kernel system had previously been used only by hackers, so there was no decent manual for novices; furthermore none of us except Per knew how to use it. Arthur Samuel came to the rescue and began to prepare an introductory manual. Meanwhile, we had special meetings with Stanford's TV network technicians, because there was no adequate way to run **METAFONT** from a classroom so as to display the results online to the audience. On April 6 I began to write `GFtoDVI`, a fairly long program that is needed to get proofsheets from **METAFONT**'s output; I knew that it would take at least two weeks to complete that program. Lynn Ruggles had already made progress on another utility routine, `GFtoQMS`, which produces fonts suitable for a new printer that we had just received. (But that printer wasn't installed yet.)

Bigelow and Southall knew that it would be a miracle if the computers were all in place on time, so they were prepared to "vamp" until I was ready. I gave my first lecture on Friday, April 13, one day after Lynn had been able to typeset the first **METAFONT**-made character on our QMS. The students had had non-computer homework to do, as mentioned above, so we were able to make it seem natural that the first computer assignment was not distributed until April 27.

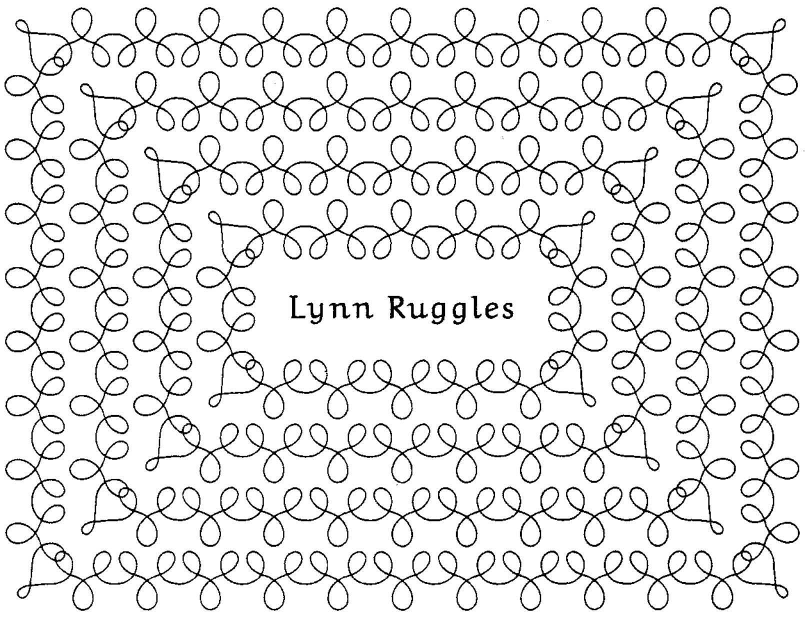
Well, the month of May was a long story, too—the computers broke down frequently because of inadequate air-conditioning, which took weeks to install—and there were plenty of software problems as I kept making new versions of **METAFONT**. But people were good natured and they tolerated the intolerable conditions; I rewarded them for this by cancelling a planned Homework #6. Teaching assistants Dan Mills and Dave Siegel did yeoman service to keep everything running as smoothly as possible throughout the nine weeks of the class.

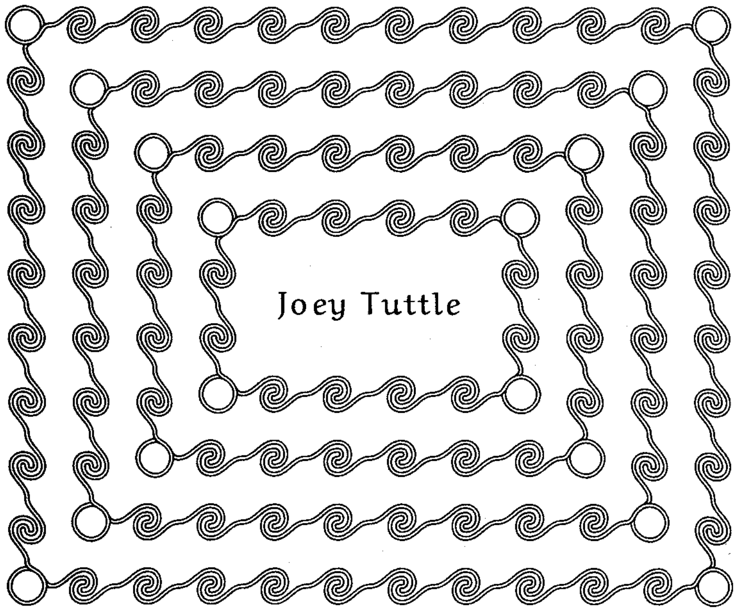
Finally the course came to a glorious finish as we took a field trip to San Francisco. We had a picnic on Font Boulevard, then toured the fascinating MacKenzie–Harris type foundry and the Bigelow & Holmes design studio. I can best convey the jubilation of that memorable day by showing a picture of the "italic" font that we all made just after lunch:

Jill Knuth

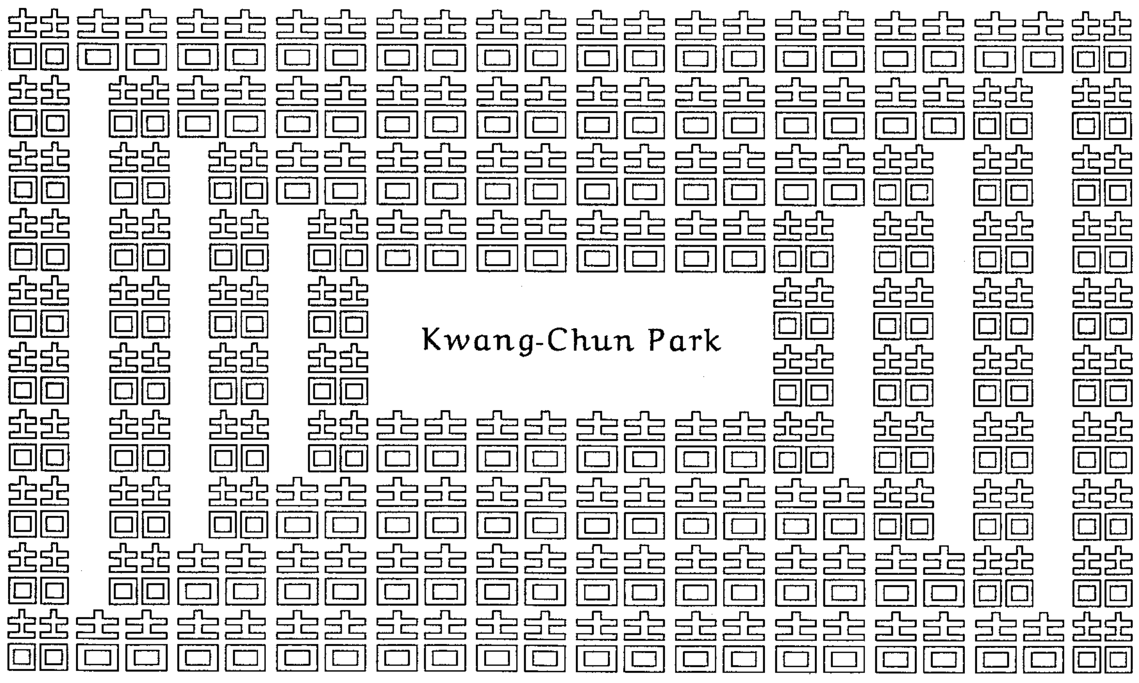
In every period there have been better or worse types
employed in better or worse ways.
The better types employed in better ways
have been used by the educated printer
acquainted with standards and history,
directed by taste and a sense of the fitness of things,
and facing the industrial conditions of his time.
Such men have made of printing an art.
The poorer types and methods have been employed
by printers ignorant of standards
and caring alone for commercial success.
To these, printing has been simply a trade.
The typography of a nation has been good or bad,
as one or other of these classes had the supremacy.
And today any intelligent printer can educate his taste,
so to choose types for his work and so to use them,
that he will help printing to be an art rather than a trade.
There is not, as the sentimentalist would have us think,
a specially devilish spirit now abroad
that prevents good work from being done.
The old times were not so very good,
nor was human nature then so different,
nor is the modern spirit particularly devilish.
But it was, and is, hard to hold to a principle.
The principles of the men of those times
seem simple and glorious.
We do not dare to believe that we, too,
can go and do likewise.

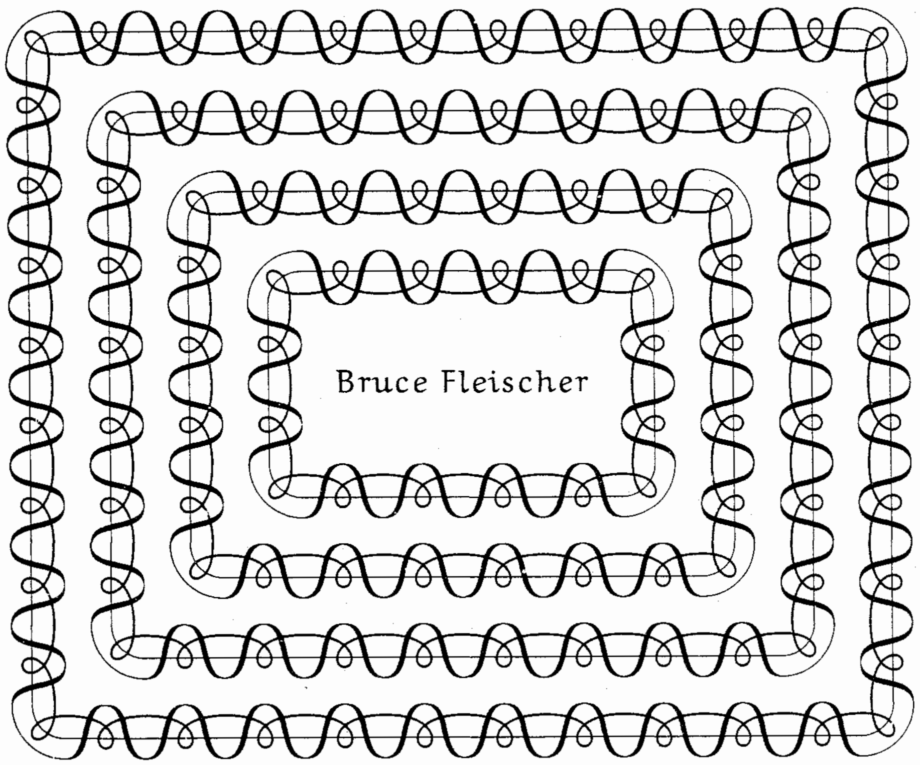
DANIEL BERKELEY UPDIKE





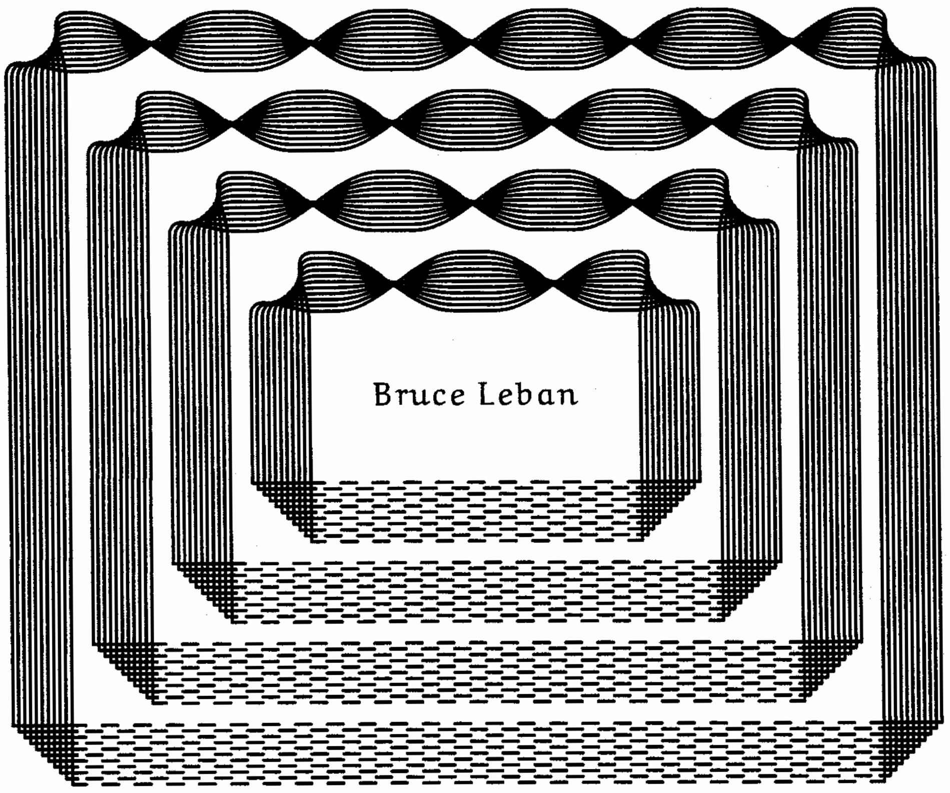
John Hershberger

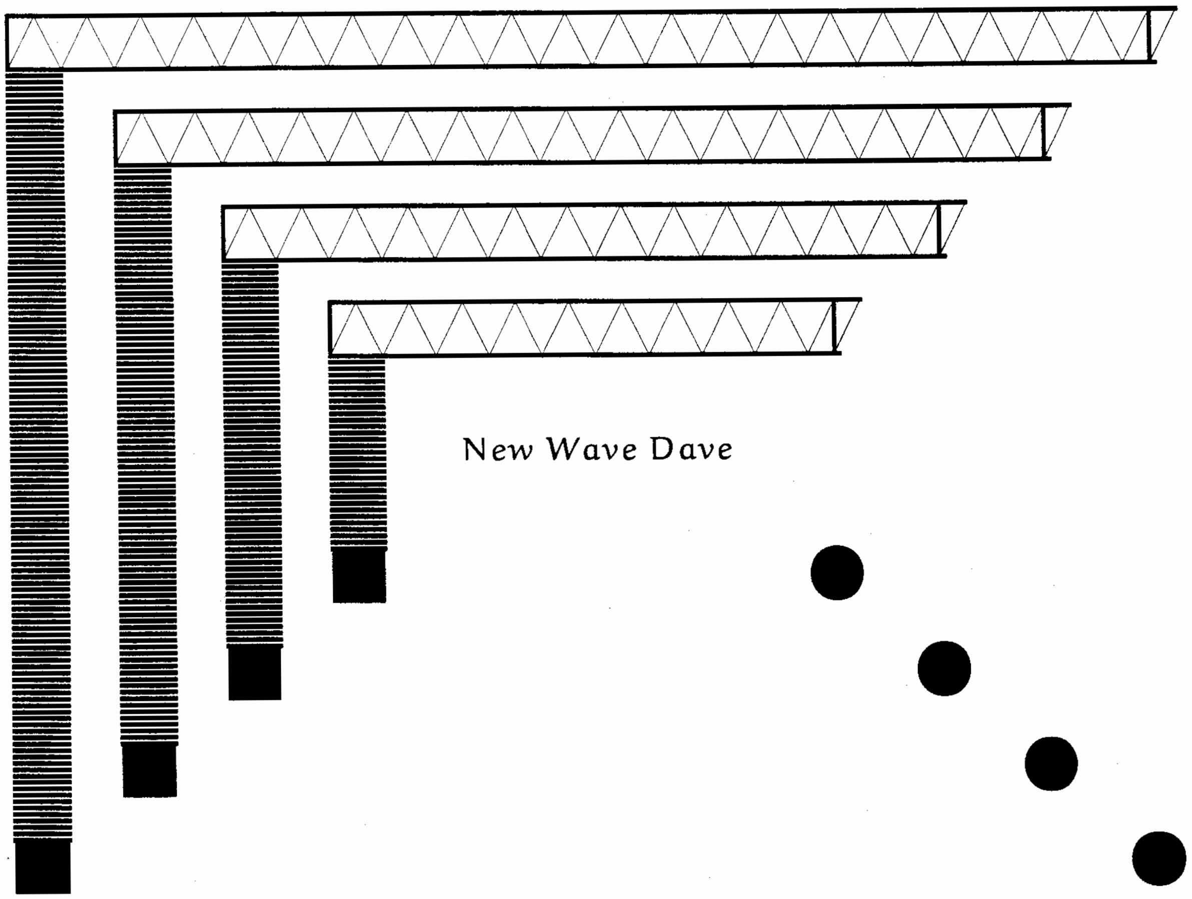


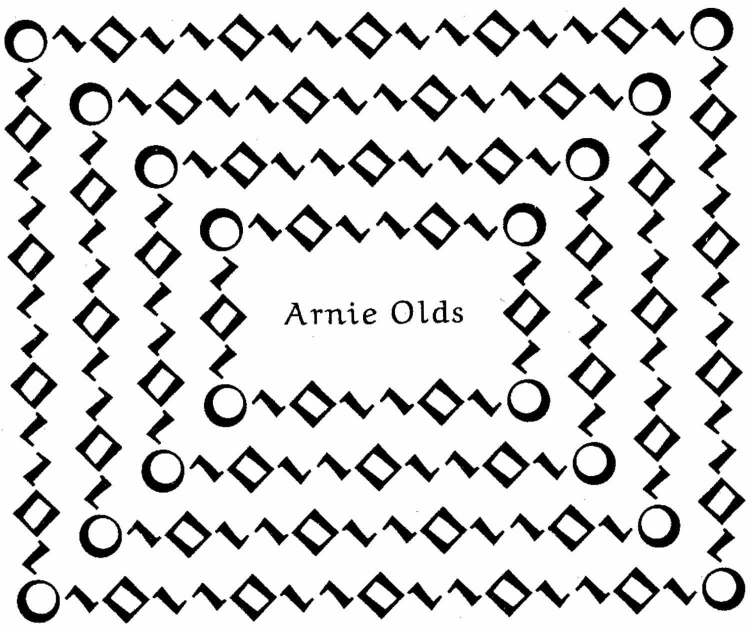


Nori Tokusue

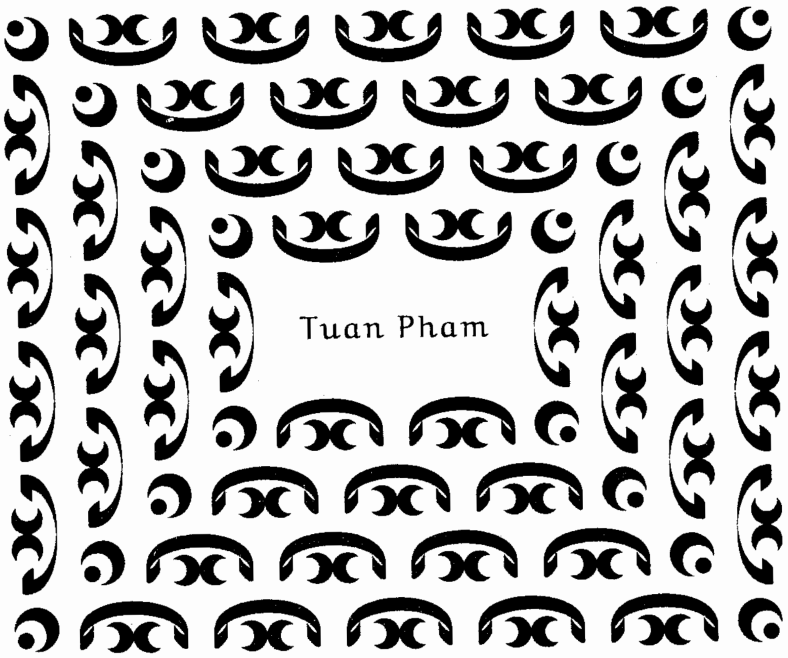
Neenie Billawala


Bruce Leban

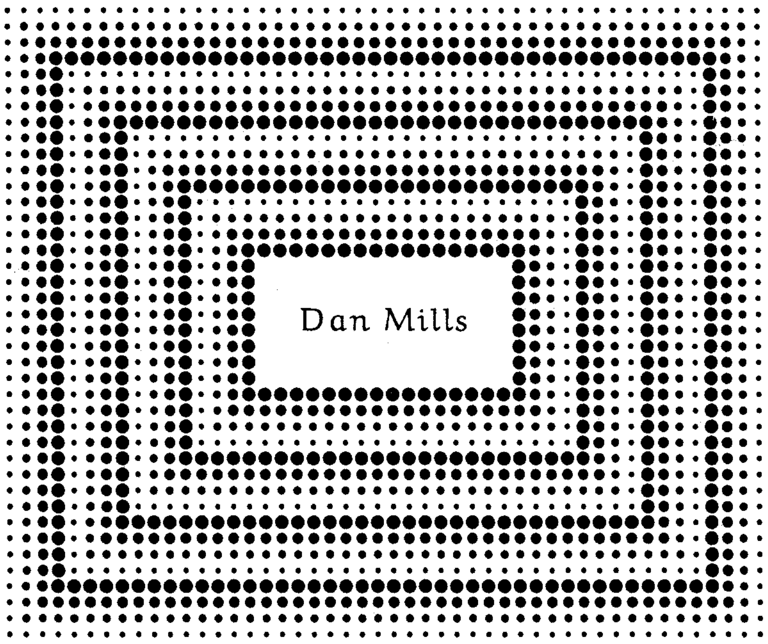

New Wave Dave

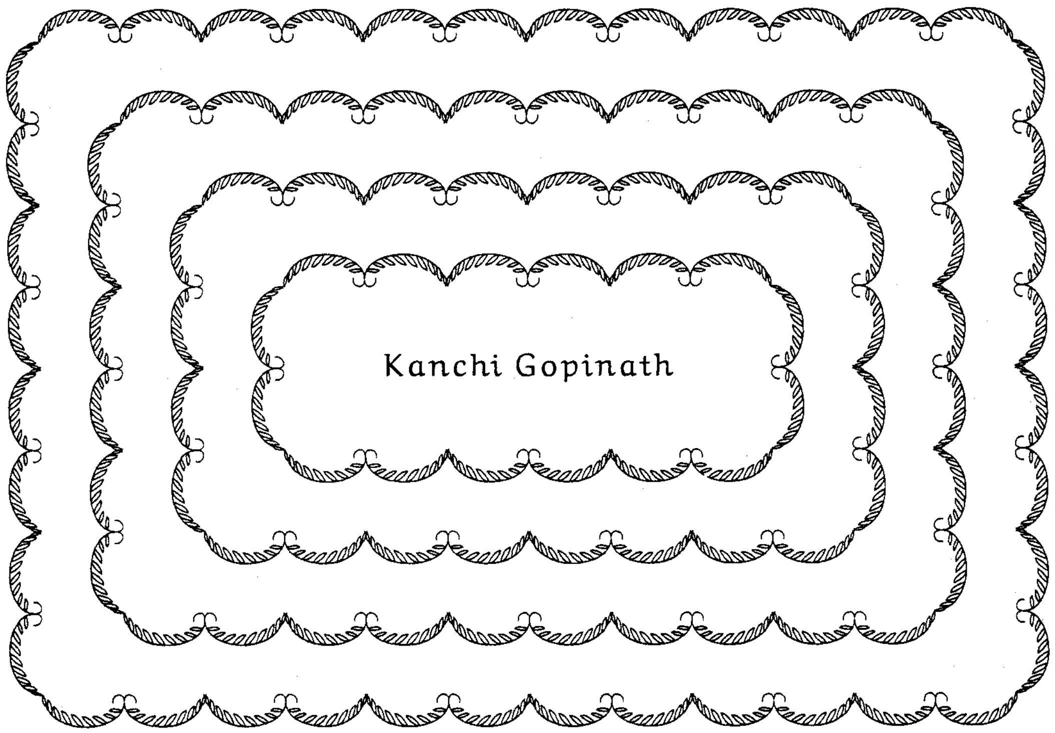


Arthur Samuel

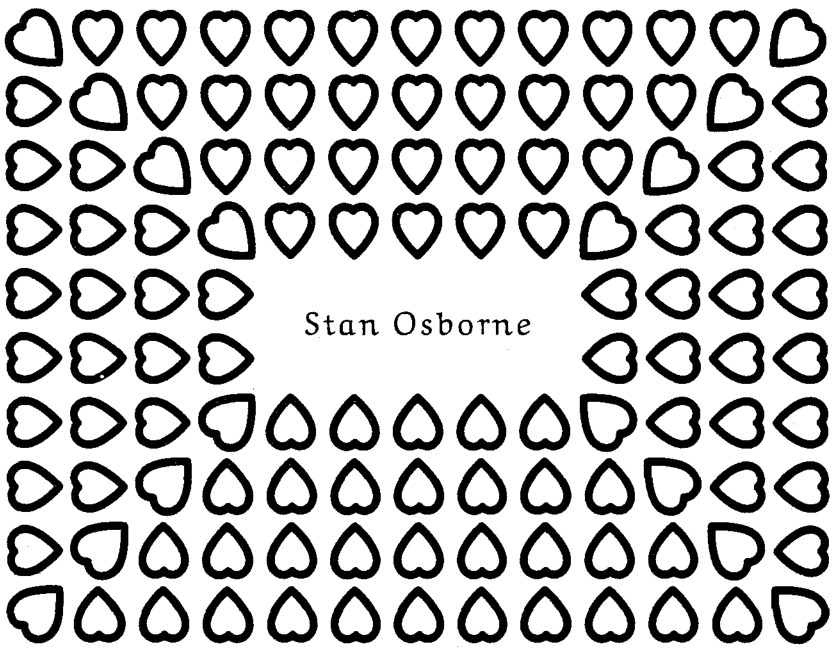


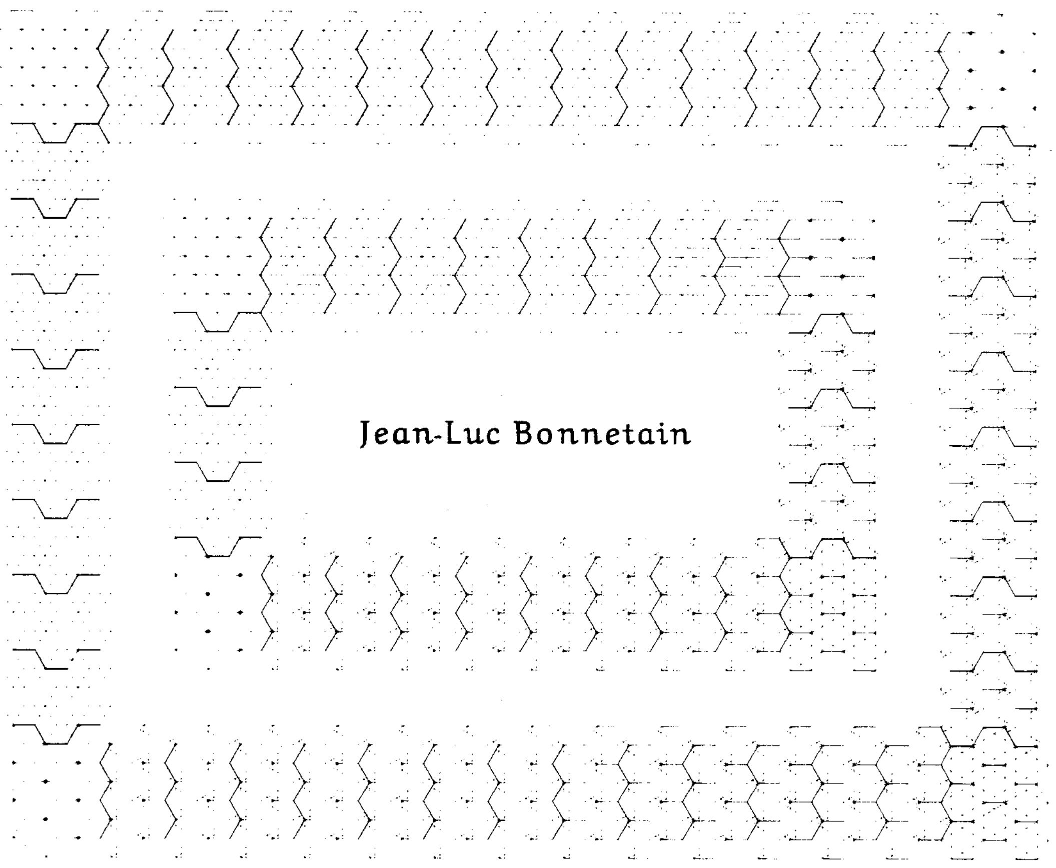
Jim Bratnober

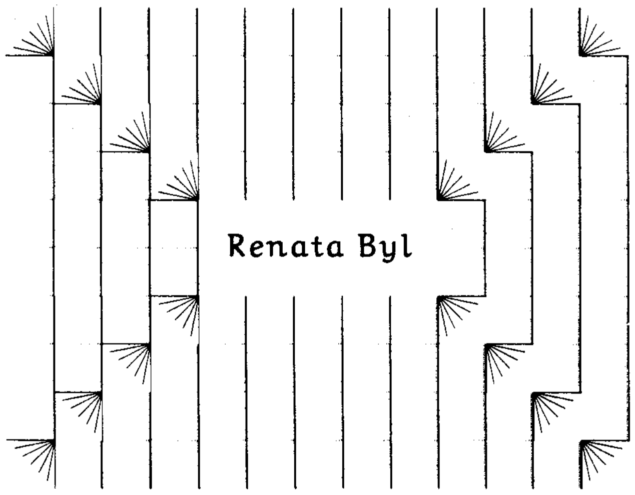


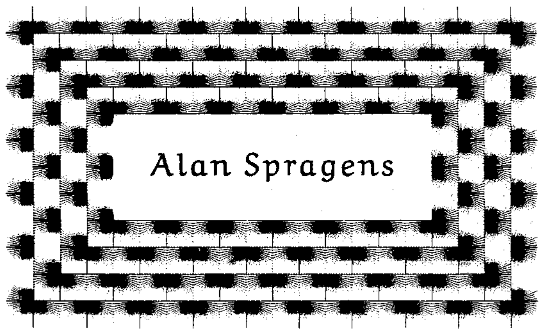


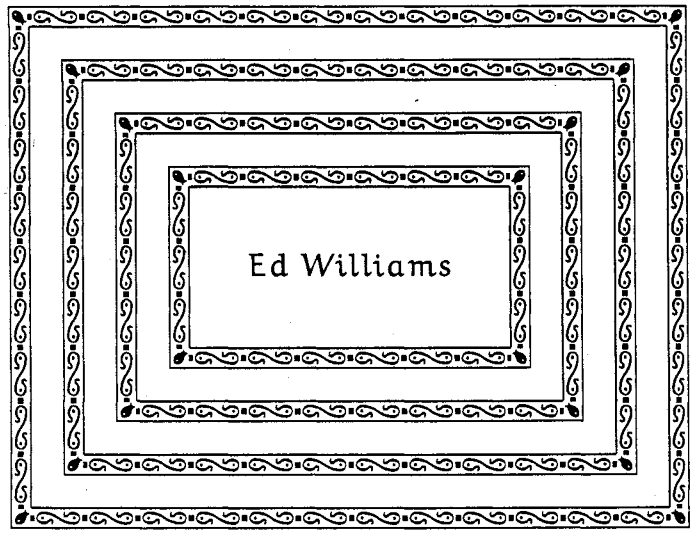
William Burley

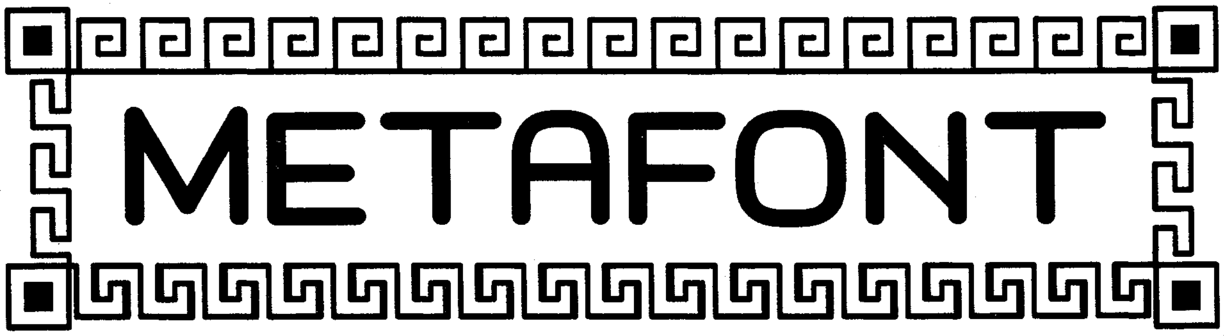



Jean-Luc Bonnetain

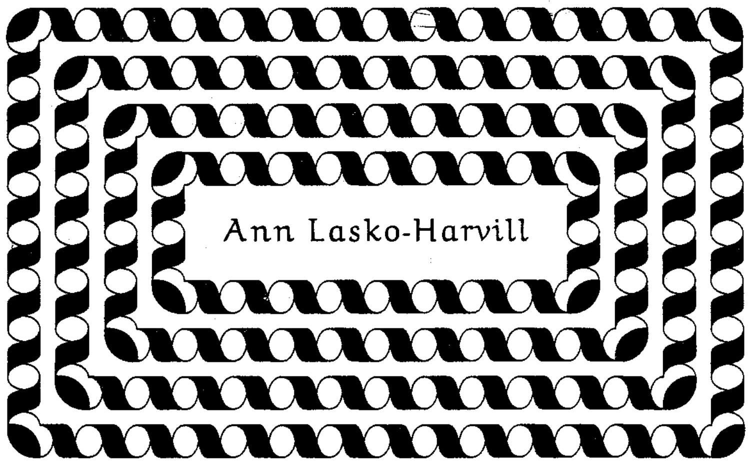

Renata Byl


Alan Spragens





Malcolm Brown



A Chinese Meta-Font

John Hobby and Gu Guoan

Abstract

METAFONT is Donald E. Knuth's system for alphabet design. The system allows an entire family of fonts or "meta-fonts" to be specified precisely and mathematically so that it can be produced in different sizes and styles for different raster devices.

We present a new technique for defining Chinese characters hierarchically with METAFONT. We define METAFONT subroutines for commonly used portions of strokes and then combine some of these into routines for drawing complete strokes. Parameters describe the skeletons of the strokes and the stroke routines are carefully designed to transform themselves appropriately. This allows us to handle all of the basic strokes with only 14 different routines.

The stroke routines in turn are used to build up groups of strokes and radicals. Special routines for positioning control points ensure that the strokes will join properly in a variety of different styles. The radical routines are parameterized to allow them to be placed at different locations in the typeface and to allow for adjusting their size and shape. Key points are positioned relative to the bounding box for the radical, and the special positioning routines find other points that must be passed to the stroke routines.

We use this method to design high quality Song style characters. Global parameters control the style, and we show how these can be used to create Song and Long Song from the same designs. Other settings can produce other familiar styles or even new styles. We show how it is possible to create completely different styles, such as Bold style, merely by substituting different stroke routines. The global parameters can be used to augment simple scaling by altering stroke width and other details to account for changes in size. We can adjust stroke widths to help even out the overall darkness of the characters. We also show how it is possible to experiment with new ideas such as adjusting character widths individually.

While many of our characters are based on existing designs, the stroke routines facilitate the design of new characters without the need to refer to detailed drawings. The skeletal parameters and special positioning routines make it easy to position the strokes properly. In our previous paper, in contrast to this, we parameterized the strokes according to their boundaries and copied an existing design. The previous approach made it very difficult to create different styles with the same METAFONT program.

This research was supported in part by National Science Foundation grant IST-8201926 and by the Systems Development Foundation.

An abridged version of this report appeared on pages 62–67 of the *Proceedings of ICTP'83*, the 1983 International Conference on Text Processing with a Large Character Set (October 1983), published by the Chinese Language Computer Society and the Information Processing Society of Japan.

0. Introduction

Chinese character generation is a very important part of Chinese language computer systems, and it is complicated by the number and complexity of Chinese characters. Even simplified characters contain an average of about twelve strokes each, and a good printing system requires all four standard styles in different sizes, with at least 8,000 characters in each. The designs can be digitized using optical scanning, but this is expensive and the resulting characters must be edited individually.

METAFONT is a system for designing alphabets for raster devices so that a single mathematical description can be used for different sizes and styles of fonts on different devices [1]. While not designed explicitly for Chinese characters, METAFONT is a general system with features useful for Chinese character design.

Knuth's idea of a "meta-font" is to describe alphabets parametrically so that one routine can produce different styles of letters. In [2], Knuth explains how this can be done for Roman alphabets. We apply the same concept to Chinese characters, except that we also describe the radicals and the strokes parametrically. While there are really only a few kinds of strokes that are fundamentally different, similar strokes can vary significantly. We can therefore produce better strokes with fewer routines by parameterizing these differences. A similar type of parameterization also applies to radicals. We create a complete hierarchy starting with eight routines for parts of strokes and thirteen routines for complete strokes. This hierarchical organization not only simplifies the design process, it leads to more uniformity in the designs. The more complicated strokes are formed in the radical routines by combining the basic strokes, and more complicated radicals call routines that draw simple combinations of strokes. The basic strokes have special parameters that specify how they are being joined together so that they can draw the special features that appear near such joins. There are also support routines for calculating points used in the constructions and for positioning certain combinations of strokes, taking into account style parameters such as the stroke width.

Designing an entire set of 8,000 characters would be a rather large project. Instead we have designed a representative sample of about 140 radicals and 128 characters. Many of the radicals appear in more than one character, and many more characters could be formed from these radicals. This work is an extension of the ideas presented in [6], which showed how **METAFONT** could be used to copy specific Song style designs. Here we also make use of these designs as well as Bold style and Long Song style from the same source [5], but we only use them to get an idea of how the strokes should be placed and to determine how to set the parameters to the **METAFONT** programs so that they can produce the various styles. The Song and Bold style designs consist of a carefully chosen set of about 125 large characters superimposed on graph paper, and the Long Song designs were taken from large scale photographs.

The global font parameters that affect size and style are used mainly in the stroke routines themselves. For Song style, there are 68 global parameters that control the slant and aspect ratio, the stroke widths and amount of taper, the size and shape of the stroke end features, and various special properties of certain strokes.

The stroke routines have to be designed carefully to work properly for all reasonable settings of these parameters, and to join together properly even when stroke widths and shapes change. To achieve this, we use control points on the skeletons of the strokes and join strokes by placing the ending control point of one on the skeleton of the other. This is simpler and more flexible than the technique used in [6], where the parameters described the edge of the stroke. The stroke routines also take fewer parameters, so that all details except the placement of the skeleton are controlled by the global font parameters.

Completely different styles can be produced from the same radical and character routines by substituting different stroke routines. It is apparent from [5] that the stroke skeletons are essentially the same in Bold style as in Song style, and many of the differences can be characterized in terms of a few simple rules that are used by the Bold stroke routines. The few differences that remain can be achieved by adding **METAFONT** conditional statements to key radical routines. There is more work to be done on how to describe the skeletal differences between styles, so we only present preliminary results here.

In section 1 we introduce **METAFONT** and LCCD. In section 2 we examine some of the basic stroke drawing routines to see how the style parameters are used. In section 3 we see how to combine the basic strokes into radicals; and in section 4 we discuss the choice of style parameters, adjustments for changing point size, and experiments with new styles. Appendix 1 presents examples of all the basic strokes in the Song, Long Song, and Bold styles. In appendix 2 we show all 128 characters at 10 and 18 points and in an example of actual Chinese text. Finally, in appendix 3 we give a sample of actual **METAFONT** programs for Chinese characters.

1. METAFONT and LCCD

METAFONT is an algebraic language with subroutines, variables, equations, conditional statements, and commands for describing letterforms. Equations are used in a declarative way to define the numerical values of variables and the coordinates of points. **METAFONT** will solve systems of linear equations, keeping track of linear constraints between variables until enough equations are given to determine their values.

Letterforms are actually created with "draw commands" that refer to points whose coordinates have been determined in the above manner. Draw commands work by moving a discrete "pen" along a path through the points and turning on all the pixels covered by the pen. The actual curve used is a piecewise cubic spline. The section of this spline between any two points i and j is defined by

$$\begin{aligned}x(t) &= x_i + (3t^2 - 2t^3)(x_j - x_i)\\ &\qquad + rt(1-t)^2\delta_{ix} - st^2(1-t)\delta_{jx}\\ y(t) &= y_i + (3t^2 - 2t^3)(y_j - y_i)\\ &\qquad + rt(1-t)^2\delta_{iy} - st^2(1-t)\delta_{jy}\end{aligned}$$

for $0 \le t \le 1$ where $(\delta_{ix}, \delta_{iy})$ and $(\delta_{jx}, \delta_{jy})$ are the direction of the spline at points i and j respectively, and r and s are additional parameters that **METAFONT** calculates. **METAFONT** has a rule for determining the directions $(\delta_{ix}, \delta_{iy})$ at each point, but it is possible for the user to give them explicitly, and this is the approach that works best for Chinese characters.

All coordinates given in a METAFONT program are in absolute raster units so that rounding to the nearest integer corresponds to rounding to the nearest pixel. This allows a METAFONT program to make rounding adjustments to help fit the characters to the raster. When a cubic spline is rounded to the raster, the curves look much better if the points where the spline is vertical occur at integer x-coordinates.

Drawing with a circular pen or "cpen" produces a constant width line with rounded ends. METAFONT also has elliptical pens and special pens that can be an almost arbitrary shape, but the most general way to draw a shape is to use the "ddraw" command to specify both sides independently and have METAFONT fill in between them.

Complex mathematical constructions can be performed in METAFONT by taking advantage of subroutines and the ability to solve linear equations. We use such constructions to define the points and directions in various subroutines that draw strokes and other parts of characters. In this way, the subroutines can have a few parameters that control what they draw, and there can also be various global parameters that control overall properties of the font and allow for differences in point size and device resolution. A good illustration of this can be found in [2], where Knuth describes in detail the constructions and parameterizations used in his Computer Modern family of typefaces. See also [7].

METAFONT subroutines have two types of parameters. Index parameters are point numbers from the calling routine and may be used as point numbers in the routine that is being called. Ordinarily, point numbers have purely local significance, but in this way, it is possible to use points that are defined in the calling routine or to define points for use in the calling routines. It is even possible to define a point partly in one routine and partly in another, by giving additional constraints that allow METAFONT to solve for the coordinates. The second type of parameters are arbitrary numeric expressions that should be "known" at the time of the call. These may be used exactly as in any programming language.

Tung Yun Mei's LCCD system for designing Chinese characters [4] is based on METAFONT. It has draw statements, variables, and pens as METAFONT does, but it does not solve implicit equations, and subroutine parameters have a different meaning. LCCD has taken the ability of METAFONT to do affine transforms and incorporated it into subroutines. Each subroutine has transformation parameters that apply to everything it draws, and the transformations are composed when one subroutine calls another. METAFONT, on the other hand, applies a global transformation matrix to each point before actually plotting it.

LCCD makes it very convenient to apply affine transformations to subroutines, but since subroutines are limited to transformation parameters, it is difficult to parameterize subroutine results in any other way. Lack of conditional statements also makes complex constructions very difficult.

Another feature of LCCD is that it has another type of pens called "tear drops," which are intended for drawing dot strokes. However, it is difficult to draw high quality dot strokes of all styles with these.

2. Stroke Drawing Routines

The routines we have constructed are carefully parameterized with just enough information to describe the skeletons of the strokes and how they are joined to adjacent strokes. Mathematical constructions are used to adapt the stroke to the length, orientation, and shape determined by its parameters and the global font parameters. These constructions can get very complex, but since a small number of routines suffice for an entire family of fonts, the time spent writing and debugging them is quite small in comparison to the whole project, even when designing just 128 characters.

The approach suggested in [4] is quite different. Tung suggests that the strokes should be drawn as affine transforms of canonical strokes. Despite its simplicity, this approach has a number of disadvantages. If the canonical stroke is rotated or stretched more than a small amount it acquires an undesirable shape. Examples in [3] show how this problem is solved by having many different versions of each stroke so that it is only necessary to transform them by small amounts. In [3] there are 108 different routines for drawing the strokes referred to here as horizontal strokes, vertical strokes, pie strokes, dot strokes, triangle strokes, f-strokes, and j-strokes. (See appendix 1.) In spite of this, the results obtainable are not as good as with the new method, where we have just a few routines that transform themselves properly.

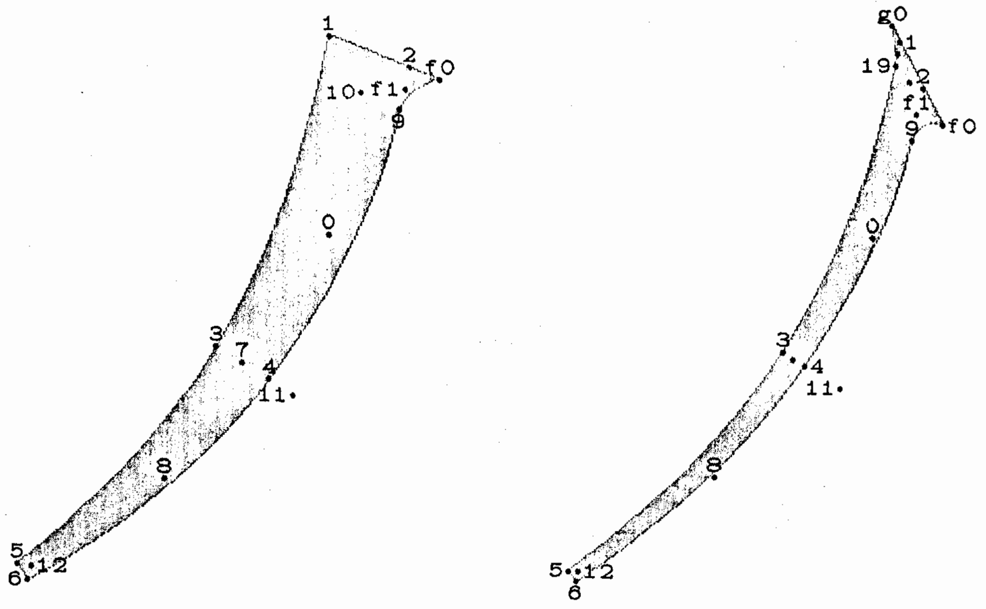


Figure 1. The pie stroke construction in Song style and in Long Song.

2.1 The Pie Stroke

In Song style, the pie stroke is controlled by three point parameters. These are points 10, 11, and 12 in figure 1. The stroke goes from point 10 to point 12, and point 11 gives the initial and final directions: from point 10 it heads toward point 11 and it approaches point 12 from the direction of point 11. Notice that the stroke overlaps points 10 and 12 by an amount equal to half the stroke width. This helps the design transform properly when stroke width changes. The exact location of the curve is determined by the *sharpness* parameter. This is used to determine a point 7 on the skeleton and another point 8 giving the tangent there. The parameter gives the ratio between the distance from point 11 to point 8 and the distance from point 11 to point 12. Similarly, it also determines point 0 where the tangent line crosses the line between points 10 and 11. Point 7 is then located so that the distance between points 7 and 8 divided by the distance between points 0 and 7 is the same as the ratio of the 11-12 distance to the 10-11 distance. The purpose of the *sharpness* parameter is to control how close the stroke gets to point 11. The construction for point 7 tends to place it near point 11 except in extreme cases where this would not yield a smooth curve.

Since there are other tapering curved strokes in Song style, most of the pie stroke is drawn by a separate subroutine. This takes as parameters the three control points for the stroke, the width near point 10, a special *taper* parameter that determines how the width is changing, the slope of the line between points 1 and 2, and the size of the flares on each side of the top part of the stroke. The width of the narrow portion near point 2 is a global style parameter so it does not need to be passed as a parameter. The routine fits a quadratic equation to the width as a function of distance along the stroke to the specified widths and taper. The function is then used to find points on the edge of the stroke. The equation gives the distance between the pairs of points 1 and 2, 3 and 4, 5, and 6. The derivative of the width function determines vanishing points that give the spline directions at each pair of points. The vanishing points are too far away to show in the figure, but they are not hard to calculate. Suppose point 7 is at distance x from point 10. Then the distance between points 3 and 4 is $w(x)$ and and the distance to the vanishing point is $-w(x)/w'(x)$. The direction of the curve at both points 3 and 4 is toward this vanishing point. There are similar vanishing points for points 1 and 2 and for points 5 and 6.

The maximum width of a pie stroke is a linear function of its length, and the coefficients of this function are style parameters. The *taper* that is used for pie strokes is also a global style parameter. It is chosen to be somewhat less than 1 so that the rate of taper at the upper end of the stroke is less than that at the lower end.

The size of the flares at points 1 and f0 are also given by style parameters. These flares are also drawn by a separate routine since similar flares also appear in other strokes. To draw the right side flare we pass points 1, 2, and 11 to the flare routine and let it draw the flare and return the point 9 where the flare stops.

2.2 The Dot Stroke

The dot stroke shown in figure 2 is halfway between the Song style and the Long Song. This should help to explain the effect of the *dotrnd* parameter that is used to interpolate between the two styles. This is a rather extreme example, because most style parameters are not so drastic in their effect or so complex in their implementation.

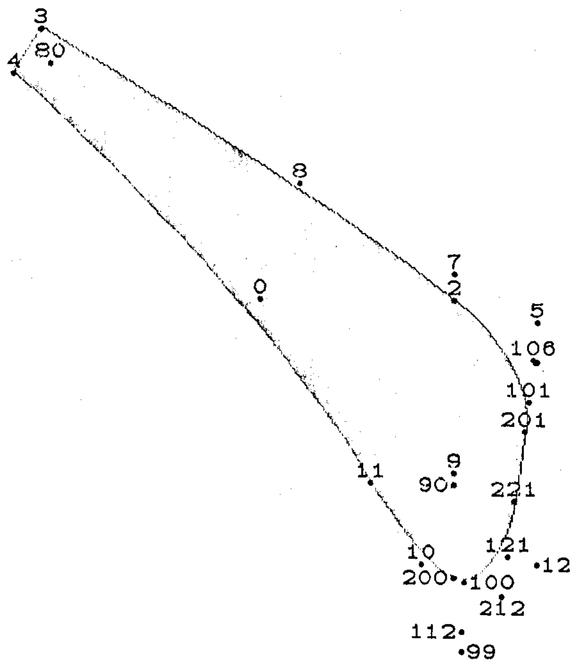


Figure 2. A dot stroke half way between the Song and Long Song styles

The basic construction is very similar to that used in [6], so we will not dwell on it here. Points 80 and 90 are parameters that control the position of the stroke. In ordinary Song style, points 10, 12, and 6 define the lower end of the stroke. Their placement relative to point 90 is fixed except for a scale factor used to control the overall width of the stroke. Lines 10-11-0 and 6-2-8 are tangent to the stroke near the lower end; lines 3-8-7-5 and 4-0-9 are tangent to the stroke at the upper end. Point 8 is a fixed fraction of the way from point 7 to point 3 and point 0 is a fixed fraction of the way from point 11 to point 4. The distance between points 5 and 6 determines the curvature of the stroke and depends on the stroke length and the *dotrnd* parameter. Finally, the distance between points 7 and 9 is fixed in terms of the scale factor that was mentioned previously.

In Long Song style, the end of the stroke should be more triangular than in Song style. Points 106 and 112 are versions of points 6 and 12 that would be more appropriate for the Long Song. Point 112 is on the tangent from point 11 and point 106 is on the tangent from point 2. These points are placed as close to points 10 and 2 as possible without violating a certain minimum separation defined by the style parameter *dotcrv*. Furthermore, we constrain the angle 11, 112, 106 to be 45°. The primary effect of the *dotrnd* parameter is that we place point 212 this fraction of the way from point 12 to point 112 and similarly for point 206 between points 6 and 106 (not shown). The edge of the stroke is tangent to the line between points 206 and 212 in two places and to the line between points 10 and 212 in one place. In ordinary Song style, there would be only one point of tangency between points 206 and 212, but we split it so that there can be a large flat spot here in the Long Song style. The *dotrnd* parameter interpolates between two sets of placements for these points of tangency. Points 100, 101, and 121 are the placements for Long Song where *dotrnd* = 1; the placements for *dotrnd* = 0 are not shown but points 200, 201 and 221 are halfway between these and Long Song placements.

2.3 The Pie Stroke in Bold Style

In Bold style, the pie stroke is controlled by three parameter points exactly as it is in Song style. In figure 3, points 10, 11, and 12 have exactly the same meaning as they do in figure 1, except that the stroke must stop somewhat before point 12 in order to have the same apparent length. For the main body of the stroke essentially the same construction is used here as for the Song style except that the width function is very different. The width appears to be constant over the length of the stroke, but actually, there is significant variation. This is controlled by two style parameters that we shall refer to as δ and σ. The widths relative to a global stroke width parameter for curved strokes are $1+\sigma+\delta/2$ near point 10, $1+\sigma-\delta/2$ near point 12; halfway in between, the width is equal to the parameter. As for the Song style, this determines a parabolic function that gives all the widths and locates the vanishing points.

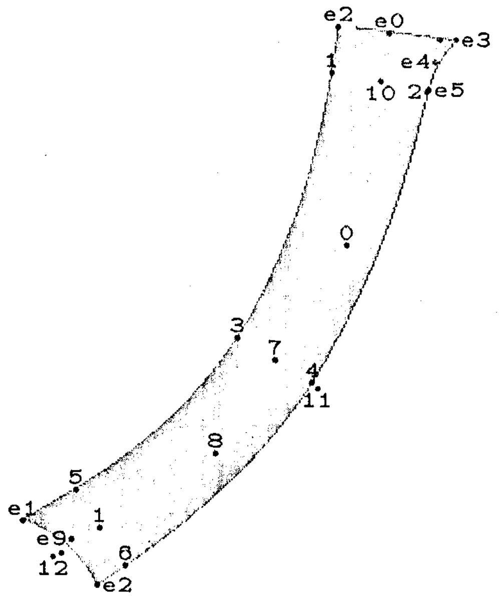


Figure 3. The pie stroke in Bold style

The upper and lower ends of the stroke are drawn by a separate subroutine that draws almost all the stroke ends in Bold style. The routine takes as parameters the stroke end parameter point, the associated vanishing point, the maximum width of the stroke, the widths of the flares on either side, and two more parameters that determine the concavity of the end of the stroke and the angle at which it is cut off. The cutoff angle near point 10 is a linear function of the angle that the 10-11 line forms with the x-axis and similarly the cutoff angle near point 12 depends on the angle of the 11-12 line. The coefficients of these linear functions are determined by style parameters.

3. Combining Strokes into Radicals

The stroke routines are designed to be as easy to combine as possible. In general, we join two strokes by placing their control points in some simple geometric relationship with each other, and by passing additional information to each of them indicating how it must adapt itself. In both Song style and Bold style, the routines have the same names and parameters but their actions are different. We will examine the problem of joining basic strokes together in Song style, since this is the more interesting of the two.

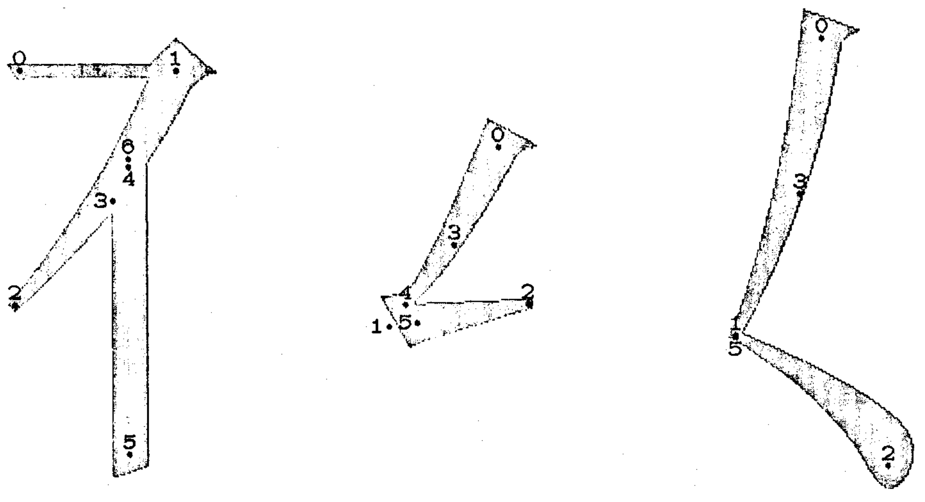
3.1 Positioning Strokes

Appendix 1 shows all the basic stroke routines along with the parameter points that control the position of each. With few exceptions each of the strokes has only enough parameters to determine its overall size and shape. This means that the radical routines only have to determine the placement of the strokes and the stroke routines handle all the other details and produce uniformly good looking strokes.

Certain stroke routines do have extra parameters to allow more generality in special cases and to simplify the process of joining basic strokes. In figure 9, the size and shape of the hooks near points 20 and 26 at the ends of e-strokes and l-strokes are determined by font parameters, so these stroke routines have no control points on the ends of the hooks. For j-strokes, the situation is similar except that the length of the hook often depends on how much room for it there is in the radical. We solve this problem by having point 29 partially determined by the stroke routine and partially determined by the radical. The radical routine sets $x_{26}-x_{29}$ to be a constant times a font parameter or a linear function of the width of the radical.

The curved strokes all use a special guide point parameter to give the initial and final directions of the curve as explained in section 2.1. When the curve is to be symmetrical it is more natural to just give a single number specifying how much the stroke curves, but sometimes a highly asymmetrical curve is desired and this requires the extra degree of freedom that is provided by the guide point. We solve this problem by having a special subroutine to calculate the guide point for symmetrical curved strokes based on the endpoints of the curve and the amount of curvature desired.

The guide point for curved strokes is also used by routines that help join strokes together. Figure 4a shows how a vertical stroke might be joined to a pie stroke. The joining routine takes points 1, 2, and 3 defining the pie stroke and points 4 and 5 defining the vertical stroke, and finds a new point 6 where the line defined by points 4 and 5 crosses the skeleton of the pie stroke. This computation is necessary to insure that the vertical stroke will always touch the pie stroke without crossing through it completely.



(*a*) (*b*) (*c*)

Figure 4. Special subroutines insure proper positioning when basic strokes are joined.

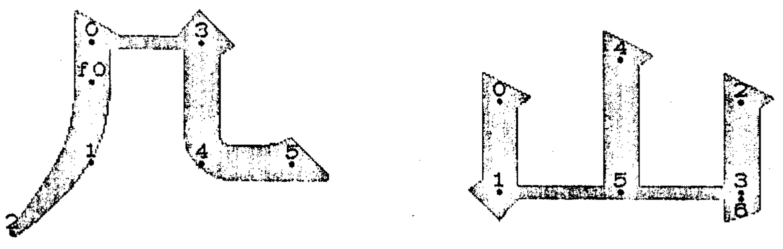
Figure 4*b* shows how a special routine is used to join pie strokes to triangle strokes. Here again, exact positioning is required to insure that the strokes meet without crossing through each other. The position of the join between strokes is controlled by point 1 where the extensions of their skeletons cross. The positioning routine takes points 1, 2, and 3 as arguments and sets points 4 and 5 so that they can be used as control points to the stroke routines. The routine also fills in a small area above point 1 in order to smooth out the corner where the strokes join and to provide an optical correction by thickening the end of the triangle stroke. The amount of this thickening depends on the width of the end of the pie stroke and this in turn is a font parameter that also controls the width of the thin portions of other strokes.

Figure 4*c* shows another stroke combination where positioning is critical. The lower left corner of the pie stroke must exactly match the upper left corner of the dot stroke. A special routine takes points 1, 2, and 3 and finds point 4 on the line between points 1 and 3, and point 5 on the line between points 1 and 2, so that the strokes will join properly if point 4 is used as the control point for the pie stroke and point 5 is used as the control point for the dot stroke. In ordinary Song style, it turns out that points 4 and 5 are almost on top of point 1 so that there is no room to show point 4 in the figure. In other styles this is not the case, but the special routine guarantees that the strokes will always join properly.

3.2 Endpoint Parameters

When strokes are joined together, the ends near the join have to change shape to adapt to the different possibilities. Fortunately, most of the basic strokes can be joined to other strokes only in a very limited number of ways. Most of them have two special parameters that determine how each end is to be joined with surrounding strokes and thus what form it should take. These parameters range over a small set of integral values that we refer to symbolically as *norm*, *join*, *corner*, etc. For most strokes, not all of these values are used and some of those that are used in Song style become synonymous in Bold style and vice versa. In general, the parameters are *norm* for isolated strokes as shown in appendix 1.

The radical in figure 5*a* is constructed from three basic strokes and it illustrates two of the most common types of joins between strokes. The relative positioning of the strokes is very simple: point 0 is passed to the *f*-stroke routine as well as the horizontal stroke routine, and similarly, point 3 is also used by the *l*-stroke routine. The turning feature near point 3 where the horizontal stroke joins the *l*-stroke is handled exactly the same way as similar features where horizontal strokes join vertical strokes, *e*-strokes, and *j*-strokes.



(*a*) (*b*)

Figure 5. Two simple radicals that illustrate the basic ways of joining horizontal and vertical strokes.

It is most convenient to draw the turning feature with a separate routine since it depends on the control points for both the horizontal stroke and the *l*-stroke. An added benefit is that this same routine can be used to draw the similar but slightly smaller feature found near point 1 in figure 5*b*. End point parameters tell the strokes not to draw their usual ending features and cause the *l*-stroke to stop short of point 3 so that it cannot cross outside of the turning feature.

The join between the *f*-stroke and the horizontal stroke is somewhat simpler, since the feature near point 0 in figure 5*a* is part of the *f*-stroke. In fact, this *f*-stroke is almost identical to isolated *f*-strokes. As before, the situation would be equivalent if the *f*-stroke were replaced by any other basic stroke having similar structure on top. The complicating factor is that there is an optical correction to help balance the height of the feature near point 0 with that of the feature near point 3, and this requires the *f*-stroke routine to know the size of the feature. This size depends on the length of the horizontal stroke and whether or not it is joined to another stroke as in the figure, so it would be awkward to provide enough information to the *f*-stroke to enable it to calculate such a quantity. We solve this problem by always drawing the horizontal stroke first and saving the information in a global variable. This is the only place where it is necessary to violate the usual stroke order, and in fact, it is the only place where stroke order matters at all.

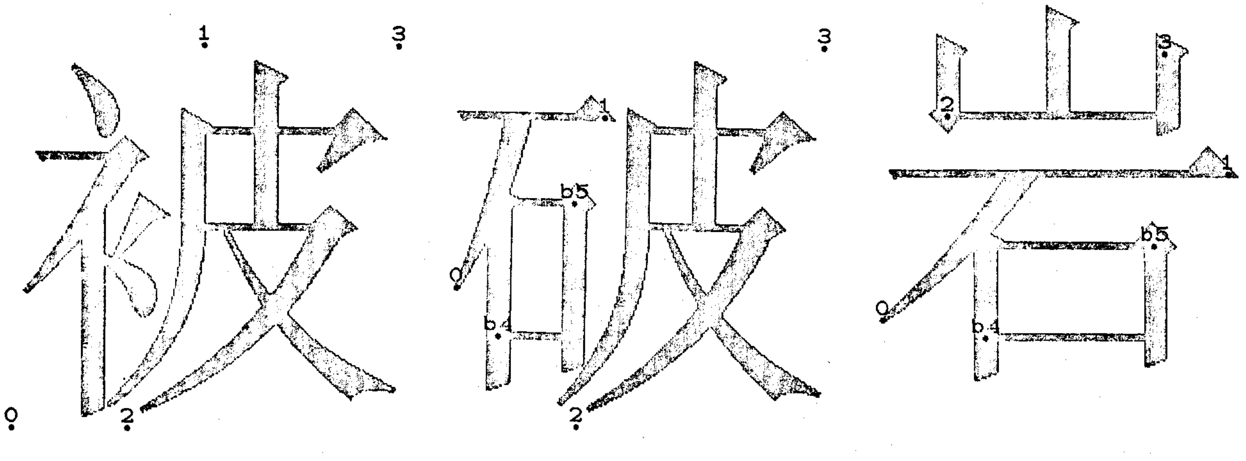
Figure 5*b* illustrates the other main ways in which basic strokes are joined. The feature near point 1 is handled exactly the same way as the similar feature in figure 5*a*, except that the lower end of the vertical stroke must be shortened and the feature is somewhat smaller.

When a vertical stroke is joined to the middle of a horizontal stroke or vice versa, the control point for the abutting stroke should lie on the skeleton line of the stroke being joined. This means that point 5 should lie on the line between points 1 and 3 and that point 3 should lie on the line between points 2 and 6. The special end point parameters are used to tell the routine for the joining stroke not to draw its usual ending feature. In the case of a vertical stroke joining a horizontal stroke as at point 5, the vertical stroke must have a squared off end flush with its control point. This is required because the vertical stroke can be much wider than the horizontal stroke that it joins and we have to guarantee that it will abut properly without crossing the horizontal stroke. The situation would be similar if the vertical stroke were to join the horizontal stroke from below, or even if the vertical stroke were replaced by a pie stroke.

It is also convenient to use end point parameters for variations not related to joining strokes together. For instance, the lower end of the *l*-stroke near point 5 in figure 5*a* is different from the more common version shown in appendix 1. The *l*-stroke routine has a parameter that tells it which form of lower end to draw.

3.3 Radical Routines

Since radicals can change size and shape when they are combined in different ways in different characters, it is necessary to design radical routines that are parameterized to allow this. The basic technique for doing this is to apply a simple geometric transformation to the control points of all the strokes. The size and shape of a radical is controlled by two points that are passed to the radical routine. Typically, these points will be two opposite corners of an imaginary box in which the



(*a*) (*b*) (*c*)

Figure 6. Examples of how radicals change their shape when they are used in different characters.

radical lies. In figure 6*a*, for instance, the left radical is controlled by points 0 and 1 and the right radical is controlled by points 2 and 3. In the radical routines, all coordinates are relative to the box defined by the control points. METAFONT has a convenient syntax for this: for instance, the x-coordinate of the vertical stroke in the left radical is referred to as $.42[x_0, x_1]$ which means $(1 - .42)x_0 + .42x_1$. (Actually, the specification is slightly more complicated than this because of the need for rounding instructions.)

The right radical in figures 6*a* and 6*b* appears to be the same size and shape in both characters, but it is actually 10% narrower in figure 6*b*. Differences of this magnitude are very common in Chinese characters and they can easily by handled by simple geometric transformations as described above.

Sometimes additional corrections are required when the amount of stretching or shrinking in each coordinate is very large. Figures 6*b* and 6*c* show how a radical undergoes such a change. The main radical routine is controlled by points 0 and 1 and it in turn calls a simpler radical that is controlled by points *b*4 and *b*5. Note that the control points are key points on the strokes in the radicals rather than the corners of surrounding boxes. This is basically an arbitrary choice, but it tends to facilitate joining additional strokes onto simple radicals to form more complicated ones. The choice of points 0 and 1 is convenient because it allows all the coordinates except y_{b4} to be specified by METAFONT expressions of the form $\alpha[x_0, x_1]$ or of the form $\beta[y0, y1]$.

The y-coordinate of point *b*4 depends on more than just y_0 and y_1. If we write y_{b4} as a METAFONT expression of the form $\gamma[y_0, y_1]$, we find that $\gamma \approx -.29$ in figure 6*b* and $\gamma \approx -.13$ in figure 6*c*. Another way to look at the problem is that if we use y_{b4} and y_1 for the control box, then points 0 and *b*5 will be too high when a radical designed for figure 6*b* is used in figure 6*c*. The solution we adopt is to make γ a linear function of $(y_1 - y_0)/(x_1 - x_0)$. This is easy to do because of METAFONT's ability to solve implicit linear equations, and the result is a much more flexible radical routine.

4. Font Parameters and Different Styles

Our goal is to have one program that can create a whole family of Chinese fonts by just changing a few parameters. It is desirable to have these parameters relatively free in the sense that, within limits, the parameters can be set arbitrarily and still produce a reasonable font. This is especially difficult for Chinese because it takes several parameters to describe each basic stroke and it is not obvious what relationships have to hold in order for all the strokes to look reasonable and appear as if they belong to the same font. Here, we emphasize the need for sufficient variability while still trying to keep the number of parameters required to a minimum. We have enough parameters to allow us to obtain all three styles from [5], but further study is still necessary to determine exactly what degrees of freedom a Chinese "meta-font" should have.

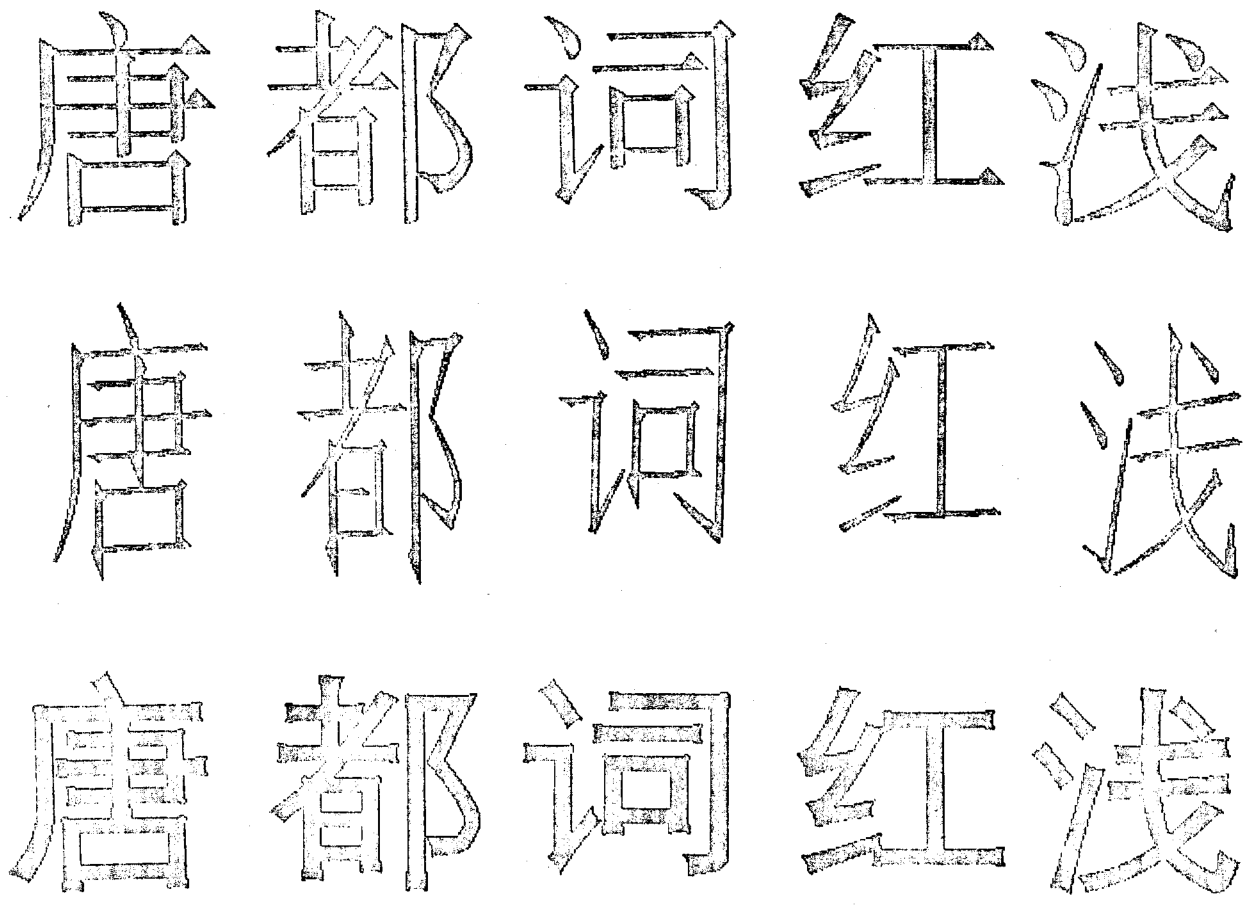


Figure 7. Five characters in Song style, Long Song style, and Bold style.

4.1 Parameterization of the Font

Figure 7 shows how five different characters appear in ordinary Song style, Long Song, and Bold style. The basic positioning of the strokes is almost the same in all three styles, but the strokes themselves are very different. The Song and Long Song style characters in the first two lines all use the same stroke routines, but a completely different set of stroke routines was used for the Bold style characters in the last line. The variation between the ordinary Song style in the first line and the Long Song in the second line is entirely due to changes in the font parameters.

The character shapes are smooth functions of most of the font parameters, although there are two cases where conditional tests are used to produce changes in structure. Notice that the character in the second column of the figure has a horizontal stroke that joins a vertical stroke in the ordinary Song style but not in the Long Song. There is a special *gap* parameter that controls the degree of shortfall in such cases. For the ordinary Song style, this parameter is zero and horizontal strokes join in the usual way. When the end point parameter is set appropriately, the horizontal stroke routine tests this parameter and shortens the stroke by the distance *gap* and draws the usual ending feature.

The characters in the first three columns show the effects of another font parameter that must be treated specially. All three characters have four basic strokes that form a rectangle. We will refer to this combination of strokes as the square radical. In the Long Song style, the horizontal stroke extends beyond the vertical stroke in the lower right corner of this radical, but in the ordinary Song style, the vertical stroke extends beyond the horizontal stroke. This is handled partly by the stroke routines and partly by the radical routines and this is controlled by the font parameter *overshoot*. End point parameters are used to tell the

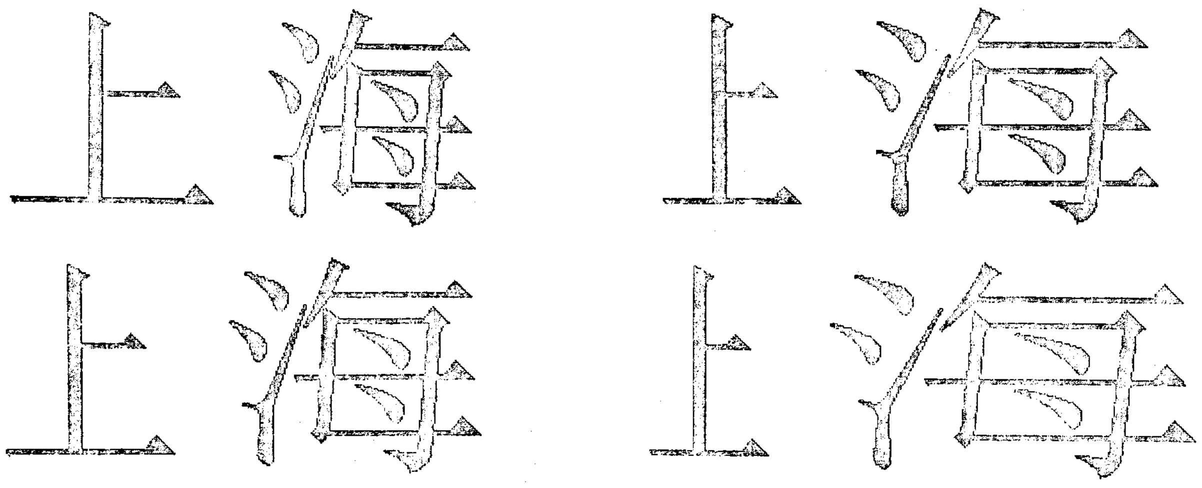


Figure 8. One possible way to experiment with a "meta-font." With each step the first character is compressed 18% and the second one is expanded by 22%.

vertical stroke and horizontal stroke routines that they are joining in this way. In this case, if the *overshoot* parameter is non-zero, the horizontal stroke is lengthened by a distance of *overshoot* and the vertical stroke is cut off at the bottom like the center one in figure 5*b*. If the *overshoot* parameter is zero, then the horizontal stroke is cut off instead. Unfortunately, the radical routine also has to test the *overshoot* parameter because the lower control point for the vertical stroke should be on the horizontal stroke in order for them to join properly in the Long Song style. Very few different radical routines have to make this test, however, because most of them just call the square radical.

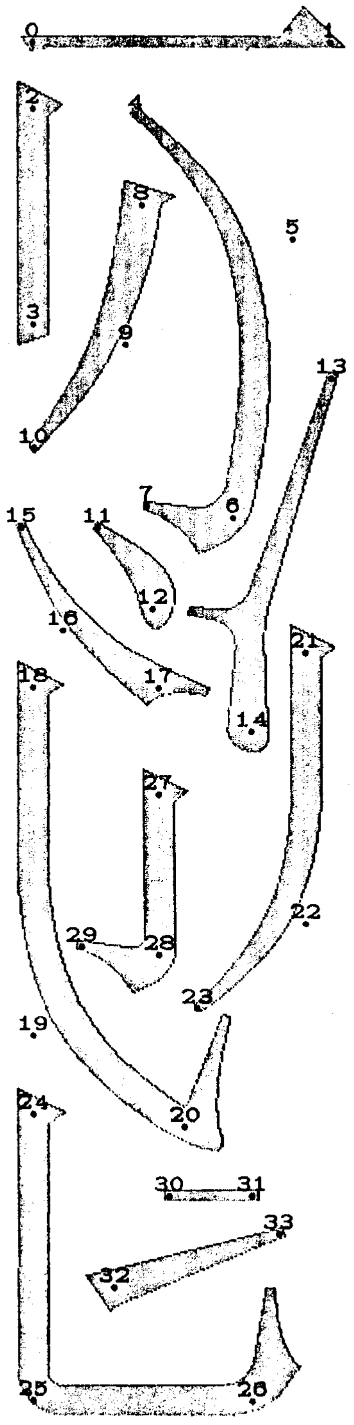
Other font parameters effect stroke widths, the sizes of various features and certain critical angles such as those that control the slopes of the ends of the strokes. Other prominent parameters are the overall height to width ratio of the characters and the slant parameter that makes the horizontal strokes slightly sloped in the Long Song style.

The Bold style stroke routines use a different, smaller set of font parameters. There are fewer special features to control, but stroke widths undergo subtle variations and there are other features such as the concavity of the ends of the strokes and the curvature of the dot strokes. Since we design the radical routines based on the Song style, stroke lengths have to be corrected slightly so that the strokes that are much thinner in the Song style will not appear too long in the Bold style. Figure 9 in appendix 1 shows how many of the Bold style stroke routines do not draw all the way to their ending control points. The magnitudes of these corrections are not true font parameters because they are calculated by the Bold style stroke routines based on the stroke width.

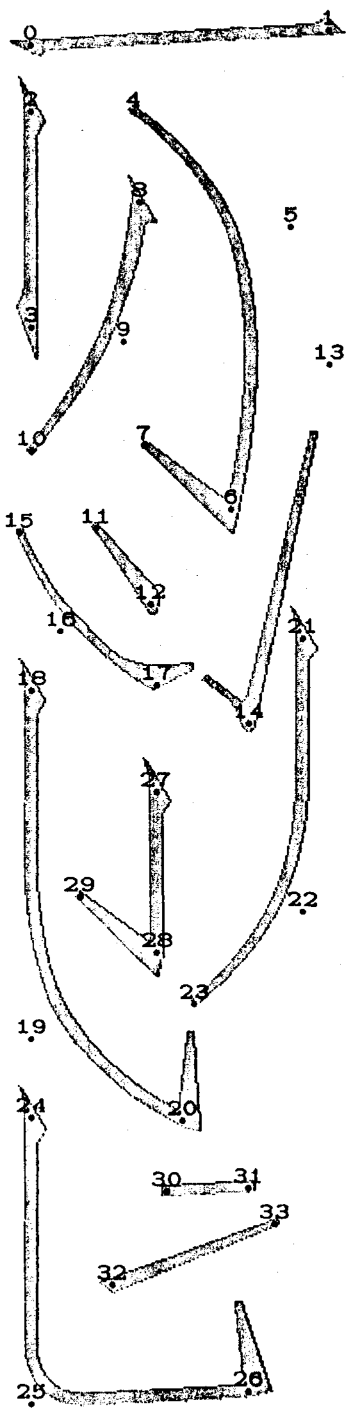
4.2 Adjusting Font Parameters

We have already seen how font parameters can be used to create different styles of characters approximating existing designs. Minor adjustments can made to change qualities like slant and boldness and to augment simple scaling to improve the appearance in small point sizes. It is also possible to experiment with new ideas and even go to ridiculous extremes.

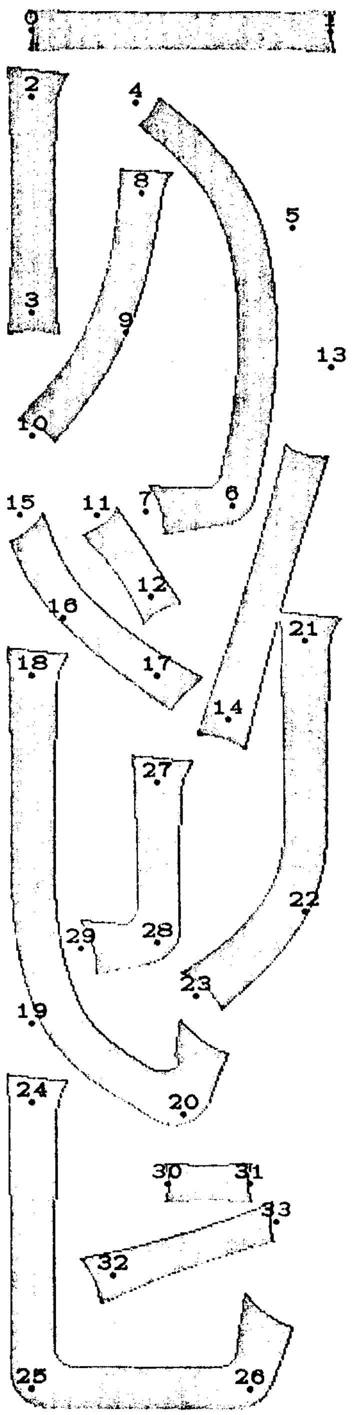
In small point sizes it is desirable to keep stroke widths more uniform so that the thin strokes will not be too hard to see and the thick strokes will not encroach upon the white space too much. In Song style, there are three main parameters that control stroke width and a few more for the thick portions of strokes that vary in width. For the 18 point characters shown in appendix 2 and the large diagrams shown in the other figures, the basic widths of the vertical strokes, horizontal strokes, and the thin parts of tapered strokes are respectively 6%, 2.2%, and 1.8% of the type size. For the 10 point characters, however, we use 6%, 3.3%, and 2.7%. This has the virtue that it tends to correct for the limitations of the printing device. The characters in the appendix were printed on a DOVER printer with a resolution of 384 dots to the inch. This means that for a 10 point font, all the characters are at most 53 pixels high. With the correction the horizontal strokes come out to be two



(*a*) Song



(*b*) Long Song



(*c*) Bold

Figure 9. The basic strokes in all three styles.

pixels wide and the vertical strokes are three pixels wide. Without the correction the horizontal strokes would be only one pixel wide and they would hardly show up at all.

Figure 8 shows one possible way to experiment with a "meta-font." We start with the characters the same width and progressively compress the simpler character while expanding the more complicated one. Note that this is not a simple geometric transformation, but a more complicated one as described in section 3.3 where the stroke widths are preserved and certain adjustments can take place. The stroke routines that cause the ending features on the horizontal strokes to become slightly larger as the character is expanded. It is possible to carry such experiments to great extremes, but milder versions may be desirable in some applications. There are innumerable possible variations to be explored.

5. Conclusion

We have shown how it is possible to use METAFONT to design Chinese characters, and to obtain many different styles from the same program just by changing a few parameters. It is possible to build up a hierarchical structure so that most of the work is not too difficult and the resulting quality can be very high.

The authors are not expert font designers. Although we had access to high quality professional designs, they did not encompass the full range of characters discussed in this work and some judgment is required in order to best adapt them to the new medium. Details such as the exact relative positioning of the radicals could probably be improved. Our goal is to provide the groundwork for further research.

Appendix 1

For convenience, the basic strokes have been given somewhat arbitrary names with unique first letters. Figure 9 shows all the basic strokes and their control points for the ordinary Song, Long Song, and Bold styles.

The strokes on the left in the figure are in ordinary Song style and the Long Song and Bold style versions are shown in the middle and on the right. The correspondence between strokes and control points is as follows: horizontal stroke $(0,1)$, vertical stroke $(2,3)$, a-stroke $(4,5,6,7)$, pie stroke $(8,9,10)$, dot stroke $(11,12)$, k-stroke $(13,14)$, na stroke $(15,16,17)$, e-stroke $(18,19,20)$, f-stroke $(21,22,23)$, l-stroke $(24,25,26)$, j-stroke $(27,28,29)$, bar stroke $(30,31)$, triangle stroke $(32,33)$.

Appendix 2

Figure 10 shows how our fonts might be used in actual Chinese text.

Figure 11 lists all 128 characters in Song style at 10 and 18 points and in Bold style at 18 points. Notice that many radicals appear in several different characters. Each radical is produced by one METAFONT subroutine, and all of the characters using a radical need only call the subroutine. This provides a substantial saving in labor and helps build uniformity into the font.

忆 江 南

白居易

(唐 朝 772–846)

江南好，风景旧曾谙：
日出江花红胜火，春来江水绿如蓝.
能不忆江南!.

Figure 10. An example of actual Chinese text.

(唐宋词一百首　上海古籍出版社)

景 旧 曾 谙 日 出 花 红 胜 火 不 来 蓝 春 水 如
忆 绿 能 白 居 易 唐 朝 宋 司 词 百 首 上 海 古
籍 版 社 一 破 迄 高 纳 德 枕 蜀 都 珞 略 明 膘
意 慓 盆 曼 分 须 杉 幔 缦 镘 漫 熳 谩 慢 鳗 清
请 鲭 情 睛 蜻 锖 倩 残 浅 钱 乾 绍 谁 秒 利 稞
裸 被 衫 钐 裡 袼 褡 锆 褚 浬 理 锂 俚 狸 犹 拘
辅 鸟 纷 粉 粝 堆 唯 呻 吩 咯 鸣 惟 帷 淮 珅 绅
伸 砷 神 祜 汶 沈 捕 哺 埔 棵 枯 赐 江 南 好 风

景 旧 曾 谙 日 出 花 红 胜 火 不 来 蓝 春 水 如
忆 绿 能 白 居 易 唐 朝 宋 司 词 百 首 上 海 古
籍 版 社 一 破 迄 高 纳 德 枕 蜀 都 珞 略 明 膘
意 慓 盆 曼 分 须 杉 幔 缦 镘 漫 熳 谩 慢 鳗 清
请 鲭 情 睛 蜻 锖 倩 残 浅 钱 乾 绍 谁 秒 利 稞
裸 被 衫 钐 裡 袼 褡 锆 褚 浬 理 锂 俚 狸 犹 拘
辅 鸟 纷 粉 粝 堆 唯 呻 吩 咯 鸣 惟 帷 淮 珅 绅
伸 砷 神 祜 汶 沈 捕 哺 埔 棵 枯 赐 江 南 好 风

景 旧 曾 谙 日 出 花 红 胜 火 不 来 蓝 春 水 如 忆 绿 能 白 居 易 唐 朝 宋 司 词 百 首 上 海 古
籍 版 社 一 破 迄 高 纳 德 枕 蜀 都 珞 略 明 膘 意 慓 盆 曼 分 须 杉 幔 缦 镘 漫 熳 谩 慢 鳗 清
请 鲭 情 睛 蜻 锖 倩 残 浅 钱 乾 绍 谁 秒 利 稞 裸 被 衫 钐 裡 袼 褡 锆 褚 浬 理 锂 俚 狸 犹 拘
辅 鸟 纷 粉 粝 堆 唯 呻 吩 咯 鸣 惟 帷 淮 珅 绅 伸 砷 神 祜 汶 沈 捕 哺 埔 棵 枯 赐 江 南 好 风

Figure 11. Three fonts that were created from the same METAFONT program.

Appendix 3

Here, we show some of the actual **METAFONT** code. The dot stroke routine for Song style is an example of one of the most complex stroke routines. Once working, however, it is very easy to use and it provides an enormous degree of flexibility.

First we have some of the support routines from which the elaborate constructions in the stroke routines are built, then we have the stroke routine itself, and finally one of the radical routines that were used to create the last column of figure 7.

```
% Set $slope = (y_i - y_j)/(x_i - x_j)$
% and also find derivatives of arc length with respect to $x$ and $y$
subroutine $fslope$(index $i$, index $j$):
new $slope$; new $dsdy$; new $dsdx$;
if $x_i = x_j$: $slope = 7423.16$;                         % a large random number
    else: if $y_i = y_j$: $slope = 1/7423.17$;
        else: $slope = (y_i - y_j)/(x_i - x_j)$;
    fi;
fi;
    $dsdx = \mathbf{sqrt}(1 + slope \cdot slope)$;
    $dsdy = \mathbf{sqrt}(1 + 1/(slope \cdot slope))$.

% Set $dist$ to the distance between points $i$ and $j$, $sqrdist$ to the square
% of the distance, and also set $dx$ and $dy$ to the $x$ and $y$ components.
subroutine $fdist$(index $i$, index $j$):
new $dx$, $dy$, $dist$, $sqrdist$;
    $dx = x_j - x_i$;
    $dy = y_j - y_i$;
    $sqrdist = dx \cdot dx + dy \cdot dy$;
    $dist = \mathbf{sqrt}\ sqrdist$.

% Specify that point $k$ is distance $d$ to the right of the line from $i$ to $j$.
% Points $i$ and $j$ should be known.
subroutine $dtoright$(var $d$, index $i$, index $j$)(index $k$):
call $fdist(i, j)$;
    $x_k \cdot (y_j - y_i) + y_k \cdot (x_i - x_j) = dist \cdot d + x_i \cdot (y_j - y_i) + y_i \cdot (x_i - x_j)$.

% Make a square end of width $d$ near $i$ for a stroke heading toward $j$. Facing from
% $i$ to $j$, $l$ is in the left and $r$ is on the right.
subroutine $sqend$(var $d$, index $i$, index $j$)(index $l$, index $r$):
no proofmode;
call $toward(-(d-1)/2, i, j, 0)$;
call $right(j, 0, l)$;
call $right(j, 0, r)$;
call $dtoright((d-1)/2, j, i, l)$;
call $dtoright((d-1)/2, i, j, r)$.

% Find point $k$, distance $d$ of the way from $i$ to $j$
subroutine $toward$(var $d$, index $i$, index $j$)(index $k$):
call $fdist(i, j)$;
    $x_k = x_i + (d/dist) \cdot dx$;
    $y_k = y_i + (d/dist) \cdot dy$.
```

```
% Specify that point $k$ is on the line between points $i$ and $j$, which should be known.
**subroutine** $online$(**index** $i$, **index** $j$)(**index** $k$):
    $x_k \cdot (y_j - y_i) + y_k \cdot (x_i - x_j) = x_i \cdot (y_j - y_i) + y_i \cdot (x_i - x_j)$.

% Find point $r$ at distance "$dist$" from $k$ so that the following lines from points
% $i$ and $r$ will have length ratio "$rat$." The lines will be tangents of a curve at
% $i$ and $r$. The tangent from $i$ passes through $j$ and that from $r$ passes through $k$.
**subroutine** $fspoint$(**index** $i$, **index** $j$, **index** $k$, **index** $r$, **var** $dist$, **var** $rat$):
**no proofmode**;
**call** $intersect(k, i, j, 1)$;
**new** $bsqr$; $bsqr = (x_i - x_1) \cdot (x_i - x_1) + (y_i - y_1) \cdot (y_i - y_1)$;
**new** $csqr$; $csqr = (x_k - x_1) \cdot (x_k - x_1) + (y_k - y_1) \cdot (y_k - y_1)$;
**call** $fslope(i, j)$;
**new** $tmpa$; $tmpa = dist \cdot rat + \mathbf{sqrt}\, bsqr$;
**new** $tmpb$;
    $tmpb = (\mathbf{sqrt}(tmpa \cdot tmpa + (rat \cdot rat - 1) \cdot (bsqr + csqr - dist \cdot dist)) - tmpa)/(rat \cdot rat - 1)/dsdy$;
    $y_0 - y_i = (x_0 - x_i) \cdot slope$;
**if** $y_j > y_i$: $y_0 - y_i = tmpb$;
    **else**: $y_i - y_0 = tmpb$;
**fi**;
**call** $fslope(0, k)$;
    $y_r - y_k = (x_r - x_k) \cdot slope$;
**if** $y_0 > y_k$: $y_r - y_k = dist/dsdy$;
    **else**: $y_k - y_r = dist/dsdy$;
**fi**.

% This draws a dot stroke compromising between two styles and changes point $n$
% by rounding its $x$-coordinate.
**subroutine** $dotstroke$(**index** $n$, **index** $t$):
**new** $tmpf$;
**if** $x_n < x_t$:
    **call** $fdist(t, n)$;
    $y_5 - y_6 = dotrnd \cdot 0.13 \cdot dist$;
    $y_{10} = y_{12} = \mathbf{round}(y_t - (.4 \cdot cf \cdot dotw - 1/2))$;
    $x_5 = x_6 = x_{12} = \mathbf{round}(x_t + (.4 \cdot cf \cdot dotw - 1/2))$;
    $y_6 - y_{12} = cf \cdot dotw - .5$;
    $x_{12} - x_{10} = 0.56 \cdot cf \cdot dotw - .5$;
    $x_2 = x_7 = x_t$;
    $tmpf = .4$;
**else**:
    **call** $roundx(n, 1)$;
    $x_5 - x_{66} = 0.13 \cdot (y_n - y_t)$;
    $x_{10} = x_{12} = \mathbf{round}(x_t - (0.4 \cdot cf \cdot dotw - 1/2))$;
    $y_5 = y_6 = y_{12} = \mathbf{round}(y_t - (0.4 \cdot cf \cdot dotw - 1/2))$;
    $x_{66} - x_{12} = cf \cdot dotw - .5$;
    **call** $fslope(t, n)$;
    $x_6 = ((.6 slope + 2)/(slope + 2))[x_{12}, x_{66}]$;
    $y_{10} - y_{12} = 0.56 \cdot cf \cdot dotw - .5$;
    $y_2 = y_7 = y_t$;                                   % at $slope > 2$, 6 moves 0.4 of the way to 12
    $tmpf = .54$;
**fi**;
```

```
call $sqend(w1, n, 6, 3, 4)$;                                 % 6 is almost the right direction
call $online(3, 5, 7)$;
    $x_8 = .37[x_7, x_3]$;
    $y_8 = .37[y_7, y_3]$;                                    % 6 to 8 is tangent on top curve
call $online(6, 8, 2)$;                                       % 2 is the tangent point
call $toward((dotrnd[1, .57])(y_2 - y_{12}) - 1, 7, t, 9)$;   % 9 is 4 tangent (divergence near $n$)
call $fspoint(4, 9, 10, 11, .48 \cdot cf \cdot dotw - .5, .64)$;   % 11 is point of inflection
if $x_n < x_t$:
    $x_{111} = x_{11}$; $y_{111} = y_{11}$;                   % we are not going to move point 11
    call $toward(cf \cdot dotcrv, 2, 6, 106)$;                % 106 is new version of 6
    call $toward(-cf(dotcrv + .19dotw), 10, 11, 112)$;        % 112 is new version of 12
    call $sin(106, 112, 111)$;
    if $acc >$ (sqrt .5): new $x_{112}, y_{112}$;             % free either 106 or 112 to move out
        call $online(10, 11, 112)$;
    else: new $x_{106}, y_{106}$;
        call $online(2, 6, 106)$;
    fi;
    $(x_{10} - x_{11} + y_{11} - y_{10})(x_{106} - x_{112})$
     $+ (x_{10} - x_{11} + y_{10} - y_{11})(y_{106} - y_{112}) = 0$;   % make 106 112 10 a 45 degree angle
    $x_{206} = dotrnd[x_{106}, x_6]$; $y_{206} = dotrnd[y_{106}, y_6]$;       % 206 is compromise version of 6
    $x_{212} = dotrnd[x_{112}, x_{12}]$; $y_{212} = dotrnd[y_{112}, y_{12}]$; % 212 is compromise version of 12
    call $toward(cf \cdot dotcrv, 212, 10, 100)$;             % 100 is new version of 0
    $x_{120} = x_{100}$; $y_{120} = y_{100}$;                 % 120=100 since point 0 doesn't split
    call $toward(cf \cdot dotcrv, 212, 206, 121)$;
    call $toward(cf \cdot dotcrv, 206, 212, 101)$;            % 101 and 121 are new versions of 1
else:
    call $toward(cf \cdot dotcrv, 10, 11, 111)$;              % 111 is new version of 11;
    call $intersect(10, 6, 8, 50)$;
    call $fdist(10, 50)$;
    call $toward(-dist, 50, 8, 112)$;                         % 112 is new version of 12
    $x_{206} = x_6$; $y_{206} = y_6$;                         % make 206 same as the original 6
    $x_{212} = dotrnd[x_{112}, x_{12}]$; $y_{212} = dotrnd[y_{112}, y_{12}]$; % 212 is compromise version of 12
    call $toward(cf \cdot dotcrv, 10, 212, 100)$;
    call $toward(cf \cdot dotcrv, 212, 10, 120)$;             % 100 and 120 are new versions of 0
    call $toward(cf \cdot dotcrv, 212, 206, 101)$;            % 101 is new version of 1
    $x_{121} = x_{101}$; $y_{121} = y_{101}$;                 % 121=101 since point 1 doesn't split
fi;
    $x_{200} = dotrnd[x_{100}, (2/7)[x_{10}, x_{212}]]$;
    $y_{200} = dotrnd[y_{100}, (2/7)[y_{10}, y_{212}]]$;
    $x_{220} = dotrnd[x_{120}, (2/7)[x_{10}, x_{212}]]$;
    $y_{220} = dotrnd[y_{120}, (2/7)[y_{10}, y_{212}]]$;      % 200 and 220 are compromises for 0
    $x_{201} = dotrnd[x_{101}, tmpf[x_{206}, x_{212}]]$;
    $y_{201} = dotrnd[y_{101}, tmpf[y_{206}, y_{212}]]$;
    $x_{221} = dotrnd[x_{121}, tmpf[x_{206}, x_{212}]]$;
    $y_{221} = dotrnd[y_{121}, tmpf[y_{206}, y_{212}]]$;      % 201 and 221 are compromises for 1
    1 ddraw $3\{x_8 - x_3, y_8 - y_3\} \mathinner{..} 2\{x_6 - x_2, y_6 - y_2\} \mathinner{..} 201\{x_2 12 - x_2 01, y_{212} - y_{201}\}$
    $\mathinner{..} 221\{x_{221} - x_{206}, y_{221} - y_{206}\} \mathinner{..} 221,$
    $4\{x_9 - x_4, y_9 - y_4\} \mathinner{..} 111\{x_{10} - x_{111}, y_{10} - y_{111}\} \mathinner{..} 200\{x_{212} - x_{200}, y_{212} - y_{200}\}$
    $\mathinner{..} 220\{x_{220} - x_{10}, y_{220} - y_{10}\} \mathinner{..} 221\{x_{206} - x_{221}, y_{206} - y_{221}\}.$
```

```
subroutine rzhe(index ll,index ur):                                % 70% of x-side
$x_0 = (19/70)[x_{ll}, x_{ur}]$;                $y_0 = 0.66[y_{ll}, y_{ur}]$;
$x1 = (62/70)[x_{ll}, x_{ur}]$;                 $y1 = 0.71[y_{ll}, y_{ur}]$;
$x_2 = (16/70)[x_{ll}, x_{ur}]$;                $y2 = 0.50[y_{ll}, y_{ur}]$;
$x_3 = (64/72)[x_{ll}, x_{ur}]$;                $y3 = 0.55[y_{ll}, y_{ur}]$;
$x_4 = \mathbf{good}_3(32/70)[x_{ll}, x_{ur}]$;   $y4 = 0.91[y_{ll}, y_{ur}]$;
$x_5 = \mathbf{good}_3(33/70)[x_{ll}, x_{ur}]$;   $y5 = 0.26[y_{ll}, y_{ur}]$;
$x_6 = (60/70)[x_{ll}, x_{ur}]$;                $y6 = 0.09[y_{ll}, y_{ur}]$;
$x_7 = (54/70)[x_{ll}, x_{ur}]$;                $y7 = 0.42[y_{ll}, y_{ur}]$;
$x_8 = (5/70)[x_{ll}, x_{ur}]$;                 $y8 = 0.09[y_{ll}, y_{ur}]$;
$x_9 = (41/70)[x_{ll}, x_{ur}]$;                $y9 = 0.90[y_{ll}, y_{ur}]$;
$x_{10} = (52/70)[x_{ll}, x_{ur}]$;             $y10 = 0.78[y_{ll}, y_{ur}]$;
call roundy(0, 2);
call roundy(1, 2);
call roundy(2, 2);
call roundy(3, 2);
call 'h hstroke(0, 1, norm, norm);
call 'h hstroke(2, 3, norm, norm);
call 'e estroke(4, 5, 6, norm);
call bendpie(7, 11, 8, 0.15);
call 'p piestroke(7, 11, 8, 7, norm, norm);
call 'd dotstroke(9, 10).
```

References

1. Knuth, Donald E., *TEX and* **METAFONT**, *New directions in typesetting*, Digital Press and the American Mathematical Society, 1979.
2. Knuth, Donald E., *The Computer Modern Family of Typefaces*, Stanford Computer Science Report STAN-CS-80-780 (January 1980).
3. Tung Yun Mei, *LCCD, A Language for Chinese Character Design*, Stanford Computer Science Report STAN-CS-80-824 (October 1980).
4. Tung Yun Mei, LCCD, A Language for Chinese Character Design, *Software Practice and Experience* **11** (December 1981), 1273–1292.
5. Unpublished Chinese character designs, *Shanghai Printing Technology Institute.*
6. Gu Guoan and Hobby, John D., Using **METAFONT** to Design Chinese Characters, *Computer Processing of Chinese and Oriental Languages* **1** (July 1983), 4–23.
7. Knuth, Donald E., The Letter S, *The Mathematical Intelligencer* **2** (1980).

Output Devices

Output Devices and Computers

Table I: Proof-Quality Devices

	Canon LBP-10	Diablo 630	Epson MX-80	Facit 4542	Fla. Data OSP	GE3000	HP2680	Imagen Imprint 10	Laser-grafix	Qume Sprint 5	screen prevue	Symbolics LGP-1	Varian	Versatec	Xerox Dover	Xerox 9700
Amdahl (MTS)								U. British Columbia								Univ. Michigan
Apollo						COS Info.		OCLC	Scan Laser		Yale			OCLC		COS Info
CDC Cyber														U. Köln		
DEC10								Stanford; Vanderbilt	Talaris					GA Techn; Vanderbilt		Univ. Delaware
DEC20					Math Reviews			SRI; Columbia	Talaris			Univ. Wash.	AMS	Univ. Wash.	CMU	
DG MV8000									Texas A&M							
Ethernet							Stanford	Imagen							Stanford	
HP1000			JDJ Wordware													
HP3000		TeXeT					TeXeT			TeXeT						
IBM (MVS)											GMD Bonn			U. Milan		CIT
IBM (VM)								SLAC						SLAC; Weizmann		Univ. Delaware
Prime									Texas A&M					Livermore		
Siemens BS2000	GMD Bonn										GMD Bonn					
Sun					Textset			Sun Inc.			Textset					Textset
VAX (Unix)								UC Irvine	Talaris			U. Wash.		U. Wash.	Stanford	
VAX (VMS)				INFN CNAF				Kellerman & Smith †	Texas A&M			Calma	Sci. Appl.	Kellerman & Smith †		

Notes:

* Still running TeX80

† Graphics supported

Most of the interfaces listed here are not on the standard distribution tapes. Some of them are considered proprietary. Information regarding these interfaces should be obtained directly from the sites listed.

Table II: Typesetters

	Agfa P400	Alphatype CRS	APS-5/Micro-5	Compugraphic 8400	Compugraphic 8600	Harris 7500	Linotron 202
Amdahl (MVS)*			Wash. St. U.*		Wash. St. U.*		
Apollo			COS Info.				
CDC Cyber *					RECAU*		
DEC 20		AMS	Textset				Adapt, Inc.
HP3000				Univ. Sheffield			
IBM 370 *			Info. Handling*				
IBM (VM)	IAM, U. Bonn						
Sun			Textset				
Univac 1100 *					U. Wisconsin*		
VAX (UNIX)						SARA	
VAX (VMS)			Intergraph †	K & S †			

Output device data is being maintained by Rilla Thedford. Anyone desiring more information or relaying new information can send it to her on the Arpanet:

`Rilla_Thedford%UMich-MTS@MIT`

Index to Sample Output from Various Devices

Camera copy for the following items in this issue of TUGboat was prepared on the devices indicated, and can be taken as representative of the output produced by those devices. The bulk of this issue, as usual, has been prepared (all with TEX82) on the DEC 2060 and Alphatype CRS at the American Mathematical Society.

- Autologic APS-5 (1440 dpi): font and border samples which appear in Donald E. Knuth, A course on **METAFONT** Programming, p. 105 ff. and elsewhere; also, the Textset advertisement, p. 159.
- Canon CX (300 dpi): The Metafoundry advertisement, p. 160.
- HP 2688A Laser Printer (300 dpi): Lance Carnes, *"small TEX"*; HP 3000.
- QMS Lasergrafix 1200 (300 dpi): QMS and Talaris advertisements, pp. 156–158.
- Versatec (200 dpi): Carlos A. Felippa, *Feedback from TEX users at Lockheed*, p. 143; VAX 11/780 (VMS).
- Xerox Dover (384 dpi): figures of Chinese characters in John Hobby and Gu Guoan, *A Chinese Meta-Font*, p. 119.

Dvi_QMS: An Example of a Driver

Tom Rokicki
Texas A&M University

This is a description of a `.dvi` driver implemented at Texas A&M University under VAX/VMS for the QMS-1200 and QMS-800 laser printers. Hopefully others will benefit from our experiences and ideas.

The code started as DVI_TYPE, and was modified by Bart Childs to drive the QMS-1200 from a Data General MV8000. Norman Naugle and I took this program and modified it to work under VMS. As I did not know WEB or TEX at the time, and the pressure was on to get a working driver, I modified the Pascal output of TANGLE. This initial program slowly evolved until now it supports dynamic as well as static downloading of fonts, graphics integration, legal size and notebook size paper, and both landscape and portrait modes of operation. It has been used rather extensively here at A&M, and has performed well.

Operation of the program is straightforward. After executing TEX and creating a `.dvi` file, users type in the command

```
$ dviqms filename
```

which then creates a file with the extension `.bit`. This file contains all the necessary commands for the laser printer to print the file, including font downloading, rule commands, character positioning, font selection, raster images, and, of course, the characters themselves. This file is then submitted to the print queue, which produces the final document.

The key to the speed of Dvi_QMS is the font downloading. There is a set of fonts that is assumed to always be downloaded—at our installation this list consists of fifteen fonts such as roman and italics at three different sizes—and is assumed to contain the fonts most commonly used. This file can be changed at any time. This list of fonts still leaves approximately seventy kilobytes of memory in the laser printer for the EUNICE `troff` fonts and dynamic downloading of TEX fonts. If a character from one of the listed fonts is used in a document, Dvi_QMS will not download or bit map it; rather, the font select and character code are sent to the laser printer directly. Of course, if the laser printer is powered down or confused, these fonts need to be re-downloaded to run TEX. This takes approximately five minutes and is a small price to pay for the resultant speed.

Since it is impossible to know which fonts a user might need, Dvi_QMS will dynamically download characters from the additional fonts on a per job basis. To do this, Dvi_QMS does a pre-scan of the `.dvi` file to determine the font usage. Based on this usage and the laser printer memory required by each font, a desirability index is calculated. Fonts are then downloaded on a priority basis as the remaining memory in the laser printer allows. Any additional characters used are bit-mapped. This technique introduces a significant reduction in both the size of the output file and the execution time of Dvi_QMS.

A true font-caching scheme was considered, but rejected for the following reasons. What we wanted was a program that would work on any VAX/VMS system with the QMS printer, regardless of other software. Secondly, the necessary interlock files and direct queue control necessary in a caching scheme would have been more difficult on a system where either several other packages make extensive use of the laser printer, or many users run TEX simultaneously. I feel that the techniques used in `Dvi_QMS` result in a more efficient scheme.

In `Dvi_QMS` I have added several options available through the `\special` primitive of TEX. Since these options are directly related to the formatting of the document, I felt they should reside in the TEX source rather than be `Dvi_QMS` run-time options.

The first of these extensions is the `\special{landscape}` option. This tells the driver that the output should be printed with the text parallel to the long side of the page. Of course, the `\hsize` and `\vsize` need to be reset to the appropriate parameters. The `\special{long}` option will instruct the driver to use legal sized (14-inch long) paper rather than the normal size. Again, the `\vsize` needs to be adjusted.

The `\special{include` *filespec*`}` command takes a file and includes it directly into the `.bit` file produced by `Dvi_QMS`. This file often consists of raster commands or vector commands output from another graphics package. Alternatively, it can be a simple company logo created by hand. Surrounding the `\special` with a `\hbox` with the appropriate dimensions in a TEX macro makes the use failsafe.

There is also a facility for including low-level commands for the printer directly in the source text. In the current implementation of `Dvi_QMS`, you would type `\special{'`*quic commands*`'}`. This allows you to create macros that draw vectors or circles on the page from the TEX level. There is also a `continue` option which will defeat the re-positioning at the end of the `\special`, to allow a long low-level instruction sequence to be created by several different `\special`s.

The program allows the starting page, total number of output pages, and number of copies to be specified as options on the calling line. This way, a single page can be output for review, or twenty copies of a document can be created without submitting twenty copies of the file into the queue.

`Dvi_QMS` supports both the QMS-800 and QMS-1200 laser printers. This selection will default, but can be overriden on the command line.

`Dvi_QMS` typically creates `.bit` files approximately three times as large as the `.dvi` file. This expansion is due to the fact that the output file is a straight ascii file, with all the positioning in decimal and raster information in hexadecimal. Any fonts dynamically downloaded are also included. For typical text, using mostly roman type in normal, magstephalf or magstep1, the running time is about half that of TEX. For files like those created by `WEB`, the execution is about 65% of TEX. Math adds a little more, but seldom does `Dvi_QMS` take more than half of the total time required to process a document.

Currently I am working on recoding the program in `WEB` to make it available on most other machines. A modified version of `Dvi_QMS` is being written that will be capable of driving most bit-mapped devices. Some GKS graphics primitives are being considered. TEX at TEXas A&M University will continue to thrive.

`\special`

Tom Rokicki
Texas A&M University

I am constantly deluged by requests for assistance with TEX from people just starting to use the Electrical Engineering VAX at A&M. I generally start them off with an analogy between TEX and compiled programming languages. TEX compiles the source, which is then 'linked' with the pixel files into a form understandable by the printer. The compiler will catch the programming errors, and the linker should be as unobtrusive as possible. Similarly, the `.dvi` driver should be very quiet, requiring as little attention from the user as possible.

It often becomes necessary, however, to use some particular features of the local output device, or to include some graphics which TEX is either incapable of or not very adept at producing. To this end, the primitive `\special` has been made a part of TEX. This command has the form

`\special{`*token list*`}`

where *token list* is expanded immediately. (Of course, a string of alphabetic characters is a token list.) Typically, the use is of the form

`\special{`*keyword arguments*`}`

Here, *keyword* is some sort of command, requiring optional *arguments*, that would be recognized and acted upon by the `.dvi` driver. As an example, the *landscape* keyword is recognized by the driver here at A&M to print the text parallel to the long edge of the paper.

Why should the uses of \special concern us? After all, only the few of us who write .dvi drivers will be able to implement uses for \special. Also, most uses of \special do not relate directly to TEX itself, but are primarily concerned with integrating other systems (especially graphics) with TEX. Finally, \special was intended as an arena for experimentation—the worst we could do would be to limit the possibilities by imposing a set of rules.

The uses of \special are of concern for several reasons. TEX source should remain compatible system to system as much as possible for obvious reasons. \special opens a Pandora's box of incompatibility. In addition, \special allows expansion of TEX to include such things as graphics in a system-independent way, if a convention can be agreed upon. Also, .dvi drivers should be written to handle foreign \special's in a reasonable fashion.

Additionally, implementing \special commands is quite easy, once it is realized that they can be used to solve a particular problem. A common example of this is the ability to include graphics in TEX that were created by a different software package. This is typically done with a \special command which inserts the output of the graphics package into the output created by the .dvi driver, insuring that the positioning of the graphics is done by TEX. For devices with the ability to change the coordinate reference, this is quite trivial on the part of the .dvi driver. It is definitely a lot easier than trial-and-error positioning or running the paper through the output device twice—once for the TEX and once for the graphics.

So, experiment with \special. Implement your own extensions or ask the local gurus about \special. Some basic implementation guidelines are obvious: ignore foreign \special's; do not make the argument a lone file name. Some other guidelines have been suggested: only put one command or group of related commands in each \special; model high-level graphics extensions after an existing standard (GKS); use a Pascal-type approach to keywords—*keyword (arg1, arg2...)*. The key here is to experiment. Perhaps by the next Users Group meeting we will have enough input and ideas to attempt some sort of standardization.

Read about A&M's driver and how it uses \special (in this issue, page 138). Let us know about your own uses or ideas. I can be reached through CSnet (or ARPAnet) at:

ROKICKI @ TAMU.CSNET

or through the post:

Tomas Rokicki
Electrical Engineering
Texas A&M University
College Station, TX 77843

Thanks to Calvin Jackson of Cal Tech, Albert B. Meador of PAR Technology, Inc., and Mark Blanford of RE/SPEC for their valuable ideas and encouragement.

Site Reports

News From the TeX Project

David Fuchs

Not too much to say this time around; all the exciting news will come next time. We do have a fairly stable version of Metafont now being distributed (0.3), and various folks have it running on Vax/VMS, Tops-10, etc. The big problem is lack of documentation; as I write this, Prof. Knuth is working on chapter 4 of The Metafontbook. Along with the new Metafont comes a "trap" test and a set of "plain.mf" macros (akin to TeX's "trip" test and "plain.tex" macros, respectively). There's also the GFtype program that deals with Metafont's "generic font" output format, and the GFtoPXL program (based on GFtype) that converts Metafont output to PXL format. For proof-mode output, there's the GFtoDVI program that produces DVI files that use the large pixels in the GRAY font to create blown-up images of characters under development. The Metafont sources to the GRAY font, plus the sources for FONT1 (the first font created with the new Metafont, one or two letters by each student in last summer's Metafont class) are also included.

Keep an eye out for the second printing of *The TeXbook*. It corrects all known errors in the first printing (well, *almost* all).

TeX Users' Activity in Germany

Bernd Schulze reported in late September that he expected about 100 people to attend a meeting in Mannheim on October 5, the third of a series of meetings for "TeX-Interessenten" in Germany. He enclosed a copy of his mailing list, which contains information on about 70 sites where TeX is either in operation, in the process of being installed, or under serious consideration.

He also identified the German coordinators for the following hardware and operating systems:

BS 2000	Michael Schrapp, Univ. Karlsruhe*
BS 3000	Georg Heygster, Univ. Bremen
IBM/MVS	Ferdinand Hommes, GMD, Bonn*
IBM/VM-CMS	Bernd Schulze, Univ. Bonn*
VAX/VMS	Peter Kobe, Univ. Bonn
CDC	Jochen Roderburg, Univ. Köln*

(Asterisks indicate TUG members.)

Persons in Germany who wish to be added to the mailing list are invited to contact
Bernd Schulze Institute for Applied Mathematics
Wegeler Str. 6 D-5300 Bonn, W. Germany

Prime Site Report

John Crawford
Ohio State University

TeX 1.0 is now running on Prime computers.

The conversion provided a very enjoyable exercise in program development and testing. TeX and the various TeX tools were compiled using the University of Sheffield's Pascal compiler. A TeX development system was designed to take the TeX Web and Primos change files, ultimately producing TeX, IniTeX, and TripTeX program modules. The tangled TeX file demanded some automated postprocessing in order to conveniently divide the Pascal program into a few tasty subprograms. Trip validation was performed as the final step in this system.

I have sent my port down to Bart Childs and Riley Rainey at Texas A&M, where Riley reports a successful Prime to QMS Lasergrafix 1200 interface. Locally, I hope to have a laser printer on hand soon. I have been rather frustrated in not having any suitable output drivers for my currently available printing machines. (The thrill of reading DVItype output has long since vanished.) When time allows, I hope to write (or obtain) DVI drivers for the TI855 matrix and Diablo 630 printers, and possibly the Printronix 300 printer. On the other end of the scale, Linda Woessner reports some early interfacing success with an Autologic APS Micro-5 phototypesetter at her site. More on this and other device news later.

I hope to get the new TeX generic tape soon, and expect to have TeX 1.1 available shortly afterwards. With luck, the latest version of Metafont will be quickly ported and available as well.

UNIX TEX SITE REPORT

Richard Furuta
Department of Computer Science
University of Washington

The most significant change to the Unix-TEX distribution since our last report in these pages has been the arrival of TEX version 1.1 and LATEX version 2.06a. Both were added to the distribution tape in late July. A faster version of the Versatec device driver was contributed by Chris Torek of the University of Maryland and was added to the distribution in early May. A DVI previewer for the Sun II workstation (version 1.0), a modification of the BBN BitGraph previewer described in the previous *TUGboat*, was created by Norm Hutchinson of the University of Washington and was included in the distribution in early July. In early August, Mike Urban of TRW sent along some changes to the Imagen driver to support the recently declared TEX standard page origin of one inch from the upper and left hand edges of the physical page.

In addition, we made a number of smaller organizational changes to the distribution, inclusing many which are intended to make it easier to find those parts of the distribution which differ between 4.1 and 4.2 bsd. Parenthetically, the VAX that serves as host to the Unix-TEX distribution has converted from 4.1 bsd to 4.2 bsd and so we are no longer able to provide TEX binaries for 4.1 bsd.

We expect that there will be more major changes to the distribution in the upcoming weeks and months. Most significantly, the WEB **METAFONT** is just becoming available. As the existing fonts are translated from the old Metafont into the new **METAFONT**, we expect to replace the .PXL files currently in the distribution with the corresponding Metafont descriptions. This will allow Unix-TEX sites to generate directly those font magnifications that they need. Also of significance will be David Fuchs' port of TEX to the C programming language, which may provide a much appreciated speed-up.

We also hope to provide a Sun-TEX version soon. Steven Correll of the Lawrence Livermore National Lab, Rusty Wright of the University of California at San Diego, and Charles Perkins and Mike Harrison of the University of California at Berkeley have all succeeded in porting Unix-TEX to the Sun II. We are currently merging their changes and expect to have them on the distribution tape by the time this report appears in print. Perkins and Harrison have also modified the Sun DVI previewer mentioned above to work within the Sun's window package (under version 1.1) and we also expect to be able to make this available within the Unix-TEX distribution.

Finally, if this is your first issue of *TUGboat*, welcome! We provide TEX for Berkeley Unix—4.1 bsd and 4.2 bsd—running on the VAX. To get a copy, send me a check for $75, made to the University of Washington (we cannot accept purchase orders), and a copy of your 4.1 or 4.2 bsd source license (the one from Berkeley, *not* any of the AT&T licenses). Please note that we recently increased the price of the tape to cover a previously undiscovered University surcharge. If you need further information, drop me a line and I'll mail you an information sheet.

VAX/VMS Site Report

Barry Smith

Well, we've been busy —

- The new VAX/VMS TEX, version 1.1, is packaged and available from Kellerman and Smith; the tape includes the LATEX, AMS-TEX, and HP-TEX macro packages.
- We think we've located the compressed-raster driver for the Versatec (it's "in the mail" at press time); this will allow the VERTEX DVI to Versatec driver to share a spooled printer with other uses. (The new VERTEX should be available about the time you read this.)
- Our Compugraphic 8600 driver produced its first pages of TEX output last week; let us know if you're interested. David has mechanized the translation from Compugraphic font width information to TFM files, so building our font library goes quickly. (Kerning information must be added by hand and eye, though.) We'll be joining the queue asking Compugraphic to add Computer Modern to their catalog.

— and the Macintosh implementation of TEX (MacTEX? Oh, please ...) continues.

Feedback from TeX Users at Lockheed

Carlos A. Felippa
Staff Scientist
Applied Mechanics Laboratory
Lockheed Palo Alto Research Laboratory
Palo Alto, CA 94304

As of this writing, TeX82 is used at the Lockheed Palo Alto Laboratory by 24 scientists, of which about 8 qualify as heavy users. It has been an underground penetration, first scorned and eventually tolerated by management. Dissatisfied with our technical typing support, late in 1981 I brought the old TeX80 (distributed by Oregon Software) in house to run under VAX/VMS and wrote a Versatec spooler. Soon I was joined by several colleagues, which now constitute the "hard core" of TeX users. Usage picked up further when TeX82 was made available on September 1983; and even more when a QMS laser printer was installed early this year and we were able to say goodbye to the foul smelling Versatec paper.

The overwhelming local use of TeX82 is for internal documents, software manuals, technical papers, memos and letters. If our experience is typical of other industrial groups (and I have no reason to think otherwise) then only a small minority of present TeX users are composing books and care (or should care) about matters of book design.

For casual users who may call on TeX perhaps once or twice a month, *language consistency* is a more important issue than artistic quality. Such users are surprised at the strange results (or lack thereof) they get. The following list has been compiled from my experience and users' feedback over the past year, and is offered in the hope that it will spotlight areas of TeX maintenance and documentation that need short-term attention. (When I mentioned some of these at the last TUG meeting, I was laughed at by the wizards, but most users are *not* wizards).

Emit blank page. There is no consistent way of getting a numbered blank page. Beginners invariably try, sensibly enough, `\eject\eject` at first; then `\vfill\eject\vfill\eject`. Since nothing happens, I got a bug report. After some experimentation, one can find that `\ \vfill\eject` does produce a blank page, but without a page number! In the old TeX80 an `\eject` always produced a page eject, which is correct from the standpoint of consistency.

That mysterious `\vglue`. Several users were baffled by the fact that a `\vskip` did not perform as expected after an explicit `\eject`. I told them to use `\vglue`, which I discovered by chance in a TeXbook Appendix (it is not mentioned in the main Chapters). Why the inconsistency, they asked. I could not explain. Incidentally, `\vglue` has curious side effects. If you type `\beginsection` after a `\vglue` you get an extra blank page.

Black hole footnotes. Footnotes inside a `\centerline` are quite common in paper headings that state authors' affiliations. Sometimes the footnote(s) vanished into thin air. For example, some authors typed something like

```
\smallskin\centerline{ B. C. Dull\footnote*{Token Scientist} and ...  }
```

where `\smallskin` is a local abbreviation for `\smallskip\noindent` and were taken aback when the footnote text vanished without warning. Another bug report. Experimentation showed that `\noindent` was responsible and that if `\smallskin` is changed to `\smallskip` the footnote reappears. The explanation is given in page 117 of the TEXbook, but how many users understand such doubly-dangerous "modal" subtleties? I believe that at least an explanatory warning message should be given.

Incorrect `\tt` *spacing.* "Verbatim listing" macros are frequently used in our documents to list input data files, insert source code fragments, etc. These macros use typewriter fonts so in theory uniform spacing should be achieved and all columns ought to align properly, right? Wrong. Distortion effects are quite noticeable on the right hand side of the page when there are character runs, *e.g.* `**************` matched against a run of blanks. Any suggestions?

Extra space after periods. The TEX rules confuse experienced typists used to inserting a double space after end of sentences but not after abbreviations. The following rule (stolen from SCRIBE) is therefore proposed: insert extra space after periods only if two or more spaces follow. For the purposes of this rule, a carriage return should count as two spaces.

Italic correction. From perusing other people's TEX sources, I believe that I am the only user here that bothers with italic corrections. Mention the subject and you get blank stares. If TEX can use dynamic programming to break up a paragraph into lines, why can't it figure out the italic correction by itself?

The `\eqalign` *surprises.* Occassionally all equations in an `\eqalign{...}` are placed flush with the left margin although plenty of space remains for centering. If this happens, I tell users to insert left spacers, for example `\qquad`, in one equation. Any thoughts on how to avoid this problem?

I would like to invite users that work in similar environments to report their TEXperiences, as well as to comment on the preceding points, in the TUGboat forum.

* * * * * * * * * * * *

"small" TEX

Lance Carnes

The world of small TEX is growing up. There are several new versions on small systems, as well as some exciting prospects for future versions.

The main news you have all been waiting for is, yes indeed folks, TEX is about to become available on your favorite home computer. There are three known projects underway to implement TEX on the IBM PC (8088 PC-DOS), and one possible version to be available for the Apple MacIntosh and Lisa.

I am working on one version for the IBM PC. At the time of writing, the project is well under way. The prototype version is targeted to run on the PC with maximum memory (640Kb). I am using the MS-Pascal compiler, running under MS-DOS.

David Fuchs is also working on TEX for the PC, and is attacking the problem in a slightly different way. His first step was to take the Pascal code produced by TANGLE and translate it into C. At the present he has a running version, but with a minimal memory model.

David claims he will win the race to bring up the first full prototype. Read the next column to find out who won.

The third effort is by Ronny Bar-Gadda of Phillips Research Laboratory in Sunnyvale. I have no further details on his project at the time of this writing.

I would appreciate a note from anyone else working on TEX for a small system.

Dave Kellerman and Barry Smith are working on a version for the Apple MacIntosh and Lisa systems.

Jaap van 't Ooster of Océ in Holland has TEX running on the PERQ.

Below is the grid showing all known implementations on small systems, both existing and in development. If there are any corrections or additions, kindly get in touch.

"small" TEX implementations

Manufacturer and model	Processor	TEX version	CPU secs. per page	Contact, organization
Hewlett-Packard 3000	16-bit	TEX82	10–30	Lance Carnes, TeXeT
Hewlett-Packard 1000	16-bit	TEX82	10–30	John Johnson, JDJ Wordware
DEC PDP-11/44 Plexus, Onyx IBM PC	16-bit Z8000 8086/88	TEX80	10–20 10–30 10–30	Dick Gauthier, TYX
Apollo	MC68000	TEX82	2–10	Thom Hickey, OCLC; Bill Grop, Yale; Pierre Clouthier, COS Information
Hewlett-Packard 9836	MC68000	TEX82	6–10	Jim Crumly, HP Boise Div.
Sun	MC68000	TEX82		Jim Sterken, Textset; Rich Furuta, U of Washington
Corvus	MC68000	TEX82[2]		
Cyb	MC68000	TEX82[1]		Norman Naugle, Texas A&M
Apple MacIntosh, Lisa	MC68000	TEX82[1]		Barry Smith, Dave Kellerman, Kellerman & Smith
Masscomp	MC68000	TEX82		Bart Childs, Texas A&M
Sage	MC68000	TEX82[2]		
Synapse	MC68000	TEX82	10–30	Dick Wallenstein, Comcon
PERQ/ICL		TEX82		Jaap van 't Ooster, Océ, Holland
IBM PC, XT, AT	8088, 80286	TEX82[1]		Lance Carnes, TeXeT
IBM XT, AT	8088, 80286	TEX82[1]		David Fuchs, Stanford
IBM PC, XT, AT	8088, 80286	TEX82[1]		Ronny Bar-Gadda, Phillips Research Laboratory

[1] in progress or recently completed [2] currently unimplementable

Letters

To the Editor:

I have come to recruit you to 'a good cause'. The Mathematics department at Imperial will take on a rather special undergraduate this coming academic year. Special, because he is both deaf and blind. Naturally, he is able, highly motivated, and knows his way round a Braille terminal. Communication, in a general sense, is straightforward, if a little cumbersome. Mathematical communication? The obvious answer is TEX. How else can you communicate mathematics unambiguously (who said APL?), in an easy to learn way (some of his lecturers are getting old), and in a way which will grow as he learns? Someone else must have tried to do this. Do you know of any such attempt? If you could publish a short note about this project in TUGboat, we would be grateful. Any bright ideas, helpful hints, even words of encouragement, will be gratefully accepted.

Malcolm W. Clark
Imperial College Computer Centre
Exhibition Road
London SW7 2BX, England
01-589 5111

Editor's note: his letter was read at the August TUG meeting, and a few members seemed to have knowledge of others working with blind students. The question was posed, why is TEX obvious? TEX, aside from its typesetting capabilities, is a system for linearizing mathematics, with a vocabulary similar to what would be used by two mathematicians communicating over a telephone. The structural and positional relationships between symbols remain intact, and the macro facility makes it possible to develop 'shorthand' forms for frequently occurring expressions.

We wish both the student and his lecturers success.

News & Announcements

Calendar

Plans are being formulated to conduct TEX for Beginners courses in the Ann Arbor, Boston, Chicago, New York and Seattle areas next summer. The possibility of conducting other TEX-related courses in these areas in conjunction with the beginning courses is being considered, also. (See TUG's advertisement for instructors on page 156 for a list of possible subjects.) A complete list of locations and dates should be available for distribution to members in January 1985. The list will be published in this column in the March issue.

1985

Jan.	9 – 12	TEX exhibit (Textset, Inc., and TUG), American Mathematical Society Annual Meeting, Anaheim Convention Center, Anaheim, Calif.
Jan.	31	TUGboat Volume 6, No. 1, Deadline for submission of manuscripts
May	16 – 17	TEX for Scientific Documentation, Varenna, Italy. For additional information, see announcement on page 147.
May	31	TUGboat Volume 6, No. 2, Deadline for submission of manuscripts (tentative)
Aug.	5 – 9	TEX for Beginners, Stanford University, Stanford, Calif.
Aug.	12 – 13	TEX Short Course (subject to be announced), Stanford University, Stanford, Calif.
Aug.	14 – 16	TEX Users Group Annual Meeting, Stanford University, Stanford, Calif.
Sep.	30	TUGboat Volume 6, No. 3, Deadline for submission of manuscripts (tentative)

TeX for Scientific Documentation

Varenna, Italy, May 16–17, 1985
Call for papers

A European Conference on the TeX System and related current applications will be held in Varenna, Italy, May 16–17, 1985. The aim is to provide a state-of-the-art survey of current work in this area and to encourage technology transfer and information exchange on the latest applications of documentation systems based on TeX.

The Conference will be supported by the following organizations:

- CNR (Italian National Research Council)
- FAST (Federation of Scientific and Technical Associations)
- Mondadori s.p.a. (Publishing House)
- UMI (Italian Mathematical Society)
- Università degli Studi di Milano, Istituto di Cibernetica

Suggested presentation topics:

- TeX system integration and improvement
- Font design
- TeX macro packages and development methodology
- Document structure standards
- End-user interfaces and work stations
- Preprinting, printing and typesetting
- Implementation of output drivers
- Integration of text and graphics
- Filing, retrieval and delivery of TeX documents
- Transporting text
- Document preparation environments
- Electronic publishing applications

Program Committee Members:

S. Bien	Univ. Warsaw (Poland)
L. Cerofolini	Univ. Bologna (Italy)
G. Degli Antoni	Univ. Milano (Italy)
B. Gaulle	CIRCE/CNRS (France)
D. Knuth	Stanford University (USA)
A. Løfstedt	RECAU, Aarhus (Denmark)
B. Shulze	Univ. Bonn (Germany)
C. Vernimb	CEE (Luxemburg)
I. Zabala	Univ. Valencia (Spain)

Organizing Committee Members:

G. Canzii	Univ. Milano (Italy)
R. Goucher	Amer. Math. Soc. (USA)
D. Lucarella	Univ. Milano (Italy)
F. Migiarra	Mondadori s.p.a. (Italy)

Papers accepted will be collected and published in Conference Proceedings by FAST. In addition a range of tutorials will be planned. Contributions in TeX source in machine readable form are encouraged. Diskette or tapes are welcome. Submit papers to

D. Lucarella
Istituto di Cibernetica
Università di Milano
via Viotti 3/5
20133 Milano, Italy

according to the following schedule:

December 1, 1984:	Submission
January 31, 1985:	Notification of acceptance
March 15, 1985:	Mailing of the final program and registration information
May 16–17, 1985:	Conference in Varenna

Official languages will be English and Italian. Simultaneous translation will be provided.

TUG 1984 Financial Report With Comparisons

September 30, 1984

	Actual 1982	Actual 1983	Budget 1984	Actuals thru 9/84	Estimated thru 12/84	Budget 1985
Income:						
Membership/Subscriptions						
Individual/Library[1,2]	$ 11,935	$ 13,629	$ 15,800	$ 15,190	$ 16,000	$ 20,000
Postage	588	1,358	-0-	-0-	-0-	-0-
Institutional Membership[1,2]						
Educational	200	5,780	5,500	6,600	7,000	8,000
Non-educational	200	3,612	5,000	9,900	10,500	11,100
Publications						
Back issue sales[3]	2,156	5,094	2,500	5,400	6,000	5,000
Other publications[4]	376	634	500	5,005	5,500	4,500
Meetings[2,5]						
Summer meeting	11,025	15,209	15,000	28,940	30,000	20,000
2-day course	27,018	14,187	15,000	17,255	17,800	20,000
5-day course	-0-	-0-	-0-	43,400	44,000	40,000
Manufacturers reps' fees	300	450	600	425	600	600
Other meetings/courses	4,500	-0-	6,000	-0-	-0-	60,000
Other						
Videotape sales/rental	2,200	4,998	3,000	1,440	2,000	2,000
Advertising/mailing list sales	-0-	325	500	777	1,000	1,000
Royalties (*The TeXbook*)	-0-	-0-	1,000	3,268	3,268	10,000
Interest income	-0-	-0-	-0-	-0-	-0-	8,000
Total income	$ 60,498	$ 65,276	$ 70,400	$ 137,600	$ 143,668	$ 210,200
Expenses:						
TUGboat (2 issues)						
Printing	$ 2,220	$ 2,620	$ 2,596	$ 1,440	$ 2,880	$ 3,120
Postage	1,143	934	944	720	1,440	1,680
Editorial services	3,460	3,772	3,835	2,040	4,200	4,200
Clerical services	180	182	177	96	180	240
Computer expenses	2,100	2,220	2,360	1,320	2,400	2,640
Meetings[*,2,5]						
Summer meeting & 2-day course	4,130	7,311	5,900	5,500	7,000	7,000
5-day course	-0-	-0-	-0-	23,000	28,000	15,000
Other meetings/courses	2,460	-0-	2,360	-0-	-0-	20,000
Other						
Other publications[4]	225	474	354	1,217	1,800	2,400
ANSI meetings[*,6]	2,503	1,280	1,652	-0-	-0-	-0-
Legal and tax consulting	-0-	-0-	590	1,017	1,500	2,000
Postage, general mailings	1,335	3,605	2,950	1,888	3,600	3,960
Printing back issues[3]	-0-	7,953	-0-	-0-	-0-	-0-
Printing, other	1,212	426	2,360	3,523	4,200	4,200
Administrative support[7]	-0-	12,923	11,800	18,000	26,400	57,600
Clerical services	3,120	5,708	6,313	2,400	2,640	3,600
Subsidies[*,8]	-0-	-0-	1,180	-0-	-0-	-0-
Video tape duplication[*]	-0-	2,553	1,180	286	960	800
Computer expenses	1,455	4,637	1,770	2,412	4,200	4,800
Programming[9]	-0-	-0-	3,009	6,012	7,200	-0-
Miscellaneous[10]	1,688	1,622	2,360	1,800	2,640	2,640
Total expenses	$ 27,231	$ 58,220	$ 53,690	$ 72,670	$ 101,240	$ 135,880
Summary:						
Balance forward	$ (8,660)	$ 24,607	$ 36,706		$ 31,663	$ 74,091
Income (Actual/Budget/Est.)	60,498	65,276	70,400		143,668	210,200
Expenses (Actual/Budget/Est.)	$ (27,231)	$ (58,220)	$ (53,690)		$ (101,240)	$ (135,880)
Balance	$ 24,607	$ 31,663	$ 53,416		$ 74,091	$ 148,411

TUG 1984 Financial Report

(Continued from preceding page)

Notes:

All 1984 expense figures include an AMS overhead charge of 20%; also, 1985 expense figures, except those with an asterisk (*), include an overhead charge of 20%. Except as indicated, these remarks apply to the 1984 year.

1. As of September 30, 1984, there were 1,022 members/subscribers, including 69 Institutional Members: 34 educational; 35 non-educational.
2. Advertising of TUG and the TUG Meeting/Courses was accomplished through a news release to 19 trade publications, several of which are known to have published the notice, in addition to direct mailings to members and former members.
3. So far in 1984 360 back issues have been sold, which already exceeds the number sold in 1983 by 60. Outside of new-issue printing, it should not be necessary to reprint back issues in 1984 and possibly 1985.
4. To date over 600 copies of Arthur Samuel's "First Grade TEX" have been sold. Only a few copies of Hewlett-Packard's "The HP TEX Macros" have been sold.
5. 153 members attended the 3-day TUG meeting at Stanford in August. In addition, 75 individuals attended the "Book Design Utilizing TEX" course taught by Leslie Lamport and Richard Southall and 58 attended the "TEX for Beginners" course taught by Arthur Keller. At Stanford during 1985 a 5-day beginning TEX course is planned for the week of August 5–9 and a 2-day short course and 3-day TUG meeting are planned for the week of August 12–16. In addition, two 3-day courses and two 5-day courses have been budgeted, to be conducted at locations to be announced.
6. Larry Beck, Grumman Data Systems, has been appointed to serve as TUG's liaison with ANSI X3J6, replacing Lynne Price.
7. During 1984 the services of Ray Goucher, amounting to approximately half time, were purchased from the AMS. He has been budgeted as full time for 1985. (The budgeted amount for 1985 includes salary and AMS benefits and overhead.)
8. The Steering Committee made this amount available to the Finance Committee to subsidize travel and membership/participation fees for individuals when appropriate.
9. Reprogramming to improve the functioning of the TUG data base.
10. Postage/express charges, telephone tolls and supplies, plus programmer and clerical services not associated with production of TUGboat.

Respectfully submitted,
Samuel B. Whidden, Treasurer

Report of the August 1984 Meeting of the TUG Steering Committee

The Steering Committee held working luncheons on August 15, 16, and 17, 1984. Attendees are listed below. (Not everyone attended every meeting.)

Lawrence Beck	William Kelly
Barbara Beeton	Donald Knuth
Lance Carnes	Pierre MacKay
Bart Childs	Norman Naugle
John Crawford	Monte Nichols
Charles Dupree	Susan Plass
David Fuchs	Barry Smith
Richard Furuta	Rilla Thedford
Ray Goucher	Joey Tuttle
Patrick Ion	Robert Welland
Arthur Keller	Sam Whidden

Lawrence Beck replaces Lynne Price as liaison with the ANSI X3J6 committee, while Patrick Ion takes over her job as editor for the TUGboat macros column. John Crawford replaces Bart Childs as site coordinator for Prime users; William Kelly augments Ralph Stromquist as site coordinator for Univac 1100 users; and Barry Smith replaces Monte Nichols as site coordinator for VAX/VMS users.

The major issues addressed at these meetings related to the changing nature and purpose of TUG. In the past, TUG members have been interested in technical details of getting TEX running on various systems and talking to various devices; in other words, they have been system implementors. However, as TEX systems stabilize, more TUG members are users of TEX than implementors, and their areas of interest are higher-level issues such as book design.

TUG has been supported in the past by fees paid by members and work donated by the American Mathematical Society (AMS). With the influx of new members, the increasing popularity of courses, and the sale of back issues of TUGboat, TUG has become financially independent. The nature of the relationship with the AMS is changing. TUG's finances will soon be completely separate from those of the AMS; however, TUG will continue to buy certain services from the AMS and will begin renting space in the AMS office as of January 1, 1985.

The Steering Committee intended to redirect the TUG organization to reflect the new constituency and direction. Therefore, special committees were set up to review projects and publications aimed at promoting TEX and educating TEX users and to handle issues related to standards and requirements for use of TEX. Each committee includes one member from the Steering Committee: the Special Projects Committee has Arthur Keller; the Publications Committee, Robert Welland, and the Standards Committee, David Fuchs.

The Steering Committee also discussed how best to make use of the financially independent status the group has achieved. It was resolved that future TUG meetings will be priced just above the estimated break-even level, with any revenues used to promote the understanding and use of TEX. Classes may still be profit-making ventures. Lease and sale of videotapes for reduced prices is under investigation; legal issues and distribution questions are open in this area.

Various options for modifying the structure and tone of TUG meetings were discussed, including the idea of having parallel tracks for different types of users, of accepting papers for presentation, and of organizing Birds-of-a-Feather sessions during the days instead of the evenings, as previously has been the case. No decisions were reached on these issues, however; so a committee headed by Joey Tuttle was established to organize next year's TUG meeting without specific requirements in these areas. The date for next year's meeting was not set, but the meeting will be held some time in August at Stanford University. [*Editor's note*: The dates have now been set; see the calendar on page 146 for details.]

Finally, the nominating committee recommended that the current holders of the offices of Vice President and Secretary, Joey Tuttle and Chuck Dupree, respectively, be renominated for another term. The Steering Committee agreed, and the nominees were duly elected at the meeting. Pierre MacKay, as TUG president, presided over the business meeting and notified the members of the attempts being made by the Steering Committee to redirect the group's actions along the lines requested by the members at the meeting. He discussed the impending separation of TUG from AMS, the financial condition of TUG, and the creation of various committees to handle work involved in promoting TEX understanding and usage. He also announced that, in view of the effective job Ray Goucher has done on a half-time basis as Business Manager, the Steering Committee would request that the AMS make his services available to TUG full time starting in January 1985. [*Editor's note*: This request was approved by the AMS in October.]

The full text of these minutes is available upon request from TUG.

These minutes respectfully submitted by
Charles Dupree

Participants, TUG Meeting and Courses

Summer Meeting and
Short Course: Book Design Utilizing TEX
Stanford University, August 13–17, 1984

In the following list, names of persons attending only the meeting are not specially marked; attendees at only the Short Course are starred; attendees at both the Meeting and the Short Course are flagged by a †.

Acuff, Richard – Ohio State University
Adler, Brian – Digital Equipment Corp.
Angerstein, Paula – Burroughs Corp.
* Annicchiarico, Julie – RE/SPEC, Inc.
† Appelt, Wolfgang – Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin
Arnson, William – McGraw-Hill, Inc.
† Azzarello, Arlene – I. P. Sharp Associates
Balance, Robert – Hewlett-Packard Co.
Barstow, Tom – Apollo Computer, Inc.
Beatty, Stuart – Hewlett-Packard Co.
Beck, Lawrence – Grumman Data Systems
† Beebe, Nelson – University of Utah
† Beeton, Barbara – American Mathematical Society
† Benson, Gary – Los Alamos National Laboratory
Bigelow, Chuck – Computer Science Dept., Stanford University
Black, William – Oxford University
† Blanford, Mark – RE/SPEC Inc.
Block, Neil – Hughes Aircraft Co.
Boes, Jeff – Lear Siegler, Inc.
* Bond, Liz – Adobe Systems, Inc.
Bratnober, James – Hewlett-Packard Co.
† Brown, Malcolm B. – Technical Support Services, Stanford University
Bryant, Sarah – McGraw-Hill, Inc.
Burgart, Cal – Talaris Systems, Inc.
† Byl, Renata – Stanford Linear Accelerator Center
Canna, Bruce – Addison-Wesley Publishing Co.
Carnes, Lance – TeXeT Co.
Cerofolini, Luigi – Istituto de Matematica Aplicata, University of Bologna
† Cheung, Lucy – Stanford Linear Accelerator Center
† Childs, Bart – Texas A&M University
Clark, Louis – Monsanto Agricultural Products

Clark, Malcolm – Imperial College
† Colman, Jane – Lawrence Berkeley Laboratory
† Conley, Walt – New Mexico State University
* Connors, Nellie – Rational
Crawford, John – Ohio State University
† Crowder, Louis – Hewlett-Packard Co.
† Cuzzo, Clint S. – Hewlett-Packard Co.
† Cymbalista, Malka – Weizmann Institute of Science
† Daniels, Susan – Hewlett-Packard Co.
DeBattista, Joe – University of California, San Francisco
Desarmenien, Jacques – Computer Science Dept., Stanford University
* Dolan, Elizabeth – Burlington-Northern Research, Inc.
Dupree, Chuck – Datapoint Corp.
† Durland, Tom – Information Handling Service
* Durst, Lincoln – American Mathematical Society
Ellickson, Bryan – University of California at Los Angeles
Eppstein, Maureen – Comptroller's Office, Stanford University
† Evans, Jim – Intergraph Corp.
Felippa, Carlos – Lockheed Missiles & Space Co.
* Feroe, John – Vassar College/Univ. Calif. San Diego
Ferguson, Michael J. – INRS-Télécommunications
† Ferris, Barry – Talaris Systems, Inc.
Fidelman, Susan – Lawrence Berkeley Laboratory
† Filippenko, Ivan – Syracuse University
† Fina, Patricia – Whitehead Institute
Fingold, Sharon – Calma Co.
Forrest, Jon – University of California, Santa Barbara
† Friedrich, Dick – Lear Siegler, Inc.
Fuchs, David – Computer Science Dept., Stanford University
Furuta, Richard – University of Washington
Gabelnick, Stephen – Argonne National Laboratory
Gaskins, Robert – Forethought, Inc.
Geddes, Cliff – Ascent, Inc.
† Genolini, Franco – Institute of Cybernetics, University of Milan
* Ghinazzi, Chris – Lawrence Livermore National Laboratory
Gordon, Peter – Addison-Wesley Publishing Co.
Gourlay, John – Ohio State University
Greenleaf, Kathy – Digital Equipment Corp.
Grosso, Paul – Textset, Inc.
Guenther, Dean – Washington State University
† Hanrahan, Tom – Oregon Software
Harrison, Michael, University of California, Berkeley
Hauck, Roger – Smithsonian Astrophysical Observatory
Heisler, Mark – Westinghouse Corp.
† Herrmann, Walter – Sandia National Laboratory
Hoffman, Mark – TYX Corp.
Ion, Patrick – Mathematical Reviews, American Mathematical Society
† Jackson, Calvin – California Institute of Technology
Jensen, Sharon – Stanford Linear Accelerator Center
Jurgensen, Helmut – The University of Western Ontario
* Karagueuzian, Dikran – SCLI, Stanford University
† Kay, David – University of California at Los Angeles
Keller, Arthur – Computer Science Dept., Stanford University
† Kellerman, David – Kellerman & Smith
Kelly, William – University of Wisconsin
† Kim, Karen – SRI International
† Kitajima, Yasuko – Aldine Press
† Kneser, Thomas – Gesellschaft für wissenschaftliche Datenverarbeitung, Göttingen
Knuth, Don – Computer Science Dept., Stanford University
† Kral, Karen – University of Delaware
† Lamport, Leslie – SRI International
† LaPlace, Bruno – Centre Inter-Régional de Calcul Électronique, C.N.R.S.
Latterner, Dan – Mathematical Reviews, American Mathematical Society
† Laurent, Kevin – U. S. Geological Survey
Leung, Sheon – Signetics Corp.
† Lucarella, Dario – Institute of Cybernetics, University of Milan
MacKay, Pierre – University of Washington
Manning, Mary Beth – SPIE
Mashruwaza, Raj – Consilium Associates
Merril, Diana – Consilium Associates
Minio, Roberto – Springer Verlag
† Monroe, Beverly K. – U. S. Geological Survey
† Morris, Thomas – University of North Carolina
Munzo, Rick – Hewlett-Packard Co.
† Naugle, Norman – Texas A&M University
† Nichols, Monte – Sandia National Laboratory
† Penny, Keith – Martin Marietta Energy Systems
Pepper, Rebecca - University of California at Berkeley
† Peyton, Evelyn – Calma Co.
Pitchford, Anthony – Burroughs Corp.
† Plass, Susan – Technical Support Services, Stanford University
† Platt, Craig – University of Manitoba
Poggio, Margaret – University of California/ Lawrence Livermore Lab
Price, Gary – Rational
Quisquater, Jean-Jacques – Philips Research Laboratory, Brussels
† Renzetti, Phyllis – U. S. Geological Survey
Resmer, Mark – Vassar College
Richert, Norman – Marquette University
Rodgers, Dave – Textset, Inc.
Rokicki, Thomas – Texas A&M University
† Saltzman, Jeff – Los Alamos National Laboratory
Samuel, Arthur – Computer Science Dept., Stanford University
Schneble, Jack – McGraw-Hill, Inc.
† Scott, Hwey – Rockwell International
Segal, Lee – Digital Equipment Corp.

† Shanmugam, Al – Hewlett-Packard Co.
† Shepherd, Gary C. – Sandia National Laboratory
Sih, Peter – IBM Deutschland GmbH
Skinner, Lydia – Krestrel Institute
Smith, Barry – Kellerman & Smith
Snodgrass, Michael – Consultant
† Southall, Richard – Computer Science Dept., Stanford University
† Spadarella, Tony – Calma Co.
† Speas, Debbie – Quality Micro Systems, Inc.
† Spencer, Dave – Oregon Software
Spragens, Alan – Stanford Linear Accelerator Center
* Steitz, Jack – IBM Corp.
Sterken, Jim – Textset, Inc.
Sturdivant, Gary – Ascent, Inc.
† Sullivan, Carol – U. S. Geological Survey
† Thedford, Rilla – Intergraph Corp.
† Thomas, Margaret – Talaris Systems, Inc.
Tobin, Georgia – Online Computer Library Center
Tobin, Richard – Online Computer Library Center
† Tuttle, Joey – I. P. Sharp Associates
Urban, Michael – TRW, Inc.
Van Camp, Warren – Information General Corp.
† van 't Ooster, Jaap – Océ-Nederland B.V.
Villere, Gary – Informatics General Corp.
† Volkmann, Suzanne – Woods Hole Oceanographic Institute
* Wade, Brad – IBM Corp.
Wadsworth, Mark – Basic Four Info. Systems, Management Assistance, Inc.
† Wallace, Richard – Los Alamos National Laboratory
Welland, Robert – Northwestern University
† Westmiller, Jane – ESL, Inc.
Whidden, Sam – American Mathematical Society
† Whitney, Ron – American Mathematical Society
Wiesenberg, Michael – Hewlett-Packard Co.
† Woessner, Linda – Computer Sciences Corp.
† Zulch, Donald – Calma Co.

TEX for Beginners
Stanford University, August 20–24, 1984

Agosta, John Mark – TEX Users Group
Alexander, Wanda – Stanford Linear Accelerator Center
Barstow, Tom – Apollo Computer, Inc.
Benson, Gary – Los Alamos National Laboratory
Briesmester, Judy – Los Alamos National Laboratory
Bryant, Sarah – McGraw-Hill, Inc.
Butler, Candy – McGraw-Hill, Inc.
Cambrelen, Diane – Acurex Corp.
Chapin, John – Lear Siegler, Inc.
Contos, Maria – Argonne National Laboratory
Cordova, Carlos – Calma Co.
Davies, Vivian – California Institute of Technology
Dysland, Diane – California Institute of Technology
Filippenko, Ivan – Syracuse University
Friedrich, Dick – Lear Siegler, Inc.
Geddes, Cliff – Ascent, Inc.
Genolini, Franco – Institute of Cybernetics, Milan
Gere, Susan – Computer Systems Laboratory, Stanford University
Goldberg, Donna – Stanford University
Gotell, Blanch – McGraw-Hill, Inc.
Harlow, Martin – Los Alamos National Laboratory
Heisler, Mark – Westinghouse Corp.
Howard, Janey – Los Alamos National Laboratory
Hung, Betty – University of Alberta
Ion, Patrick – TEX Users Group
Jackson, Mark – IBM Corp.
Jenkins, Donna – Fairchild Corp.
Jenness, Jeanette – Lawrence Livermore Laboratory
Jensen, Sharon – Stanford Linear Accelerator Center
LaPlace, Bruno – Centre Inter-Régional de Calcul Électronique, C.N.R.S.
Lawrence, Mark – Technical Support Services, Stanford University
Lorch, John – Naval Weapons Center
Mashruwaza, Raj – Consilium Associates
McPartland, Marie – GTE Laboratory
Melewski, Lorraine – Digital Equipment Corp.
Merrill, Diana – Consilium Associates
Moffat, Shannon – Technical Support Services, Stanford University
Monroe, Beverly K. – U. S. Geological Survey
Nilson, Grace – ESL, Inc.
Ochoa, Janette – Graduate School of Business, Stanford University
Olson, Karen – Lawrence Berkeley Laboratory
Páez, Marlen – Stanford University
Peric, Branko – University of British Columbiia
Perry, Linda – Stanford University
Peterson, Roger – Consolidated Data Services
Plass, Susan – Technical Support Services, Stanford University
Ridenour, Ruth – California Institute of Technology
Saltzman, Jeff – Los Alamos National Laboratory
Shepp, Lois – McGraw-Hill, Inc.
Shoemaker, Kathy – RE/SPEC, Inc.
Sigl, Jill – Computer Systems Lab, Stanford University
Slezak, Mike – Quality Micro Systems, Inc.
Speas, Debbie – Quality Micro Systems, Inc.
Sullivan, Carol – U. S. Geological Survey
Swenson, Marcy – TEX Users Group
Telford, Kathleen – Lawrence Livermore National Laboratory
Van Riper, Ken – Los Alamos National Laboratory
van 't Ooster, Jaap – Océ-Nederland B.V.
Wall, Mary – RE/SPEC, Inc.
Walters, Jane – National Bureau of Standards
Worden, Floy – Lawrence Livermore National Laboratory
Zulch, Donald – Calma Co.

Profile of TEX Installations Available to TUG Members

October 12, 1984

The TEX Users Group records show the following about institutions where TEX is either running or in the process of being installed.

	academic institutions	non-academic organizations
U.S. and Canada	64	86
elsewhere	51	22

This information comes from the "Request for Information" section on membership forms. Since many membership renewals are processed early in the year, much of this information is not current. Therefore, the sites marked [ip] to indicate "installation in progress" may actually have TEX running by now. Furthermore, many members have omitted mentioning whether TEX is actually available at their site, so this should be considered a "minimum" list.

For computers which may run under different operating systems, particularly IBM mainframes and VAX, the operating system has been specified where it is known. Members using such computers are requested to be specific in their responses on the renewal form; this information will be used to create the grouping by computer in the membership list.

TEX is running on the following computers:

- Amdahl, under IBM operating systems and MTS
- Basic Four
- Burroughs
- CDC Cyber
- Data General MV8000
- DEC 10, DEC 20; Foonly (a DEC lookalike)
- Honeywell CP-6
- IBM, various models under MVS and VM/CMS operating systems
- Perkin-Elmer
- Prime
- Siemens
- Univac 1100
- VAX, under UNIX and VMS operating systems
- Various 68000-based microcomputers/workstations:
  - Apollo
  - Cyb
  - Hewlett-Packard 9000 Series 200
  - Masscomp under UNIX
  - PCS M68000 under Siemens UNIX
  - Sun
  - Synapse
- Other micros and workstations:
  - Hewlett-Packard HP 1000, 3000
  - Onyx (still running TEX80)
  - PERQ
  - Stratus

Installations of TEX are being attempted on the following computers:

GEC 4190	Microdata
Harris H800	Microcomputers
Hitac	Pyramid
IBM PC	Wang VS

TEX is running or being installed at these U.S. and Canadian academic institutions:

- Boston University: DEC 20; VAX (VMS)
- Brown University: VAX (UNIX)
- California Inst of Technology: VAX (UNIX); VAX (VMS)
- Carleton University: Honeywell CP-6
- Carnegie-Mellon University; DEC 20; VAX (UNIX)
- Colorado State University: VAX (VMS) [ip]
- Columbia University: DEC 20; Nevis Labs: VAX (VMS)
- Dalhousie University: VAX (UNIX)
- Dartmouth College: VAX (UNIX) [ip]
- Hawaii Inst of Geophysics; Harris [ip]
- Indiana University: VAX (UNIX)
- Inst for Defense Analyses: VAX (UNIX)
- Marquette University: VAX (VMS)
- Mass Inst of Technology: DEC 10; DEC 20
- McGill University: VAX (VMS) [ip]
- New Mexico State University: HP 1000
- New York University: VAX (UNIX)
- Northeastern University: VAX (VMS) [ip]
- Ohio State University: DEC 20; Prime
- Princeton Plasma Physics Lab: DEC 10
- Princeton University: IBM PC XT [TEX80 in C]; VAX (VMS)
- Purdue University: VAX (UNIX)
- Smithsonian Astrophys Observ: VAX (VMS)
- Space Telescope Science Inst: VAX (VMS) [ip]
- St Olaf College: VAX (UNIX) [ip]
- Stanford Linear Accelerator Ctr: IBM (VM/CMS)
- Stanford University: DEC 10; DEC 20; IBM (MVS); Sun; VAX (UNIX); VAX (VMS)
- SUNY at Stony Brook: VAX (VMS)
- Syracuse University: DG MV8000
- Texas A & M University: DG MV8000; IBM 370; Prime; VAX (VMS)
- Trinity College: VAX
- Univ of Calgary: VAX (UNIX) [ip]
- Univ of California, Berkeley: Magnuson M80/43 [=IBM 370] (MVS); VAX (UNIX)
- Univ of California, Irvine: DEC 20; Honeywell; VAX (UNIX)
- Univ of California, Los Angeles: VAX (UNIX)
- Univ of Chicago: Pyramid [ip]; VAX (UNIX) [ip]
- Univ of Delaware: IBM (VM/CMS); VAX (UNIX)
- Univ of Hawaii: VAX (VMS)
- Univ of Illinois at Chicago: IBM (VM/CMS)

Univ of Manitoba: Amdahl (MVS); IBM (MVS)
Univ of Massachusetts: Sun; VAX (VMS)
Univ of Minnesota: CDC Cyber
Univ of Michigan: Amdahl (MTS)
Univ of North Carolina, Chapel Hill: VAX (UNIX) [ip]
Univ of Oklahoma: VAX (VMS)
Univ of Oregon: VAX (UNIX) [ip]
Univ of Pennsylvania: IBM (VM) [ip]; Moore School: VAX (VMS)
Univ of Rochester: DEC 10; DEC 20; IBM (VM/CMS) [ip]; VAX (VMS)
Univ of Tennessee: VAX (VMS)
Univ of Texas at Austin: VAX (VMS)
Univ of Toronto: VAX (VMS)
Univ of Utah: DEC 20; VAX (VMS)
Univ of Washington: DEC 10; VAX (UNIX); VAX (VMS) [ip]
Univ of Waterloo: IBM (VM/CMS) [ip]
Univ of Western Ontario: VAX (VMS) [ip]
Univ of Wisconsin, Madison: Univac 1100; VAX (VMS)
Vanderbilt University: DEC 10
Vassar College: VAX (VMS)
Villanova University: VAX (VMS) [ip]
Virginia Polytechnic Inst: VAX (VMS) [ip]
Washington State University: Amdahl (MVS)
Washington University: VAX (UNIX); VAX (VMS)
Yale University: Apollo; DEC 20; VAX (VMS)
York University: VAX (VMS) [ip]

TEX is now running or being installed at these academic institutions outside of North America:

Åbo Akademi, Datacentralen: Univac 1100 [ip]
Adelaide University: VAX (VMS)
Australian National University: DEC 10
CERN: VAX (VMS)
CIRCE/CNRS: NAS 9080 [=IBM] (MVS)
E.S.I.E.E. (France): Prime [ip]
ETH (Swiss Federal Inst Tech): VAX (VMS)
Flinders U of South Australia: Prime [ip]
Gesellschaft Math und Daten: IBM (MVS)
Helsinki Univ of Technology: DEC 20
Imperial College Computer Ctr: CDC Cyber
INFN - CNAF: VAX (VMS)
Inst of Operations Research (Bonn, Fed Rep Germany): IBM (VM)
Keio University: VAX (UNIX)
KTH (Royal Inst of Tech): DEC 20; VAX (VMS) [ip]
The Open University (Oxford, England): DEC 20 [ip]
Otaru Univ of Commerce: Hitac [ip]
Oxford University: VAX [ip]
Queen Elizabeth College (London): VAX (VMS)
Queen's Univ of Belfast: PERQ [ip]
RECAU, Aarhus Univ: CDC Cyber
St Patricks College: VAX (VMS)
Stockholm University: DEC 10; DEC 20
Tampere Univ of Technology: VAX (VMS)
Technical Univ of Berlin: PCS M68000 (UNIX) [ip]
Technical Univ of Darmstadt: IBM (VM/CMS) [ip]; Siemens
Univ of Antwerpen: VAX (VMS)
Univ of Bergen: Univac 1100 [ip]
Univ of Bochum: CDC Cyber
Univ di Bologna, Fac di Ingegn: VAX
Univ of Bonn, Inst fur Angewandte Mathematik: IBM (VM/CMS)
Univ of Canterbury (New Zealand): Burroughs
Univ of Hamburg: DEC 10
Univ of Jyväskylä: Univac 1100 [ip]
Univ of Kent at Canterbury: VAX (UNIX)
Univ of Köln: CDC Cyber [ip]
Univ of Linköping: DEC 20
Univ of London: VAX (VMS)
Univ of Münster: IBM (VM/CMS) [ip]
Univ of New England (Australia): DEC 20 [ip]
Univ of New South Wales: DEC 10
Univ of Oslo: DEC 10
Univ des Saarlandes: Siemens [ip]
Univ of Sheffield: HP 3000
Univ of Sydney: VAX (VMS)
Univ of Western Australia: DEC 10
Universita Studi Milano, Istituto di Cibernetica: IBM (MVS)
Universita Studi Roma, Dipartimento de Fisica: VAX (VMS) [ip]
University College Dublin: DEC 20; IBM (VM/CMS) [ip]
University College London: GEC 4190 [ip]
Weizmann Institute of Science: IBM (VM/CMS)

Non-academic organizations in North America where TEX is running or being installed:

Acurex Corporation: VAX (VMS)
Adapt, Inc: DEC 20
Adobe Systems, Inc: VAX (UNIX) [ip]
Apollo Computer, Inc: Apollo [ip]
Argonne National Laboratory: VAX (VMS)
Arthur D Little, Inc: IBM (VM/CMS) [ip]
AT&T Bell Laboratories: VAX (UNIX) [ip]
Bell Northern Research, Inc: DEC 20
Bendix Guidance Systems: VAX (VMS)
Boeing Aerospace Company: VAX (VMS)
Brookhaven National Lab: VAX (VMS) [ip]
Cadtec Corp: VAX (VMS)
Calma: Apollo; VAX (VMS)
Candle Corp: IBM
Charles Stark Draper Lab, Inc: IBM (MVS) [ip]
Communications Research Centre (Ottawa): VAX (VMS)
Consilium Associates, Inc: VAX (VMS) [ip]
Consolidated Data Services, Ltd: Univac 1100 [ip]
Crawford Service Co: VAX (VMS) [ip]
Demonics, Ltd: Masscomp (UNIX)
Digital Equipment Corp: VAX (VMS)
E I DuPont de Nemours, Inc: DEC 10; VAX (VMS)
Electrocon International, Inc: Apollo
ESL, Inc: IBM (VM/CMS)
Fairchild: VAX (VMS)
G A Technologies, Inc: DEC 10
General Electric Co: VAX (UNIX)
Great Pacific Computer Works: VAX (VMS)

GTE Laboratories: VAX (VMS) [ip]
Hewlett-Packard: HP 3000; HP 9000 Series 200; VAX (UNIX)
Hughes Aircraft Company: Perkin-Elmer; VAX (VMS)
Imagen Corp: VAX (UNIX)
Independence Industries: Microdata [ip]
Informatics General Corp: VAX (VMS)
INRS - Télécommunications: VAX (VMS)
Intergraph Corp.: VAX (VMS)
JDJ Wordware: HP 1000
Kellerman & Smith: VAX (VMS)
Kestrel Institute: Foonly (TOPS-10)
Las Vegas Sun Newspaper: VAX (UNIX) [ip]
Lawrence Berkeley Lab: VAX (VMS)
Lawrence Livermore National Lab: Cray; Foonly; Prime; VAX (UNIX); VAX (VMS) [ip]
Liberty Mutual Research Ctr: VAX (VMS)
Lockheed Palo Alto Res Lab: VAX (VMS)
Logicware (Toronto): VAX (VMS) [ip]
MAI/Basic Four: Basic Four
Martin Marietta Energy Sys, Inc: DEC 10; IBM (MVS) [ip]
Massachusetts Computer Associates: VAX (VMS)
Mass General Hospital: VAX (VMS)
Microelectronics Center of N C: VAX (UNIX), VAX (VMS)
McDonnell Douglas: VAX (VMS) [ip]
McGraw-Hill, Inc: IBM (MVS)
Monsanto Co: DEC 10; VAX (VMS)
NASA Ames Research Center: VAX (VMS)
NASA Langley Research Center: Prime
Natl Ctr for Atmospheric Res: Masscomp
Natl Radio Astronomy Observatory: VAX (VMS)
Naval Research Lab: VAX (VMS)
OCLC, Inc: Apollo
Optical Sciences Co: Perkin-Elmer
Oregon Software: VAX (VMS)
PAR Technology Corp: VAX (VMS)
Polygon Associates, Inc: VAX (VMS) [ip]
Quality Micro Systems, Inc: VAX (UNIX)
RCA: (unknown) [ip]
RE/SPEC, Inc: DG MV8000; VAX (VMS)
Rockwell International: Intergraph; VAX (VMS)
Sandia National Laboratories: VAX (VMS)
Santa Cruz Operation, Inc: Onyx [TEX80 in C]
SAS Institute: IBM (MVS); VAX (VMS)
Schlumberger (various, numerous): VAX (VMS)
Science Applications, Inc: DEC 10; VAX (VMS)
Spartacus Computers: Nixdorf, Spartacus [=IBM 370]
SRI International: Foonly; HP 3000 [ip]
Stratus Computer, Inc: Stratus
Talaris Systems, Inc: DEC 10; DEC 20; VAX (UNIX, VMS)
Tektronix, Inc: VAX (UNIX) [ip]
Textset, Inc: Amdahl (MTS, U Mich); Sun
TYX Corp: Onyx [TEX80 in C]
Varian Associates: VAX (VMS)
Voelker-Lehman Systems, Inc: VAX (UNIX)
The Waite Company: Prime [ip]
Westinghouse Electric Co: VAX (VMS) [ip]
Wise Corp: Alpha Micro [ip]
Woods Hole Oceanographic Inst: VAX (VMS)
Zehntel Inc: VAX (UNIX)

Non-academic organizations outside North America where TEX is running or being installed:

AEG/Telefunken Research Inst (Ulm): VAX (VMS)
ASCII Corp (Tokyo): VAX (UNIX) [ip]
AT&T and Philips Telecomm Ind (Netherlands): VAX (VMS)
Bord Fáilte Eireann (Dublin): VAX (VMS)
CILEA (Milan): Univac 1100
CSIRO: CDC Cyber; VAX (VMS)
DESY (Hamburg): IBM (MVS)
Enea Data Svenska AB: VAX (UNIX) [ip]
Enel–Centro Ricerca Automatica (Milan): IBM (MVS)
Gesellschaft Wissen Datenverarbeitung (Göttingen): VAX [ip]
Hartmann + Heenemann KG (Berlin): Siemens [ip]
IBM Finland: IBM (VM/CMS)
Information Science Lab (Japan): Facom [=IBM 370] [ip]
International Telecommun Union (Geneva): Siemens [ip]
Mt Stromlo Observatory (Australia): VAX (VMS)
Mitsubishi Electric Corp: DEC 20
Océ-Nederland B V: PERQ
Philips & MBLE Assoc (Brussels): VAX (VMS)
Quantics (Paris): IBM (VM/CMS) [ip]
Scan Laser (England, Netherlands): Apollo
Schlumberger (England, France, Indonesia, Japan, Singapore, Venezuela): VAX (VMS)
Springer-Verlag (Heidelberg): VAX (VMS)

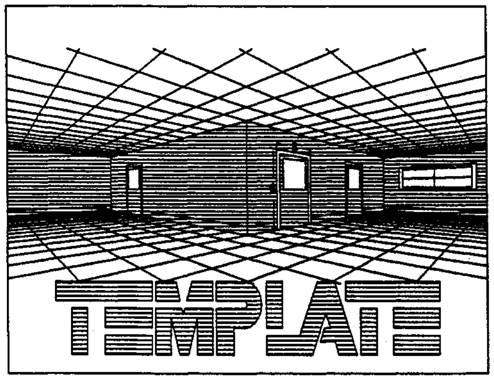
SYSTEMS INC.

TALARIS TALKS TeX

We talk your language. Talaris Systems Inc. offers TeX users full TeX support through its line of TeX-compatible text and graphics software, its family of high quality laser printers, and its extensive library of METAFONT-generated TeX and LATeX fonts.

TeX AND GRAPHICS

We offer QDRIVE™, a TeX-compatible text and graphics merging tool. QDRIVE processes TeX-formatted text pages and merges them with automatically scaled and positioned embedded graphics, like so:



and/or overlaid graphics (see our logo above) for letterheads, borders, forms, grids and "rubber stamp" emulation. QDRIVE is available on VAX/VMS, DEC-10 and DEC-20.

LASER JOY

Talaris offers three fast, affordable, high quality laser printers: the Talaris 2400 (24 pages per minute), the Talaris 1200 (12 pages per minute) and the Talaris 800 (8 pages per minute, desk-top). All are full page bit-map, 300-dots-per-inch text and graphics laser printers. Each functions as a line printer, a plotter (both vector and raster graphics), a Diablo-style printer and a typesetter.

TeXSUPPORT

TeXsupport is our software kit for TeX82 on DEC-10, DEC-20 and VAX/VMS based Talaris laser printers, and for TeX82 on VAX/Berkeley 4.1 or 4.2 UNIX based Talaris laser printers. Each TeXsupport kit comes with QTeX, our TeX82 DVI file post-processor.

TeXsupport comes with a complete library of 185 TeX and LATeX fonts (183 down-loadable, 2 in ROM), spooler or symbiont enhancements, and TeX version 1.1, LATeX and SLiTeX at no extra charge. We also offer 4 optional TeX ROM font sets. (Our TeX fonts are completely compatible with Stanford's.)

FONTS FLASH!

Now available on the VAX/VMS: The **VMS Print Server**, also known as the **Font Daemon**. The Font Daemon provides automatic down-loadable font memory management. While your document is being printed, the Print Server dynamically loads for you any font it encounters in your text. It first checks to see if the referenced font has already been loaded, and if not, automatically loads it for you. If font memory is too full, it deletes the oldest unused font before loading the new one. (Bonus: the Font Daemon works with any text file, not just TeX text files.)

TEMPLATE is a registered trademark of Megatek Corporation.

SYSTEMS INC.

WHY QDRIVE?

Question: Why would a TEX user want to use QDRIVE, our text and graphics merging program? **Answer**: Because QDRIVE has far more expanded capabilities than QTEX.

QTEX is our regular TEX driver; however, its power is limited. QDRIVE, on the other hand, has these expanded features:

- it accepts a variety of files as input—TEX DVI, word processing and ASCII (not just TEX DVI files, as with QTEX);
- it has powerful text and graphics merging capabilities (QTEX has none); and
- it gives far more control over output page parameters (you can specify no less than 11 of them, compared with QTEX's one).

WORDS AND PICTURES

QDRIVE simplifies two types of text and graphics merging applications:

(1) "Internal overlays." These are figures to be embedded in TEX text on the output page. All you have to do is specify, from within the TEX input file, the size available for the figure and the location of the graphics file using the Talaris "grafix" macro. QDRIVE then scales the figure and prints it in the correct position on the output page.

(2) "External overlays." These are whole page overlays that enable you to layer several different pieces of information onto the same page—much like transparent overlays used in viewgraphs. They're useful for layering logos, borders, forms, etc., onto TEX text.

QDRIVE also allows you to:

- mix vector, raster and "native" (printer-dependent) graphics on the same page;
- overlay TEX output onto TEX text; and
- mix page orientations.

OUTPUT CONTROL

For those of us who have not yet mastered Chapter 23 of Dr. Knuth's *The TEXbook* ("Output Routines"), QDRIVE can be an invaluable aid in controlling TEX output generation.

Using QDRIVE's output parameters menu you can do the following:

- turn off automatic centering of TEX pages;
- control the placement of the TEX output page onto the printed page (by specifying the left, top and binding margins); and
- specify page dimensions and orientation.

By layering several TEX output files onto one page, you can create page layouts which would otherwise require writing a new TEX output routine. For instance, you can mix together, on the same page, output generated with both the dual-column and the plain TEX output routines. This greatly increases your productivity when creating unique, one-time page layouts.

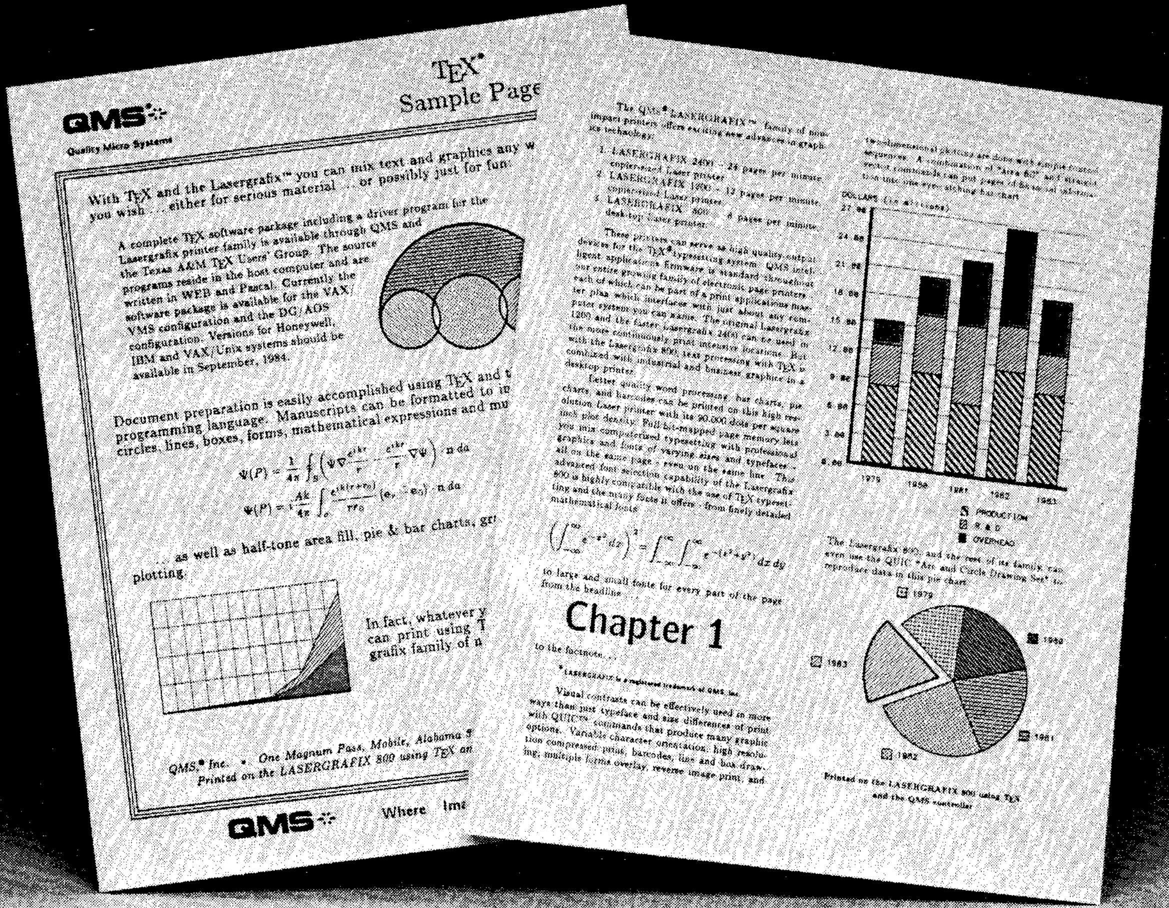
Why QDRIVE? Because of its powerful text and graphics merging capabilities, its versatility as a page layout tool, and its friendliness and ease of use.

For more information on our TEX-related products, call or write us today.

WE TALK TEX!

P.O. Box 8309, La Jolla, CA 92038 (619) 454-3363

COMBINING TEX AND GRAPHICS TAKES IMAGINATION.



QMS Quality Micro Systems

TEX* Sample Page

With TEX and the Lasergrafix™ you can mix text and graphics any w
you wish . . . either for serious material . . . or possibly just for fun:

A complete TEX software package including a driver program for the Lasergrafix printer family is available through QMS and the Texas A&M TEX Users' Group. The source programs reside in the host computer and are written in WEB and Pascal. Currently the software package is available for the VAX/VMS configuration and the DG/AOS configuration. Versions for Honeywell, IBM and VAX/Unix systems should be available in September, 1984.

Document preparation is easily accomplished using TEX and t
programming language. Manuscripts can be formatted to in
circles, lines, boxes, forms, mathematical expressions and mu

$$\Psi(P) = \frac{1}{4\pi}\int_S\left(\Psi\nabla\frac{e^{ikr}}{r} - \frac{e^{ikr}}{r}\nabla\Psi\right)\cdot \mathbf{n}\,da$$

$$\Psi(P) = i\frac{Ak}{4\pi}\int_o \frac{e^{ik(r+r_0)}}{rr_0}(\mathbf{e}_r - \mathbf{e}_0)\cdot \mathbf{n}\,da$$

. . . as well as half-tone area fill, pie & bar charts, gr
plotting.

In fact, whatever y
can print using T
grafix family of n

QMS,® Inc. • One Magnum Pass, Mobile, Alabama 3
Printed on the LASERGRAFIX 800 using TEX an

QMS Where Ima

The QMS® LASERGRAFIX™ family of non-impact printers offers exciting new advances in graph[illegible] technology:

1. LASERGRAFIX 2400 - 24 pages per minute copier-sized laser printer
2. LASERGRAFIX 1200 - 12 pages per minute copier-sized laser printer
3. LASERGRAFIX 800 - 8 pages per minute desk-top laser printer

These printers can serve as high quality output devices for the TEX® typesetting system. QMS intelligent applications firmware is standard throughout our entire growing family of electronic page printers, each of which can be part of a print applications master plan which interfaces with just about any computer system you can name. The original Lasergrafix 1200 and the faster Lasergrafix 2400 can be used in the more continuously print intensive locations. But with the Lasergrafix 800, text processing with TEX is combined with industrial and business graphics in a desktop printer.

Letter quality word processing, bar charts, pie charts, and barcodes can be printed on this high resolution laser printer with its 90,000 dots per square inch plot density. Full bit-mapped page memory lets you mix computerized typesetting with professional graphics and fonts of varying sizes and typefaces - all on the same page - even on the same line. The advanced font selection capability of the Lasergrafix 800 is highly compatible with the use of TEX typesetting and the many fonts it offers - from finely detailed mathematical fonts

$$\left(\int_{-\infty}^{\infty} e^{-x^2}\,dx\right)^2 = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-(x^2+y^2)}\,dx\,dy$$

to large and small fonts for every part of the page from the headline . . .

Chapter 1

to the footnote . . .

* LASERGRAFIX is a registered trademark of QMS, Inc.

Visual contrasts can be effectively used in more ways than just typeface and size differences of print with QUIC™ commands that produce many graphic options. Variable character orientation, high resolution compressed print, barcodes, line and box drawing, multiple forms overlay, reverse image print, and

two-dimensional graphics are done with simple control sequences. A combination of "Area Fill" and straight vector commands can put pages of financial information into one eye-catching bar chart

The Lasergrafix 800, and the rest of its family, can even use the QUIC "Arc and Circle Drawing Set" to reproduce data in this pie chart

Printed on the LASERGRAFIX 800 using TEX and the QMS controller

With TEX software and a QMS Lasergrafix™ non-impact printer you can print just about anything you can imagine.

Documents with arcs, circles, lines, boxes, forms, mathematical expressions. Manuscripts with multiple fonts, halftones, pie charts, bar charts, graphs and vector plotting.

The reason?

All QMS Lasergrafix printers including the 800, 1200 and 2400 are equipped with the firmware necessary to merge high-resolution graphics and letter-quality text on the same page. In any configuration. At a remarkable 90,000 dots per square inch.

All it takes is imagination to combine TEX and graphics. As long as you have a printer with intelligence.

Ask for a demonstration of a QMS Lasergrafix printer today and we'll show you what we mean.

QMS®
QUALITY MICRO SYSTEMS
WHERE IMAGINATION LEADS™

___ Please contact me to arrange a demonstration.
___ Please send me more information on the QMS Lasergrafix™ Family of Non-Impact Printers today. Be sure to include samples of graphics applications.
Name ________ Title ________
Company ________
Address ________
City ________ State/Zip ________ Phone ( ) ________
Type of system you are now using ________
Number of print workstations you are considering ________
Print application for printer ________

P.O. Box 81250, Mobile, Alabama 36689
Telephone (205) 633-4300 • Telex RCA 26 60 13

TB

TEXTSET, Inc. P.O. Box 7993, Ann Arbor, Michigan 48107
direct-typesetting services and software

TEXTSET offers a full range of services and products to TEX users.

DIRECT-TYPESETTING

Direct-typesetting from TEX DVI files, TEX source, and other formatting systems on an Autologic APS–5. Macro consultation and design. Wide range of fonts available including Computer Modern.

TEX SYSTEM CONSULTING

Expert consulting and programming services for in-house TEX users. Design and implementation of complete working TEX systems.

- preprocessors
- TEX installation
- macro package design
- device drivers
- font assistance
- system integration
- special applications support
- programmer level instruction

TEX SOFTWARE

TEX Package software for Sun Workstations.
TEX Screen Preview for Sun Workstations.
DVIAPS, TEX DVI-to-printer driver for Autologic APS–5 series phototypesetters.
DVIXER for Xerox 8700 and 9700.
Device drivers for other selected printers.
Multiple copy discounts, site licenses, and maintenance agreements available.

For more information call Bruce Baker at (313) 996–3566.

Typeset using TEX

More than just a pretty face.

The Metafoundry is dedicated to the creation of beautiful digital typefaces with METAFONT. But, Metafoundry fonts are much more than beautiful. Each has a TEX Font Metric (tfm) file, containing all the kerning, ligature and character size information needed to get the most accurate typesetting. Not only are Metafoundry fonts fully TEX-compatible, they can also be adapted to other systems.

Using METAFONT helps us keep down the cost of changing resolutions and we pass the savings on to you. When you license our fonts, you are entitled to a substantial discount on those fonts in different resolutions.

ROMAN
Chel Sans Serif
Copperplate
Schoolbook

The above are the major families in our catalogue. Also available are:

- Roman and sans serif monospace groups,
- Mathematical and scientific symbols, including extensions,
- Foreign language alphabets (Greek, Cyrillic, Azeri and Slavic) including diacritics,
- The International Phonetic Alphabet.

The Metafoundry's fonts are available in point sizes ranging from 4 to 96 points, at resolutions of 100, 200, 240, 300, 400 and 480 dpi (custom resolutions negotiable). Most fonts are available in text and italic, and in four weights (regular, medium, bold and extra bold). And The Metafoundry's designers are willing to undertake custom and logo design work.

The Metafoundry™

6565 Frantz Rd. • Dublin, OH 43017 • (614) 764-6087

The original of this ad was printed with Metafoundry fonts at only 300 dpi.

TEX CONSULTANT

Michael Spivak

Author of the
AMS-TEX macro package

Write: c/o Publish or Perish, Inc.
1001 Jefferson Plaza, Suite 112
Wilmington, DE 19801
Phone: (713) 523-5700

INSTRUCTORS WANTED – PART TIME

The TEX Users Group is accepting applications from qualified individuals to teach TEX-related two-, three- and five-day seminars at various locations in the U.S. and possibly abroad. Subjects include, but are not limited to, TEX for beginners, TEX for intermediate users, AMS-TEX, font design, output devices, document design, output routines, macro writing, TEX macro packages, integration of text and graphics, and tutorials for TEX users.

Fee, plus expenses. Send resumé, with description of qualifications, to Ray Goucher, TEX Users Group, P. O. Box 9506, Providence, RI 02940-9506, U.S.A.

Request for Information

The TEX Users Group maintains a database and publishes a membership list containing information about the equipment on which members' organizations plan to or have installed TEX, and about the applications for which TEX would be used.

Please answer the questions below, in particular those regarding the status of TEX and the computer(s)/operating system(s) on which it runs or is being installed. (Especially for IBM and VAX, the operating system is more relevant than the model.)

If it has not yet been done for your site, please also answer the questions about output devices on the other side of this form, obtaining information from the most knowledgeable person at your installation if necessary. If this information has already been provided by another TUG member, please indicate that member's name, and the information will be repeated. If you need more space than is provided here, feel free to use additional paper.

If your current listing is correct, you need not answer these questions again. Your cooperation is appreciated.

- *Send completed form with remittance* (checks, money orders, UNESCO coupons) to:
  TEX Users Group
  P. O. Box 594
  Providence, Rhode Island 02901, U.S.A.
- *For foreign bank transfers* direct payment to the TEX Users Group, account #002-610871, at:
  Rhode Island Hospital Trust National Bank
  One Hospital Trust Plaza
  Providence, Rhode Island 02903-2449, U.S.A.
- *General correspondence* about TUG should be addressed to:
  TEX Users Group
  P. O. Box 9506
  Providence, Rhode Island 02940-9506, U.S.A.

Name: ____________________
Home [ ]
Bus. [ ] Address: ____________________
____
____
____

QTY	ITEM	AMOUNT
	1985 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **North America** New (first-time): [ ] $20.00 each Renewal: [ ] $30.00; [ ] $20.00 – reduced rate if renewed before January 31, 1985	
	1985 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **Outside North America** * New (first-time): [ ] $25.00 each Renewal: [ ] $35.00; [ ] $25.00 – reduced rate if renewed before January 31, 1985	
	TUGboat back issues, $15.00** per issue, circle issue(s) desired: 1980 (v. 1) #1; 1981 (v. 2) #1, #2, #3; 1982 (v. 3) #1, #2; 1983 (v. 4) #1, #2; 1984 (v. 5) #1, #2 (after 12/31/84)	
	First Grade TEX: A Beginner's TEX Manual by Arthur L. Samuel – $6.00 each	
	User's Guide to the HP TEX Macros by Susan Daniels – $6.00 each	
	TEX and Metafont: Errata and Changes (final edition, September 1983) – $4.00 each	
	The TEXbook: Errata and Changes (included with TUGboat) – additional copies $3.00 each	
	TEX Lectures on Tape (*Prices reduced* – see cover 3, Vol. 5, No. 2)	

* Air mail postage is included in the rates for all subscriptions and memberships outside North America.
** Discount: 5–7 copies, 10%; 8 or more, 15%

TOTAL ENCLOSED: ____________
(*Prepayment in U.S. dollars required*)

* * * *

Membership List Information

Institution (if not part of address):

Title:
Phone:
Specific applications or reason for interest in TEX:

My installation can offer the following software or technical support to TUG:

Please list high-level TEX users at your site who would not mind being contacted for information; give name, address, and telephone.

Date:
Status of TEX: [ ] Under consideration
[ ] Being installed
[ ] Up and running since
Approximate number of users:
Version of TEX: [ ] SAIL
Pascal: [ ] TEX82 [ ] TEX80
[ ] Other (describe)

From whom obtained:

Computer(s) and operating system(s):

Please answer the following questions regarding output devices used with TEX unless this form has already been filled out by someone else at your installation. Use a separate form for each output device.

Name ______________________ Institution ______________________

A. Output device information
Device name
Model
1. Knowledgeable contact at your site
Name
Telephone
2. Device resolution (dots/inch)
3. Print speed (average feet/minute in graphics mode)
4. Physical size of device (height, width, depth)

5. Purchase price
6. Device type
[ ] photographic [ ] electrostatic
[ ] impact [ ] other (describe)

7. Paper feed [ ] tractor feed
[ ] friction, continuous form
[ ] friction, sheet feed [ ] other (describe)

8. Paper characteristics
a. Paper type required by device
[ ] plain [ ] electrostatic
[ ] photographic [ ] other (describe)

b. Special forms that can be used [ ] none
[ ] preprinted one-part [ ] multi-part
[ ] card stock [ ] other (describe)

c. Paper dimensions (width, length)
maximum
usable
9. Print mode
[ ] Character: ( ) Ascii ( ) Other
[ ] Graphics [ ] Both char/graphics
10. Reliability of device
[ ] Good [ ] Fair [ ] Poor
11. Maintenance required
[ ] Heavy [ ] Medium [ ] Light
12. Recommended usage level
[ ] Heavy [ ] Medium [ ] Light
13. Manufacturer information
a. Manufacturer name
Contact person
Address

Telephone
b. Delivery time
c. Service [ ] Reliable [ ] Unreliable

B. Computer to which this device is interfaced
1. Computer name
2. Model
3. Type of architecture *
4. Operating system

C. Output device driver software
[ ] Obtained from Stanford
[ ] Written in-house
[ ] Other (explain)

D. Separate interface hardware (if any) between host computer and output device (e.g. Z80)
1. Separate interface hardware not needed because:
[ ] Output device is run off-line
[ ] O/D contains user-programmable micro
[ ] Decided to drive O/D direct from host
2. Name of interface device (if more than one, specify for each)

3. Manufacturer information
a. Manufacturer name
Contact person
Address

Telephone
b. Delivery time
c. Purchase price
4. Modifications
[ ] Specified by Stanford
[ ] Designed/built in-house
[ ] Other (explain)

5. Software for interface device
[ ] Obtained from Stanford
[ ] Written in-house
[ ] Other (explain)

E. Fonts being used
[ ] Computer Modern
[ ] Fonts supplied by manufacturer
[ ] Other (explain)

1. From whom were fonts obtained?

2. Are you using Metafont? [ ] Yes [ ] No

F. What are the strong points of your output device?

G. What are its drawbacks and how have you dealt with them?

H. Comments – overview of output device

* If your computer is "software compatible" with another type (e.g. Amdahl with IBM 370), indicate the type here.

Revised 9/84

Request for Information

The TEX Users Group maintains a database and publishes a membership list containing information about the equipment on which members' organizations plan to or have installed TEX, and about the applications for which TEX would be used.

Please answer the questions below, in particular those regarding the status of TEX and the computer(s)/operating system(s) on which it runs or is being installed. (Especially for IBM and VAX, the operating system is more relevant than the model.)

If it has not yet been done for your site, please also answer the questions about output devices on the other side of this form, obtaining information from the most knowledgeable person at your installation if necessary. If this information has already been provided by another TUG member, please indicate that member's name, and the information will be repeated. If you need more space than is provided here, feel free to use additional paper.

If your current listing is correct, you need not answer these questions again. Your cooperation is appreciated.

- *Send completed form with remittance* (checks, money orders, UNESCO coupons) to:
  TEX Users Group
  P. O. Box 594
  Providence, Rhode Island 02901, U.S.A.
- *For foreign bank transfers* direct payment to the TEX Users Group, account #002-610871, at:
  Rhode Island Hospital Trust National Bank
  One Hospital Trust Plaza
  Providence, Rhode Island 02903-2449, U.S.A.
- *General correspondence* about TUG should be addressed to:
  TEX Users Group
  P. O. Box 9506
  Providence, Rhode Island 02940-9506, U.S.A.

Name: ______
Home [ ]
Bus. [ ] Address: ______

______

______

______

QTY	ITEM	AMOUNT
	1985 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **North America** New (first-time): [ ] $20.00 each Renewal: [ ] $30.00; [ ] $20.00 – reduced rate if renewed before January 31, 1985	
	1985 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **Outside North America** * New (first-time): [ ] $25.00 each Renewal: [ ] $35.00; [ ] $25.00 – reduced rate if renewed before January 31, 1985	
	TUGboat back issues, $15.00** per issue, circle issue(s) desired: 1980 (v. 1) #1; 1981 (v. 2) #1, #2, #3; 1982 (v. 3) #1, #2; 1983 (v. 4) #1, #2; 1984 (v. 5) #1, #2 (after 12/31/84)	
	First Grade TEX: A Beginner's TEX Manual by Arthur L. Samuel – $6.00 each	
	User's Guide to the HP TEX Macros by Susan Daniels – $6.00 each	
	TEX and Metafont: Errata and Changes (final edition, September 1983) – $4.00 each	
	The TEXbook: Errata and Changes (included with TUGboat) – additional copies $3.00 each	
	TEX Lectures on Tape (*Prices reduced* – see cover 3, Vol. 5, No. 2)	

* Air mail postage is included in the rates for all subscriptions and memberships outside North America.
** Discount: 5–7 copies, 10%; 8 or more, 15%

TOTAL ENCLOSED: ______
(*Prepayment in U.S. dollars required*)

* * * *

Membership List Information

Institution (if not part of address):

Title:
Phone:
Specific applications or reason for interest in TEX:

My installation can offer the following software or technical support to TUG:

Please list high-level TEX users at your site who would not mind being contacted for information; give name, address, and telephone.

Date:
Status of TEX: [ ] Under consideration
[ ] Being installed
[ ] Up and running since
Approximate number of users:
Version of TEX: [ ] SAIL
Pascal: [ ] TEX82 [ ] TEX80
[ ] Other (describe)

From whom obtained:

Computer(s) and operating system(s):

Please answer the following questions regarding output devices used with TEX unless this form has already been filled out by someone else at your installation. Use a separate form for each output device.

Name ______________________ Institution ______________________

A. Output device information
Device name
Model
1. Knowledgeable contact at your site
Name
Telephone
2. Device resolution (dots/inch)
3. Print speed (average feet/minute in graphics mode)
4. Physical size of device (height, width, depth)

5. Purchase price
6. Device type
[ ] photographic [ ] electrostatic
[ ] impact [ ] other (describe)

7. Paper feed [ ] tractor feed
[ ] friction, continuous form
[ ] friction, sheet feed [ ] other (describe)

8. Paper characteristics
a. Paper type required by device
[ ] plain [ ] electrostatic
[ ] photographic [ ] other (describe)

b. Special forms that can be used [ ] none
[ ] preprinted one-part [ ] multi-part
[ ] card stock [ ] other (describe)

c. Paper dimensions (width, length)
maximum
usable
9. Print mode
[ ] Character: ( ) Ascii ( ) Other
[ ] Graphics [ ] Both char/graphics
10. Reliability of device
[ ] Good [ ] Fair [ ] Poor
11. Maintenance required
[ ] Heavy [ ] Medium [ ] Light
12. Recommended usage level
[ ] Heavy [ ] Medium [ ] Light
13. Manufacturer information
a. Manufacturer name
Contact person
Address

Telephone
b. Delivery time
c. Service [ ] Reliable [ ] Unreliable

B. Computer to which this device is interfaced
1. Computer name
2. Model
3. Type of architecture*
4. Operating system

C. Output device driver software
[ ] Obtained from Stanford
[ ] Written in-house
[ ] Other (explain)

D. Separate interface hardware (if any) between host computer and output device (e.g. Z80)
1. Separate interface hardware not needed because:
[ ] Output device is run off-line
[ ] O/D contains user-programmable micro
[ ] Decided to drive O/D direct from host
2. Name of interface device (if more than one, specify for each)

3. Manufacturer information
a. Manufacturer name
Contact person
Address

Telephone
b. Delivery time
c. Purchase price
4. Modifications
[ ] Specified by Stanford
[ ] Designed/built in-house
[ ] Other (explain)

5. Software for interface device
[ ] Obtained from Stanford
[ ] Written in-house
[ ] Other (explain)

E. Fonts being used
[ ] Computer Modern
[ ] Fonts supplied by manufacturer
[ ] Other (explain)

1. From whom were fonts obtained?

2. Are you using Metafont? [ ] Yes [ ] No

F. What are the strong points of your output device?

G. What are its drawbacks and how have you dealt with them?

H. Comments – overview of output device

* If your computer is "software compatible" with another type (e.g. Amdahl with IBM 370), indicate the type here.

Revised 9/84

AMERICAN MATHEMATICAL SOCIETY

P. O. Box 6248

Providence, RI 02940-6248

Ordered by: ______________________ Mail to (if different): ______________________

______________________ ______________________

______________________ ______________________

______________________ ______________________

QTY	CODE	AUTHOR and TITLE	PRICE
	TEXBKT	Donald E. Knuth: The TEXbook @ $15 each	$
	JOYTT	Michael Spivak: The Joy of TEX @ $10 each (revised preliminary edition, 1982, with AMS-TEX82 supplement)	
		Shipping and Handling ☐ Surface ☐ Air	
		Total due (**All orders must be prepaid**)	$

To order using VISA or MasterCard (for **book orders only**)

☐ VISA ☐ MasterCard, Account Number ______________________

Expiration Date ____________ Signature ______________________

Ordering, Shipping and Handling

Books are sent via surface mail (UPS to U.S. addresses and printed matter elsewhere) unless air delivery is requested. The shipping and handling charges for book orders are shown below.

	First Book	Each Additional	Maximum
Surface	$2	$1	$ 25
Air	$5	$3	$100

Ordering Information: Individuals may use **VISA** or **MasterCard** to order these books by mail or phone. For callers in the continental U.S., the Society's toll-free number, 800-556-7774, is attended from 8:00 a.m. to 4:15 p.m., Eastern Time, Monday through Friday except on holidays. Send **charge orders** to the AMS, P.O. Box 6248, Providence, RI 02940-6248. Send orders being paid by **foreign transfer** to Rhode Island Hospital Trust National Bank, Account #000-753-111, One Hospital Trust Plaza, Providence, RI 02903, U.S.A.

10/84

Staple here

Fold here

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 554B PROVIDENCE, RI

POSTAGE WILL BE PAID BY ADDRESSEE

AMERICAN MATHEMATICAL SOCIETY
P.O. Box 1571
Annex Station
Providence, RI 02901-1571

Fold here

Fasten PAYMENT securely

TEX82 ORDER FORM

The latest official versions of TEX software and documents are available from Maria Code by special arrangement with the Computer Science Department of Stanford University.

Nine different tapes are available. The generic distribution tape contains the source of TEX82 and WEB, the test program, a few "change" files, the collection of fonts in TFM format, and other miscellaneous materials; a PASCAL compiler will be required to install programs from a generic tape. The AMS-TEX macro package is included on the TEX distribution tapes; other macro packages, including LaTEX and HP TEX, will be added as they become available. The special distribution tapes are for the indicated systems only, and should be ordered for these systems instead of a generic tape. Two tapes are PXL font collections covering various magnifications at 200/240 dots/inch and 300 dots/inch respectively. The METAFONT tape contains the SAIL source for the METAFONT program and includes the .MF source files.

Each tape will be a separate 1200 foot reel which you may send in advance or purchase (for the tape media) at $10.00 each. Should you send a tape, you will receive back a different tape. Tapes may be ordered in ASCII or EBCDIC characters. You may request densities of 6250, 1600 or 800 (800 is discouraged since it is more trouble to make).

The tape price of $82.00 for the first tape and $62.00 for each additional tape (ordered at the same time) covers the cost of duplication, order processing, domestic postage and some of the costs at Stanford University. Extra postage is required for first class or export.

Manuals are available at the approximate cost of duplication and mailing. Prices for manuals are subject to change as revisions and additions are made. It is assumed that one set of manuals will suffice you. If you require more than two sets, please write for prices since we must ask for more money for postage and handling.

Please send a check or money order (payable on a US bank) along with your order if possible. Your purchase order will be accepted, as long as you are able to make payment within 30 days of shipment. Please check this out before sending a purchase order since many large firms seem to be unable to make prompt payment (or don't worry about it).

The order form contains a place to record the name and address of the person who will actually use the TEX tapes. This should *not* be someone in the purchasing department.

Your order will be filled with the most recent versions of software and manuals available from Stanford at the time your order is received. If you are waiting for some future release, please indicate this. Orders are normally filled within a few days. There may be periods (like short vacations) when it will take longer. You will be notified of any serious delays. If you want to inquire about your order you may call Maria Code at (408) 735-8006 between 9:30 a.m. and 2:30 p.m. West Coast time.

If you have questions regarding the implementation of TEX or the like, you must take these to Stanford University or some other friendly TEX user.

Now, please complete the order form on the reverse side.

TEX82 ORDER FORM

**** TAPES **** density (6250, 1600 or 800) = ____________

TEX generic distribution tapes (PASCAL compiler required):

________	ASCII format	________	EBCDIC format

TEX distribution tapes in special formats:

________	VAX/VMS Backup format	________	IBM VM/CMS format
________	DEC 20/Tops-20 Dumper format	________	* IBM MVS format

* Not yet available; call before ordering

Font tapes:

________	Font library (200/240 dots/inch)	________	Font library (300 dots/inch)
________	METAFONT (SAIL compiler required)		

________ Total number of tapes.

Tape costs: $82.00 for first tape; $62.00 for each additional.

Tape cost = $ ____________

Media costs: $10.00 for each tape required.

Media cost = $ ____________

**** MANUALS ****

________	TEX82 – $20.00	________	Test Manual – $8.00
________	WEB – $10.00	________	TEXware – $8.00
________	TEXbook – $20.00	________	LaTEX (preliminary edition) – $8.00

Manuals cost = $ ____________

California orders only: add sales tax = $ ____________

Domestic book rate: no charge.
Domestic first class: $2.50 for each tape and each manual.
Export surface mail: $2.50 for each tape and each manual.
Export air mail to North America: $4.00 each.
Export air mail to Europe: $7.00 each.
Export air mail to other areas: $10.00 each.

Postage cost = $ ____________

(make checks payable to Maria Code) Total order = $ ____________

Name and address for shipment: Person to contact (if different):

____________________ ____________________

____________________ ____________________

____________________ ____________________

____________________ ____________________

Telephone ____________

Send to: Maria Code, DP Services, 1371 Sydney Dr., Sunnyvale, CA 94087

5/84