題名	デジタル印刷術
Title	Digital Typography
著者名	ドナルド・アーヴィン・クヌース
Author(s)	Donald Ervin Knuth
言語 Language	日本語・英語　Japanese, English
書名	稲盛財団：京都賞と助成金
Book title	The Inamori Foundation: Kyoto Prizes & Inamori Grants
受賞回	12
受賞年度	1996
出版者	財団法人 稲盛財団
Publisher	The Inamori Foundation
発行日 Issue Date	11/1/1997
開始ページ Start page	82
終了ページ End page	109
ISBN	978-4-900663-12-3

デジタル印刷術

ドナルド・アービン・クヌース

私は物心のついた頃からずっと、本をこよなく愛好してきました。最初、私は両親からたくさんの本を読み聞かせてもらったのですが、それは当時のアメリカとしては珍しいことでした。といいますのは、1940年代には、小さいうちに知的なことに接した子どもは、後に学校に上がるときにそれに飽きてしまう、ということが広く信じられていたからです。この両親のおかげで、私は4歳で、ミルウォーキー公立図書館の「本の虫クラブ」の最年少会員となったのです(photo 1)。

おそらく、私が教育を受けた期間を通してずっと、一度として退屈した覚えがないのは、このように小さな頃から本と親しんできたことに起因しているのでしょう。実際、私は、現代社会は「退屈」という観念にすっかり巻き込まれていると思います。人は互いに退屈していると言い合いますが、私にとってそれは、けしからぬ、恥ずべき告白であります。どうして、他人に楽しませてもらわなければならないのでしょう。自分のしていることに何の興味も見いだせず、常に外からの刺激や楽しみを必要としている人間は、人生の喜びの多くを見失っているのです。

私に関してはいつもその逆でした。私にはつい違った方向にいってしまう傾向があります。読書や勉強をしているときに、その本の第1章に夢中になって、最終章を読む時間が足りなくなってしまうこともしばしばです。

私が5歳のときのことですが、両親は、私をひとりで路面電車に乗せて、ダウンタウンにある図書館に行かせたことがありました。私はそこですっかり児童書のとりこになってしまい、時間になっても家に帰ってこない私を心配した両親は、図書館に電話をかけました。そして、夜警員のひとりが書庫まで探しにきて、喜々として本を読んでいる私を発見したのです。私は、図書館が閉館されて誰もが家に帰ってしまっていたことなど、まったく気づかなかったのでした。現在でも、私の妻は、私が図書館へ行ったら、まず帰宅が遅くなることを知っています。

事実、私は常に本を愛してきましたが、そればかりでなく、その中にある「文字」の一つ一つをも愛してきました。これは、私が小さい頃いちばん初めに出会ったABCアルファベットの本の1ページです(photo 2)。奇妙なことですが、私は、活字のセリフ(ひげ)一つ一つに、小さくxで印をつけて、セリフの数を数えたのでした。アルファベットKの文字にはセリフが7つ、P(photo 3)には4つあり、O(photo 4)にはセリフはありません。

このことから、私が文字と同様、数字も好きなことがおわかりいただけるでしょう。私がスタンフォード大学の教授になった頃には、自分の主たる才能を生かせるのはコンピュータープログラミングであると悟り、私自身の本の執筆を始めました。私は、

DIGITAL TYPOGRAPHY

Donald Ervin Knuth

I have been in love with books ever since I can remember. At first, my parents read to me a lot—an unusual practice in America at the time, because the prevailing "wisdom" of the 1940s was that a child who is exposed to intellectual things at an early age will be bored later when entering school. Thanks to my parents, I became at age four the youngest member of the Book Worm Club at the Milwaukee Public Library [photo 1].

That early experience with books is probably responsible for the fact that I don't remember ever being bored, throughout my education. In fact, I think contemporary society is all mixed up in its concept of "boredom": People often say to each other that they are bored, but to me this is almost a shocking, shameful admission. Why should it be somebody else's duty to entertain us? People who can't find anything of interest in what they are doing, who constantly need external sources of stimulation and amusement, are missing most of life's pleasures.

With me it has always been the opposite: I tend to err in the other direction. I often get so interested in Chapter 1 of the books that I'm reading or studying, I don't have much time to read the final chapters.

Once, when I was five years old, my parents let me take the streetcar to the downtown library by myself, and I was absolutely fascinated by the children's books. When I didn't come home on time, my parents were worried and phoned the library. One of the night staff went looking and found me in the stacks, reading happily—I had no idea that the library was closed and that everyone else had gone home! Even today my wife knows I go to the library, I'll probably come home late.

In fact, not only have I always loved books, I've also been in love with the individual *letters* in books. Here's a page from the first ABC alphabet book that I had when I was little [photo 2]. Curiously, I marked each serif in the letters with a little x, and I counted the serifs: The letter K has 7 serifs. The letter P [photo 3] has 4; the letter O [photo 4] has none.

From this you can see that I like numbers as well as letters. By the time I became a professor at Stanford I had learned that my main talents were associated with computer programming, and I had begun to write books of my own. My first book, Volume 1 of *The Art of Computer Programming,* came out in 1968, and Volume 2 was ready a year later [photo 5].

I was excited to see these volumes not only because I was pleased with the information they contained, but also because of the beautiful typography and layout. These books were produced with the best, time-tested methods known for the presentation of technical material. They appeared in the same classic style

最初の本である『The Art of Computer Programming』の第１巻を1968年に出版し、第２巻をその翌年に完成させました(photo 5)。

私はそれらの本を見て、そこに書かれた情報に満足したからだけでなく、本の組版とレイアウトの美しさのために感激を覚えました。それらの本は、技術的資料の表現で有名な、長い年月をかけて証明された最良の方法で作られていて、私のお気に入りの大学の教科書に使われているのと同じ伝統的な形式をとったものでした。ですから、それらを眺めることは、読むのと同様に楽しいことだったのです。

この本は、２種類の機械から成るモノタイプと呼ばれる19世紀の技術を使って作成されました。まず、284のキーのついた複雑な空気式鑽孔キーボードが使われました(photo 6)。この機械で、自動演奏用のピアノロールのような穴の空いた紙テープを作りました。写真の上にあるのがこのテープです。次に、このテープは、特別な鋳植機(photo 7)にかけられ、溶解した高温の鉛から一つ一つの活字を作るための制御に使用されました。

この機械を使って数学テキストを植字する工程は非常に複雑でした(photo 8)。まず、熟練した植字工が、２工程で数式を組みました。最初に基準となる行の文字や記号と上付き添字（写真の黄色で強調してある部分）を組み、次に下付き添字を組みます（写真の中ではij)。キーパンチャーは、下付き添字を適切に配置するため充分な間隔を取れるよう、各文字の幅を把握しておく必要がありました。数式が地金に鋳植された後、もう一人の熟練工が、残りの大きな記号（大かっこや、ΠやΣ等の記号）を手作業で挿入しました。モノタイプを用いて数式を組む方法を知る人は、世界中でもわずか数十人ほどでした。私は以前、第１巻と第２巻のキーパンチをしてくれた植工のエリックと会う機会に恵まれました。そこで私は、彼がアメリカに住み、世界で最も高等な数学の本を何冊か組版したにもかかわらず、強いロンドンの下町訛りで話すことを知って驚きました。

コンピューター科学に関する本のために、印刷機がすでに数式を組む際に直面していた問題はさらに複雑になりました。コンピューター科学の研究者たちは、計算機が扱う文を表現するために、タイプライター体と呼ばれる特別な書体を使う必要があります。例えば、こちらの、第２巻から抜粋したページの、コンピュータープログラムの一部をご覧ください(photo 9)。私は「ＯＦＬＯ」のようなタイプライター体と、通常の書体を組み合せなければなりませんでした。当初、私は、特殊なアルファベット文字はモノタイプでは印字できないと聞いていました。従来の数学の公式に関して、すでにモノタイプ技術は、その限界にきていたからです。しかし後になって、エリック


Too young to read books himself, Donald Knuth, 6, of 2961 N. 18th st. is nevertheless the youngest member of the public library's Ancient Order of Book Worms. His father reads storybooks to him. Wednesday he gave oral reports on his "reading" in the young people's room of the library. Some of the stories he reported on were "Country Bunny," "Pokey Bear," "When the Rest Children Wake Up" and "Baker the King."

photo 1

photo 2

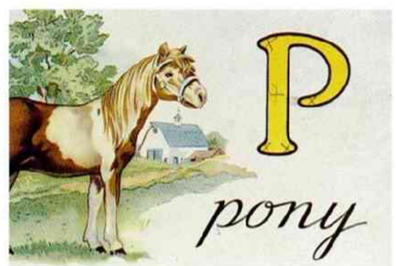


photo 3

photo 4

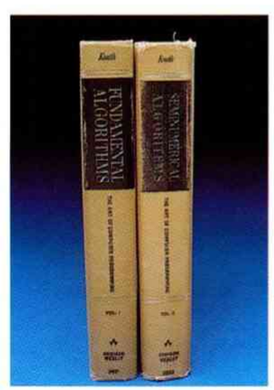


photo 5

photo 6

と彼の上司は、その問題の解決法を見いだしました。私は、O（アルファベットのオー）と0（数字のゼロ）をはっきりと区別するために、角張った新しいタイプライター体の文字Oを必要としていたことを強調したいと思います。

1960年代には、モノタイプのような鉛版鋳植機に代わって、写真技術を用いた新しい機械が登場し始めました。この新しい機械は写真乾板に一文字ずつ感光させてページを作成するもので、回転文字盤とレンズを精巧に組み合せることで文字を正しく配置していきます。1973年、『The Art of Computer Programming』の第3巻が出版されたすぐ後に、出版社はモノタイプ機を売却し、エリックは職を失いました。第1巻と第3巻の新版は、それまでの版で読者が見つけた誤植を修正して1975年に出版されました。この改訂版は、まだモノタイプ技術が使われていたヨーロッパで活字を組みました。

第2巻の第2版の準備も終えていましたが、本全体をもう一度植字し直すことが必要でした。出版社は、1969年と同じ方法で1976年に本を印刷すると、費用が非常に高くつくことに気づきました。そのうえ、初版で用いた書体は写真植字機では使えませんでした。そこで私はカリフォルニアからマサチューセッツに飛び、緊急の話し合いをもちました。出版社は良質の組版が最重要であることを認め、翌月から初版のものに匹敵する新しいフォントを手にいれるため力を尽くしてくれました。

しかし、結果はたいへん残念なものでした。例えば、これが新しいフォントを「調整した」第2バージョンの活字です(photo 10)。最初のものよりはずいぶん改善されたものの、まだまだ満足いくものではありませんでした。「NORM」の「N」の字は傾いていますし、「effect」の「ff」は色が濃すぎ、「multiple」の「ip」は接近しすぎているといった具合です。

私は途方に暮れました。15年も費やして書いた本が恐ろしい姿になろうとしているなら、もう本を書く気が失せてしまいます。こんな本に誇りを持てるでしょうか。

この窮地から脱する方法を思いついたのは数か月後のことです。印刷技術においてもう一つの画期的な変化がもたらされたことを知ったのです。最新の印刷機はアナログではなくデジタル方式で、フィルムに像を写し出しました。これはテレビと本物の映画との違いのようなものです。文字は小さなドットから形作られ、電子パルスがオンかオフかによって決まります(photo 11)。そうです。これなら私の理解の範疇にあります。野球場のスコアボードのライトと同じで、とても単純です。

冶金学や鉛版鋳植はいつも、私にはまったく不可解なものでした。レンズや機械的に軸を調節する装置も私には理解できませんでした。しかし小さなドットでできた文

that had been used in my favorite college textbooks. So it was a pleasure to look at these volumes as well as to read them.

They were produced with 19th-century technology called Monotype, involving two kinds of machines. First, there was a complex pneumatic keyboad with 284 keys [photo 6]. This machine produced a punched paper tape something like a player-piano roll; you can see this tape at the top of the picture. The paper tape was then used to control at special casting machine [photo 7] that produced individual pieces of type from hot molten lead.

The process of typesetting mathematics with such machines was very complicated [photo 8]. First, a specially trained typist would key in the formula by making two pases: One for the letters and symbols on the main line, and their superscripts (the characters '$|\det(a)| \leqq a^{2}$' high-lighted in yellow); a second pass was made for the subscripts (the characters '$_{ij}$' in this example). The keyboard operator had to know the width of each character so that he could leave just enough space to make the subscripts line up properly. After the formula had been cast into metal, another specially trained technician inserted the remaining large symbols (the big parentheses and symbols like Π and Σ) by hand. Only a few dozen people in the world knew how to typeset mathematical formulas with Monotype. I once had the privilege and pleasure of meeting Eric, the compositor who did the keyboarding for Volumes 1 and 2; I was surprised to discover that he spoke with a very strong London-Cockney accent, although he lived in America and was responsible for some of the world's most advanced books in mathematics.

Books on computer science added a new complication to the difficulties that printers already faced in mathematical typesetting: Computer scientist need to use a special style of type called typewriter type, in order to represent the textual material that machines deal with. For example [photo 9], here's another portion of a page from Volume 2, part of a computer program. I needed to combine typewriter type like the word 'OFLO' with the ordinary style of letters. At first I was told that an extra alphabet would be impossible with Monotype, because traditional math formulas were already stretching Monotype technology to its limits. But later, Eric and his supervisor figured out how to do it. Notice that I needed a new, squarish looking letter O in the typewriter style, in order to make a clean distinction between O (oh) and 0 (zero).

New machines based on photography began to replace hot-lead machines like the Monotype in the 1960s. The new machines created pages by exposing a photographic plate, one letter at a time, using an ingenious combination of rotating disks and lenses to put each character in its proper position. Shortly after

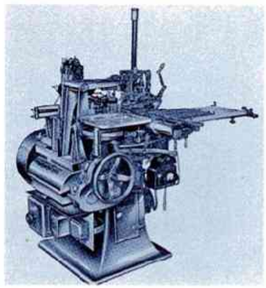
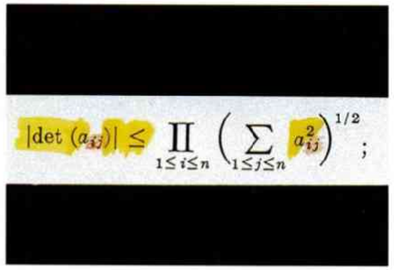


photo 7



$$|\det(a_{ij})| \leq \prod_{1\leq i\leq n}\left(\sum_{1\leq j\leq n} a_{ij}^2\right)^{1/2};$$

photo 8

```
      LDA A:  ADD B;  SUB C;  STA D;                    (7)
espond to the floating-point coding sequence
STA ACC;  LDA B,  JMP FADD;  LDA C,  JMP FSUB;  STA D.
(Addition, subtraction, and normalization). The following p
ine for Algorithm A, and it is also designed so that the normal
be used by other subroutines which appear later in this secti
m and in many other programs throughout this chapter, OFLO
utine which prints out a message to the effect that MIX's o
unexpectedly found to be "on."

  EQU   1:1        Definition of exponent field.
B STA   TEMP       Floating-point subtraction subroutine:
  LDAN  TEMP       Change sign of operand.
```

photo 9

```
(Addition, subtraction, and normalization). The fol
ne for Algorithm A, and it is also designed so that th
be used by other subroutines which appear later in
n and in many other programs throughout this
subroutine which prints out a message to the ef
gle was unexpectedly found to be "on." The byte si
tiple of 4. The normalization routine NORM assum
f, where rA = 0 implies rX = 0 and rI2 < b.

EQU    1:1      Definition of exponent field
STA    TEMP     Floating-point subtraction
LDAN   TEMP     Change sign of operand.
```

photo 10

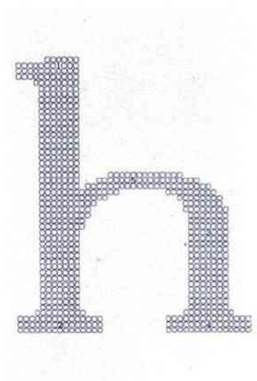


photo 11

Volume 3 of *The Art of Computer Programming* came out in 1973, my publisher sold its Monotype machine and Eric had to find another job. New printings of Volume 1 and Volume 3 were published in 1975, correcting errors that readers had found in the earlier printings; these corrections were typeset in Europe, where Monotype technology still survived.

I had also prepared a second edition of Volume 2, which required typesetting that entire book all over again. My publishers found that it was too expensive in 1976 to produce a book the way it had been done in 1969. Moreover, the style of type that had been used in the original books was not available on photo-optical typesetting machines. I flew from California to Massachusetts for a crisis meeting. The publishers agreed that quality typography was of the utmost importance; and in the next months they tried hard to obtain new fonts that would match the old ones.

But the results were very disappointing. For example [photo 10], here's some of the type from the second, "tuned up" version of their new fonts. These were much improved from the first attempt, but still unacceptable. The "N" in "NOAM" was tipped; the "ff" in "effect" was much too dark; the letters "ip" in "multiple" were too close together; and so on.

I didn't know what to do. I had spent 15 years writing those books, but if they were going to look awful I didn't want to write any more. How could I be proud of such a product?

A possible way out of this dilemma presented itself a few months later, when I learned about another radical change in printing technology. The newest machines made images on film by digital instead of analog means—something like the difference between television and real movies. The shapes of letters were now made from tiny little dots, based on electronic pulses that were either ON or OFF [photo11]. Aha! This was something I could understand! It was very simple, like the lights on a scoreboard at a sports match.

Metallurgy and hot lead have always been complete mysteries to me; neither have I understood lenses or mechanical alignment devices. But letters made of little dots—that's computer science! That's just bits, binary digits, 0s and 1s! Put a 1 where you want ink, put a 0 where you don't want ink, and you can print a page of a book!

I had seen digital letterforms before, but only on crude machines. Computer scientists had been experimenting for many years with a machine called the Xerox Graphics Printer, which had been invented in England about 1961 but not controlled by computers until the 70s. This machine made letters out of dots, but

字なら、これはコンピューター科学です。ビットと同じもの、2進数、つまり0と1です。インクをのせたいところは1に、のせたくないところは0にすれば、これで本の1ページが印刷ができるのです。

それまでのデジタルによる字体を見たことがありましたが、その機械は不完全でした。コンピューター科学の研究者たちは長年かけてXGP（ゼロックス・グラフィック・プリンタ）と呼ばれる機械の試験を行っていました。XGPは1961年にイギリスで発明されましたが、コンピューター制御になったのは1970年代のことでした。この機械ではドットで文字を作っていましたが、ドットはあまり小さくありませんでした。1インチ当たり180ドットしかなく、文字の端がギザギザでした。XGPを扱うのは楽しいことでしたが、この機械で実際の本を作ることができるとは思いませんでした。単純すぎて安物のイミテーションしか作れそうになかったのです。電子楽器と本物のピアノやバイオリンの音色とが違うように……。

しかし、1977年2月、初めて私は高品質のデジタル植字機による印刷物を目にしたのです。1インチ当たりのドット数は1,000以上あり、印刷は完全で、それまで見た最高の地金組版と比べてまったく遜色ありませんでした。そのとき私は、ドットが充分に小さければ、物理的に考えてなめらかなカーブを描けることを悟ったのです。そして、人間の目は個々の棒細胞と円錐細胞からできた、本来デジタルなものであることを思い出しました。こうして、デジタル植字機で実際に最高品質の本を印刷できることが初めてわかったのです。

デジタルカメラは、従来の写真のように細部を正確にとらえることはできません。また高品位テレビでも、画質はビスタビジョンの映画にかないません。しかし、紙の上の印刷についてなら、デジタル方式は適しています。

つまり、本を美しく印刷するという問題は、冶金技術の問題から光学の問題へ、そしてさらにコンピューター科学の問題へと移りました。グーテンベルグが活版印刷で本を印刷したという事実は、500年前に起こった歴史上の過去の遺物になってしまいました。新しい植字機の登場で、古い機械的なやり方はまったく無用のものとなってしまいました。組版の将来は0と1の組み合わせを生み出せる人間、つまり数学者とコンピューター科学の研究者にかかっています。

このことに気づいてからは、組版の問題に自ら取り組まずにはいられませんでした。自分が取り組んでいたあらゆることをやめ、私は第4巻の初めの100ページを書き終えたばかりだったのですがそれもやめて、出版社と私が第2巻の新版に必要としている0と1の組み合わせを作り出すプログラムを書こうと決心したのです。

the dots weren't very small. There were only about 180 dots per inch, so the letters had lots of jagged edges. It was fun to play with the Xerox Graphics Printer, but I never expected that such a machine could produce real books. It seemed too simple, capable only of making cheap imitations—like the difference between an electronic synthesizer and a real piano or violin.

But in February 1977 I saw for the first time the output of a high-quality digital typesetter, which had more than 1,000 dots per inch... and it looked perfect, every bit as good as the best metal typography I had ever seen. Suddenly I saw that dots of ink will form smooth-looking curves if the dots are small enough, by the laws of physics. And I remembered that human eyes are inherently digital, made from individual rod and cone cells. Therefore I learned for the first time that a digital typesetting machine was indeed capable of producing books of the highest conceivable quality.

Digital cameras don't capture all the sharp details of traditional photographs. High-definition television can't match the quality of a Vista Vision movie. But for ink on paper, a digital approach is as good as any other.

In other words, the problem of printing beautiful books had changed from a problem of metallurgy to a problem of optics to a problem of computer science. The fact that Gutenberg had made books from movable metal type was suddenly only a 500-year-long footnote to history. The new machines have made the old mechanical approaches essentially irrelevant: The future of typography depends on the people who know the most about creating patterns of 0s and 1s; it depends on mathematicians and computer scientists.

When I realized this, I couldn't resist tackling the typography problem myself. I dropped everything else I was doing—I had just finished writing the first 100 pages of Volume 4—and decided to write computer programs that would generate the patterns of 0s and 1s that my publishers and I needed for the new edition of Volume 2.

At first I thought it would be easy; I expected that the job could be done in a few months. In March of 1977 I wrote to my publishers that I thought I would have first proofs ready in July. Boy, was I wrong! All my life I have underestimated the difficulty of the project I've embarked on, but this was a new personal record for being too optimistic.

In the first place, almost nobody else in computer science was doing this kind of work, so it was difficult to get financial support. The typesetting machine was very expensive, too much for our university budget. Moreover, that machine was designed to be run 24 hours per day by trained operators; I was just a single

当初私は簡単なことだと考えていました。この仕事はたかだか数か月で終わるだろうと思っていたのです。1977年の3月に、出版社に宛てて第1回の校正刷りは7月に用意できるだろうと書き送りました。とんでもありませんでした。私は着手した仕事の困難さをいつも過小評価してきましたが、これほど楽観的だったのは新記録です。

まず第一に、コンピューター科学の世界でこの種の仕事に携わる者はほかにいなかったために、財政的援助を得るのが困難でした。植字機は大学の予算からすると非常に高価なものでした。また植字機は、熟練工の手によって1日24時間運転し続けるように設計されていました。私は奇妙な数学的発想を持っているだけの人間で、印刷業界での経験は一切ありませんでした。それでも、私の書いたプログラムが動けばどこかのデジタル植字機を使わせてもらえるだろうと信じていました。

また、ニワトリが先か、卵が先か、という問題もありました。文字や数学記号のフォントがなければ印刷することはできませんが、私の必要とするフォントのデジタル版は存在しませんでした。一方、フォントを印刷しなければフォントのデザインをすることは容易ではありません。一度に二つのことが必要だったのです。デジタル化されたフォントはほかにありましたが、私は自力でフォントを定義しようと決心し、純粋な数学公式を用いて自分で制御しました。そうすれば、何か技術的な変更があったときでも初めからやり直す必要がなくなります。私のプログラムがページの上の0と1をすべて管理すれば、本の印刷をはっきりと定義することができます。

出版社から、第1巻の初版で用いたモノタイプの印刷の原版を受け取りました(photo 12)。そして、文字の形を描くための数学公式を見つけるのは容易だろうと思いました。ジョン・ワーノックがゼロックス研究所で同じようなことをしているのを知っていたので、私のフォント作成にゼロックス社の研究施設を使わせてもらえないだろうかと頼みました。返事はイエスでしたが、落とし穴がありました。ゼロックス社は、私がゼロックス社の設備を使って開発したフォントのすべての使用権を主張したのです。もちろん彼らに特権はあるでしょうが、こんな主張は受け入れられません。数学の公式は決して誰の「所有物」でもないのです。数学は神のものです。

そこで私は、スタンフォード大学の人工知能研究所へ赴きました。研究所にあるテレビカメラを使えば、文字を拡大したり文字をデジタル化して記録することができます。しかし残念ながら、テレビカメラでは正確な画像は得られず、像は歪んでいました。さらに悪いことに、部屋の照明の明るさがわずかに違っただけでテレビの像は大きく変化しました。これでは一文字ごとに一貫したデータを得ることはできません。テレビカメラによるフォントは、私が以前却下したアナログ機のフォントよりもずっ

individual with strange mathematical ideas and no experience in the printing industry. Still, I assumed that if I could get my computer program working, I'd be able to borrow time on some digital typesetting machine.

There also was a chicken-and-egg problem. I couldn't set type until I had fonts of letters and mathematical symbols, but the fonts I needed did not exist in digital form. And I could not readily design the fonts until I could set type with them. I needed both things at once. Other fonts had been digitized, but I had resolved to define the fonts by myself, using purely mathematical formulas under my own control. Then I could never have to face the problem that another change in technology might upset the applecart again. With my own computer program controlling all aspects of the 0s and 1s on the pages, I would be able to define the appearance of my books once and for all.

My publishers provided me with original copies of the Monotype images that had been used to make the first edition of Volume 1 [photo 12]. So I thought it would be easy to find mathematical formulas to describe the shapes of the letters. I had seen John Warnock doing similar things at Xerox Research Center, so I asked if I could use Xerox's lab facilities to create my fonts. The answer was yes, but there was a catch: Xerox insisted on all rights to the use of any fonts that I developed with their equipment. Of course that was their privilege, but such a deal was unacceptable to me: A mathematical formula should never be "owned" by anybody! Mathematics belongs to God.

So I went to Stanford's Artificial Intelligence lab, which had a television camera that I could use to magnify the letters and capture them in digital form. Unfortunately, the television camera did not give a true picture—the image was distorted. Even worse, a tiny change in the brightness of the room lights made a tremendous change in the television images. There was no way I could get consistent data from one letter to another. With that TV camera my fonts would look much worse than the fonts I had rejected from the non-digital machine.

I tried photographing the pages and magnifying them by projecting the images on the wall of my house, tracing the enlarged outlines with pencil and paper. But that didn't work either.

Finally, a simple thought struck me. *Those letters were designed by people.* If I could understand what those people had in their minds when they were drawing the letters, then I could program a computer to carry out the same ideas. Instead of merely copying the form of the letters, my new goal was therefore to copy the intelligence underlying that form. I decided to learn what type designers knew, and to teach that knowledge to a computer.

と見劣りしました。

私は各ページを写真に焼き付け、像を家の壁に映し出して拡大し、輪郭を紙と鉛筆でなぞることを試みました。しかしこれも失敗に終わりました。

そして最後に、単純な考えが浮かびました。「文字は人間がデザインしたものだった」と。もし人が文字を描くときに心の中で何を思っているのか理解できれば、コンピューターに同じ考え方を実行させることができます。こうして、ただ単に文字の形をコピーするのではなく、文字の形の中に潜む知的情報をコピーすることが新たな目標となりました。活字設計者の頭の中を学び、その知識をコンピューターに教え込むことにしました。

この考えの結果、METAFONTと呼ばれるコンピューターシステムが生まれることになります。これを今、皆さんにお見せしたいと思います。例えば、これは私が最終的に決定したコンピュータープログラムを使って文字Aを作る方法です。この文字の重要となる点は、すべてここに表示されているグリッドに基づいています(plate 1)。しかし、もちろん実際には目に見えません。このグリッドや通常のテキストのフォントの仕様に基づき、コンピューターはまず太縦線のストロークを描きます(plate 2)。上のほうが太すぎるのでこのストロークの一部を削る必要があります(plate 3)。そして次に左に斜行するストロークを加え(plate 4)、横棒を加えます(plate 5)。今度は左下にセリフを加えてみます(plate 6、plate 7)。次に下のほうを少し削ります(plate 8)。これはセリフによって文字に重い感じを与えないようにするためです。右下にもセリフを同様に描きます(plate 9~plate 11)。こうして文字Aができあがります。

同じプログラムでも仕様を変えることで、無数の異なった文字Aを描くことができます。例えば、濃いボールド体の変形体を見てみましょう(plate 12~plate 22)。次に、印字が綺麗にできる小さい文字のAを見てみましょう(plate 23~plate 33)。元のAを単純に50%に縮小するだけでは、読みやすい小さい文字を作れるわけではありません。すなわち、高品質の組版のために、同じ文字でも小さい文字は、サイズの大きい文字とは微妙に異なった形にする必要があります。

タイプライター体のAも同様のプログラムで描くことができます。今回は太い線も細い線も同一にし、セリフの角は丸くするよう指定します(plate 34~plate 42)。こうして作られたAという文字は、私の最初のタイプライター体フォントになりましたが、後になってこのAが本来のものよりも少し濃いことがわかりました。この問題を解決するために二つの斜めの線を少し離し、内側に「刻み目」を入れ、少し広げました(plate 43~plate 49)。これが現在私が使っている奇麗なタイプライター体のAです。私はこの

That train of thought led to my computer system called METAFONT, which I want to try to show you now. Here is the way I finally desided to create the letter A, for example, using a computer program. All the key points of the letter are based on a grid that is displayed here [plate 1], although of course the grid is really invisible. Based on this grid and the specification of a normal text font, the computer first draws the main stem stroke [plate 2]. Part of this stroke needs to be erased because it's too thick at the top [plate 3]. Then the left diagonal stroke is added [plate 4], and the crossbar [plate 5]. It's time now to add a serif at the bottom left [plate 6, plate7], and to erase a little at the bottom [plate 8] so that the serif doesn't make the letter too heavy. A similar serif is drawn at the bottom right [plate 9-plate 11]. This completes the letter A.

The same program will draw infinitely many different A's if we change the specifications. For examples, here's darker, boldface variant: [plate 12-plate 22]. And here's a small A suitable for fine print [plate 23-plate 33]. Simply shrinking the original A by 50% would not produce such a legible character at a small size; good typography requires small letters to have subtly different shapes from their larger cousins.

Even the typewriter style A can be drawn with same program. This time we specify that the thick strokes and thin strokes are identical, and the corners of the serifs are rounded [plate 34-plate 42]. The resulting A went into my first typewriter-style font, but I learned later that such an A was a bit darker than it should be. To solve the problem, I moved the two diagonal strokes slightly apart, and cut a "notch" in the interior so as to open the inside a bit [plate 43-plate 49]. This is the nice typewriter-style A that I use today. I didn't learn such tricks until several years after I started to study type design.

Here is an example of the way my first draft fonts looked on the Xerox Graphics Printer, about one and a half years after I had begun to work on typography [photo 13]. Two years later, with some financial help from my publishers, my project was finally able to obtain a high-resolution digital typesetting machine, and I could print the new edition of Volume 2 [photo 14]. The proofs for that book looked so much better than the xerographic proofs I had been working with, I thought my goals for quality typography had finally been reached.

But when I received the first printed copy of the new Volume 2 in its familiar binding, and opened the pages, I burned with disappointment. The book did not look at all as I had hoped. After four years of hard work, I still hadn't figured out how to generate the patterns of 0s and 1s that are demanded by fine printing. The published second edition didn't look much better than the version I had rejected

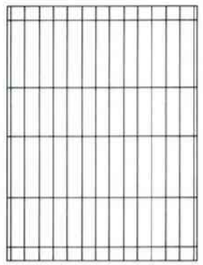


plate 1

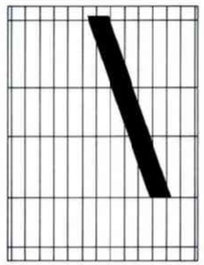


plate 2

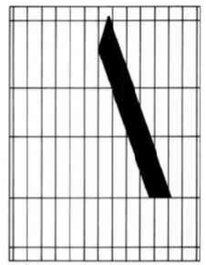


plate 3

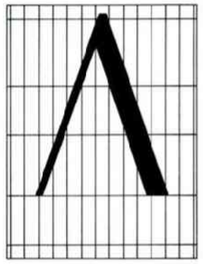


plate 4

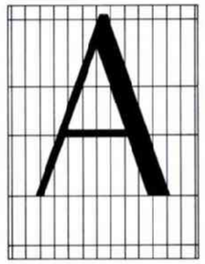


plate 5

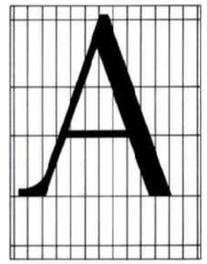


plate 6

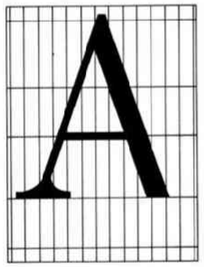


plate 7

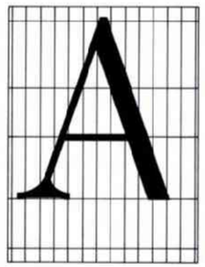


plate 8

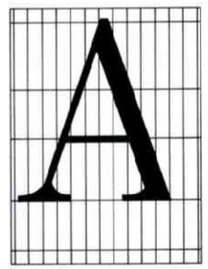


plate 9

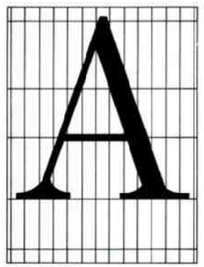


plate 10

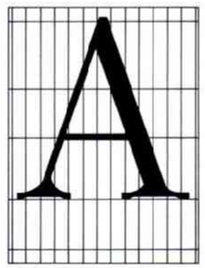


plate 11

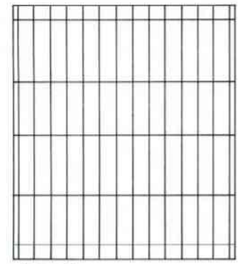


plate 12

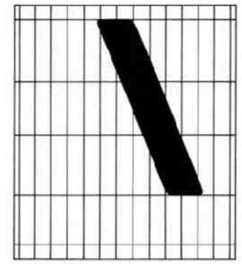


plate 13

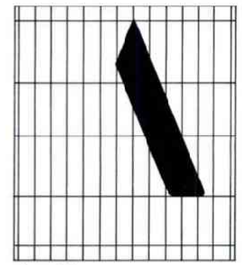


plate 14

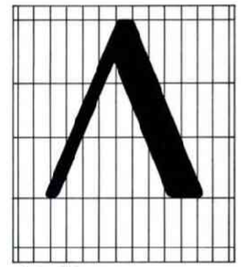


plate 15

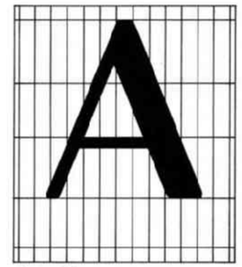


plate 16

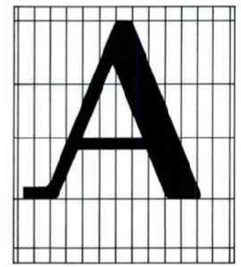


plate 17

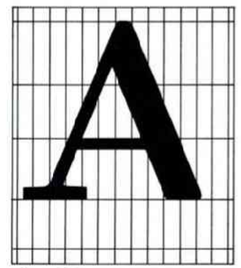


plate 18

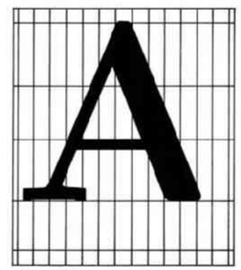


plate 19

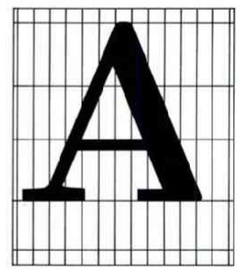


plate 20

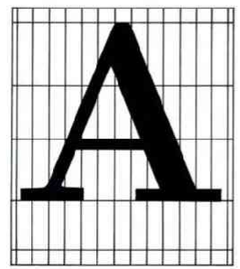


plate 21

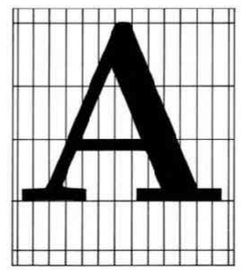


plate 22


plate 23


plate 24


plate 25


plate 26


plate 27


plate 28


plate 29


plate 30


plate 31


plate 32


plate 33

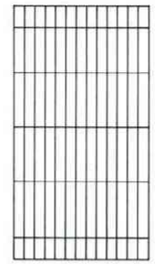


plate 34

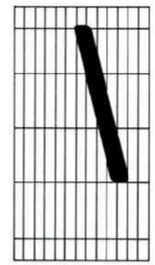


plate 35

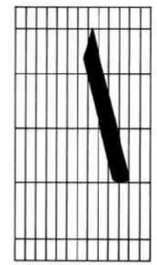


plate 36


plate 37


plate 38


plate 39


plate 40


plate 41


plate 42

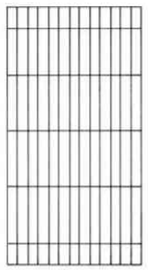


plate 43

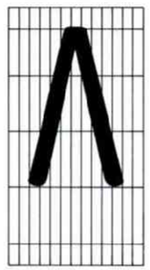


plate 44

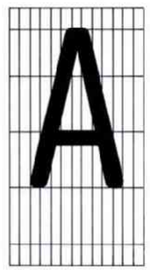


plate 45

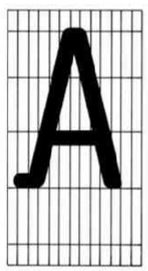


plate 46

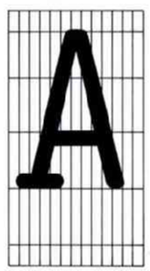
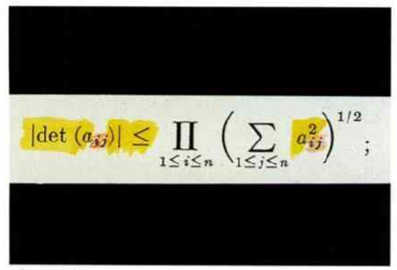


plate 47

plate 48

plate 49



$$|\det(a_{ij})| \leq \prod_{1\leq i\leq n}\left(\sum_{1\leq j\leq n} a_{ij}^2\right)^{1/2};$$

photo 12

Program A (*Addition, subtraction, and normalization*). The following program is a subroutine for Algorithm A, and it is also designed so that the normalization portion can be used by other subroutines that appear later in this section. In this program and in many other programs throughout this chapter, OFLO stands for a subroutine that prints out a message to the effect that MIX's overflow toggle was unexpectedly found to be "on." The byte size b is assumed to be a multiple of 4. The normalization routine NORM assumes that $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where $\mathrm{rA} = 0$ implies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

```
00  BYTE  EQU   1(4:4)      Byte size b
01  EXP   EQU   1:1         Definition of exponent field
02  FSUB  STA   TEMP        Floating-point subtraction subroutine:
03        LDAN  TEMP        Change sign of operand.
```

photo 13

A, STA ACC; LDA B, JMP FADD; LDA C, JMP FSUB; STA D. (8)

A (*Addition, subtraction, and normalization*). The following program itine for Algorithm A, and it is also designed so that the normalization n be used by other subroutines that appear later in this section. In am and in many others throughout this chapter, OFLO stands for a that prints out a message to the effect that MIX's overflow toggle was dly found to be "on." The byte size b is assumed to be a multiple normalization routine NORM assumes that $\mathrm{rI2} = e$ and $\mathrm{rAX} = f$, where iplies $\mathrm{rX} = 0$ and $\mathrm{rI2} < b$.

```
EQU   1(4:4)      Byte size b
EQU   1:1         Definition of exponent field
STA   TEMP        Floating point subtraction subroutine:
LDAN  TEMP        Change sign of operand.
```

photo 14

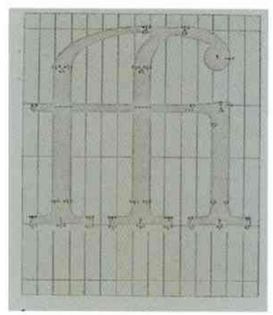


photo 15

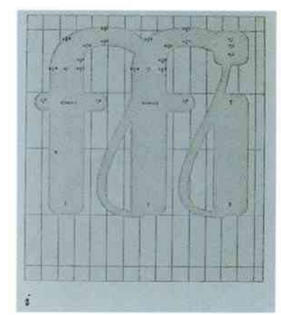


photo 16

技術を活字設計を学んで数年後に初めて知りました。

これが、私が初めて考案したフォントをゼロックス・グラフィック・プリンターで出力した一例です(photo 13)。組版を研究し始めて約1年半たった頃です。2年後、出版社からの資金援助を受け、私のプロジェクトはついに高解像度デジタル写植機を手に入れ、第2巻の新版を印刷することができました(photo 14)。その本の校正刷りは、以前の乾式複写のものより数段仕上がりが良く、高品質の組版という私の目標はついに達成されたと思いました。

しかし、いつものように装丁された第2巻の新版を受け取り、ページを開いたとき、愕然としました。私が望んでいたのとはまったく違う本のように見えたからです。4年間の苦労にもかかわらず、私はまだ、良質な印刷に要求される0と1の組み合わせを作り出す方法を理解していなかったのです。出版された第2版は、私が組版プロジェクトを始める前に却下した版と代わり映えしませんでした。

一方、私は世界で一流の活字設計者の方々の多くにお会いする幸運に恵まれました。私は彼らから貴重な指導や批評を受けることができ、改良を重ねました。5年余たった後に、皆さんに誇れる本を作ることができました。

しかし皆さんに、この9年間の研究が退屈な苦労にすぎなかったという印象を持っていただきたくありません（先にも述べましたが、私はめったに退屈することはありません)。フォント設計は実際たいへん楽しい仕事です。特に、失敗したときなどはなおさらです。コンピューターはよく、人間が夢にも思いつかないような楽しい創造的な画像を描きます。私はこれを「meta-flops」と呼んでいます。例えば、ここにffi合字の組み合わせがあります(photo 15)。ここでは、左にあるfは右にあるiの点までずっと続いています。ここにも別の変わったffiがあります(photo 16)。私はこれを「the ffilling station」と呼んでいます。

大文字のタイプライター体のYを初めて試作したとき、私は上部右のセリフを間違った場所に置きました(photo 17)。誓って言いますが、そのとき、私は日本円（¥）のことを考えていたのではありません。

METAFONTはローマン活字と同様に、日本語の文字でも使えるのでしょうか。私は使えると思いますが、アジアの文字の形については眼識がありません。そこで、私の学生であるジョン・ホビーと上海印刷会社のグ・グァンが共同で見込みのある実験を行いました。彼らの実験を少しお見せしたいと思います。まず初めに、彼らは基本となるストロークのために、13のコンピュータープログラムを書きました。例えば、ここにそのプログラムの中の一つで作った「涙」の形をしたものが二つありま

before starting my typography project.

Meanwhile I had had the good fortune to meet many of the world's leading type designers. They graciously gave me the instruction and criticism I needed as I continued to make improvements. After five more years went by, I finally was able to produce books of which I could feel proud.

I don't want to give the impression that those nine years of work were nothing but drudgery. (As I said before, I raely seem to get bored.) Font design is in fact lots of fun, especially when you make mistakes. The computer tends to draw delightfully creative images that no human being would ever dream up. I call these "meta-flops." For example [photo 15], here's an ffi ligature combination in which the f at the left reaches all the way over to the dot on the i at the right. And here's another weird ffi [photo 16]: I call it "the ffilling station."

In one of my first attempts to do a capital typewriter-style Y, I put the upper right serif in the wrong place [photo 17]. I swear that I was *not* thinking of yen when I did this!

Does METAFONT work for Japanese characters as well as for Roman letters? I think it does, but I haven't been able to develop a good eye for Asian letterforms myself. My student John Hobby did some promising experiments together with Gu Guoan of the Shanghai Printing Company, and I'd like to give you a taste of what they did. First they wrote 13 computer programs for basic strokes. For example, here are two "teardrop" shapes produced by one of their programs [photo 18]. A font designer specifies the top, the bottom, and the edge of the bulb; the computer does the rest. Here [photo 19] are some more examples of teardrops, together with variants of three other basic strokes.

Hobby and Gu used their stroke routines to design 128 Chinese characters. And they did it in such a way that you could get three different styles of letters simply by using three different versions of the 13 basic strokes. Here [photo 20] are five characters rendered in Song style, Long Song style, and Bold style. And here [photo 21] are examples of the 13 basic strokes in all three styles.

With the METAFONT system for type design, and the TEX system for putting letters and symbols into the right positions on a page, anybody who wants to write a beautiful book can now do so singlehandedly with a reasonable amount of effort. These systems give an author total control over the patterns of 0s and 1s that are needed to define the pages. I have made special efforts to ensure that TEX and METAFONT will give exactly the same results on all computers, and to ensure that they will give the same results 50 years from today as they do today. Furthermore I have published all of the details and put all of my programs

(photo 18)。フォント設計者が、先端、底、球の端を指定し、残りをコンピューターが指定します。これは、その他三つの基本ストロークの変形体と涙の形をした数例です(photo 19)。

ジョン・ホビーとグ・グァンはこうしたストロークの手順を用いて、128の漢字を設計しました。その仕組みは、13の基本ストロークに対して三つの変形バージョンを用いることにより、3種類の書体を生み出すという単純なものです。これはソング体及びロング・ソング体、ボールド体で書かれた五つの文字です(photo 20)。また、これは13の基本ストロークを3種の書体で表わしたものの例です(photo 21)。

活字設計のためのMETAFONTシステムと、文字や記号をページ上に正しく配置するためのTEXシステムによって、美しい本を書きたい人は誰でも、ある程度努力をすればそれがひとりでできるようになりました。これらのシステムでは、ページを定義するのに必要な0と1の組合せは完全に著者の手に委ねられます。私はTEXとMETAFONTが、どんなコンピューターでもまったく同じ結果を出せるように、そして今から50年後にも同じ結果を出せるように特に努力してきました。さらに、私は詳細のすべての著作権をフリーにし、私のプログラムをすべての人が無料で使用できるようにしました。もちろん、今後多くの人々が付加サービスを提供し、その専門技術については有料となることはあるでしょう。しかし肝心なことは、献身的な著者が、以前は高嶺の花であった製本をする力を手に入れたということなのです。

最近、著者自らが私に送ってきた何冊かの本をお見せしましょう。ここに、チェコ共和国からのものがあります(photo 22)が、METAFONTで作られたフォントの例を示しています。これはエチオピアからのもので、TEXシステムの使い方をその国の人々に教えています(photo 23)。これはTEXに関する私の著書をロシア語に翻訳した一部です(photo 24)。そして、これは同じ部分の日本語訳です(photo 25)。ところで、私が日本に住んでいたら、きっとTEXやMETAFONTを制作する気にはならなかっただろうと思います。なぜなら、その必要を感じなかっただろうと思うからです。日本の組版の水準は、アメリカやヨーロッパのように低下しませんでした。しかし、TEXが日本の出版界で広く使われているのを見て、たいへんうれしく思っています。

私の元には、TEXやMETAFONTがなければ、決して存在しないだろうと思われるたくさんの良質の本が送られてきます。私のお気に入りのものは、ここに示されている、エスキモー語の民話を国際語で書いたテキストなどの学術書です(photo 26)。同様に、これはギリシャ語のテキストの校訂版からの脚注です(photo 27)。ほかには、アラビア語(photo 28)、サンスクリット語(photo 29)で書かれたものがあります。

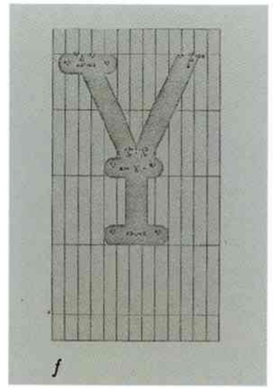


photo 17

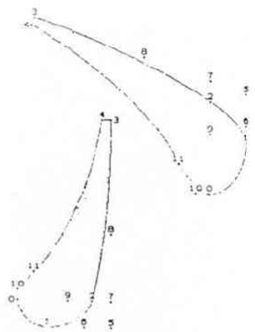


photo 18

photo 19

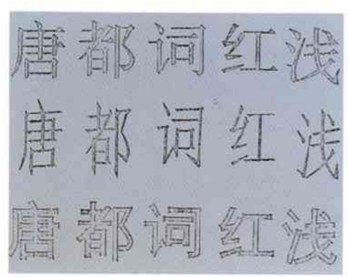


photo 20

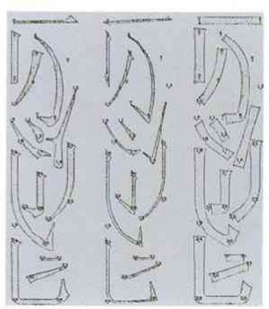


photo 21

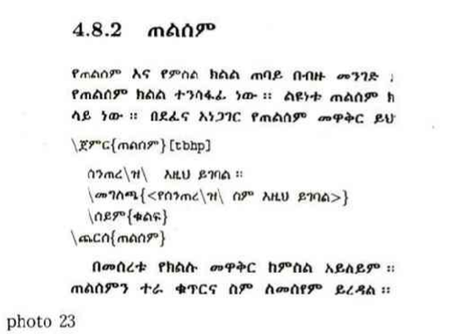
Kapitola V

Knihu zapečetěnou Beránek přijal. 11. protož od andělů, zvířat, starců 13. a ode všeho stvoření, spolu s sedícím na trůnu, jednostejnou čest, slávu a chválu sobě činěnou měl.

1 Viděl jsem po pravici sedícího na trůnu knihu popsanou vnitř i zevnitř, zapečetěnou sedmi pečetmi. 2 A viděl jsem anděla silného, volajícího hlasem velikým: Kdo jest hoden otevříti tu knihu a zrušiti pečeti její? 3 I nemohl žádný, ani na nebi, ani na zemi, ani pod zemí, otevříti té knihy, ani pohleděti do ní. 4 Protož já plakal jsem velmi, že není nalezen žádný, kdo by hoden byl otevříti a čísti tu knihu, ani pohleděti do ní. 5 Tedy jeden z těch starců řekl mi: Neplač. Aj, svítězil lev, ten kterýž jest z pokolení Judova, kořen Davidův, aby otevřel tu knihu a zrušil sedm pečetí jejích. 6 I pohleděl jsem, a aj, mezi trůnem a čtyřmi těmi zvířaty a mezi těmi starci Beránek stojí jako zabitý, maje sedm rohů a sedm očí, jenž jsou sedm duchů Božích, poslaných na všecku zemi. 7 I přišel a vzal tu knihu z pravice toho, kterýž seděl na trůnu.

8 A jakž vzal tu knihu, hned těch čtvero zvířat a těch čtyřmecítma starců padlo před tím Beránkem, majíce jeden každý z nich harfu a báně zlaté plné vůně, jenž jsou modlitby svatých. 9 A zpívali píseň

photo 22



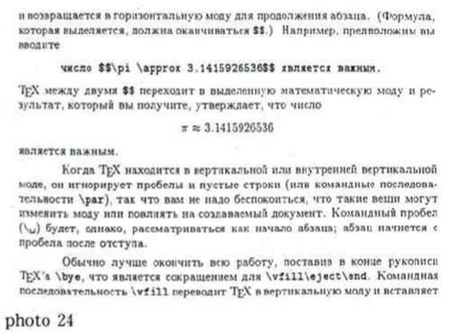
4.8.2 ጠልሰም

የጠልሰም አና የምስል ክልል ጠባይ በብዙ መንገድ ፣ የጠልሰም ክልል ተንሰፋፊ ነው ። ልዩነቱ ጠልሰም ክላይ ነው ። በደፈና አነጋገር የጠልሰም መዋቅር ይህ

```
\ጀምር{ጠልሰም}[tbhp]
  ስንጠረ\ዝ\  አዚህ ይገባል ።
  \መገለጫ{<የሰንጠረ\ዝ\ ሰም አዚህ ይገባል>}
  \ሰይም{ቁልፍ}
\ጨርስ{ጠልሰም}
```

በመሰረቱ የክልሱ መዋቅር ከምስል አይለይም ። ጠልሰምን ተራ ቁጥርና ስም ለመሰየም ይረዳል ።

photo 23



и возвращается в горизонтальную моду для продолжения абзаца. (Формула, которая выделяется, должна оканчиваться $$.) Например, предположим вы вводите

число $$\pi \approx 3.1415926536$$ является важным.

TEX между двумя $$ переходит в выделенную математическую моду и результат, который вы получите, утверждает, что число

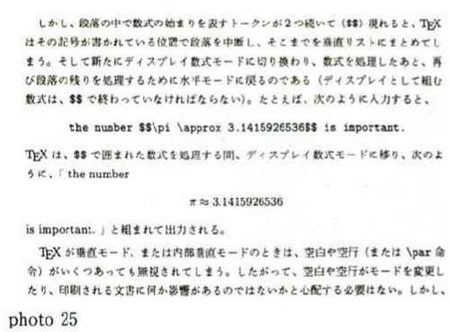
$$\pi \approx 3.1415926536$$

является важным.

Когда TEX находится в вертикальной или внутренней вертикальной моде, он игнорирует пробелы и пустые строки (или командные последовательности \par), так что вам не надо беспокоиться, что такие вещи могут изменить моду или повлиять на создаваемый документ. Командный пробел (\␣) будет, однако, рассматриваться как начало абзаца; абзац начнется с пробела после отступа.

Обычно лучше окончить всю работу, поставив в конце рукописи TEX'a \bye, что является сокращением для \vfill\eject\end. Командная последовательность \vfill переводит TEX в вертикальную моду и вставляет

photo 24



しかし、段落の中で数式の始まりを表すトークンが2つ続いて（$$）現れると、TEXはその記号が書かれている位置で段落を中断し、そこまでを垂直リストにまとめてしまう。そして新たにディスプレイ数式モードに切り換わり、数式を処理したあと、再び段落の残りを処理するために水平モードに戻るのである（ディスプレイとして組む数式は、$$ で終わっていなければならない）。たとえば、次のように入力すると、

the number $$\pi \approx 3.1415926536$$ is important.

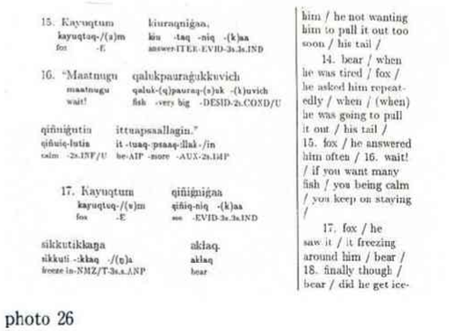
TEXは、$$ で囲まれた数式を処理する間、ディスプレイ数式モードに移り、次のように、「the number

$$\pi \approx 3.1415926536$$

is important.」と組まれて出力される。

TEXが垂直モード、または内部垂直モードのときは、空白や空行（または \par 命令）がいくつあっても無視されてしまう。したがって、空白や空行がモードを変更したり、印刷される文書に何か影響があるのではないかと心配する必要はない。しかし、

photo 25



15. Kayuqtum kiuraqniġaa,
kayuqtuq-/(s)m kiu -taq -niq -(k)aa
fox -E answer-ITEE-EVID-3s.3s.IND

16. "Maatnugu qalukpauraġukkuvich
maatnugu qaluk-(q)pauraq-(s)uk -(k)uvich
wait! fish -very big -DESID-2s.COXD/U

qiñiñiġutin ittuapsaallagin."
qiñuiq-lutin it -tuaq-psaaq-llak-/in
calm -2s.INF/U be-AIP -more -AUX-2s.IAP

17. Kayuqtum qiñiñiġaa
kayuqtuq-/(s)m qiñiq-niq -(k)aa
fox -E see -EVID-3s.3s.IND

sikkutikkaŋa akłaq.
sikkuti -akłaq -/(t)a akłaq
freeze in-NMZ/T-3s.s.INP bear

him / he not wanting him to pull it out too soon / his tail /

14. bear / when he was tired / fox / he asked him repeatedly / when / (when) he was going to pull it out / his tail /

15. fox / he answered him often / 16. wait! / if you want many fish / you being calm / you keep on staying /

17. fox / he saw it / it freezing around him / bear / 18. finally though / bear / did he get ice-

photo 26

in the public domain, so that nobody has to pay for using them. Of course, many people who offer additional services will charge money for their expertise, but the main point is that a dedicated author now has the power to prepare books that previously were prohibitively expensive.

I can't resist showing you samples from some of the books that I've received in recent years from their authors. Here's one from the Czech Republic [photo 22], showing another font done with the METAFONT SYSTEM. Here's one from Ethiopia [photo 23], telling the people of that country how to use the TEX system. Here [photo 24] is part of the Russian translation of my own book on TEX. And here [photo 25] is the same passage in Japanese translation. By the way, if I had lived in Japan, I'm sure I never would have been inclined to invent TEX or MATAFONT, because I wouldn't have felt the need: The standards of typography in this country never declined as they did in America and Europe. However, I'm glad to see that TEX is now widely used in Japanese publishing.

People have sent me many fine books that probably never have existed without TEX and METAFONT. My favorite examples are scholarly publications, such as the interlingual text of an Eskimo language folk tale shown here [photo 26]. Here, similarly, are some footnotes from a critical edition of a Greek text [photo 27]; another, in Arabic [photo 28]; another in Sanskrit [photo 29].

Ever since I began working on TEX in 1977, I've kept a record of all the errors, large and small, that I found and removed from the program with the help of volunteers around the world. This list has now grown to 1,276 items. Perhaps TEX has thereby become one of the most thoroughly checked computer programs ever written.

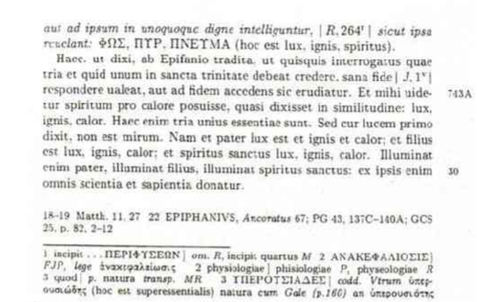
I would like to conclude this talk by quoting one of my favorite poems, written by the Danish sage Piet Hein. He calls it a "grook"—it's sort of a Danish variant of *haiku*. My wife and I like it so much, we commissioned a British stonecutter to carve it in slate for the entryway of our house [photo 30]. It goes like this:

The road to wisdom? Well it's plain
and simple to express:
Err
and err
and err again
but less
and less
and less.

1977年にＴＥＸの研究を始めて以来、私は世界中の有志の方の力添えによって、プログラムから見つけ、削除したすべてのエラーを大小問わず記録しています。このエラーリストは今や1276項目を収めるに至っています。このように、おそらくＴＥＸは、世の中で最も完璧にチェックされたコンピュータープログラムの一つとなったと言えるでしょう。

今回の講演を、デンマークの賢人ピエット・ハインによって書かれた私の大好きな詩を引用して終わりたいと思います。彼はこの詩を「グルーク」と呼んでいます。これはデンマーク版の俳句のようなものです。妻と私はたいへんこの詩が気に入り、イギリスの石工に依頼して、家の通路にある石板にこの詩を彫ってもらいました(photo 30)。次のような詩です。

知恵への道とは？
それは明白で単純なこと。
間違いをして、
そして、間違いをして、
そして、また間違いをして、
でも、間違いを減らして、
そして、減らして
そして、減らすこと。



aut ad ipsum in unoquoque digne intelliguntur, | *R*, 264' | sicut ipsa resclant: ΦΩΣ, ΠΤΡ, ΠΝΕΤΜΑ (hoc est lux, ignis, spiritus).
Haec. ut dixi, ab Epifanio tradita, ut quisquis interrogatus quae tria et quid unum in sancta trinitate debeat credere, sana fide | *J*, 1° | respondere ualeat, aut ad fidem accedens sic erudiatur. Et mihi uidetur spiritum pro calore posuisse, quasi dixisset in similitudine: lux, ignis, calor. Haec enim tria unius essentiae sunt. Sed cur lucem primo dixit, non est mirum. Nam et pater lux est et ignis et calor; et filius est lux, ignis, calor; et spiritus sanctus lux, ignis, calor. Illuminat enim pater, illuminat filius, illuminat spiritus sanctus: ex ipsis enim omnis scientia et sapientia donatur. 743A 30

18–19 Matth. 11, 27 22 EPIPHANIVS, *Ancoratus* 67; PG 43, 137C–140A; GCS 25, p. 82, 2–12

1 incipit . . . ΠΕΡΙΦΥΣΕΩΝ] *om. R*, incipit quartus *M* 2 ΑΝΑΚΕΦΑΛΙΟΣΙΣ] *PJP*, *lege* ἐνκεφαλίωσις 2 physiologiae] phisiologiae *P*, physeologiae *R* 3 quod] *p.* natura *transp. MR* 3 ΤΗΕΡΟΤΣΙΑΔΕΣ] *codd. Vtrum* ὑπερουσιώδης (hoc est superessentialis) natura *cum Gale (p.160) an* ὑπερουσιότης

photo 27

photo 28

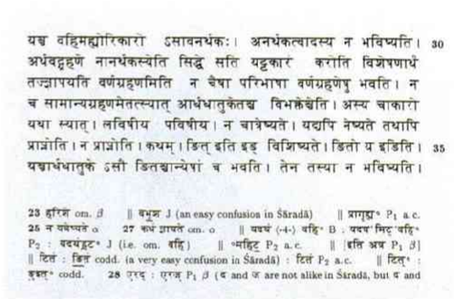


photo 29

photo 30

稲盛財団1996——第12回京都賞と助成金

発　　行　1997年11月1日
発 行 所　財団法人稲盛財団
　　京都市下京区四条通室町東入函谷鉾町88番地　〒600
　　電話〔075〕255-2688
製　　作　㈱ウォーク
印刷・製本　大日本印刷株式会社
ISBN4-900663-12-3 C0000