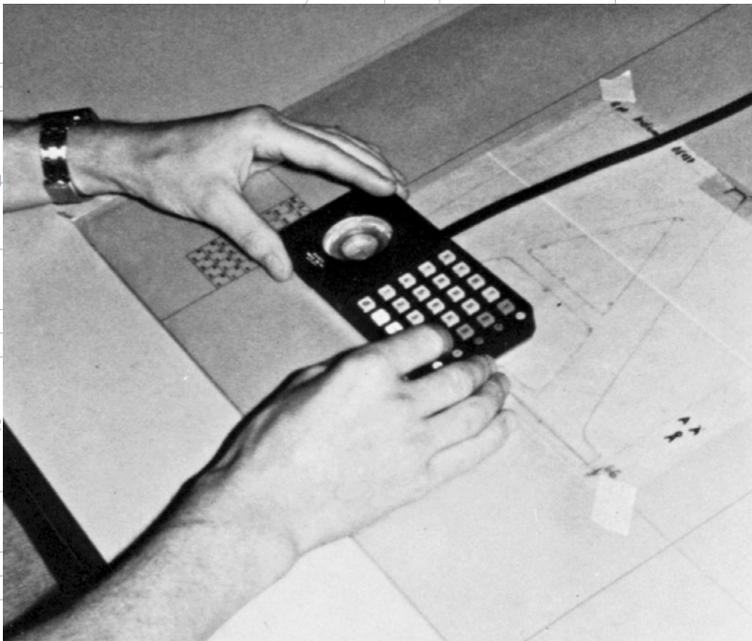


愛

Dr. Peter Karow

Strokes

Digital Typography & Artificial Intelligence



401
心

ON OCCASION OF THE PRESENTATION OF THE THIRD
DR. PETER KAROW AWARD FOR FONT TECHNOLOGY AND
DIGITAL TYPOGRAPHY TO DR. DONALD E. KNUTH

305
父

AT THE ATYPI AMSTERDAM 2013
CONFERENCE

401
802
1102

*On occasion of
the presentation of the third
Dr. Peter Karow Award
for Font Technology & Digital Typography
to **Dr. Donald E. Knuth**
at the ATypI Amsterdam 2013
conference*

Ikarus Master - Sheldon.IK

File Edit View Tools Functions Window Help

Sheldon.IK Sheldon.IK - 307 (0x133)

Char Info:

L	129
CW	8100
R	610
TW	8839
C	3
P	99
XMin	0
XMax	8100
YMin	-2840
YMax	7732

Preview

C= 3 P= 43 x=6521.0,y=6491.0

g

g h i j

IK B

Edit XY List 307

Sheldon.IK

Points(1)	4023	7432	Start Point
Points(2)	3360	7282	Curve Point
Points(3)	2808	6749	Curve Point
Points(4)	2420	5825	Curve Point
Points(5)	2294	4727	Curve Point
Points(6)	2437	3572	Curve Point
Points(7)	2836	2654	Curve Point
Points(8)	3341	2238	Curve Point
Points(9)	4108	2082	Curve Point
Points(10)	4868	2283	Curve Point
Points(11)	5380	2832	Curve Point

Dr. Peter Karow

Digital Typography & Artificial Intelligence

Adobe
Dutch Type Library
2013

Introduction

In this highly energetic and volatile era of rapid developments in the field of Information Technology, it's good to honor those who stood at the cradle of these developments, as it is also good to stimulate present-time innovative contributions.

The *Dr. Peter Karow Award for Font Technology & Digital Typography* has been established for this purpose, and it focuses on two mutually related matters that perhaps objectively can be considered small elements within the field of Information Technology, but which have a huge impact on the way we communicate: *font technology* and *digital typography*.

More than forty years ago Dr. Karow began to digitize the outlines of characters using an Aristo tablet with lens cursor, which resulted in the IKARUS format. For roughly two decades the IKARUS format was the *de facto* standard for digitizing type. Subsequently, letters were mostly manually digitized before the rise of desktop publishing. With the 'democratizing' of digital typography every aspect of the related market changed, from the manufacturing of type setting machines to the production of fonts.

In the wake of Adobe's PostScript format, a new generation of relatively low-cost Bézier-oriented font tools for personal computers (starting with Jim von Ehr's *Fontographer* in early 1986) made manual digitizing less obvious.

Nowadays there are a wide range of inexpensive or even free powerful font tools available, and never before in the short history of digital type has it been so easy to produce and distribute fonts. And never before have the options for digital typography been as sophisticated as they are nowadays. The exciting current development of webfont technology and the application of fonts in apps and e-books assure us that even forty years after fonts became digital, the end of innovation is still nowhere in sight.

In the course of time the importance of manual digitizing may have been diminished a bit, but to this day it is still used in *IkarusMaster*, a part of the *DTL FontMaster* suite that is developed at URW++ Design & Development GmbH under supervision of Dr. Jürgen Willrodt. This set of tools finds its origin in a cooperation between URW (and its later successor URW++) and the Dutch Type Library, which started back in 1991, when I was introduced at the Hamburg-based firm by my friend and colleague Albert-Jan Pool, who was running URW's font production back then. In that year I met Dr. Karow for the first time and I recall, above all, his generosity and hospitality when, as a fledging company, DTL needed every form of support it could get. It was especially Peter Rosenfeld, at that time manager of the URW fontstudio and who later became Managing Director of URW++, who immediately noticed the value and potential of DTL's type library and who took the company under his wings.

Although this introduction is not really a place to plead for manual digitizing, I think that drawing by hand and hence manual digitizing is still, in many cases, the best start for a digital typeface, if only because it reduces the pace a bit leaving more room for contemplation. My opinion on this seems to be shared by others in the field, evidenced by the growing number of requests we receive at DTL and URW++ for the support of different tablets with lens cursors.

But manual digitizing is only one of the numerous contributions that Dr. Karow made to digital type and typography, and one can't think of many technological aspects that have not been investigated or described by him during his long career. From hinting to grayscaling, from kerning to optical scaling, from Kanji-separation to paragraph composition, et cetera, et cetera, his influence in all of these can be seen in this booklet. For the paragraph composition he developed the *hz*-engine program together with Hermann Zapf, which was implemented by Adobe in *InDesign* in 1995.



*Peter Karow and
Hermann Zapf at the
Rochester Institute of
Technology in 1989*

The cooperation between the scientist, Dr. Karow, and the very skilled type designer Zapf was very fruitful, and a good example of how font-related technology was and is developed. Based on my PhD research at Leiden University on the origins of harmonic and rhythmical patterns in Latin type (one of the first times I gave a talk on this subject was in June 1993 at a meeting of the Dutch TeX users group), I think it is plausible that movable type was developed by craftsmen and entrepreneurs like Johannes Gutenberg and Nicolas Jenson as a ‘font format’, i.e. a technological vehicle for type designs, which was as much adjusted to the then existing aesthetic preferences, as that it created new ones. Probably it was the result of an interaction between scientists/technologists and designers, like in case of forenamed *hz-engine* program.

In an e-mail exchange last year, Adobe’s Senior Manager of Type Development, David Lemon, wrote to me: *‘It seems to me that the great leaps forward in the history of type and printing have stemmed from a sort of dialog between technology and art. Each figure in that history may have been more artist than technician or more technician than artist, but they all had both qual-*

ities. The great designers were necessarily also master craftsmen, fully in control of the technologies they used. And the great inventors were never mere technicians; their inventions arose from seeing opportunities that new technology could address – so an awareness of the artistic aspect was a precondition for the invention.

In the case of PostScript, John Warnock had strong connections to the world of graphic arts and publishing. John’s wife was a professional graphic designer (who created Adobe’s original logo), and when John got money from Adobe’s success he spent quite a bit of it on rare books. For example, when he acquired a Kelmscott Chaucer edition, he brought it to Adobe so the type team could enjoy looking through it. That’s not something a simple “computer scientist” would do.’

This acknowledgement of the interaction between technologist/scientist and artist characterizes the Dr. Peter Karow Award. Thus, it will not come as a surprise that, at Adobe, there was a keen interest in the award, which resulted in a considerable support, both organizational as well as financial, for the third Dr. Peter Karow Award, to be presented on Saturday 12 October 2013 at the ATyPl conference in Amsterdam. What especially appealed to Adobe, is that the purpose of the award is not only to honor, but also to stimulate. It is not a lifetime achievement award as such, although it can be presented to someone whose achievements span a lifetime, of course.

The Dr. Peter Karow Award is presented once every five years to a person who makes an exceptional and innovative contribution to the development of digital type and typography related technology. The award was established by the Dutch Type Library in 2003. In that year, the first award was presented to Dr. Karow himself at the third DTL FontMaster Conference at Castle Maurick in the Dutch city of Vught.

It took an extra year (six years) before the second Dr. Peter Karow Award was presented to Thomas Milo at the fourth DTL

FontMaster Conference at the Steigenberger Kurhaus Hotel in The Hague in November 2009.

Thomas Milo and his company DecoType developed ACE, which is an acronym for ‘Arabic Calligraphic Engine’, a new advanced technology for Arabic text setting, a language group which requires a far more sophisticated approach than systems like the Latin script, based on a thorough analysis of the Arabic script. Not only did Milo’s typographic research serve as the fundament for the ACE technology, clearly it also formed a basis for the development of the OpenType layout model, although this is a less known and acknowledged fact.

Thomas Milo’s contributions to the field of digital type and typography are clear, and he rightfully deserves a position next to Dr. Karow. As one consultant of the award jury stated: *‘Dr. Karow made type digital in a way we know today (description of shapes as outlines, rasterization, hinting, greyscaling, plus page-layout improvements). Thomas Milo added the “smartness” needed for scripts that ask for a more sophisticated behavior than Latin.’*

The award was presented to Thomas Milo by Dr. Peter Karow himself and in presence of Dr. Renk Roborgh, Director-General at the Ministry of Education, Culture and Science. The ceremony followed on an inspiring talk by Milo on the history of his company DecoType and the development of the ACE technology. Milo underlined that without the invaluable assistance and support of his colleagues at DecoType, who are his wife Mirjam Somers and his brother-in-law Peter Somers, his smart font technology would not have emerged.

The third Dr. Peter Karow Award has been unanimously awarded to Dr. Donald E. Knuth by this year’s jury, a group composed of Dr. Peter Karow, Thomas Milo (DecoType), David Lemon (Adobe), Peter Rosenfeld (URW++), Dr. Jürgen Willrodt (URW++), and chairman Frank E. Blokland (Dutch Type Library).

Dr. Knuth's highly valuable contributions to digital typography and font technology are numerous. Every reader of this booklet will be aware of the existence of TeX (from Greek 'techne' = art/craft, the stem of 'technology'), a powerful typesetting system, especially suited for technical texts. Extensions of TeX are still the standard tools for composing most scientific texts. Equally famous is his METAFONT, a computer language for creating letterforms by mathematical means, which blazed a trail for the future of parameterized type design that is still being explored today.

Like Dr. Karow and Thomas Milo, Dr. Knuth developed key pieces of the foundation of digital typography. And like Dr. Karow, Dr. Knuth came to value a close working friendship with Hermann Zapf, starting even before they collaborated on the Euler types in Metafont, for the American Mathematical Society. Dr. Knuth's appreciation of the calligraphic underpinnings of type is also evident in the collection 3:16 that he curated and published, asking calligraphers around the world to apply their art to an arbitrary set of verses from the Bible.

With Prof. Charles Bigelow he established a Masters course in digital type at Stanford University, a joint production between Stanford's Art and Computer Science departments that sadly was able to last only two years. Three of the graduates of the Stanford M.S. Digital Typography program worked at Adobe: Carol Twombly, Daniel Mills, and Cleo Huggins.

In an interview of Charles Bigelow conducted in 2012 and published Summer 2013 in *TUGboat*, Bigelow explains that Dr. Knuth's mathematical approach of generating type didn't imply that it was restricted from an artistic point of view: *'Although Knuth says his goal was to imitate a metal typeface called Monotype Modern 8A, Computer Modern has many original ideas underlying its forms. In visual form, the basic serifed version of Computer Modern did imitate Monotype Modern, but in conception and technical implementation, Computer Modern was original.'* In the same paragraph he praises Dr. Knuth's

non-commercial approach: *'It is noteworthy and commendable, too, that Knuth published all the Metafont code for his designs. For commercial reasons, most typefaces are marketed with intellectual property restrictions, but Knuth saw his typographic work as part of a greater goal, the publication of scientific literature and the dissemination of knowledge. He did the same with his TeX system for mathematical composition, publishing the source code for wide usage. A paragon of enlightened generosity.'*

In an e-mail to me dated 24 August 2013 Dr. Karow summarized Dr. Knuth contributions to digital typography: *'TeX was written by Donald Knuth and released in 1978 when DTP was yet not invented. Together with the Metafont language for font description TeX was designed with following goals: to allow anybody to produce theses and even books with typographical quality using a minimal amount of effort, and to provide a system giving the same results on all computers, now and in the future. The AMS Euler was designed and created by Hermann Zapf in 1980–81 with the assistance of Donald Knuth and added to the system.*

After 1980 all this went around the world installed on all larger computer systems at the Universities and was the first commonly used "DTP program". The first time, students could write their theses without the help of secretaries, typewriters, glues, scissors and copies. The paragraph justification was excellent, never achieved before, also the typography of formulas.'

In the *METAFONT* book, which was first published in 1986, there is a dedication that clearly underlines the forenamed interaction between scientists and type designers: *'To Hermann Zapf: Whose strokes are the best'*.

Frank E. Blokland, August 2013

DIGITAL TYPOGRAPHY
& ARTIFICIAL INTELLIGENCE

Summary

Digital Typography transpired in type *design* and text *layout*. It has changed font *production* and text *composition* in their entirety.

I was involved in many of the demands for *digital typefaces* which came into existence from 1972 through 1997. These issues included formats, variations, interpolation, rasterizing, hinting, autotracing, grayscaling, and element separation.

Modern text composition was mostly influenced by programs such as WordStar, Word, PageMaker, QuarkXpress, and FrameMaker, which replaced writing, typesetting and printing in offices on the one hand and at home on the other. In current times, text is composed much less manually than in the past, but not as digitally generated as its potential.

Within modern text composition, *digital text* is a special part that should proceed without manual assistance and human layout. Up to now, the milestones were these: kerning, optical scaling, paragraph composition (*hz*-program), chapter composition (chapter fit), and digital ads.

As is known, a good deal of engineering endeavors has already been implemented in regards to digital typography. However, distinct challenges still exist such as refinements to autotracing, autohinting, element separation, kerning, optical scaling, chapter fit, and automatic text composition. Many sophisticated tasks are still left to be executed, they belong more to artificial intelligence than to engineering.

Digital Typefaces

When did typefaces become digital?

At the end of 1972, I met Walter Brendel³⁸ – my first client – and started to ‘crunch’ typefaces. I began to digitize the outlines of characters using a digitizing tablet. I performed this similarly to the people at the company of Aristo in Hamburg³¹, where they digitized forms of electronic layouts, automobiles, and ships (see fig. 1).

fig. 1

Typical desktop in the seventies with digitizer tablet, ‘mouse’, keyboard (below), direct VDT, alphanumeric terminal, digitizer electronic (above)



I added these basic features: [1] four sorts of control points (SECT: start, edge [corner], curve, tangent), and [2] editing a single control point somewhere in between. The connecting line was calculated by a spline interpolation (see fig. 2, left). This editing-feature is fundamental for automatic refinements, (re)design on screens, and the generation of variations like ‘shadowing’ and ‘contouring’.

At that time, creative typesetting professionals produced variations manually using their photographic experience and a photo-typesetting machine to make fonts with contours and shadows (see below *remark 1*). Naturally, shadowing as well as contouring were my next features, which I started on 26 February 1973. This date might be regarded as the birth of

‘digital typefaces’: new forms were generated automatically (see fig. 2, right).

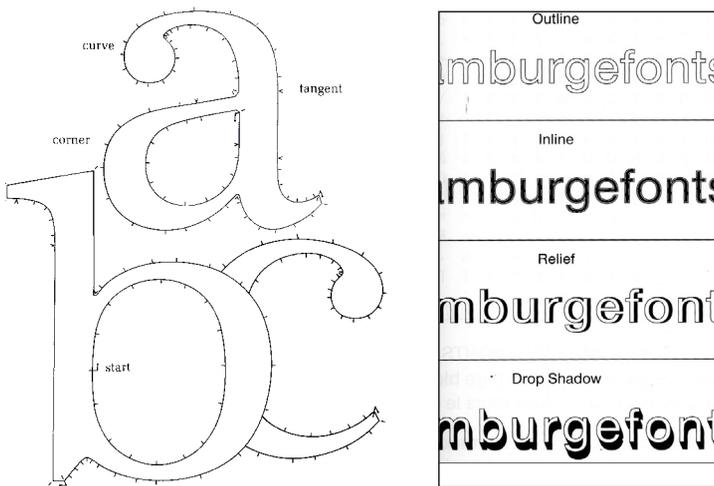


fig. 2
 IKARUS-Format was started 1972 (left). Contouring used to make Outline and Inline versions, shadowing and contouring used to make Relief and Drop Shadow (right)

I spent a lot of time trying to generate a lighter and/or heavier typeface version out of one master. I attempted a lot of tricks, compromises and heuristic rules to achieve it. I never got it. My programs crashed quite often and generated substandard looking results. That influenced my decision to call the software IKARUS¹⁵. However, on the 25 May 1973, my partner Gerhard Rubow³⁴ proposed to use two masters and interpolation, which immediately worked well, but needed twice the amount of work for input.

In the beginning I said to my partners: ‘I can program it in two months.’ It took me four years. So, be cautious with programmers: their estimates are usually wrong. Perhaps as a rule, these estimations should be taken to the next higher time unit and then multiplied by two: *2 months* → *4 years*.

These days, everyone regards all fonts on computers as digital fonts since they are stored in digital formats such as OpenType. In the early seventies, we had long discussions with famous designers. They argued that pure mapping from analogue to digital is not changing the basic quality of a typeface (old properties), namely its type, appearance, effect, expres-

sion and congeniality. Therefore they asserted that the typefaces only had digital images and were therefore still analogue. ‘Digital’ at that time was regarded as a pseudo-property. The designers were anxious and envisioned their typefaces as losing their luster and personalities once they were ‘crunched by numbers’.

Today, the property ‘digital’ is not only accepted but also embraced. It serves as an additional characteristic which doesn’t interfere with the old properties and holds an extremely high significance regarding a font. It allows and creates new and important functions which did not exist before.

Formats

In 1960, the company of Aristo³¹ introduced *stroke fonts* in their Perthronic controller – at first for the characters 0...9 – in order to inscribe the drawings on their plotters for the design of buildings, cars, ships, and surveying. Until the late eighties, stroke fonts were also used on graphic display terminals (direct VDTs, storage displays). Tektronix was considerably successful with their devices. Stroke fonts consist of just one line; the ‘stem thickness’ is the thickness of the drawing pen or electronic beam respectively.

Bitmaps started as dot matrices, which computer scientists needed for line printers, and much later for dot matrix printers and VDTs. Typical sizes were 6×8 bytes for Latin upper case characters or 16×16 for Kanji in Japanese. The smaller the dots became in line printing, the larger the dimensions of the matrices grew.

With the invention of the Digiset by Rudolf Hell in 1965, typefaces were digitized for the first time. No additional ideas were put into place other than using them 1:1 for typesetting on the Digiset, scaled linearly and displayed at resolutions between 1,000 and 2,400 dpi. The dot matrix became a bitmap, stored as *run length format* with dimensions of 50×60, 100×120, 200×240, 400×480, and 800×960.

Since the invention at Xerox in 1971, laser printing has developed quickly. Xerox, IBM, Siemens, Fujitsu, Canon, and others started office automation. All manufacturers – and I emphasize *all* – introduced their own formats for bitmaps (or Bytemaps).

At the end of 1972, I started with the *IKARUS-format*¹¹, a high-resolution description of the outline of characters as mentioned before (see fig. 3).

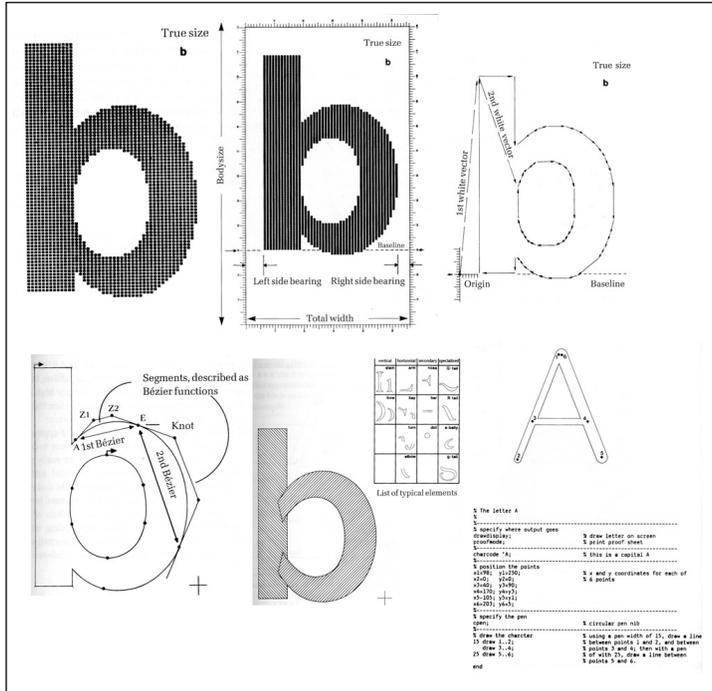


fig. 3

Six representations of typical formats (2 rows, from left to right): bit-map, run length, vector, PostScript, element separation, Metafont

In May 1973, the so-called Rockwell patent was filed. It claimed code compression for the run length code by using an outline described piecewise by vectors and circle segments as a *vector format*. Among other inventions, it claimed linear scaling from smaller to larger point sizes. A patent was given on that –unbelievable! Around the same time others introduced their own vector format to reduce the needed storage and to save costs.

In the late seventies, John Warnock worked on the PostScript language to describe the layouts of printed pages. In

1983, Adobe introduced the *PostScript format* for typefaces. In 1987, Apple came to me asking for help with a new format. Two years later they finished the development of the *TrueType format*, which was introduced by Apple and Microsoft in 1991. Neither liked paying for the licenses in order to use PostScript from Adobe.

After 2000, *stroke fonts* had a revival³⁶ as global fonts, which can consist of 50,000 characters or more. A stroke font reduces the storage needed by more than 80%. It does not need to incorporate things such as serifs and other embellishments of usual typefaces. Storage (= money) is still of high interest in the field of navigation systems and multilingual presentation of geographic maps.

In 2005 at a conference in London, David Lemon received the first-ever Linotype Font Technology Award for his decades-long work as a font developer: ‘Sumner Stone hired David Lemon to help Adobe turn typefaces into fonts. At Adobe, David became an expert in Type 1 font development, then multiple master fonts, and eventually had a chance to help direct the development of OpenType format during the years 1996 through 1999. As manager for the Adobe type team, David continues to work for the synergy of art and technology.’³⁰

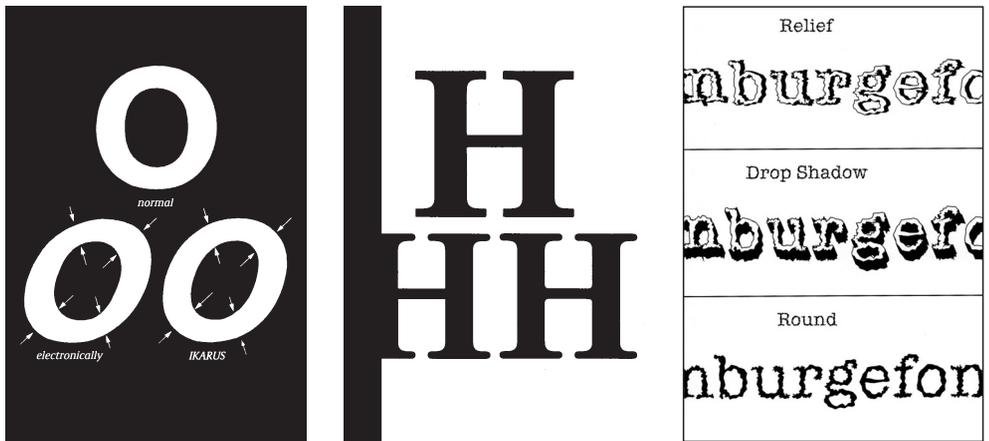
And so, finally, formats are recognized as part of digital typefaces. OpenType is the consummation of continuous development.

Variations

First of all, there were *contouring* and *shadowing*. Contouring could be inside or outside of the area, which the master character covers. It could be generated twice or even more often with varying thicknesses of the contour lines¹⁰. Shadowing could be implemented in any direction; however, people often constructed a shadow which had a length less than the stem

thickness and a direction of minus 45° (to the right and downwards: a drop shadow).

Italicizing gives a typeface the version ‘italic’. Of course, a real italic typeface is a new design and probably could not be generated by a program. However, there are numerous slanted versions of fonts as italic which are in use these days, and some of them have become quite famous. Statistics show: for italic, the angle 12.5° is taken quite often, and 7° up to 20° seem to be tolerable²² (see fig. 4).



You can replace all sharp corners of the characters by round segments. We called this *rounding*.

When you displace the control points of a character arbitrarily you get a contour shape, which is more or less distorted. We called this variation *antiquing*. Subtle distortions enable the characters to look like used hot metal characters, while large distortions give the appearance of frozen characters. One of the innumerable versions we referred to as ‘IceAge’.

Interpolation

Interpolation needs two statements in a loop, which runs for all digitized points of a contour. An outer loop runs for all contours of a character, and a third for all characters of the two masters of a typeface family. The statements are:

fig. 4

Italicizing (left) creates bad slanting effects, the *IKARUS* software corrects them automatically. *Rounding* (middle) rounds all corners of the characters. *Antiquing* (right) destroys deliberately smoothed contours, the generated font was called: ‘IceAge’

$$X_{new} = (X_{master2} - X_{master1}) * factor + X_{master1}$$

$$Y_{new} = (Y_{master2} - Y_{master1}) * factor + Y_{master1}$$

X and Y are the coordinates of the control points in a two-dimensional plane, and the factor could be set between 0,...1. Additionally, factors between -0.2,...0 will work for intrapola-tion and factors between 1,...1.2 for extrapolation. A factor of 0.5 generates a typeface in the ‘middle’ of the two masters. When the weight of master1 is ‘normal’ and the weight of master2 is ‘bold’, the interpolated version is referred to as ‘demi-bold’ (see fig.5).

fig.5
Interpolation (left)
between two weights of
a typeface, interpolating
hybrids (right) in be-
tween a light roman
(antiqua) and a bold sans
serif (grotesque) font



Interpolation is exceedingly simple for a programmer; however, it is a colossal tool for professional typeface production at the places of manufacturers for typesetting equipment. Interpolation became the basis for large font families. Sometimes there were up to ten weights in a family. The managers said, ‘interpolation only eats electricity, no spaghetti’.

One should know that the drawing of a new weight of a Latin typeface consists of at least a month of work or 170 man-hours. However, for a Kanji font it takes 100 months of 170 man-hours. Therefore interpolation was and is an enormous point of consideration in countries like China, Japan, and Korea (CJK).

When one takes a sans serif (grotesque) font as master₁ and a roman (antiqua) font as master₂, then one interpolates *hybrids* (see fig. 5). Most people don't find these very appealing.

For many years, only font designers and manufacturers used interpolation. It was a special function in programs like IKARUS, Fontographer (introduced 1986 by Jim von Ehr) and other software. In 1987, it was the now well-known company Adobe which put interpolation into the hands of all users by providing MultipleMaster fonts.

Rasterizing

Human eyes can resolve more than 600 dpi (dots per inch). A dot represents a displayed pixel or any other entity consisting of the smallest element of representation. For coarse displays with 60–120 dpi, effects of rasterizing show up immediately and create disorientation – even on printers with 300 dpi. These effects aren't visible on devices with 1200 dpi or finer resolutions¹³.

In November 1975, I started rasterizing fonts for the famous Digiset of the company Hell (see below remark 2). As long as the resolution of the device is high (600 dpi or more), arbitrary effects of rasterizing don't matter very much (see fig. 6). High-end typesetters can work with those 'bad scans'. But manufacturers like Hell and Linotype even edited their high-resolution images of characters by hand, wanting perfection and the elimination of a selling argument for the competitor.

Laser printers had low resolutions (180 up to 400 dpi). They needed hand-edited bitmaps of typefaces. In *office automation* the big printers didn't need a lot of typefaces or point sizes. These printers were relatively expensive and therefore generated the money to pay for the editing costs of bitmaps. Therefore, office automation didn't lead to any improvements in rasterizing worth mentioning.

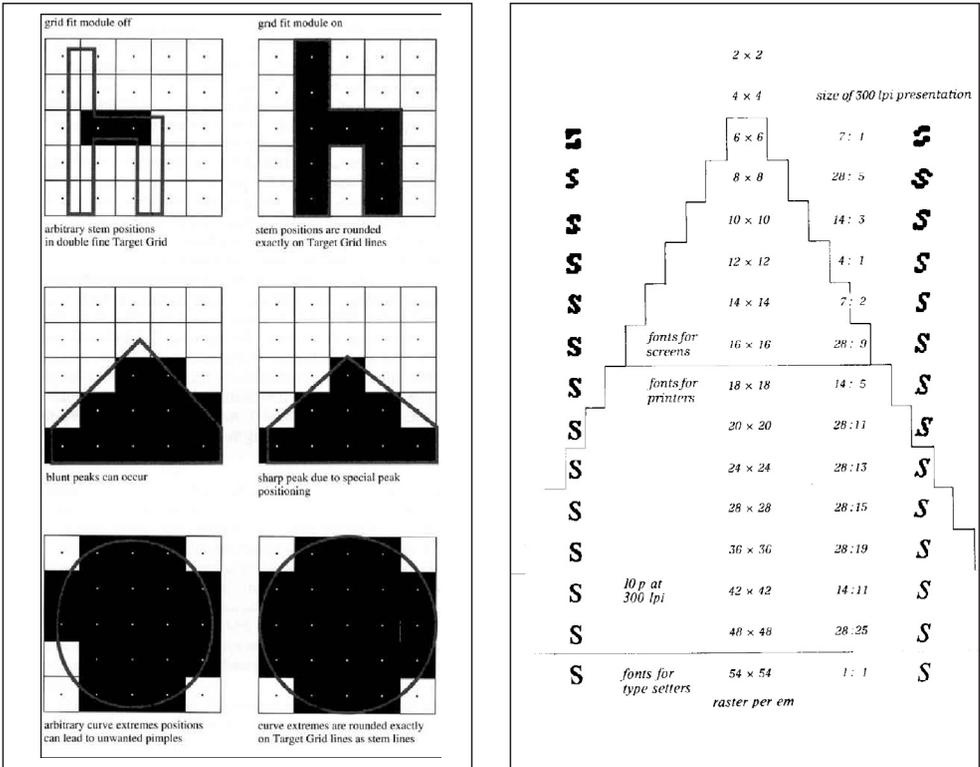


fig. 6
Bad rasterizing effects may occur (left side of left illustration). Comparison of coarse with fine rasterizing (right): the coarser the grid, the narrower the scope of design (resolution funnel)

The real impact came from *desktop publishing* and ‘Triple A’: Apple, Adobe, and Aldus. One wanted everything ‘wysiwyg’ (what you see is what you get). The VDTs (video display terminals) had a resolution of about 60 dpi and later 120 dpi (see below *remark 3*). They should have been used as raster scan VDTs similar to the famous Xerox Star in 1980. One needed many typefaces and many point sizes!

All point sizes had to be hand-edited in order to obtain reasonable representations for 9, 10, 12, 14, 18, and 24 pt. fonts. In the beginning, in order to reduce the editing job, one selected around 11 typefaces: Helvetica, Times, Courier, Symbol (the 13 ‘core fonts’), Helvetica narrow, Palatino, Century Schoolbook, Avant Garde, Bookman, Zapf Chancery, and Zapf Dingbats. Besides the 3 typefaces Symbol, Zapf Chancery, and Zapf Dingbats, all of the other 8 typefaces had four versions: normal, bold, italic, and bold italic. Altogether there were

$3 + 8 \times 4 = 35$ fonts, and $35 \times 6 = 210$ Bitmaps. Quite a lot. A compromise for the beginning, of course, but not for long.

Hinting

Already in 1976, I had started to automate the hand editing of bitmaps because our company had spent a lot of time producing bitmaps for low resolution (see below *remark 4*).

The IKARUS program put out a bitmap per character per point size per typeface per version, formatted for one of the various types of laser printers or typesetters (see fig. 7). Needless to say, it generated the bitmaps slowly. We never thought of doing it ‘on the fly’.



fig. 7
Rasterizing 3 different
fonts at 9 pt, 18 pt, 36 pt
for 72 lpi without (left)
and with hinting (right)

Nevertheless, IKARUS performed auto-hinting. At first its abilities were limited and it could only find stems and bows. Step by step we improved it and finally *IkarusM* (the Macintosh version) could find most all of the 17 hints automatically (see fig. 8). Here again we met with artificial intelligence (see below element separation), which has yet to be completed.

I taught rasterizing and our version of ‘hinting’ in order to avoid hand-editing and storing bitmaps to Compugraphic and Linotype (Bitstream) in 1980. In 1982, the IKARUS program made its way to the former company Autologic near Los Angeles – and to its production manager Sumner Stone.

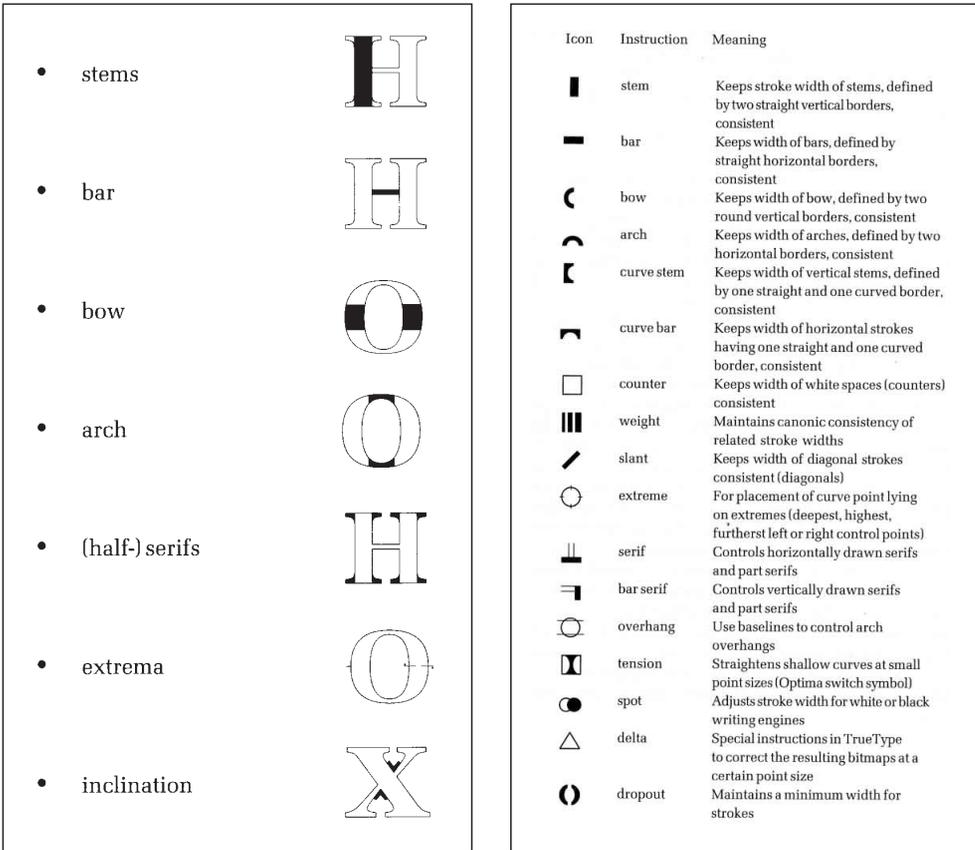


fig. 8

8 hints in the beginning (1981, IKARUS) 17 hints later (1987, PostScript, TrueType)

In 1983 Adobe announced PostScript. I met with John Warnock the first time during the ATypI working seminar at the Stanford University. A year before, he and Chuck Geschke had founded Adobe. They knew best what desktop publishing meant for PostScript: many characters and many point sizes and many typefaces, that hand editing didn't have a chance, and that the rasterizing should be fast enough to allow it 'on the fly' in order to avoid expensive memory for the equipment in DTP.

In 1983, I told John Warnock about the IKARUS system at Autologic, its auto-hinting and their type director Sumner Stone. In 1984, Adobe hired him. He was then at Camex in Boston, after Autologic. With him, *John Warnock and Bill Paxton made the real invention: hinted fonts rasterized on*

the fly. Since 1985, hinted fonts could be sold by Adobe and rasterized on its PostScript RIPS. Naturally it became a big success.

In 1988, Apple sent some software engineers to URW in Hamburg in order to learn about rasterizing and hinting. Later in 1990, we made our own rasterizing software Nimbus¹⁷, which existed among others like those of Bitstream and Compugraphic who all learned from us.

Autotracing

After the IKARUS system was established in the field of typesetting and printing, many managing directors came to us and asked for the automation of hand-digitizing, especially the Japanese. Their companies had stored lots of characters as large high quality images. Quite naturally, they wanted to scan and convert them into outlines automatically.

Another impact came from users of illustration programs like Illustrator and FreeHand. They also wanted to acquire the outlines of scanned objects.

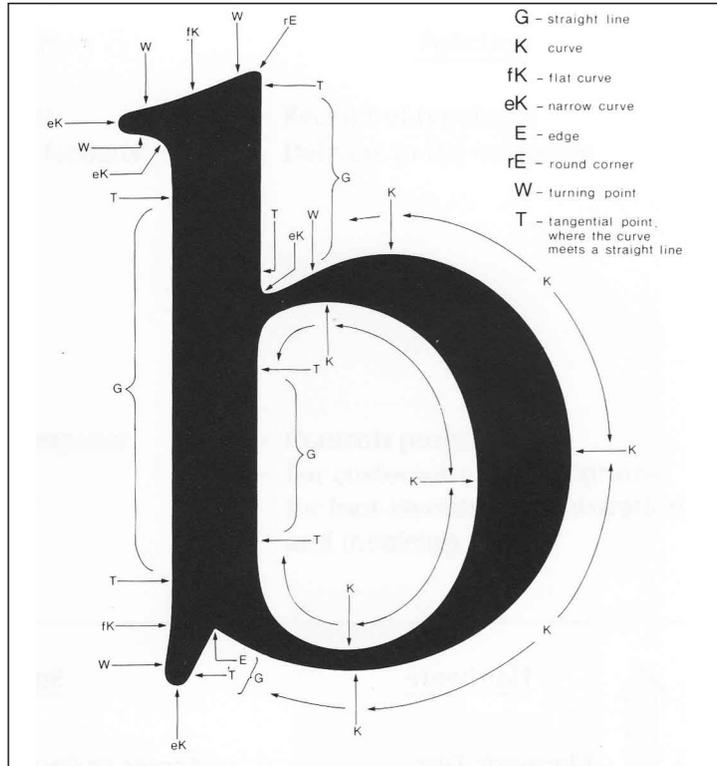
In 1980, we started autotracing and called the software ‘Outliner’³⁹. We tried it several times. Step by step, year by year. Every time, we believed we had a better idea as the solution. Every time, we underestimated the task (see fig. 9).

Scanning is full of arbitrary local effects, especially for corners and flat curves. Design decisions must be made: [1] is this scanned corner still round or already a sharp corner, [2] is this flat curve still a curve or already a straight stem, [3] is this an acutely angled joining of strokes – a so-called ink trap – or not, [4] is this a curve or a flat joining of stems?

As human beings, we are biased by our own ability to recognize shapes and especially to guess the details of a shape and to make design decisions correctly. We can do it fast and think that it is easy. We all know that we calculate much slower than computers. And here begins our misunderstanding. We

fig. 9

Autotracing should distinguish between straight lines and flat curves, corners should be recognized either as sharp, or round or as narrow curves, and finally autotracing should find tangential control points



believe therefore that computers can make design decisions (pattern recognition) as fast as man can do it or even faster. But that is not true, at least not today.

Since 1969, I was involved in pattern recognition, OCR, and image processing. Autotracing belongs to this chapter on artificial intelligence. Programming work and the experience of more than one generation is needed. Today in autotracing, we have achieved a certain level, and yet we are still far away from what could be achieved in the future.

Grayscaleing

The line printers of IBM were dot matrix printers for all of those years. The VDTs could only write in one character size, and sometimes only using the upper case character set. All programmers were used to it at the time, they were happy to have

achieved the first representation of various point sizes and typefaces in DTP.

Looking at the Xerox Star in 1980, then at Apple's first Lisa in 1983 and Macintosh in 1984, we saw the jagged bitmap representations of characters. A group of people argued against grayscaling⁴³, some called it fuzzy².

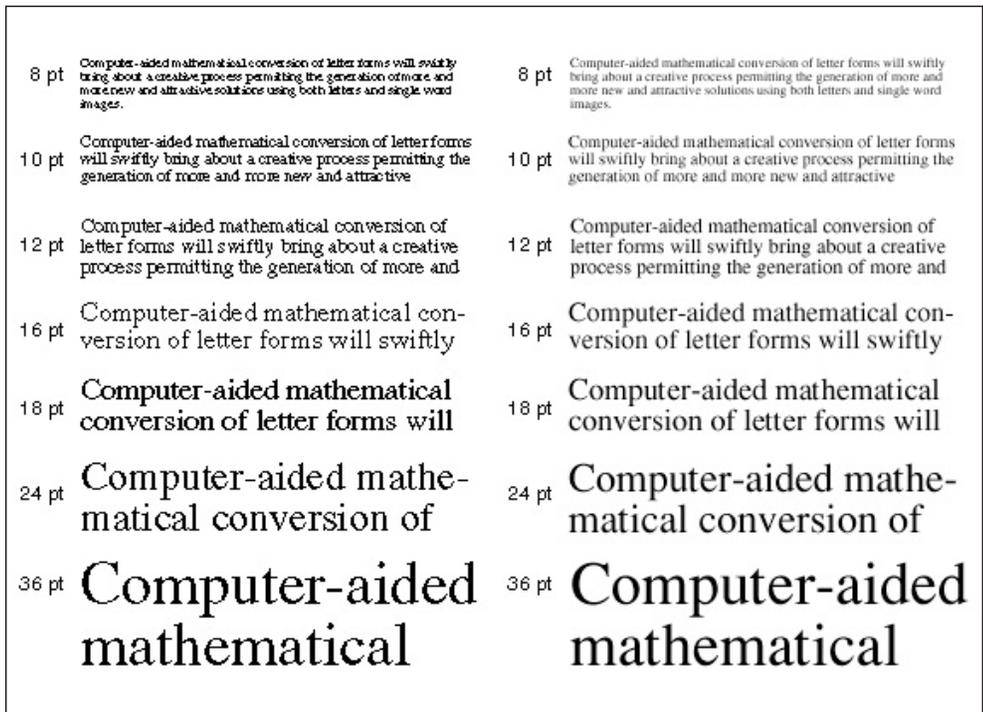
In 1980 John Warnock⁴⁰ worked on the display of characters using gray level sample arrays. In 1985 Avi Naiman³² published his model for grayscaling.

I started grayscaling in 1981, not in order to achieve a better vDT, but to build a video terminal for proofing typesetter output without wasting film. Memory was expensive and limited at that time. In 1984 one could only dream of having more memory than 128 KB. Therefore, I wanted it as additional equipment to Hell's Digiset (see below *remark 5*).

Normal TV guided me. I took a picture of a piece of printed paper with a video camera in order to get a grayscaled rep-

fig. 10

The left column is bitmapped, the right is grayscaled. Especially at lower point sizes, the grayscaled text is much easier and faster to read



resentation of text with 9 TV-lines corresponding to 9 pt text at 72 lpi. I saw that a very simple method would do it: generating a graymap by converting 4×4 fine pixels. 2×2 is too coarse, 8×8 is not necessary.

In 1993 we finished the software ScenMaster^{19|21} that could generate more than 10,000 grayscaled characters per second on a Macintosh – again on the fly (see fig. 10).

For a long time in DTP, people were not interested in grayscaling because of additional costs for memory and additional computation time. Relatively late – meanwhile computers had much more memory and were much faster – grayscaling came in 1994 to Adobe as well as Microsoft³. In 1995 grayscaling became an integral part of Acrobat and ATM. Today it is the standard.

Element separation (Shape components)

In 1975 Philippe Coueignoux⁶, a student of William Schreiber at MIT – used shape components (a part of element separation) in one of the first fully digital typesetter controllers. It is the earliest work known to describe typographic characters by digital shape elements (see fig. 3, row below, in the middle).

In 1979, Donald Knuth published Metafont, the next innovation done in the field of parametrizable fonts^{25|26|28}. The main components such as horizontal, vertical, diagonal strokes, and round parts are described as the path of a pen with given orientation and pen width. A sequence of them with individual pen stresses, positions and directions describes a character and generates its outline.

In 1998, Uwe Schneider³⁷ published his work on ‘Infinifont’, an object-oriented model for the hierarchical composition of letterforms in typeface design.

In 2001 Roger Hersch and Changyuan Hu, following Hu’s publication in 1998⁵, published a proposal for a new, highly flexible font description using assemblies of parametrizable

shape components. Their system can derive fonts that vary in weight, condensation, and shape.

In 1990, URW started collaboration on element separation with the company of Fujitsu. Jürgen Willrodt⁴² and I aimed at these aspects: computer-aided design of Kanji, reducing the needed storage space, and speeding up the rasterizing of Kanji. We separated Kanji into elements, elements into strokes, and strokes into parts. We programmed tools to disassemble and to assemble them.

This looks like an easy task, but it isn't. Many Kanji characters – especially those of bolder fonts – have elements and strokes that overlap. Naturally, the parts of strokes are connected. Therefore, it took a lot of programming to implement all of the functions which were needed to get the right locations for separation and smooth junctions for the assembly of Kanji without interactive assistance.

Separation needs recognition of parts similar to auto-hinting, and the opposite – the assembly – has to compensate those parts of the counter for discontinuities where parts have been put together.

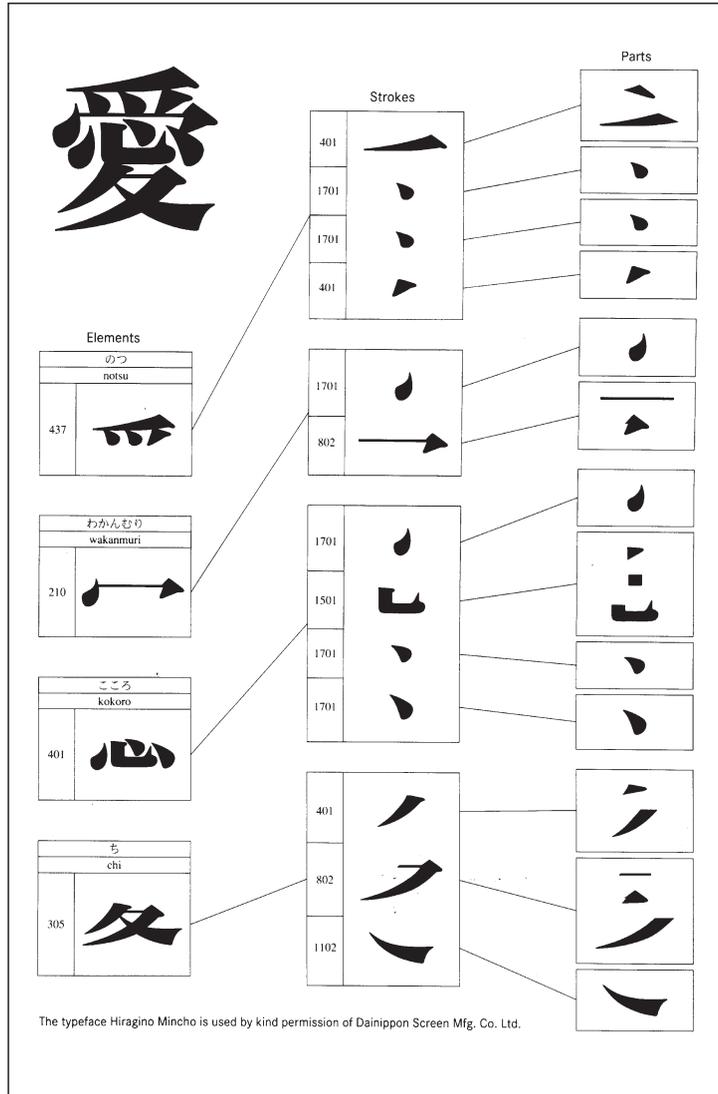
The next challenge was to instruct a program to decide typographically whether parts are the same or strokes, or whether elements are the same. Again it turned out that we wanted to automate something, which is relatively easy for a designer, but belongs to artificial intelligence in computer science. It ended up as a big success, but also with a lot of compromises with respect to automation of design decisions (see fig. 11).

Perfect element separation belongs to artificial intelligence. Moreover, on the one hand it will contribute to the computation and production of pretentious handwriting, and on the other to the reading of human handwriting.

	# being different	# being real different (4% deviation)
Kanji	7 000	7 000
Elements	21 000 (x.3)	17 000 (80%)
Strokes	70 000 (x.3.3)	14 600 (23%)
Parts	210 000 (x.3)	6 600 (5%)
	▲ Showing 4% deviation due to handwork (h) and digitization (d)	
	▲ Regularized to keep out effects of h & d	
Needed memory	Image 6.3 MByte Instructions 2.0 MByte Administration 0.4 MByte Total - 8.5 MByte	Image 0.26 MByte Instructions 0.13 MByte Administration 1.126 MByte Total - 1.5 MByte

fig. 11

The separation of Kanji characters into elements, strokes, and parts worked quite well. It saved storage space and served to repair arbitrary effects of hand-drawing



Digital composition

When is text composition digital?

Great fundamental changes happened to text composition in comparison with the analogue situation: text editing, text templates, animation, and others. Editing was really annoying when changes to a large typewriter document were necessary. People had a lot of work to do in order to write a new document based on text already written in other forms of context.

Now there are many tools in the menus, also many powerful ‘assistants’. Some of them never show up, but rather they work like wizards in the background. Text composition is still done manually, but it is quite facilitated. However, the text is not composed digitally.

Digital text – in the sense as I see it – is composed by programs without manual assistance. It takes pure text and a composition model as input and then generates the text and its layout (typography) automatically. Here are the following components:

Kerning

In hot metal printing, kerning had to be created by using pre-designed ligatures for ‘fi’, ‘fl’, and so on. In photo typesetting kerning was created on a broad basis. One generated it manually or used kerning tables to change the individual distance of pairs of characters⁷. David Kindersley filed a patent in 1965²³, but he was not very successful with his system.

Together with Margret Albrecht I started automatic kerning¹³ in 1980. We wanted to save money because the generation of kerning tables along with left and right side bearings took a lot of time in our typeface production. As in other cases of artificial intelligence, we had to go through several approaches throughout the years until 1987. We mixed programmed ideas and heuristic parameters gained by processing a lot of existing kerning tables manufactured by different companies. We expanded kerning to get overlapping and blending of characters in tightly composed words²⁰ (see fig. 12).

In 1993¹⁶, our program could ‘kern’ 15,000 characters per second on a Macintosh at that time. Therefore, we called it ‘kerning on the fly’. It finally worked with digital text composition. In 1995, ‘kerning on the fly’ was implemented in InDesign by Adobe.

‘Kerning on the fly’ kerns better than a human being can do with respect to one typeface and all its various applications.

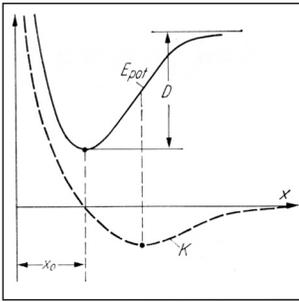
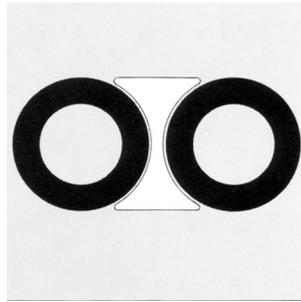


fig. 12

Kerning can be regarded as a power that repulses characters the nearer they come to each other, and that attracts characters the farther they get from each other (left). Kerning can be used also to calculate character positions for overlapping and touching of text

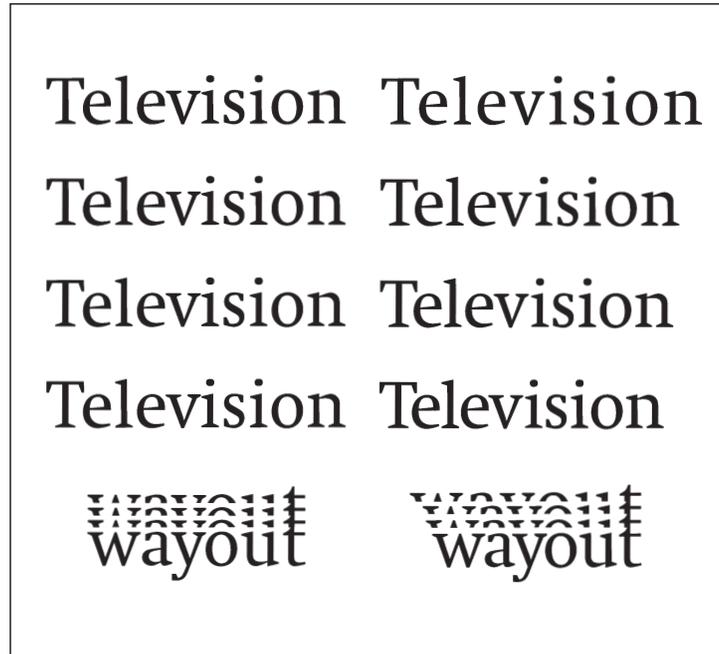


This has statistical reasons; human beings are biased in regards to their actual aspects and impressions. However, with respect to one specific word such as a logotype written out of a special typeface, some designers may kern better.

Here begins the area of taste. And as we all know, we can discuss taste endlessly. I am not astonished that the US Patent Office filed patents on kerning matters recently^{4|33}. Still, it is possible that kerning could be improved (see fig. 13).

fig. 13

Kerning supports the writing of different pointsizes (6 pt, 12 pt, 36 pt, 72 pt from top to bottom): the left column is without, the right with kerning



Optical Scaling

In hot metal printing, optical scaling was usual⁹. In any case one had to cut the point sizes individually, so it was a matter of knowledge, but not of money. This changed when photo type-setting came up and the possibility of linear scaling came into existence. Optical scaling didn't play a role in the beginning of DTP, however, a lot of people wrote about it¹.

In 1991 at URW, I made the following approach for text fonts:

The smaller the type size:

1. the wider the composition
2. the thicker the strokes
3. the broader the characters, especially the lowercase.

The larger the type size in titles:

1. the more compact the composition
2. the thinner the strokes, especially the hairlines
3. the narrower the characters, especially the lowercase.

Simplified, one applies the rule that space and stroke width of light fonts (text fonts) are reduced or enlarged by 7% on the average if the point size is enlarged or reduced by a factor of 2¹⁸ (see fig. 14). For bold fonts the opposite is true.

To my knowledge, optical scaling was not employed for bold fonts in the past because they weren't (and still aren't) used very often, and if so, they were cut just for these special cases in certain point sizes as a special effort.

Optical scaling is left for refinement and artificial intelligence. Some unique individuals may not need it. But among us normal readers the following is true: *linear scaling isn't sufficient, we want optical scaling as an intrinsic quality of text composition.*

fig. 14

*Different point sizes,
which have been
generated at the same
size for the purpose of
comparison*



Paragraph Composition (hz-program)

The *hz*-engine – developed with and named after Hermann Zapf^{14|44} – uses a justification per paragraph system^{24|27}, along with ‘kerning on the fly’ and expanding/condensing of characters in order to obtain margin lines for a column that is optically straight (optical margins), and achieve typeset spaces among words within lines of text that are fairly constant in order to avoid rivers and creeks.

Rivers run vertically through poorly spaced words in consecutive lines of text when the spaces between the words have the same space or a greater space than the distance between the

baselines of the text. In contrast, a creek is a less severe form where the spaces between words are accidentally too wide within one line.

The basic feature of the *hz*-engine, which was programmed by Margret Albrecht, is to regard all lines of a given paragraph at once – as described by Donald E. Knuth in *Metafont*. At first, all words or syllables are distributed to the lines together in a manner where each line gets a line length nearest to its given individually parametrized width (as default there is usually column width). The following optimization is controlled by minimizing the typographical demerits, which are obtained from a function of the actual line lengths, given line lengths, given line widths and tolerances of the layout parameters.

If hyphenation is turned on, words are replaced by syllables. The *hz*-engine has to follow a lot of exceptions and to provide solutions for them, e.g. ligature substitution, consecutive hyphens and good or bad locations for hyphenation within a word. This level of text/typographic detail promotes a better fit and contributes to the reader's comfort (see fig. 15).



fig. 15

A comparison between the *hz-engine* and today's typical composition tools demonstrates the superiority of the *hz-engine* (see fig.16). In 1995, the *hz-engine* was implemented in InDesign by Adobe. This has been a big step in digital text composition, but truly it is not the last one.

fig. 16

Hermann Zapf 1986:		
<p>Magazine Composition</p> <p><i>left two columns:</i></p> <p><i>hz-program (most left) compared with today's software,</i></p> <p><i>narrow columns, hyphenation on.</i></p> <p><i>right two columns:</i></p> <p><i>hz-program compared with today's software (most right),</i></p> <p><i>narrow columns, hyphenation off which is unusual and used as a test.</i></p>	<p>Writing is the visual reproduction of the spoken word, its primary objective being to convey a text to the reader without difficulties, or distraction, and without disturbing the flow of reading with unnecessary embellishments. The letters have no self-fulfilling purpose, neither are they a medium for self-presentation of a designer. Everything which makes reading difficult or time-consuming, or is detrimental because of its unusual form, has to be avoided. The new technical possibilities of type composition - with all its limitations - also determined the form of the letters. The infinite possibilities provided by today's electronics are used for example to develop types of our time, without historical hangovers. Ideally, the <i>hz-program</i> comprises (1) kerning on the fly, (2) optical spacing, (3) expanding and condensing plus optical scaling from Multiple Master fonts and (4) justification per paragraph. It is the non-plus-ultra in typography.</p>	<p>Writing is the visual reproduction of the spoken word, its primary objective being to convey a text to the reader without difficulties, or distraction, and without disturbing the flow of reading with unnecessary embellishments. The letters have no self-fulfilling purpose, neither are they a medium for self-presentation of a designer. Everything which makes reading difficult or time-consuming, or is detrimental because of its unusual form, has to be avoided. The new technical possibilities of type composition - with all its limitations - also determined the form of the letters. The infinite possibilities provided by today's electronics are used for example to develop types of our time, without historical hangovers. Ideally, the <i>hz-program</i> comprises (1) kerning on the fly, (2) optical spacing, (3) expanding and condensing plus optical scaling from Multiple Master fonts and (4) justification per paragraph. It is the non-plus-ultra in typography.</p>
<p><i>hz: 10 hyphens, 3 lines less</i></p>	<p><i>Today's software: 12 hyphens, 3 more lines, bad in line 11 and 12</i></p>	<p><i>Today's software: spaced out words are worst typography, line 14 has a ligature!</i></p>

Chapter Composition (chapter-fit)

The initial idea of chapter-fit was to apply the same kind of automation to chapters as is available for paragraphs¹². Whereas the *hz-engine* handles and optimizes the layout of

paragraphs (sentences) / lines / words (syllables) / characters, the chapter-fit handles the layout of *chapters / pages / paragraphs (sentences) / lines* in order to optimize the presentation of text working at two levels higher. *Chapter-fit* calculates all lengths of paragraphs (similar words) and lengths of pages (similar lines) and balances them for a chapter (similar paragraphs) in a manner that each individual page (column) gets an optimal amount of paragraphs which can be managed and fitted with a minimum of imperfections into the given layout. A special paragraph-fit and page-fit are needed.

Paragraph-fit composes a paragraph in a manner that its last line gets a certain desired length compared with the line (column) width. Last lines that are shorter than a certain minimum are not desired. They open too large a space between paragraphs; they cause 'orphan paragraphs'. Likewise, lines which are longer than a certain maximum are also not desired because they give no clear optical indication for the end of a paragraph; they cause a 'widow paragraph' (see fig. 17).

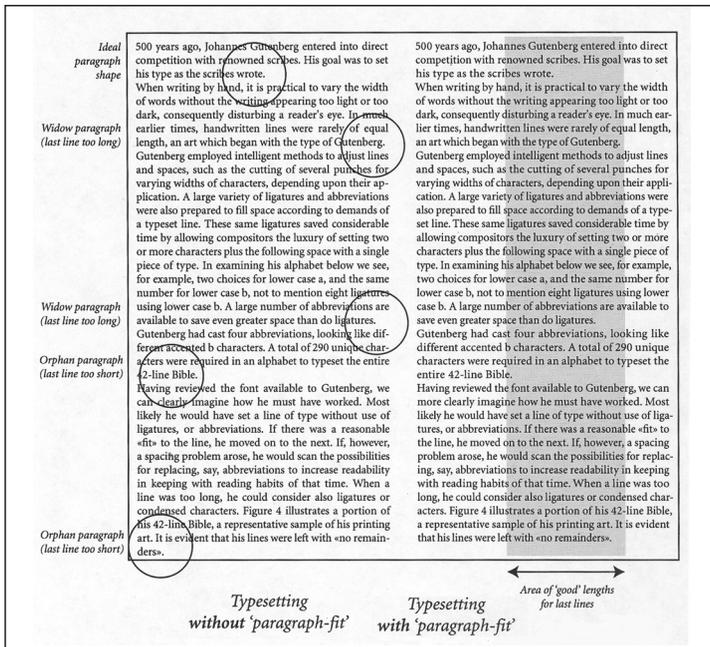


fig. 17

The right column shows paragraphs that end with a last line of 'typographical length'

Page-fit processes all lines and paragraphs within the text according to the original parameters for typesetting, such as point size, line leadings, line width, etc. Then all paragraphs are distributed and counted. It aims to terminate each page with the end of a paragraph under consideration of the potential for shrink or spread of the given text.

Some of the last paragraphs on the pages have to be divided into two parts. A special feature called *hyphenation of paragraphs* performs this. Paragraphs are divided best at a punctuation mark such as a full stop (see fig. 18).

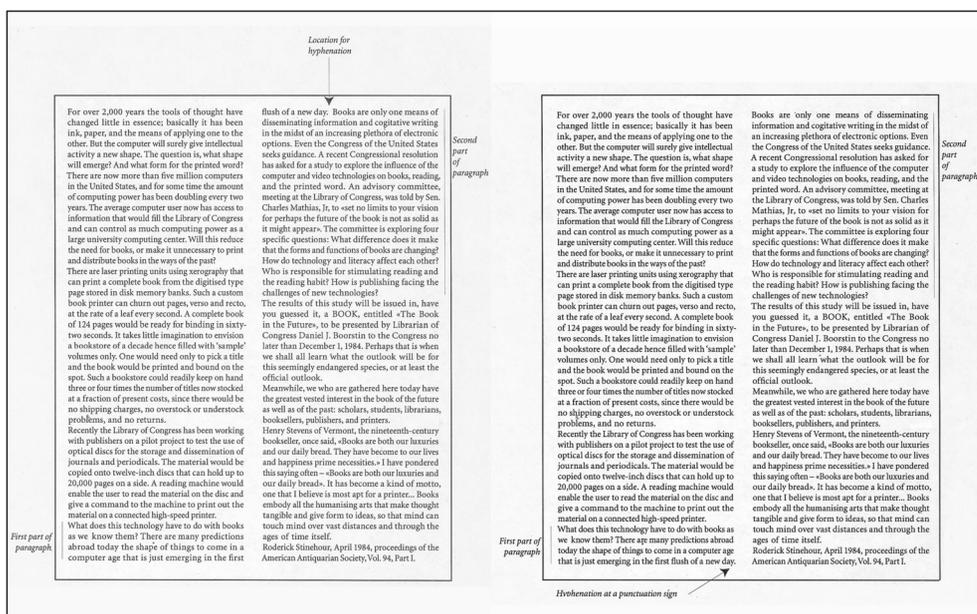


fig. 18

The left illustration shows a hyphenated paragraph without paragraph-fit, the right with paragraph-fit

Depending on the length and typographic complexity of a chapter, this may cause many time-consuming trials because of necessary iterations and tests to find the optimal distribution. The *chapter-fit* also avoids a chapter ending with a page that is too long which would create insufficient room for footnotes ('widow chapter'), or one having a last page which is nearly empty ('orphan chapter').

Especially important is that *chapter-fit* should obtain an even number of pages to allow the start 'on the recto' (right

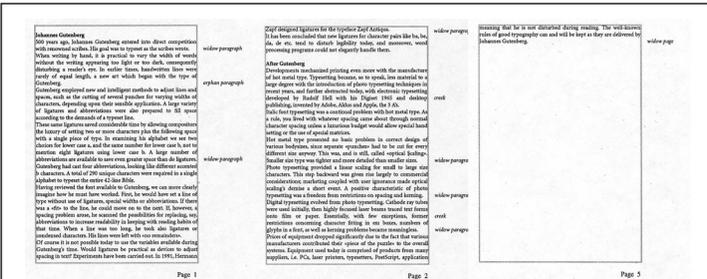
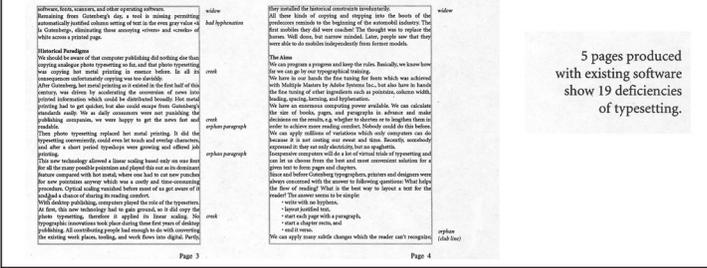


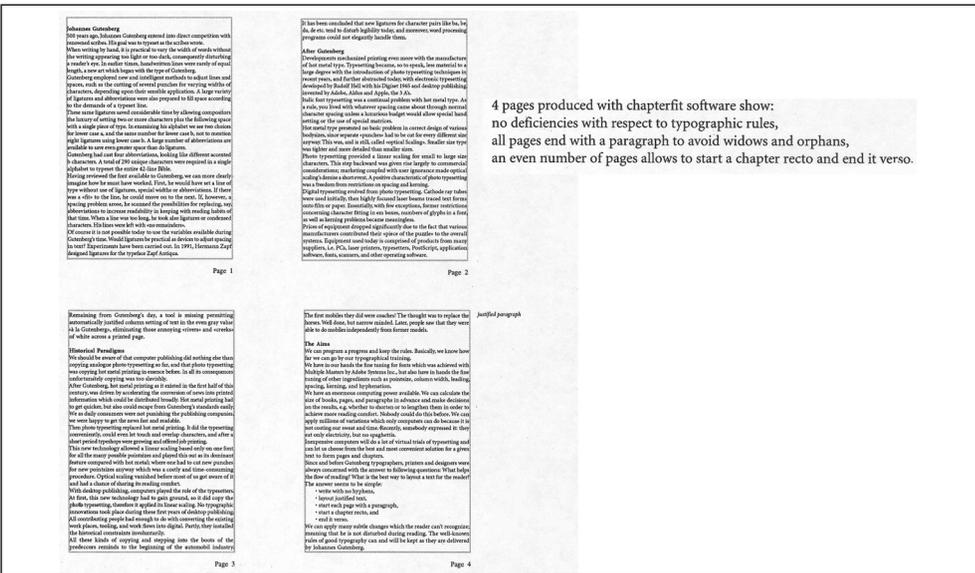
fig. 19
A chapter with 5 pages before chapter-fit



5 pages produced with existing software show 19 deficiencies of typesetting.

fig. 20
Same chapter as in fig. 19 with 4 pages after chapter-fit. The example has paragraphs which have reasonable lengths of last lines formed by a paragraph-fit. It has pages which let it start on the recto and end on the verso, and it was typeset without hyphens. Each page starts with a paragraph: No widows, orphans, creeks, or rivers are to be seen

page) and the end 'on the verso' (left page). Compare fig. 19 and fig. 20. Still, page-fit could be improved in order to handle the 'famous exceptions' which are mentioned by developers so often. In reality, the handling of these exceptions belongs to artificial intelligence. I think the long list of these anomalies makes it interesting for computer scientists.



4 pages produced with chapterfit software: no deficiencies with respect to typographic rules, all pages end with a paragraph to avoid widows and orphans, an even number of pages allows to start a chapter recto and end it verso.

AdMaster let us come down to earth. Gerhard Rubow and Jochen Lau implemented a system which had a hierarchical format for the layout (see fig. 21). They called it IG-format. Heredity of attributes and typographical rules were the fundamental ingredients. Generic templates were the cornerstones (see fig. 22).

fig. 22

The software is based on 8 levels of hierarchy (left), terms of the heredity of attributes (right)

Levels of Hierarchy	
• Point	anchor points, determining the shape of a contour
• Contour	inner and outer contours of a character
• Character	a letter or a logo
• Line	one or more characters
• Group	one or more lines, headline text and body text
• Layer	one or more groups
• Graphic	one or more layers
• Ad	one or more graphics, the complete ad

Terms	
• Subobject	object beneath the current object, on lower level
• Superobject	object above the current object, on higher level
• Predecessor	object before the current object, on same level
• Successor	object behind the current object, on same level
• Production mode	setting of ads with reduced levels of hierarchy

Meanwhile, this system was further developed as VI&VA and became a system of Lufthansa Systems AS²⁹. It is used by large newspaper companies in order to generate small ads interactively and to show them to the clients for acceptance (see fig. 23 and 24). It is the beginning of digital ads.

fig. 23

A picture is changed. It results in the layout of three groups of text, which is generated automatically.



‘The make-up is pre-programmed’ is still a dream. Design decisions for layouts are much harder to program than problems like rasterizing, autotracing, element separation, and kerning.

fig. 24

Two obituaries, the above with more text in the upper group of text, below with less text. The software balances the layout of all text lines automatically

	<p>S t a t t K a r t e n</p> <p>Nimmer vergeht, was Du liebend getan. Adding text and even more text and another line and still more lines lines lines lines we find the bodysize of the text decreased.</p> <p>Wir trauern um</p> <p>Nicole Hand</p> <p>geb. Tergani * 13. 4. 1934 † 27. 7. 2002</p> <p>Wir werden sie nie vergessen.</p> <p>In stiller Trauer Ludwig Weide Maria Schiedel geb. Weide Markus Schiedel Lara und Sebastian</p> <p>Ronnenberg, den 28. Juli 2002 Die Trauerfeier findet am 1. 8. 02 auf dem Ronnenberger Friedhof statt. Bestattungsinstitut Muster, 12345 Hannover</p>
	<p>S t a t t K a r t e n</p> <p>Nimmer vergeht, was Du liebend getan.</p> <p>Wir trauern um</p> <p>Nicole Hand</p> <p>geb. Tergani * 13. 4. 1934 † 27. 7. 2002</p> <p>Wir werden sie nie vergessen.</p> <p>In stiller Trauer Ludwig Weide Maria Schiedel geb. Weide Markus Schiedel Lara und Sebastian</p> <p>Ronnenberg, den 28. Juli 2002 Die Trauerfeier findet am 1. 8. 02 auf dem Ronnenberger Friedhof statt. Bestattungsinstitut Muster, 12345 Hannover</p>

What will the future bring?

On the one side, most of the designers dislike digital text since they are fearful that it may dissolve their professional existence. On the other hand, the manufacturers of software know this and slow down. But work is going on in the field. It will bring back the optical adjustments and contextual variations which have been removed by computers since the seventies.

Hermann Zapf said this: 'Pleasant looking, easily readable text has been the aim of typographers since scribes began to write and Gutenberg perfected printing with movable type. Text transports information. It must be accomplished with a minimum of communication errors. But it does even more than this; its typographic appearance establishes the care with which it is presented – its authenticity and its persuasive power. The goal is the creation of a harmony in which text, images and graphics are related in a controlled manner.'

I am anxious to see the next steps. I am sure that we will have the benefits of the analogue past and the digital present in the near future. Artificial intelligence will take place and give us more digital text. And this will be a remarkable accomplishment.

In the fifties, people like me thought that motors in automobiles were perfect and needed no more improvement after having 50 years of development behind them. A comparison of those engines with today's motors shows how wrong they were. Some of us think similarly today that perhaps digital typefaces and especially digital text don't need improvements anymore. But they haven't been perfected yet; rather, they still need our engagement and talent.

Remark 1***Master disks for photo typesetting***

My first customers used digital outlines to cut vinyl foils (rubylith sheets) on flatbed plotters. Rubylith consists of two layers. The lower is transparent, the upper coloured. The upper was cut and peeled off where one wanted to have transparent areas. Character by character was cut and peeled. Together they were placed in a large grid on a wall to be photographed as a font which could be reproduced on plates to make the master disks for photo typesetting machines.

Remark 2***How to imagine rasterizing***

To understand rasterizing, one should imagine a character drawn in black on a white sheet of paper. Then one visualizes rasterizing as putting a transparent grid foil over this character and painting all cells black which cover more than 50% of the black area of the underlying character. A black cell is called a black raster pixel. The character size should be regarded as fixed for the present, but the grid size is flexible. A larger (coarser) grid would give less raster pixels, a smaller (finer) grid more. So, a coarser raster generates a smaller character, while a finer grid generates a larger character on a display device, which – typically – has a fixed resolution.

After one has rasterized characters for the first time, one immediately learns the job: avoiding arbitrary effects. In the case of an antiqua lower case m for example, one can usually realize this: none of the three vertical stems have the same width counted in raster pixels nor do the two white spaces in between. Furthermore, the six parts of the bottom serifs have different forms and the two upper ones have different round bows. The upper serif of the first stroke is different from that of character n. Many of the sharp corners are not really sharp because raster dots are missing. And so on.

Remark 3***What does 72 dpi mean to the variety of typefaces?***

One inch measures 72 pts approximately. A point size of 9 pt for a 72 dpi display is taken as a rough but typical example. We assume quadratic raster cells. Under these assumptions, we have just 9 pixels for the body size, 6–7 for the cap(ital) height, 3–2 for descender length, and 4–5 for lower case height.

We raster the lower case character e: it has a horizontal bow at the top, a horizontal stem in the middle, and a horizontal bow at the bottom. In between these three black strokes there are two white spaces which separate them. Therefore, we need at least 5 grid lines to represent the image. We don't have the freedom to represent the thin horizontal stem in the middle of the e by half a pixel or a quarter of it.

Consequently, one can imagine that it isn't possible to represent e's of different typefaces at 9 pt for 72 dpi differently. And one realizes that characters at that size need hand-editing at least for the purpose of working out coarse differences.

Remark 4***How hinting works***

We imagine rasterizing as putting a transparent grid foil over an underlying character drawn black on white paper, and painting all grid cells black which cover more than 50% of the black area of the character. If one shifts a part of the outline of the character in any direction one gets different rasterizing results. If one doesn't do it arbitrarily, but rather in the right manner, one improves the rasterizing result.

As an example, we can vision the character m of a very light sans serif typeface. The m has three vertical stems and two vertical white spaces in between. We want to generate it for the size of 9 pt on a 72 dpi display. In this case, nine pixels are available as the maximum to the total width. Now we have to make a decision about the design of the bitmap: we decide to give

a width of 1 pixel to each of the three vertical stems, two pixels to the two white spaces in between them, one pixel to the front of the m as left side bearing, and one pixel to the back as right side bearing. Altogether, we have distributed nine pixels: $(1+(1+2+1+2+1)+1)$.

We assume that the real left side bearing of the m is 1.4 pixels at this size and the stem width 0.85 pixels. We put the raster foil on it and fit the farthest left grid line with the front of the m. It is marked as the front of the left side bearing. So, stepping to the right in the middle of the m-height, the first vertical grid line lies in the white area before the m, but the second lies just in the middle of the first black stem, namely leaving 0.4 of a black pixel on its left side and 0.45 of a black pixel on its right side. So, we count no black pixels on the second and the third vertical grid line. We have a rasterizing accident!

Hinting now plays its role and helps with the rasterizing. It identifies the left and right outline of the first stem of the m. Also, it knows how to make design decisions such as those above. Therefore, it knows that the left side bearing should get one pixel and the stem width also one pixel.

At first, it shifts the total m to the left until the left outline of the first stem is positioned exactly under the first grid line. Then it shifts all parts of the outline of the m, which are located to the right of the first grid line. It shifts them until the right outline of the first stem is positioned exactly under the second grid line. This only demonstrates one implementation concerning the execution of hinting, in reality it is much more complex.

Hinting is executed consecutively and really fast, stem-by-stem, white space-by-white space, bow-by-bow, and serif-by-serif. Arbitrary effects are avoided step by step.

There are at least 17 different hinting instructions: stem, bar, bow, arch, curve stem, curve bar, counter, weight, slant, extreme, serif, bar serif, overhang, tension, spot, delta, and dropout^{11, P.112}

Remark 5***Grayscale is important for the fast transport of text***

Our sight is based on our eyes and an admirable system of pattern recognition in our brain. The eyes focus on a relatively small area (to get more information) and simultaneously catch an overall larger surrounding of it (to get less information)⁴¹. Logically, this reduces the flow of imaged details to the brain and is adapted to its enormous, but still limited power of performance.

Images are probably decomposed into locally received differences of light and color signals with the help of the retina⁸, sent as picture elements via nerves to the brain where the images are improved at the edges, reconstructed, compared with primitive patterns in the beginning, evaluated according to previous experience, and assessed to formerly received images. Finally, this process forms the «real» image which we see individually.

While reading, experience has trained us to extract the information itself as early as possible and to discard most of the received images in order to avoid an «overflow» in our memory. For fast reading, we rely on familiar typefaces in familiar media like books, newspapers, and letters⁴¹.

Today, computers are also becoming a familiar medium as well. Therefore, its representation of text has to be as familiar and as typical as possible.

References

1. André, A., Vatton, I.: 'Dynamic Optical Scaling and Variable-Sized Characters', *Electronic Publishing* 7(4): 231–250, 1994
2. Bigelow, Ch., *Principles of Type Design for the Personal Workstations*, ATypI Congress Kiel, Germany, 1985
3. Brassell, L. H., Pathe P. D., Kohlen, E., *Method of font rendering employing grayscale processing of grid fitted fonts*, US-Patent number: 5684510, Microsoft, 1994
4. Browne, C. B. et al, *Automatic kerning of text*, US-Patent number: 6829748, Canon, 1999
5. Changyuan Hu, Roger D. Hersch, *Parameterizable Fonts Based on Shape Components*, IEEE Computer Graphics and Applications, v.21 n.3, p.70–85, May 2001
6. Coueignoux, P., *Generation of Roman Printed Fonts*, Ph. D. Thesis, MIT, Cambridge, Mass., June 1975
7. Freund, H., *Font of Matrices for Kerning or Producing Composite Type Characters*, US-Patent number: 2672794, Inter-type, 1954
8. Grüsser, O. J., *Gesichtssinn und Okulomotorik*, Kap. 11, p. 256–299, *Physiologie des Menschen*, R. F. Schmidt, G. Thews, 20. Aufl., Springer Verlag Berlin Heidelberg New York, 1980
9. Johnson, B., *Optical Scaling*, Master's Thesis for RIT, Rochester, New York, 1994
10. Karow, P., *Contouring*, brochure, URW Verlag, Hamburg, 1987 (avail. as PDF)
11. Karow, P., *Digital Typefaces*, Springer Verlag, Heidelberg, 1994
12. Karow, P., 'Extending Control of Digital Typography', *Visible Language*, Vol. 32.2, Providence, Rhode Island, 1998
13. Karow, P., *Font Technology*, Springer Verlag, Heidelberg, 1994
14. Karow, P., *hz-Program*, brochure, URW Verlag, Hamburg, 1992 (avail. as PDF)

15. Karow, P., *Ikarus*, German brochure, URW Verlag, Hamburg, 1975 (avail. as PDF)
16. Karow, P., *Kernus*, brochure, URW Verlag, Hamburg, 1993 (avail. as PDF)
17. Karow, P., *Nimbus*, brochure, URW Verlag, Hamburg, 1993 (avail. as PDF)
18. Karow, P., *Optical Scaling*, brochure, URW Verlag, Hamburg, 1991 (avail. as PDF)
19. Karow, P., *ScreenMaster*, brochure, URW Verlag, Hamburg, 1993 (avail. as PDF)
20. Karow, P., *Set theory for characters*, brochure, URW Verlag, Hamburg, 1987 (avail. as PDF)
21. Karow, P., 'The quality display of text on computer screens', *Type*, a Journal of ATypI, Volume 1, Number 1, Spring 1997
22. Karow, P., *Typeface Statistics*, URW Verlag, Hamburg, 1993 (avail. as PDF)
23. Kindersley, D. et al, *Index Printed Material*, US-Patent number: 3361048, Kindersley, 1965
24. Knuth, D. E., Plass, M. F., 'Breaking paragraphs into lines', *Software-Practice & Experience*, 11/11), 1119–1184, Nov. 1982
25. Knuth, D. E., *Mathematical Typography*, Stanford University, Cal., February 1978
26. Knuth, D. E., *Metafont: A System for Alphabet Design*, Stanford University, Cal., September 1979
27. Knuth, D. E., *The TeXbook*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1984, reprinted as Vol. A of *Computers & Typesetting*, 1986
28. Knuth, D. E., *TeX and Metafont*, American Math. Society and Digital Press, Bedford, Mass., 1979
29. Lau, H.-J., *VI&VA-Anzeigeneditor*, Lufthansa Systems SA, Hamburg, private communication, 2007
30. Linotype TypoTechnica 2005, conference in London, <http://www.linotype.com/2358-20699/speakerspresentations.html>
31. Matthes, W., *Geschichte der Zeichenmaschine*, Aristo Graphic Systeme GmbH & Co. KG, Hamburg, private communication, 1984

32. Naiman, A., *High-Quality Text for Raster Display*, Dep. of Comp.Science, University of Toronto, January 1985
33. Neville et al, *Method and apparatus for automatic, shape-based character spacing*, US-Patent number: 5803629, Neville, 1998
34. Rubow, G., *Interpolation*, URW Software & Type GmbH, Hamburg, private communication, 1973
35. Rubow, G., *MasterDesign*, URW Software & Type GmbH, Hamburg, private communication, 1994
36. Rosenfeld, P., *Stroke fonts*, URW++ Design & Development GmbH, Hamburg, private communication, 2007
37. Schneider, U. *An Object-Oriented Model for the Hierarchical Composition of Letter Forms in Computer-Aided Typeface Design*, Proc. Seventh Int'l Conf. Electronic Publishing (EP '98), pp. 109–125, 1998.
38. Stiehl, U., *TypeShop Collection*, cited on Page 1 of his Report, Heidelberg, 2006, <http://www.sanskritweb.net/forgers/brendel.pdf>
39. URW Software & Type GmbH, *Linus Modern Logo Processing*, URW spectrum, Hamburg, 2/1987 (avail. as PDF)
40. Warnock, J. E., 'The Display of Characters Using Gray Level Sample Arrays', *Computer Graphics*, Volume 14, Number 3, July 1980, pp. 302–307, Siggraph Proceedings, 1980
41. Wendt, D., 'Semantic Differentials of Typefaces as a Method of Congeniality Research', *The Journal of Typographic Research*, Vol. II, 1, 1968
42. Willrodt, J., *Element separation*, (URW)++ Design & Development GmbH, Hamburg, private communication, 2007
43. Gerald Murch, Tektronix: 'Visual Fatigue and Operator Performance with DVST', *Proc. of the Society for Inf. Display*, Vol. 24, No. 1, 1983
44. Zapf, H., *Alphabet Stories A Chronicle of Technical Developments*, Mergenthaler Edition Linotype GmbH, Bad Homberg, 2007

Peter Karow was born as a farmer's son in Stargard, Pomerania, on November 11, 1940. From school and after his civil service he progressed to the University of Hamburg, to study high energy physics leading to a dissertation in 1971.

With Rubow and Weber he founded the company URW Software & Type GmbH. Digital type became his life's work and has continued to be a most enjoyable and exciting task to the present day.

Several contacts with type designers, especially Hermann Zapf, had great impact on his development of the IKARUS program. Since 1988, both worked on the 'hz-program' in order to use the power of computers to improve the micro-typography of texts.

Peter Karow gave international presentations on numerous conferences, wrote many articles for journals, got 7 patents for DTP-related methods, and wrote several books, among them: *Digital Typefaces* and *Font Technology* (both Springer-Verlag, Berlin Heidelberg New York, 1994), *Typeface Statistics* (URW Verlag, Hamburg, 1993).



ACKNOWLEDGEMENTS

Special thanks go to David Lemon for the initiation and realization of Adobe's support for the Dr. Peter Karow Award, to Nicole Minoza and Caleb Belohlavek at Adobe for their organizational and editorial contributions, and to Charles Bigelow for his kind advice and the valuable additional information for this booklet's introduction.

Dr. Karow's text dates from June 2007 and it was originally written for a presentation at the 3rd International Conference on Typography and Visual Communication in the Greek city Thessaloniki, that same year.

Frank E. Blokland was responsible for the design and typography and the two applied typefaces are *Adobe Minion*, created by Robert Slimbach, and *DTL Prokyon*, created by Erhard Kaiser.

The frontispiece shows a screendump of DTL IkarusMaster, a recent sibling of the IKARUS system.

Ando BV in The Hague, (NL) printed it all on 120 and 240 gsm Bioiset FSC paper.