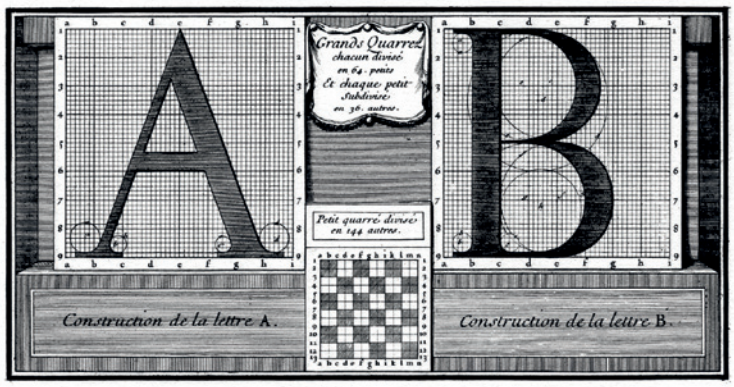
COMPUTED TYPE DESIGN



Christoph Knoth

Theory part of the MA Type Design thesis by Christoph Knoth

Thanks to

Magda, my parents, Tom, Konrad, Kathl, Luke, Louise, my ecal class mates, my teachers: François Rappo, Roland Früh, Ian Party, Frederik Berlaen, André Vladimir Heiz and all the people who answered my questions and helped me getting further with my research: Changyuhan Hu, Ingo Preuß, Tim Ahrens, Karl Leuthold and everyone I forgot.

This document is my master thesis at the ECAL / École cantonale d'art de Lausanne and is by no means complete or a hundred percent exact. Please send corrections and additions to mail@christoph-knoth.de

When this should ever get published I hopefully will have attempted to contact all copyright holders, but I am not sure this will be possible in all instances. So I already apologize for all the mistakes and promise to correct everything with the next print run.

Typeface: Computer Modern in all its variety.

Printed on a laserprinter on paper.

Christoph Knoth in March 2011, Lausanne, Switzerland.

Contents

Abstract

A lot of tasks in font design are interlinked and a change on one letter will maybe create hours of work on others. The idea of a parametrical typeface could minimize those problems and would allow to design an infinite number of typefaces at the same time.

I will try to understand why this way of designing a font never got widely adopted. If it is possible to create a more easy to use program to design western characters. And finally if this approach to type design would help to create new and interesting curves and shapes for letterforms.

Introduction

Type design is a long and tedious process. Just to design the basic letters takes days and it sometimes takes years for a full character set. The process has changed over time with technology evolving giving the designer more and more possibilities, at the same time making everything far more complex and complicated. As a result of this only a small group of people are able to design what is nowadays seen as a proper typeface. Besides the broad knowledge that is needed to design a font the learning curve for a novice can be quite long, even with the help of scripting which can do a lot of tasks that normally would have to be done by hand.

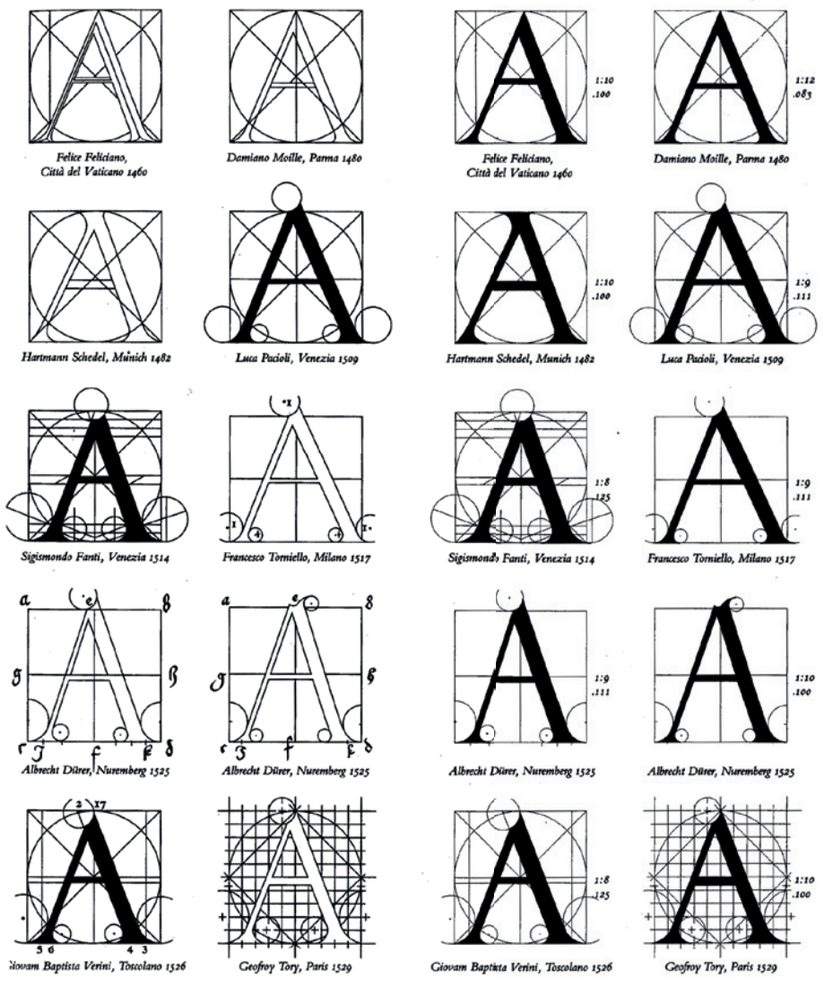
In the 70s Donald Knuth developed Metafont. A program that worked with the idea of parametrical fonts and would allow to design an infinite number of typefaces. But why this way of designing never adopted by more then a handful of people? Is it possible to create a far more easy to use program to design western characters by trying to analyze the strongness and weakness of other approaches? And does a programmatic approach to type design help to create new and interesting curves and shapes for letterforms something that would not have been imagined before?

History

To understand how type design works today one has to understand the history of type design. That is why I have collected some early historical samples that show first approaches for a mathematical notation and a systematical modification and variation of fonts in a pre computer era.

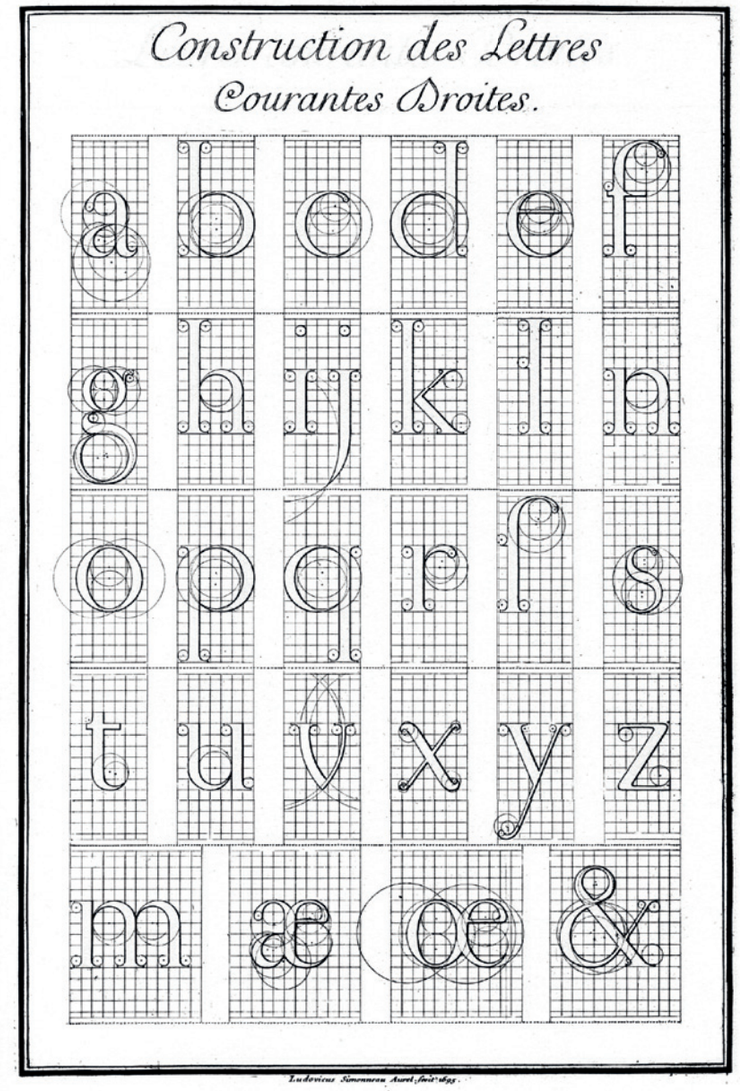
Followed by a short chapter about the curve and another chapter where I will try to shed some light on the changes that the computer brought to the type design industry.

Some Historical Samples



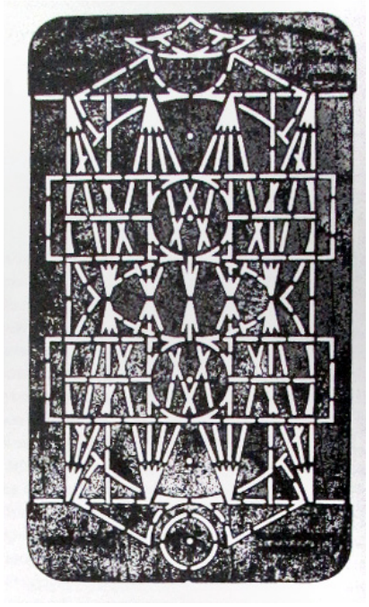
A collection of 12 constructed “A”s dating from 1460 to 1529.[1]

1 Ian Party. Le Romain du Roi – calligraphie ou construction géométrique?. 2006. KABK. Page 51



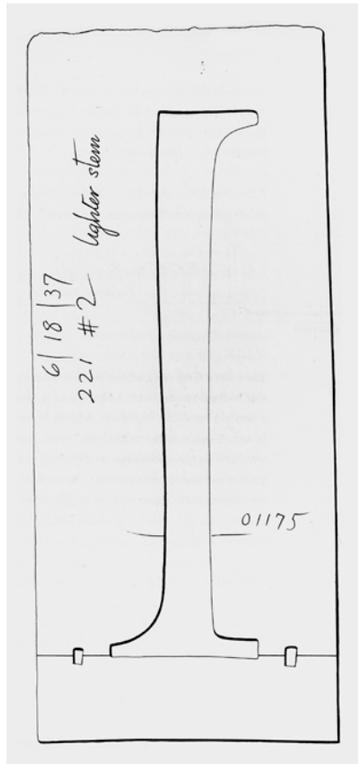
Le Roman de Roi, a font that was drawn for Louis the XIV, is considered as one of the most detailed and exact notations for the mathematical construction of a font.[1]

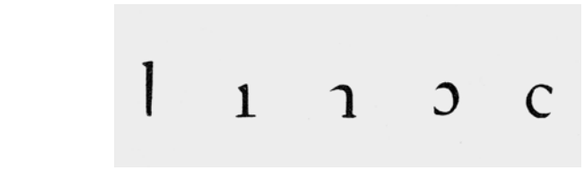
1 Even though research conducted by Ian Party has shown that there is quite a difference between this idea and the final typeface. But this has more to do with the limited mathematical possibilities at this time than the unability to create a “good” font.

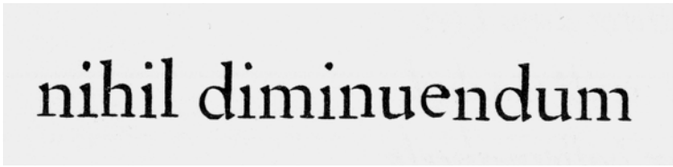


The “Plaque Découpée Universelle” a modular stencil device[1], from around 1879.

Oswald Coopers Experiment “15 Serifs” shows an early idea for a modular typeface.[1]







“Falcon stencils” and “Letters built up from the above elements”[1]

On the left the “cardboard templ[a]te for making pencil-outline pattern drawings”.

1 Kindel, Eric. The “Plaque Découpée Universelle”. Typography papers 7. 2007. Page 72. Print.

1 Bilak, Peter. History of History. <http://www.typotheque.com/articles/history_of_history>. Web.

1 Department of Printing and Graphic Arts in the Harvard College Library Cambridge

About Curves

"In mathematics, a curve (sometimes also called a curved line) is, generally speaking, an object similar to a line but which is not required to be straight." "This entails that a line is a special case of curve, namely a curve with null curvature."[1] And now we have gone in a full circle and are not much smarter then before, even though this is the best definition I found. And the search for it was very interesting but will probably not help type designers to produce better fonts.

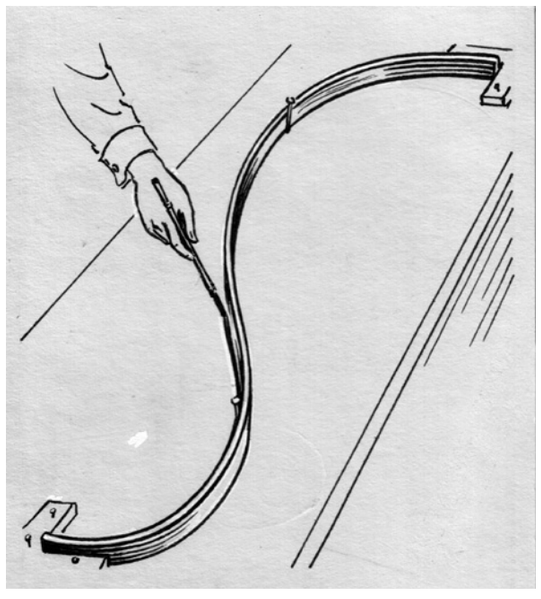
More interesting is that defining a curve as a geometric object in the time before computers was also a nontrivial task. It was needed to fixate a sketch for the planks of a boat or the wing of a plane. Where you needed to achieve a certain kind of accuracy if you wanted to do a second curve of the same kind. A common procedure to "write down" a curve was to define points that would be on the curve and then fixate them with nails and framing squares. Afterwards thin wooden strips (called "splines") would be placed in between the nails in a way that the inner force of the spline would create the desired curve.

Today we are using a series of Bézier curves (more on page 15) where the last point of one curve coincides with the starting point of the next curve and call them "a Bézier spline".[2] "In computer graphics splines are popular curves because of the simplicity of their construction, their ease and accuracy of evaluation, and their capacity to approximate complex shapes through curve fitting and interactive curve design."[3] Even though the concept of off-curve points are harder to understand then the old "nails on-curve concept". But perhaps Spiro curves (page 22) will be one day a usable alternative.

1 http://en.wikipedia.org/wiki/Curve

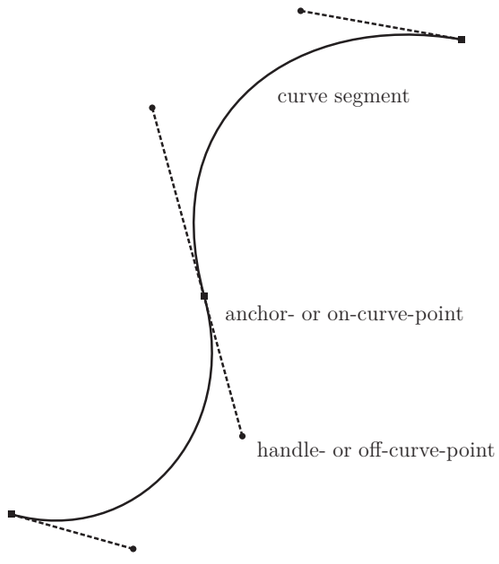
2 Compare with http://en.wikipedia.org/wiki/B%C3%A9zier_spline

3 http://en.wikipedia.org/wiki/Spline_curve



Pearson Scott Foresman

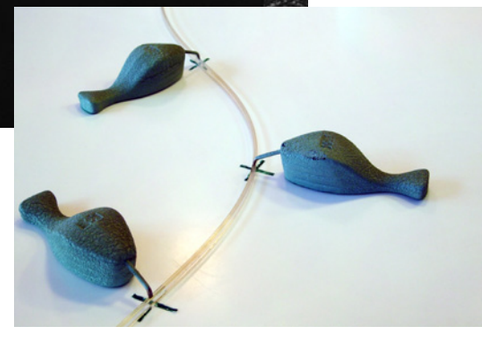
A wooden spline fixated with nails and framing squares.



A Bézier spline made up from two Cubic Bézier curves each consisting of two on-curve points and two off-curve points.

Boeing

A boeing draftman splining a curve and his tools on the right.



Carl de Boor

The History of Electronical Type Design

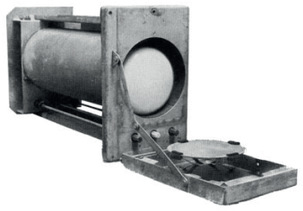
It is true, the development of type design has been heavily influenced by printing technology[1]. Until at one point type became more and more independent from material matter. The time when letters could be easily stored, manipulated and arranged inside a computer, changed not only the world of graphic design, but it more or less changed the way how everybody communicates, works and creates. And it did not only change the visual image of the world but also its all driving structure.

Because the output that shapes this new time is still dependent on the limitations of the technology, recapitulating the development from the very early computer graphics to the rise of global collaborative font design will reveal the strong and weak ideas of digital type design and may help to find new ideas to change it again. That is what the first chapter is for.

1 Peter Bilak in http://www.typotheque.com/articles/in_search_of_a_comprehensive_type_design_theory "The development of type has always been inextricably connected to the development of printing technology."

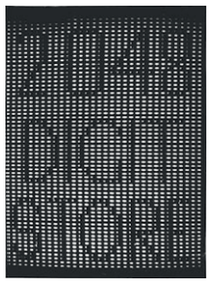
Graphical Pioneers

1946



Freddie Williams and Tom Kilburn developed a **cathode ray tube (CRT)** that could electronically store binary data. At the same time the tubes "monitor" functioned as a visual representation of the memory.[1]

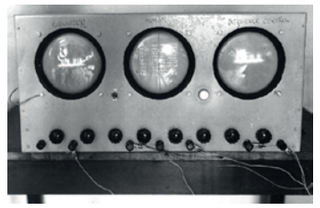
1947



The picture above is taken from Kilburns' report to Telecommunications Research Establishment (TRE) Malvern, of December 1947.[2] And can probably be seen as the **first electronical generated digital and rasterized type on a screen**.

1949

Whirlwind went operational in 1949. It was the first digital computer capable of displaying real time text and graphics on a video terminal, which was an oscilloscope screen.

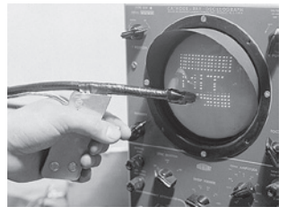


The **EDSAC memory display** was the first to use cathode ray tubes to display information. The center display shows the contents of the memory.[3]

1954

When the **IBM 740 CRT Recorder** became available it could be used with the IBM 701, IBM 704, and IBM 709 computers to draw vector graphics images on 35 mm photographic film. With the help of this device non-permanent computer images that were created digitally could be fixated.

1952



A handmade prototype of a **"light gun"** as part of the Whirlwind Project at MIT.[4] It was one of the first electronical input devices and was used to change the state of the memory.

The names of the objects shown on this radar screen are in line art and are therefore real vector letters.

1957

The picture above shows one of the first digitally scanned letters displayed on an oscilloscope. It was fed into a computer with a rotating drum scanner at the National Bureau of Standards.[5]

1 http://en.wikipedia.org/wiki/File:Williams-tube.jpg

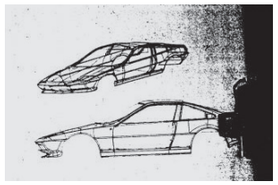
2 http://www.digital60.org/rebuild/50th/gallery/gallery1/index.html#bits2048

3 http://www.webbox.org/cgi/_timeline50s.html

4 Image from the courtesy of The MITRE Corporation Archives

5 http://www.webbox.org/cgi/1957%20First%20image-processed%20photo.html – article from paper: R. A. Kirsch, L. Cahn, L. C. Ray, and G. H. Urban, Experiments in processing pictorial information with a digital computer, Proceedings of the Eastern Joint Computer Conference, Dec. 9-13, 1957, Institute of Radio Engineers, New York (1958).

1959



Paul de Casteljau developed what was later known as **Bézier curves** using de Casteljau's algorithm. The ideas there are based on were widely publicized in 1962 by the French engineer Pierre Bézier, who used them to design automobile bodies.[6] [7]

They were a primer part for the later to come postscript language.

1960

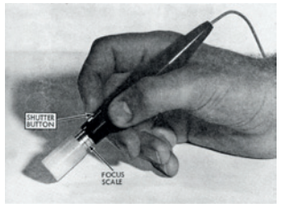
William Fetter (1928–2002) coined the term **Computer Graphics**.

1962-1963

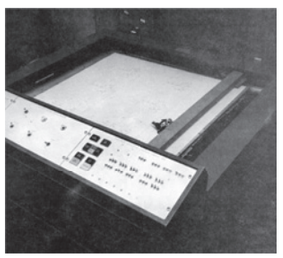
Sketchpad is a computer program written by Ivan Sutherland. It was the first program ever to utilize a complete graphical user interface.

The program used line art that was displayed on a CRT and not a pixelated screen as we know it today. The reason for this has nothing to do with its progressiveness, the reason has much more to do with the lack of memory for a buffer to store all the pixels that would be needed.

It had lots of great ideas inbuilt that are not even standard functions in today's programs. It was object oriented and had the possibility that objects could be scaled and reused, geometrical constructions could be made that would depend on each other, it also had magnetic lines and options to make lines parallel and perpendicular.



The further development of the light gun resulted in a **light pen** that was a used to draw and drag-and-drop shapes. Which turned out to be quite tiring because one had to have their arm all the time in a lifted position.[8]



Because all the graphics were made up of line art they could also be outputted by a plotter that could draw straight lines and circles.

When Ivan Edward Sutherland was asked: "How could you possibly have done the first interactive graphics program, the first non-procedural programming language, the first object oriented software system, all in one year?" He replied: "Well, I didn't know it was hard."

1964

The IBM 2250 Graphics Display Unit was announced [...]. Similar to the start screen of sketchpad that reads "INK". "Characters were built of line segments specified by display list subroutines. Thus any character set or font could be displayed, although fonts were generally extremely simplified for performance reasons. The computer altered the display by changing the display list. As the display list got longer, the refresh time got longer too and eventually the display would start to flicker."[9]

6 http://en.wikipedia.org/wiki/B%C3%A9zier_curve

7 Picture taken from http://www-gmm.insa-toulouse.fr/~rabut/bezier/DocumentsBezier/SectionHistoireUsinesRenault/IdeeBiz2.htm

8 Pictures from Ivan Edward Sutherland. Technical Report Number 574, Computer Laboratory – Sketchpad: A man-machine graphical communication system. September 2003. New preface

9 Wikipedia contributors, 'IBM 2250', Wikipedia, The Free Encyclopedia, 16 December 2009, 10:35 UTC, <http://en.wikipedia.org/w/index.php?title=IBM_2250&oldid=332007122> [accessed 11 May 2010]

The Commercialization and Freeing of Electronical Type

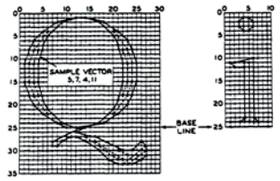
1965

Hell's 50T1 Digiset, the first digital typesetter, was made commercially available.

"The process of digitizing the letters happened in an early state of development in photo typesetting. The first fully digitized typesetting system has been the "Digiset", invented by Hell in Kiel, Germany. The system was presented in 1965. For the first time, a letter has been built up upon little spots. The available fonts were quite rough in resolution and were stored as bitmaps, not as Bézier curves or vectors. Every letter was activated point by point from out of the memory and put together to the whole black and white page containing pictures too.

A mnemotechnical code was used to build up the page, somewhat similar to HTML. For example, a command like "dz12" was used to set the leading between lines, "sg9" meant a point size of 9. With this method, complete newspaper pages were generated, processed in the mainframe computer and after that send to the output device."[10]

1967



In their paper **Three Fonts of Computer-drawn Letters** M. V. Mathews, Carol Lochbaum, and Judith A. Moss describe three groups of letters drawn with a vector display on a cathode ray tube. It is probably the first serious attempt to draw outlines of letters.[11]

Allen V. Hershey published the paper **Calligraphy for Computers** describing how he used computers and CRT printers to create a large repertory of digitized characters.[12]

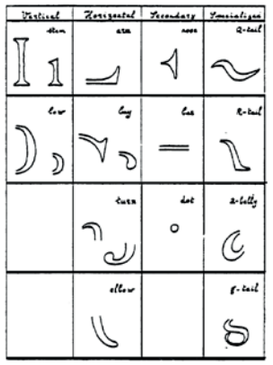
ITSELF (InTeractive Synthesizer of LetterForms), was designed by H. W. Mergler and Y. M. Vargo. A set of parameters could be altered to control the geometric design of the typeface.
It can probably be seen as the first digital parametrical letter design.[13]

1968

DigiGrotesk

DigiGrotesk was one of the first commercially available digital fonts and was designed in seven weights from light to bold by the Hell Design Studio.[14]

1975



CSD(Character Simulated Design) was the Ph. D. dissertation of Phillippe Coueignoux. He derived primitives such as stems, arms, and noses and defined the spatial relationships between them. From the evaluation of these primitives, he generated a grammar to describe the implicit structure of the characters in a font.[15]

1975

IKARUS from URW in Hamburg programmed by Peter Karow was introduced at the ATypI in Warsaw. At the beginning it was primarily used to digitize analog letters and not so much for actually drawing type itself. To do that a digitizer tablet and a puck was used to mark

10 http://www.global-type.org/Digiset.721.0.html

11 Letterform Design Systems by Lynn Ruggles, Page 5

12 Letterform Design Systems by Lynn Ruggles, Page 6

13 Letterforrn Design Systems by Lynn Ruggles, Page 7

14 http://www.designhistory.org/Digital_Revolution.html

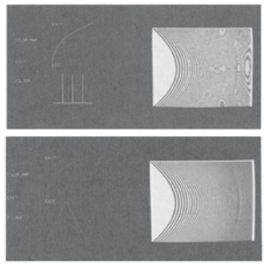
15 Letterforrn Design Systems by Lynn Ruggles, Page 7

a point on all the curves at around every 30°. Further adjustments to every letter could then be made on a screen. One advantage of Ikarus was that it could save lots of letters in a big database.

Ikarus later supported interpolation, and could create a rounded and shaded version of a typeface. It was used for the digitalization of letters in a bigger scale.[16]

At the end of the same year Linotype stopped to produce metal type setters.[17]

1977



Franklin C. Crow developed one of the first practical solutions for the anti-aliasing problem.[18]

16 Picture and information from the talk: Re-Inventing Technology, Peter Rosenfeld, DTL FontMaster Conference, The Hague, November 2009

17 http://www.print.ch/home/page.aspx?page_id=550&archive_type_id=92&person=&categories=&archive_id=2611&from=756

18 compare with http://en.wikipedia.org/wiki/Franklin_C._Crow

October 1977

A. A. Beers from the Academy of Science in the Soviet Union wrote a program that could read a bitmap character and generate an outline contour using straight and diagonal lines and four different kinds of curves.[19]

1977-1979

Mathematical Typography

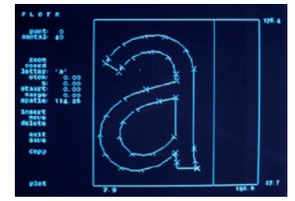
Donald Knuth developed his first version of his layout engine **TeX** together with the font design engine **Metafont** (reworked in 1984). It is a scripting based system for desktop publishing and font creation. It has a pen and an outlined model (page 26). Could the interpreter back then only output bitmap fonts it is now also possible to use **Metatype1** and produce PostScript fonts.

All those programs and most of the fonts created with them are available under a free software licence.

19 Letterforrn Design Systems by Lynn Ruggles, Page 6

mid 1980s

Fontastic (a bitmap font editor by Altsys)



Plotr by Petr van Blokland used the URW Ikarus algorithm. It was the starting point for the later program **Pika** that would run on MacOS.[20] It later got sold to URW and was the foundation for MacIKARUS (Ikarus M) and later FontMaster.

1981

Bitstream was founded by Mike Parker, the head of Mergenthaler's font design activity, that had left ITC taking with him three colleagues including the type designer Matthew Carter his former Linotype colleague.[21]

They are the first to call themselves a digital type foundry.

"Through Bitstream, Carter and Parker proposed to license digital typeface designs to typesetting equipment manufacturers, a revolutionary step in detaching the activity of designing digital fonts from that of building the machines upon which to set those fonts."[22]

20 http://www.petr.net/index2/-/p-332

21 compare with Emily King, New Faces – Type design in the first decade of device-independent digital typesetting (1987-1997), PhD Thesis (Chapter One: Technological and Industrial Change: Setting the Scene)

22 Emily King, New Faces – Type design in the first decade of device-independent digital typesetting (1987-1997), PhD Thesis (Chapter Three: The East Coast – Matthew Carter)

The Age of Device-Independent Design – GUI & WYSIWYG on your Desktop

1982

Adobe was founded in December 1982 and develops and publishes PostScript (PS) (a page description language) in the end of the 1983. A picture and line art description language that is still used today.[23]

It can be seen as the start of "the age of device-independent digital typesetting technology"[24].

"PostScript was the first software which allowed fonts to be designed and distributed independently of the manufacture of the systems on which they were to be printed. Pages of type described in the PostScript computer language could be printed on any output device equipped that understand the language."[25]

To store fonts Adobe used the Type 1 format. It was a simplification of the PostScript system and was not a complete language. That is why it could only store outlines and font information like names and spacing (kerning was stored in a PFM (Printer Font Metrics) file). Adobe would then sell licenses for the Type 1 fonts technology and also offer Type 3 fonts, as a lower-cost implementation of Type 1, which had no hinting support.[26]

This new storage format and the software to read it lead to many changes in the world of graphic design. It made it very easy to copy typefaces, there was an enormous increase in quantity of typefaces and an "anglicization of type culture"[27] could be observed.

1983

Telefont from Linotype allows the transfer of digital typefaces via the telephone line.

1984

"The first Macintosh was introduced on January 24, 1984; it was the first commercially successful personal computer to feature a mouse and a graphical user interface rather than a command-line interface."[28] [29]

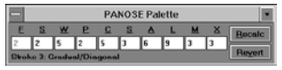
1985

Apple introduces **LaserWriter** a laser printer with built-in PostScript interpreter. It had four typefaces or respectively nine font masters already built in (Adobes adapted versions of Times (Italic, Medium, Bold, Bold-Italic) Helvetica (Medium, Fat) Courier (Medium, Fat) Symbol (Medium)).[30]

And finally the industrialization, which was criticized by the private press movement, made the private press possible.

"The Apple LaserWriter was a relatively low resolution printer, 300-dot-per-inch, but provided a lot more graphic and typesetting flexibility than similar machines that were on the market, for example the Hewlett Packard's LaserJet"[31]

In July 1985 PageMaker was introduced by Aldus Corporation. It was one of the first desktop publishing programs[32], initially for the then new Apple Macintosh and in 1987 for PCs running the then new Windows 1.0."[33] [34]



Also in 1985 Benjamin Bauermeister developed the "PANOSE System. "It is a method for classifying typefaces solely on their visual characteristics. It can be used to identify an unknown font from a sample image or to match a known font to its closest visual neighbor from a font pool. The word PANOSE is compound from letters taken from the six classes in which the creator of the system organized the Latin alphabet."[35] (read more about it in the text "characterizing typefaces" on page 32).

23 compare with Robin Kinross. Postscript – Übergangsmomente: Schrifttypen von 1968 bis 1997. Page 18

24 Emily King, New Faces – Type design in the first decade of device-independent digital typesetting (1987-1997), PhD Thesis

25 Emily King, New Faces – Type design in the first decade of device-independent digital typesetting (1987-1997), PhD Thesis (Chapter One: Technological and Industrial Change: Setting the Scene)

26 http://en.wikipedia.org/wiki/Type_1_and_Type_3_fonts#History

27 compare with Andreas Pawlik. Postscript – GhostScript. Page 25

28 picture from http://www.mac-history.net/wp-content/uploads/2008/10/apple_macintosh_1984_high_res.jpg

29 http://en.wikipedia.org/wiki/Macintosh

30 compare with Robin Kinross. Postscript – Übergangsmomente: Schrifttypen von 1968 bis 1997. Page 19

31 photo taken from http://museumvictoria.com.au/collections/itemimages/237/714/237714_large.jpg

32 Adams, Peter (2004-03-16). "PageMaker Past, Present, and Future". http://www.makingpages.org/pagemaker/history/. Retrieved 2007-06-27.

33 http://en.wikipedia.org/wiki/Adobe_PageMaker

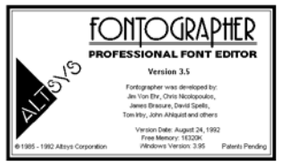
34 picture taken from http://www.prepressure.com/images/postscript_pagemaker.gif

35 http://en.wikipedia.org/wiki/PANOSE

late 1985

Adobe began to sell fonts independently of output devices. Marking the beginning of the access to device-independent typesetting technologies.[36]

1986



When Altsys released **Fontographer**[37] it was the first successful program in the age of the device-independent type design technologies and lead to a "severe erosion and lose of the coherent professional body that lay behind type design" and a "trend toward independent small-scale activity".[38] Because the "design and distribution of type had been liberated from the large-scale manufacture of typesetting systems" which lost more and more of their control and market.

"The type designer does not depend on technicians anymore" involving proofs and corrections of those proofs which increased the speed of type design.[39] More and more novices[40] get access to the type design world. Resulting in more custom typefaces for every kind of purpose.

At the same time we see a lot of "new" fonts that are just renamed copies with no or little changes in the design.

36 http://www.typotheque.com/articles/new_faces_%28chapter_one_technological_and_industrial_change_setting_the_scene%29

37 Fontographer in the later Version 3.5 (picture taken from http://www.guidebookgallery.org/splashes/fontographer)

38 New Faces (Chapter One: Technological and Industrial Change: Setting the Scene), thesis for the degree of Doctor of Philosophy 1999, Kingston University, http://www.typotheque.com/articles/new_faces_%28chapter_one_technological_and_industrial_change_setting_the_scene%29

39 Fred Smeijers, Type now Page 27

40 Emily King, New Faces – Type design in the first decade of device-independent digital typesetting (1987-1997), PhD Thesis

1988

IBM invents subpixel rendering. It is an improvement of the rasterization and therefore the display quality of type on the screen.

FontShop was founded by Erik Spiekermann. Later, when the typefoundry Berthold went out of business FontShop overtook their font library.[41]

late 1988 and early 1989

Rob Friedman, president of Bitstream, announced that his company had cracked Adobe's encryption. PostScript printers would now accept and process Bitstream fonts as if they were Type 1 fonts. This finally allowed anyone with the necessary tools to create Type 1 fonts and Adobe lost controls as the only manufacturers of output devices.

In October 1989, Adobe opened the PostScript page description language and ended the "**font war**".

1989-1990

1989 and 1991 are remembered as the "glory years", "when type revenue was going through the roof."[42]

41 http://fontfeed.com/archives/celebrating-20-years-of-fontshop-with-erik-spiekermann/ 'The FontFont logo designed by Neville Brody; left in the original FontFont corporate colour PMS 187, and right in the current FontShop yellow.'

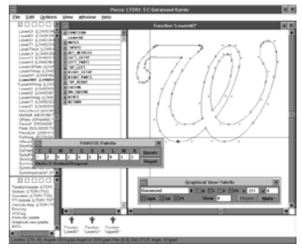
42 http://www.typotheque.com/articles/new_faces_%28chapter_two_the_west_coast%29#_ftn298

Type Outside the Box

1990

The first "random" typeface[43], called Beowolf was created by Just van Rossum and Erik van Blokland. The letter shapes changed during printing.[44] "Beowolf was also the first font in the digital font library FontFont."[45]

1991?



ElseWare developed **Infinifont** a parametrical font constructor, whose actual main goal was not to ease font design but to save disk space. To achieve this the program used an internal font structure that would create a font according to a given **Panose** code. After which each glyph had to be reworked with the writing of "cryptic" lines of code to further "bend" the shapes to match the desired font.[46] At the end of this process only those lines had to be saved resulting in a much smaller font file.

May 1991

"The cost of the licensing was considered very high at this time, and Adobe continued to stonewall on more attractive rates. It was this issue that led Apple to design their own system, **TrueType**, around 1991. [TrueType was released with the launch of Mac OS System 7 in May 1991 and later in March 1992 in Windows 3.1.] Immediately following the announcement of TrueType, Adobe published the specification for Type 1 font for-

43 http://www.wired.com/wired/archive/3.07/letterror.html

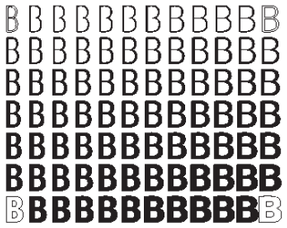
44 Shown here is the static version FF Beowolf OT R24

45 http://www.letterror.com/content/nypels/introduction.html

46 Interview conducted via email with Karl Leuthold

mat. Retail tools such as Altsys Fontographer (on January 1995 acquired by Macromedia, owned by FontLab since May 2005) added the ability to create Type 1 fonts. Since then, many free Type 1 fonts have been released; for instance, many of the fonts used with the TeX typesetting system are available in this format."[47]

1992



Apple introduced Advanced Typography (AAT) with **Multiple master fonts**. These are "Type 1 font programs that include two or more "master" fonts within a single font file. It allows users to interpolate many intermediate "instances" of the typeface. The fonts have one or more "axes" which might typically represent the weight, width, or optical size of the font."[48]
Seen as to unpractical it was widely adopted and used.

1993

FontLab 2.0 for Microsoft Windows was released.

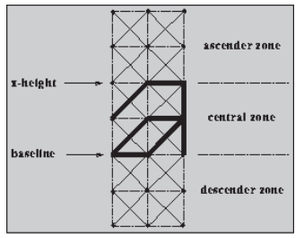
January 1995

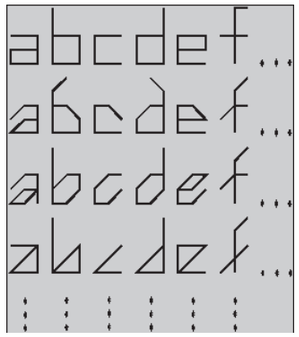
Altsys was acquired by Macromedia which included then a new version of Fontographer in the Macromedia Graphics Suite, which helped Fontographer's wider adoption.

Fontographer could now be used with **RoboFog** made by Erik van Blokland and Just van Rossum that brought, together with FogQ, Python scripting to Fontographer.

With this combination repetitive tasks could be automated. (When later Fontographer was discontinued and a lot of people switched to FontLab. For it RoboFog was rewritten and called **RoboFab**).[49]

September 1995





"The **Letter Spirit** project is an attempt to find out more about [...] the creative act of artistic letter-design. The aim is to model how the 26 lowercase letters of the roman alphabet can be rendered in many different but internally coherent styles."

"Starting with one or more seed letters representing the beginnings of a style, the program will attempt to create the rest of the alphabet in such a way that all 26 letters share that same style, or spirit."[50]

In 2002 the Swiss design studio Norm published a similar attempt with their program **Sign-generator 1.0.**[51]

1996

The specifications for **OpenType** got announced by Microsoft and Adobe.[52] This new font format is a subset of both the PostScript Type 1 and the TrueType format.

It is platform independent, has many typographical feature possibilities and a better unicode support.[53]

But it took longer than 2000 before the first OpenType typefaces were commercially available.

1997

Monotype Typography was taken over by AGFA forming AGFA Monotype.

1998

FontLab 3 for Mac was released.

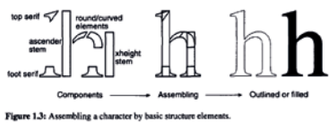
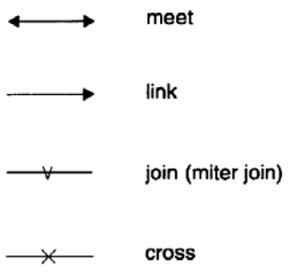


Figure 1.3: Assembling a character by basic structure elements.



The images above show some types of connections used in the internal structure of a font design tool developed by Changyuhan Hu for his Ph. D. thesis **Synthesis of Parametrisable Fonts by Shape Components** at the EPFL in Lausanne.

47 http://en.wikipedia.org/wiki/Type_1_and_Type_3_fonts#History

48 http://www.adobe.com/devnet/opentype/archives/mult_master.html

49 http://www.petr.net/index2/-/p-254 http://www.petr.net/index2/-/p-253

50 http://www.cogsci.indiana.edu/farg/mcgrawg/lspirit.html http://www.cogsci.indiana.edu/farg/mcgrawg/thesis.html

51 http://www.norm.to/pages/generator_3.html

52 http://news.cnet.com/Short-Specification-for-OpenType-available/2110-1001_3-226558.html

53 compare with http://de.wikipedia.org/wiki/OpenType#Vergleich_mit_TrueType_und_PostScript

It is interesting because he tried to deconstruct font characters into their essential parts and then used this imagined structure to reassemble a font just from it parts. A system that then worked for very different fonts.

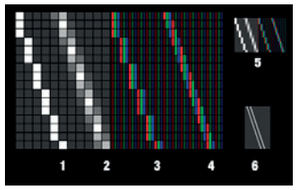
1999

Mac OS 9 integrated **TrueType** outline font support. Fonts could now be installed using drag-and-drop.

November 7th, 2000

PfaEdit (renamed to **FontForge** in 2004) was released by George Williams. It is an attempt to create an open source and free for all font editor.

2001



Microsoft introduces **ClearType** in Windows XP (though it was not activated by default until Windows Vista). It is an improvement of the former rendering of fonts especially on color LCD displays. This is quite important for fonts that are not hinted well.

23 August 2002

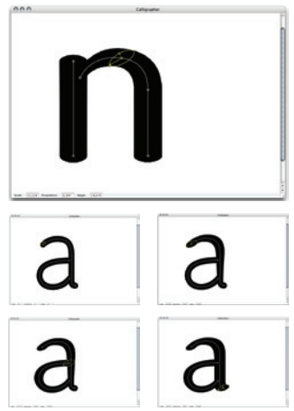
Mac OS X. Version 10.2 introduces **subpixel rendering** a technique similar to **ClearType**.

October 2004

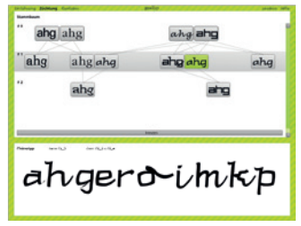
Release of the **Unified Font Object (UFO)** at the ATypI meeting in Prague by Erik van Blokland and Just van Rossum. It is an open, application independent, human read- and editable font format.[54]

These characteristics made it possible for lots of people to create their own tools (some among others are Superpolator, MetricsMachine, Rounding Ufo).

2004



Here we see a more calligraphic approach to font design by Jürg Lehni and François Rappo where a little program called Calligrapher will allow the application of different pens to the bone structure of a font.55



GenoTyp is an experiment into genetic typography by Michael Schmitz. "Different fonts can be mixed as desired and their genomes can be manipulated. New fonts are generated according to genetic rules."[56]

2005

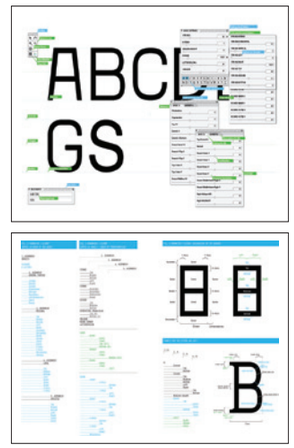
Gustavo Ferreira presents **Elementar** at the ATypI 2005. It is a parametric system of pixel fonts generated by Python scripts.

2006

FontLab releases **PhotoFonts**, a plug-in for Photoshop to edit and output bitmap fonts. Plug-ins for other DTP programs and browsers follow.

The screenshot above from Kalliculator shows an approach that is similar to Calligrapher but much more advanced. Kalliculator was developed by Frederik Berlaen in his master project at the Type & Media course in The Hague.

He is using the ideas and possibilities of the broad nib and the pointed pen following the imaginary line that the hand would take when drawing a letter. It is possible to change the skeleton of every letter and the angle and the thickness of the pen for every node.57



Type Generator designed and conceptualized by Remo Caminada and Ludovic Varone and programmed by Patrick Vuarnoz is a program

54 http://unifiedfontobject.org

55 Source: We Make Fonts, ECAL – Typography, 2006 (ISBN: 978-2-949271-76-4)

56 http://www.genotyp.com/

57 Source: http://typemytype.com

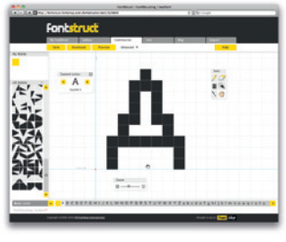
that is able to generate letter forms in real time. Because the characters are defined by mathematical formulas a lot of different parameters can be adjusted and then will change the whole alphabet or just one letter. Afterwards the typeface can be exported as a vector path and used in other programs.58

May 2007

Font Constructor is a tool made by Frederik Berlaen that makes it easy to “play” with elements in order to build a typeface.

Spiro was released by Raph Levien, a new and more intuitive way of drawing “smooth”[59] curves.

April 2008



FontStruct from FontShop is a free online font-building tool based on geometrical shapes, which can be arranged in a grid pattern.[60]

August 2008

Frank E. Blokland released the **DTL Letter Modeller.** It is a font construction program that is based on a calligraphic model that Edward Johnston used in the 1920s. The model has metaness features built in to make it possible to change certain variables and output a basic lower-case alphabet. Furthermore Frank E. Blokland has the idea that the model that he is using can also be taken to “measure” typefaces by comparing their characteristics to the most basic configuration of the DTL Letter Modeller. That is why he is working on a letter measurer, so that he can find out more about the rhythm (spacing, width) and the legibility of letters.[61]

2009

“**typism** is a web-based font editor. It is a public site where anyone can create a font for others to use and to study, to modify and to copy.”[62]

October 2009

Web Open Font Format (woff) for Firefox 3.6. Other browsers plan to introduce it.[63]

2010

As part of a larger collection of scripts for FontLab the **RMX Harmonizer** from Tim Ahrens gives the possibility to harmonize Bézier curves and so helps to speed up the design process.

Opening up again the debate about whether the computer is taking away too much of the responsibility from the type designer.May 2007

Spiro created by Raph Levien is a toolkit for curve design, especially for font design. Spiro is licensed under the GPL and shipping with FontForge and Inkscape.[64] Trying to establish a more intuitive on-curve-point vector design (on the left) in comparison to the off- and on-curve-point based Bézier curves (on the right).

August 2008

Frank E. Blokland released the **DTL Letter Modeller.** It is a font construction program that is based on a calligraphic model that Edward Johnston used in the 1920s. The model has metaness features built in to make it possible to change certain variables and output a basic lower-case alphabet. Furthermore Frank E. Blokland has the idea that the model that he is using can also be taken to “measure” typefaces by comparing their characteristics to the most basic configuration of the DTL Letter Modeller. That is why he is working on a letter measurer, so that he can find out more about the rhythm (spacing, width) and the legibility of letters.[65]

2009

“**typism** is a web-based font editor. It is a public site where anyone can create a font for others to use and to study, to modify and to copy.”[66]

October 2009

Web Open Font Format (woff) for Firefox 3.6. Other browsers plan to introduce it.[67]

2010

As part of a larger collection of scripts for FontLab the **RMX Harmonizer** from Tim Ahrens gives the possibility to harmonize Bézier curves and so helps to speed up the design process.
Opening up again the debate about whether the computer is taking away too much of the responsibility from the type designer.

58 Source: Type Generator, User Instructions, Alphabet + Programm, 2006 HGK Zürich, 1 Screenshot, Type Generator workspace, 2 User Utility Programm, S. 5, 6, Parameter overview

59 http://typophile.com/node/52821

60 http://fontstruct.fontshop.com/learn_more

61 http://typophile.com/node/48736

62 http://typism.appspot.com/fonts/index

63 http://en.wikipedia.org/wiki/Web_Open_Font_Format

64 http://www.levien.com/spiro/

65 http://typophile.com/node/48736

66 http://typism.appspot.com/fonts/index

67 http://en.wikipedia.org/wiki/Web_Open_Font_Format

Technical Approaches in Type Design

There are many ideas and concepts towards font design and how the design of a typeface should be tackled. That is why there are numerous ways on how programmers laid out software for the development of digital typefaces.

The two main tasks when designing type are the definition of shapes and the combination of them. Therein we can find different approaches that will be analyzed in this chapter. It will be pointed out which program uses which approach with a discussion of what are the assets and drawbacks.[1] Some programs, one being Metafont, even allow the combination of two or more of those approaches.

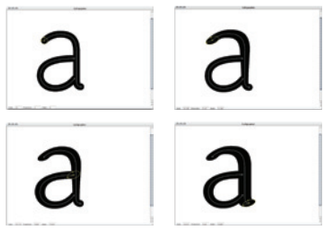
1 Starting points for this kind of comparison can be found in the "research project generating fontdesign" by Frederik Barlean, Jürg Lehni, Ian Party, François Rappo and Ludovic Varone.

Defining Shapes

The first signs left by man were the prints of their foot steps in sand and soft soil. The first signs consciously created by man were probably written with the index finger or a stick in the sand, then later carved in wax, wood and stone which would finally preserve the traces for centuries.

The foot would leave a complete shape with one print, making it the ancestor of today's often used **outline approach**. The finger however, would create a thick line that would follow its movement making it the forerunner of the **calligraphic approach**. Later shapes would also be drawn from the outside what took more time but would also give new stylistic possibilities. So from the very beginning both approaches would coexist as calligraphy and punchcutting coexisted.

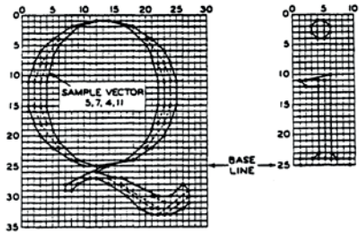
Also found objects like stones would be used to create shapes and are today known as the **mosaics or grid approach**.



Calligraphic or Skeleton Approach

How it works
Originated from linear-drawing, writing and calligraphy where different kinds of pens (pointed, broad-nib, ...) get applied to a skeleton resulting in different kind of characters depending on the pen.

Example Programs
» **Illustrator** has many tools to make fonts with a bone structure (and all kind of other vector deformation), but there is no font export
» **Metafont** uses formulas and parameters to change the bone structure
» **Calligrapher** from Jürg Lehni and François Rappo
» **Kalliculator** uses hand placed skeletons with a very sophisticated brush model



Outline Approach

How it works
Originated from the drawing of letters rather then the writing of them, different kinds of tools allow to change the outline of each glyph. Sometimes referred to as "sculpting"[2] this approach has not changed much since its introduction in Fontographer.

Example Programs
» **FontLab** and **Fontographer** (Bézier curves) require a lot of experience to make the curves harmonic.
» **Ikarus** needs a point each 30°, which results in lot of points, it is a good program to digitize a font.
» **Metafont** uses curves and formulas together with parameters, but is not very intuitive, because there is no graphical user interface. If one outline has metaness included then it can be used to generate numerous shapes, but they are at the same time very hard to control.
» **Spiro curves** (see also page 22) are a more intuitive approach than Bézier curves, they are harder to control, but have better joints.
» **ARAP** (As-rigid-as-possible) is used for a very intuitive shape deformation, but it is at the same time probably harder to achieve very exact results.[3]

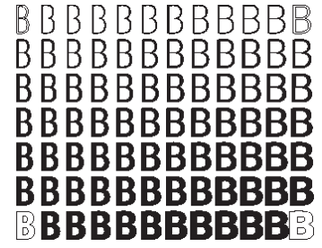
2 http://typophile.com/node/29008

3 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.3205&rep=rep1&type=pdf

Mosaic or Grid Approach

How it works
A grid defines where different kind of strokes or parts can go. Originally used to cover surfaces with tiles and called a mosaic many grids have been refined and adapted to better suit the construction principles of fonts.

Example Programs
» **Pixelfonts** were a necessity when displays changed from vector to bitmap. In a way it is still the standard today for screen displays. But today the pixelized letters are derived from vector versions. And only very few fonts get a special so called hand hinted version, where the letters get optimized for special pixel sizes.
» **Fontstruct** is an online tool which makes it very easy to share your font design.
» **DutchLettermodeller** is programmed to use relations based on a calligraphic model but the actual rendering is more related to a grid approach.



Inter- and Extrapolation

How it works
A software tries to analyze already existing glyphs and inter- and extrapolates between their maxima. It works best to interpolate in between different weights and widths of the same font but it can also be used to interpolate in between different fonts. Most of the time it produces an unforeseen and more experimental new font.

Today it is mostly used to interpolated in between outlines but one can also imagine it to be used to interpolate in between skeletons.

Example Programs
» **Multiple Master Fonts**
» **Extrapolating functions in Fontlab**
» **Superpolator**
» **GenoTyp**

Combining Shapes

The combination of shapes is an approach that can already be observed with the use of counter punches in the early metal type design. Its scope can be advanced by looking at Oswald Cooper's work from 1936 (page 8), "which involved applying 15 serifs [...] to stems of similar weight to test their influence in letter design"[4] or Matthew Carter's work for the Walker Art Center[5].

But the process of combining shapes is not only important in the design process but can also be used in the type setting program. With the use of different OpenType features it can provide alternate letters or ligatures. This is especially important for Arabic typefaces or lots of typefaces that try to simulate handwriting.

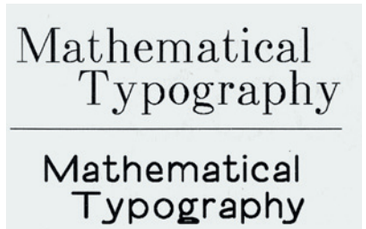
4 The history of History, Peter Bil'ak, http://www.typotheque.com/articles/the_history_of_history

5 Matthew Carter http://design.walkerart.org/detail.wac?id=2098&title=Articles

Placing (Linked) Shapes by Hand

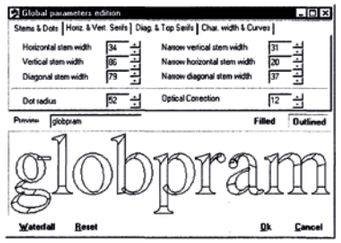
How it works
After drawing a shape you can drag it around, save it somewhere and reuse it for the next shape. It is a function that most people used to and that is probably the most "natural". For programming reasons it is usually not possible to nest shapes in an infinite number. Some things, like the placing of accents for diacritics, are often automated.

Example Programs
» **FontLab**
» **Font Constructor**
» **Fontstruct** (shapes can only be placed in a very rough grid)



Automated Shape Placing

Example Programs
» **Metafont** allows different kind of parametrical components to be used in a font. What is not only useful for accents but also comes in handy for Kanji characters.
» In 2008 at the ECAL in Lausanne, David Keshavjee and Julien Tavelli wrote a **script for FontLab** to generate different basic shapes with a freely chosen contrast. After which the script could assemble those components to form a font. Every position where those parts had to be placed was hand coded.



Self Organizing Shapes

» The method that Changyuhan Hu introduced with his PhD-thesis "**Synthesis of Parametrisable Fonts by Shape Components**" at the EPFL in Lausanne in 1998 offers a flexible font description. Characters are derived by an "assembly of structure elements (stems, bars, serifs, round parts, and arches)"[6].

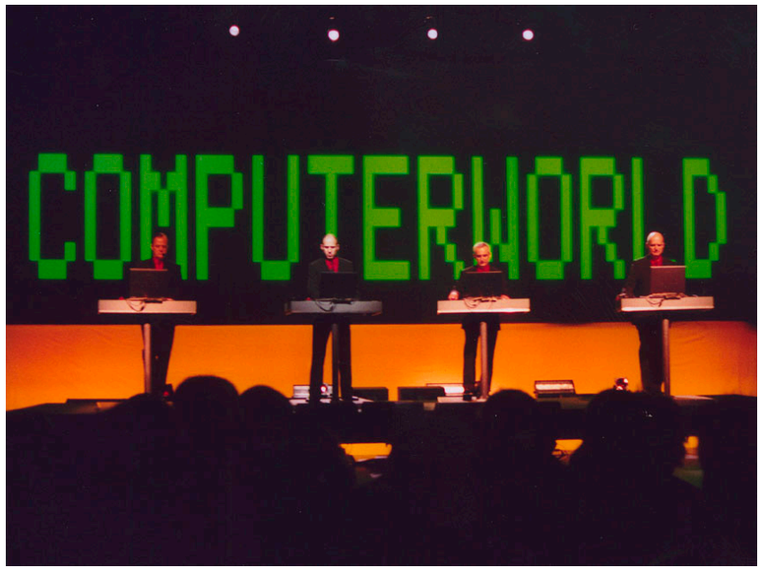
In my eyes the underlying model is a great idea because it captures the fundamental model of type design. And it definitely takes care that the font gets a high consistency. I am not sure though if the implementation is as flexible as the ground model as I have not seen the program in use. It is obviously quite good in recreating fonts, but it is not clear if it can create new fonts.

6 Changyuan Hu, Roger D. Hersch. Ecole Polytechnique Fédérale de Lausanne, Switzerland. Parameterizable Fonts Based on Shape Components. Page 1

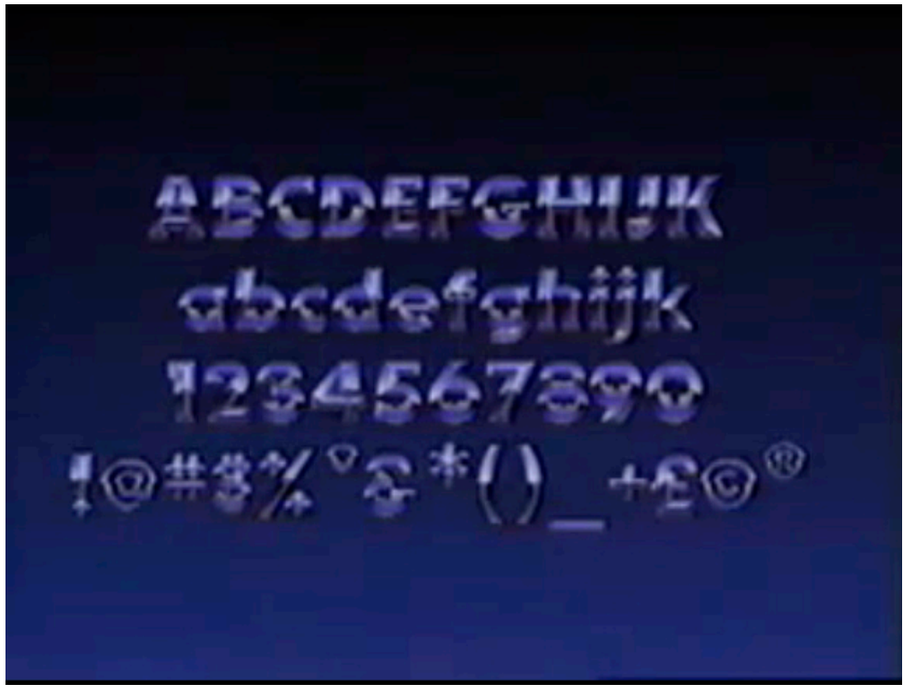
How the ~~tool~~ computer shapes the letter.

There is a certain idea about how computer generated fonts look like. This originated from the look and feel of technically created typefaces which are most of the time constructed or designed in a way that they are computer readable like OCR-A (1966) from American Type Founders and OCR-B (1968) from Adrian Futiger.[1] Inspired from those type faces other designers created fonts which are not computer readable but share the same style.

» Another characteristic is an obvious grid which is associated with the way a computer works. One can observe this in the "New Alphabet" (1967) by Wim Crouwel which is the most famous example of this style.
» Fonts that are created solely from lines or geometric shapes like pixel fonts.
» The first PostScript fonts could not have too many points. An outcome of this constraint is among others the font Charter (1993) from Matthew Carter.
» Fonts that try to emulate materials with a filterish like feeling.
» Repetitive geometric shapes like the fonts generated with the FontLab script from David Keshavjee and Julien Tavelli.



a pixel font

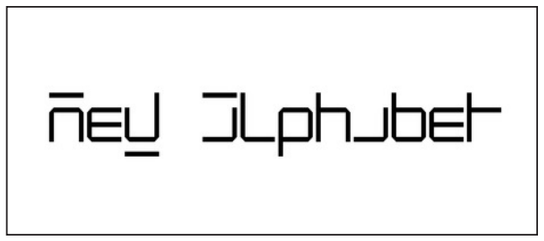


Amiga Video Toaster 2.0 Demo from Newtek

metallic type filter

1 http://myfonts.wordpress.com/2006/09/18/typographic-abbreviations-series-1-ocr/

OCR A



Wim Crouwels New Alphabet

Characterizing and Classifying fonts

There have been many attempts in the history of type design to find classification systems. An early and widely used one was developed by Maximilien Vox in 1952. It groups fonts depending on their characteristics typical for the centuries they appeared in the first time.

The Panose System developed by Benjamin Bauermeister that you can see on the right side solely relies on visual characteristics and leaves out the historical references. One example could be: “Family Kind: Latin text = 2, Serif Style: Cove = 2, Weight: Medium = 6, Proportion: Modern = 3), Contrast: Medium low = 5, Stroke variation: Transitional = 4, Arm style: Straight arms = 5, Letterform: Round = 2, Midline: Standard = 3, X-height: Large = 4”, which would give you the code 2263545234. This is very handy for a database but does not help you when you are talking with a human about type faces.

Being dissatisfied with the existing approaches Indra Kupferschmid proposed together with Max Bollwage and Hans Peter Willberg in 1998 another classification system that takes the structure and impression into account.[1]

Another approach is to describe fonts by grouping them with similar fonts because one very known font name can imply a lot of characteristics.[2]

Categories make sense for a general overview but proved to be useless in special cases because nowadays when type designers mix styles and characteristics belonging to different categories to elude their type designs from being put in one group human readable classifications are more or less obsolete or only suitable for older typefaces.

To get a better overview I tried to collect as many as possible tags and categories. Characterizations are taken from the websites of big font sellers like linotype, myfonts, a lot are from the type cooker[3] and some are taken from memory. They are regrouped shortend and extended when I thought this was necessary.

Construction
roman, italic, caps and small caps, capitals with roman, capitals with italic, continuous script, casual brush script, nothing

Width
compressed, condensed, narrow, normal, extended, wide, very wide, extremely wide

Height and Height-Relations
ascender to x-height to descender, ascender/descender is normal, longer than normal, shorter than normal, much shorter than normal, none at all

Contrast type
broad nib, pointed nib, transitional, speedball nib, brush, can’t be determined, the relation of the inner curve to the outer curve

Contrast amount
inverted contrast, slightly inverted contrast, no contrast at all (thick equals thin), no visible contrast, very low contrast, low contrast, some contrast, visible contrast, quite some contrast, a lot of contrast, high contrast, very high contrast, extreme contrast

Stems
straight, some ductus, not a single straight line, not a single straight angle, flared

Stroke endings
straight, no serif, rounded, no serif, wedge shaped serifs, slab shaped serifs, some serifs here and there, only serifs at the top end, only serifs at the bottom end

Stroke weight
hairline, very thin, thin, extra light, light, book, plain, medium, semi bold, bold, extra bold, black, heavy

Intended application
unknown, multi-purpose, newsprint, smooth offset printing, engraving, signage, packaging, subtitles on television, anti-aliased bitmaps, rubber stamping, way finding[4]

Intended size
very small, reading sizes, display sizes, very large sizes

Other things
many/few details (corresponds to small and display sizes), discontinuous, many/few peculiarities, randomness in type setting (through letter substitution), squarishness, thick/thin stroke endings

Special
only straight lines, octagonal construction, rough contours, casual, sketchy appearance, cut as stencil without drop-out counters, must contain at least 1 ligature, must contain at least 2 ligatures, initial and terminal swash variations

Qualities
readable, neutral, most efficient [5], suitable for small type[6], best to learn to read[7], fraud resistant[8], for low resolution printing (Bitstream Charter and Lucida typeface)

From a Certain Period
Humanist, Garald, Transitional, Modern (Didone, Mechanistic, Lineal)

1 Indra Kupferschmid. Buchstaben kommen selten allein (ISBN 3-7212-0501-4), Hans Peter Willberg. Wegweiser Schrift (ISBN 3-87439-569-3), Max Bollwage. Typografie kompakt (ISBN 3-540-22376-2)

2 http://typophile.com/node/9757

3 http://typecooker.com/parameters.html

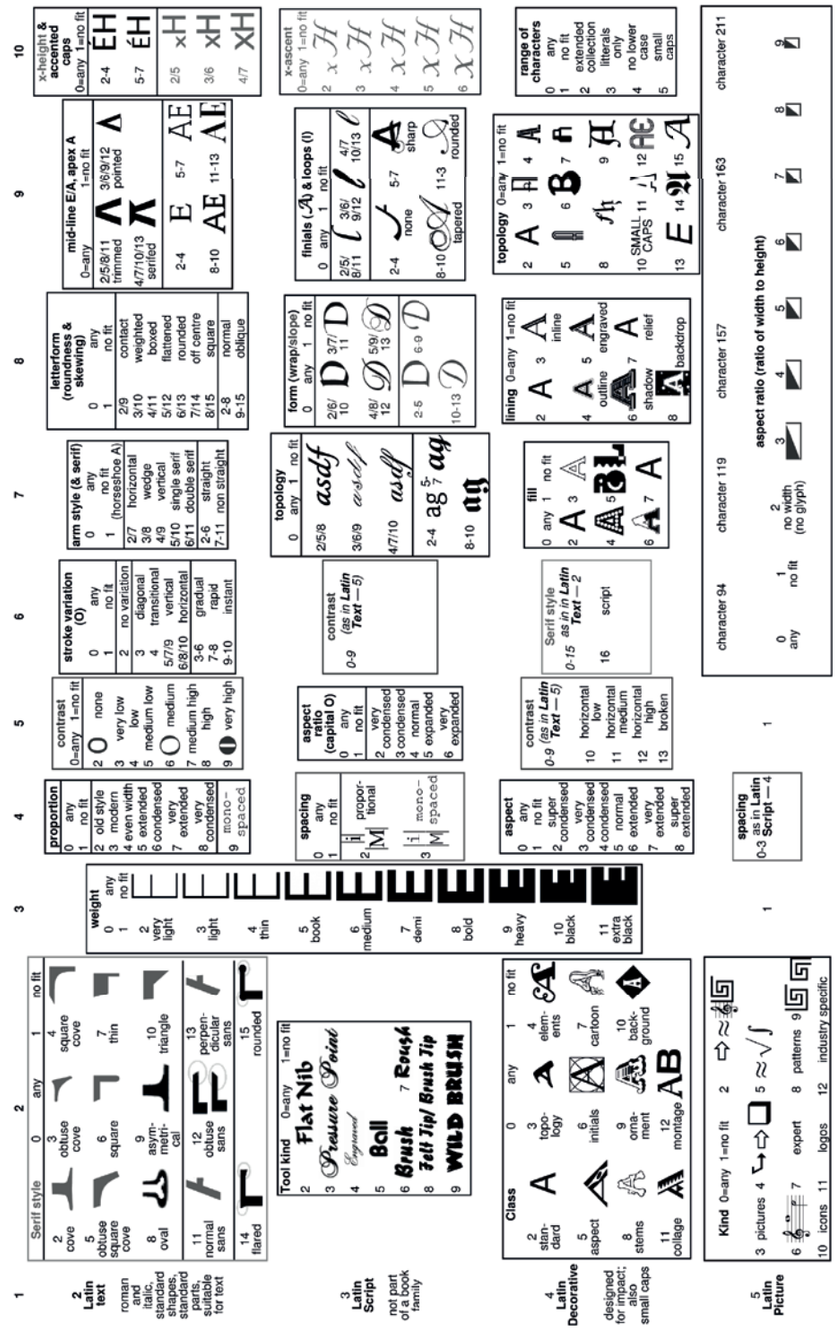
4 http://opentype.info/blog/2009/09/02/designing-the-ultimate-wayfinding-typeface/

5 http://www.gerardunger.com/fontstore/store-gulliver.html

6 http://www.fontshop.com/fontlist/applications/small_text/?utm_source=NewsletterApr272010&utm_medium=web&utm_content=FontsForSmallSizes&utm_term=em&utm_campaign=HelvCompanions

7 http://www.clubtype.co.uk/fonts/sas/sassoonfonts.html

8 http://en.wikipedia.org/wiki/FE-Schrift



An overview of the panose system created by Alan Bernard Hughes.[1]

1 http://forum.high-logic.com/viewtopic.php?t=941

Making a parametrical type design tool.

Why make your own tools?

"I try not to get too engaged with designing and defining the technology by itself. I see too many gifted designers getting too deeply into the abstract questions of formulations of technology, so they have no time left to actually use the technology in a creative way. I suppose there is a lot of creativity in making the tools, but I also want to reserve the time for the simple making of artifacts: books, posters, typefaces, exhibitions, etc."[1]

Peter Bilak may have a point with his opinion because tools need a lot of time to be conceptualized and to be build but if you are done with it you save a lot of time and can create something unique you were unable to do before. For example mass customization of an item with the help of a script that overtakes repetitive task.

Furthermore the limits of the programs you have installed on your hard disk often dictate what you can and will do with it. But it would be better to tell the machine what to do, not the other way around.[2]

Maybe the point of making your own programs is also that already the thinking and researching about a possible tool that you would need and trying to produce it can change how you think about your profession and the way you work. And it is much better then to wait for the update of your favorite program that will never come anyway.

Why is there no good parametric type design tool?

Donald Knuth says that: "A top-notch designer of typefaces needs to have an unusually good eye and a highly developed sensitivity to the nuances of shapes. A top-notch user of computer languages needs to have an unusual talent for abstract reasoning and a highly developed ability to express intuitive ideas in formal terms. Very few people have both of these unusual combinations of skills; hence the best products of Metafont will probably be collaborative efforts between two people who complement each other's abilities. Indeed, this situation isn't very different from the way types have been created for many generations, except that the role of "punch-cutter" is now being played by skilled computer specialists instead of by skilled metalworkers."[3]

In this quote a weakness of MetaFont already becomes obvious. You need to be skilled in two things and as a matter of fact many people are not. So being skilled only in programming will not get you started with designing a font so you better start getting good in drawing curves rather then learning to program.

Today the situation may have changed considering that the many "super families" with OpenType features are the product of programming help from someone other than the type designer and are often part of a collaboration. But there is still a lot of hand work included.

Learning from Font Generators

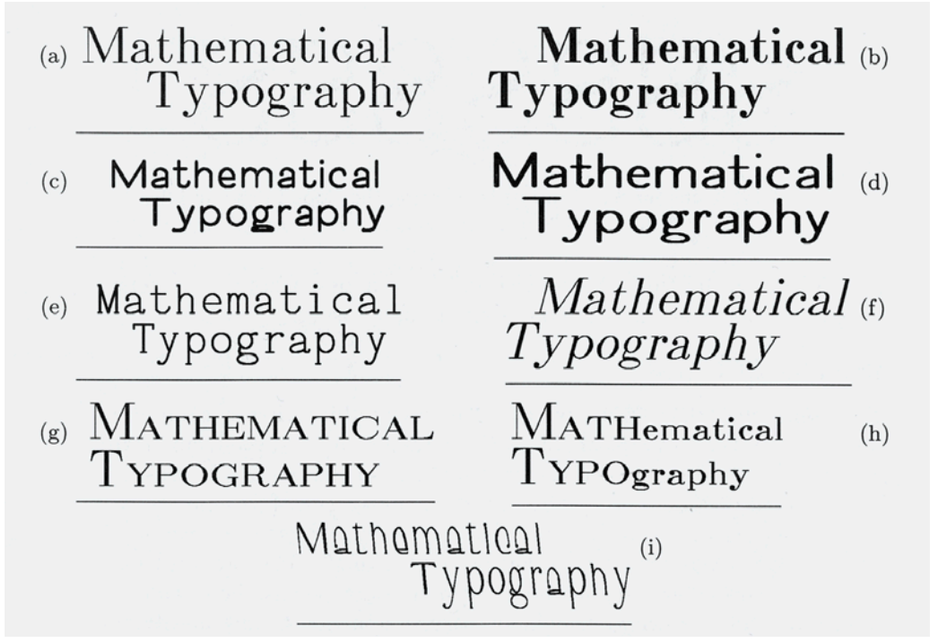
Only a few designers where ever interested in improving the technology of type design. They just used the available tools and more or less handcrafted with an enormous time consuming process lots of typefaces. Most of the programmers on the other hand just saw the possibilities of the technology but never thought about what a type designer really needs.

Metafont can probably be seen as a good example for this assumption. It was designed to give rasterized letters at the end. This was not a problem at the time Donald Knuth programmed TeX but is a big disadvantage now that every font is made out of Bézier curves. And it seems that it is almost impos-

1 Jürg Lehni interviews Peter Bilak , Typeface as Program

2 http://www.youtube.com/watch?v=TMJsELe2Rh0

3 Donald E. Knuth. "The METAFONTbook. 2000



Different styles of type obtained by varying the parameters of Metafont: (a) Computer modern roman; (b) Computer modern bold; (c) Computer modern sans-serif; (d) Computer modern sans-serif bold; (e) Computer modern typewriter; (f) Computer modern slanted roman; (g) Computer modern roman with small caps; (h) Computer modern roman with small caps and "small lowercase"; (i) Computer modern funny.[1]

1 Knuth, Donald Ervin. **Digital Typography.** Stanford, Calif.: CSLI Publications, 1999. Page 50. Print.

Computer Modern (BKM-cmr10)

abcdefghijklmnopqrstuvwxyz

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Century

abcdefghijklmnopqrstuvwxyz

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Monotype Modern

abcdefghijklmnopqrstuvwxyz

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Comparison between Donald Knuth's Computer Modern and the presumable archetypes.

sible to convert “the formulas” as of which the letters are made to those Bézier curves. The only practical process is tracing the rasterized letters.

Dave Crossland is stating the idea that type design is probably not very abstract and logical [...] its more emotional than logical.[4] Making the following statement from Donald Knuth more clear: “asking an artist to become enough of a mathematician to understand how to write a font with 60 parameters is too much”.

Adam Twardoch is one of many type designers that say Knuths font Computer Modern (see on the page before) is “simply hideous” and because he is so “brilliant” it proves that his “concept has massive flaws”.[5]

So I guess that most of the designers assume that it is much faster to sculpt letters than to program them.

On the following pages I started to collect examples of computer generated fonts to make future comparisons between presumable archetypes and “good” role models in type design.

The Difference between Spiro curves and Bézier curves

When I started my research and I discovered Spiro Curves (page 22) I first thought you would need less points if you define a font with the more intuitive spiro curves. But after trying it out for a while I am not so sure anymore.

If you edit with Bézier curves you only edit two off-curve-points between two endpoints defining the inbetween-curve. A change on a spiro points on the other hand affects multiple parts of the whole shape because it is harmonized. In this way it sometimes is quite uncontrollable. It kind of misses a function for maxima points. Something that works naturally with Bézier points but always gets out of control with Spiro curves.

How could it work?

It would be nice if it would be possible to treat typefaces more like colors[6]. So that it would be possible to choose some and then mix in between them.

A good parametrical type design tool should have in my eyes an internal font structure that incorporates the most basic features similar to the one that Changyuhan Hu came up with. It should be possible to add tags to every node or connection to allow special treatment trough the program. Tags could be for example x-axis, top-extrema, baseline, pen-turn, smooth-connection et cetera.

Categories derived from the chapter about categorization and classification (page 32) will be used to make first decisions about the appearance of the typeface. Further tags can be added to specific parts to change details.

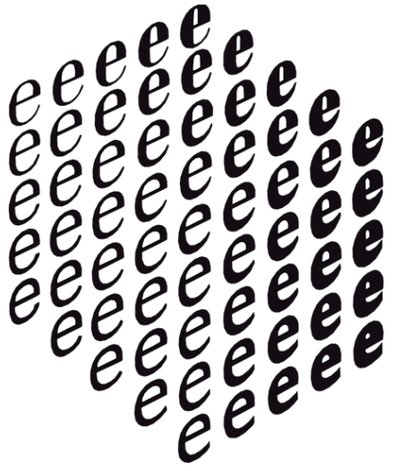
Out of this ground structure the software would then create a bone structure made up from Beziér curves for all the letters. The tags help to put the curves extreme points in the right positions. It will be then possible to choose the roundness of the curves and to add in between points that will help to influence the overall curve shape.

If the user wants to edit the letters I imagine a global and a local mode. If the user is in the global mode and changes something on one letter all other letters will change correspondingly. If he or she is in the local mode the change will only happen on the letter he or she is working on. Every value that is changed will always be stored in relation to the original model. I am also thinking about a relative mode where you can define point-to-point-relations or point-to-baseline.

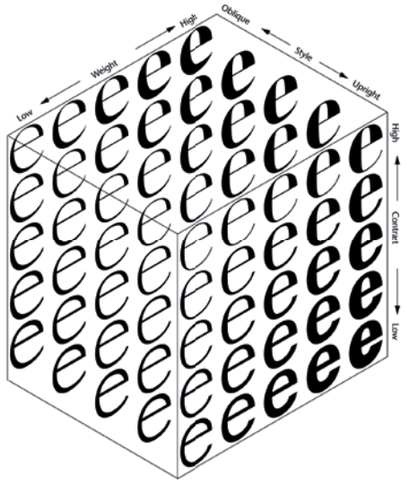
4 http://understandinglimited.com/2008/07/24/why-metafont-didnt-catch-on/

5 Adam Twardoch as twardoch on http://typophile.com/node/29008

6 http://typophile.com/node/9757



Original Noordzij Cube



regenerated by Changyuhan Hu

hamburgefont
hamburgefont
hamburgefont
hamburgefont

Apple Times from Linotype
Apple Times Bold from Linotype
Apple Helvetica from Linotype
Bauer Bodoni from Linotype

hamburgefont
hamburgefont
hamburgefont
hamburgefont

regenerated by Changyuhan Hu

The solution could also lie in the creation of a graphical user interface for Metafont but maybe it would be easier to just take some of the ideas and port them to a new program.

Another good ground structure could be provided by FontForge (page 21) because many things like drawing shapes and exporting to numerous font formats are already included. Plus it is free and open source software. On the other hand the graphical user interface is a little messy. But maybe this could be solved at the same time.

I am asking myself if also my software should be open source and free for all to. So other people could help to design and program it. And this would also create a bigger chance for it to get used.

It should be easily expandable and flexible enough able to accommodate change and to create new letter forms, but stiff enough to make letter forms recognizable.

As most of the parametrical type design programs we have seen just recreated older designs there is the question if there really can be something new or if the program is stuck in the in between of already preconceived design? This question can only be answered by making the program.

Possible problems that I will encounter and could imagine already: font looks to equal or boring and special characteristics have to be hand coded.

Too Many Fonts – Too Many Possibilities

If you have a great amount of typefaces there is a point where you can only find a lot of grunge fonts and sloppy digitalizations from the early 90s. It is like looking for a needle in a haystack. This problem also has to be considered if one makes a parametric type design tool that can output endless amounts of fonts.

So maybe one idea would be to create a brainstorm option where you can see the actual state of your font with up to 8 automatically created variations like in some programs from Adobe.[7] If you click on one of the variations the whole font will develop in that direction giving you again 8 new variations. So after time you can effortless refine your font until you reached your favorite result.

Or you just do it as Michael Bierut: “I left my first job. Suddenly I could use any typeface I wanted, and I went nuts. On one of my first projects, I used 37 different fonts on 16 pages.”[8]

How do I want to use a parametric font design program?

At the beginning as long as the program is not so much advanced I want to do some quick experiments and fast try outs. Using it as font inspiration machine.

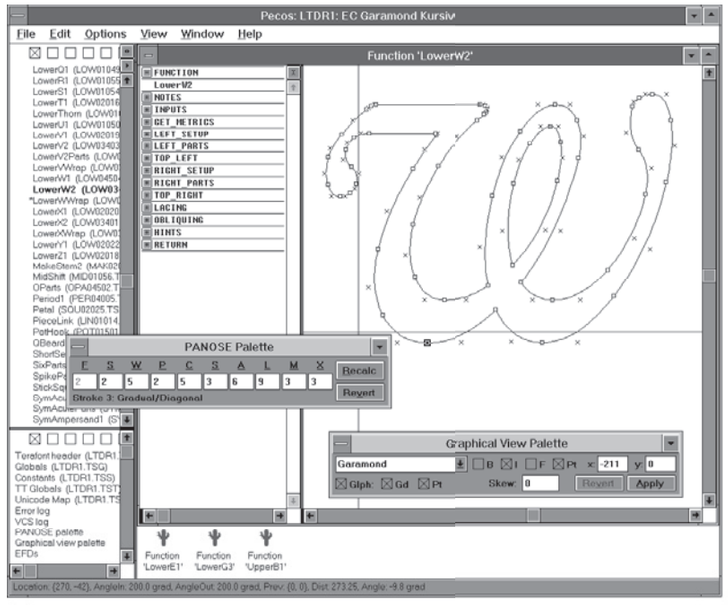
Later I imagine it as a tool to react on daily news. A kind of “Fast Fashion”-tool for type design. Maybe even offering a daily font for download.

Big news sites and smaller blogs feel more and more like things that are just made for consuming. A real interaction or surprise that is not just defined by the content but furthermore by the design does not currently exist. The layout only plays a role when someone visits a site for the first time.

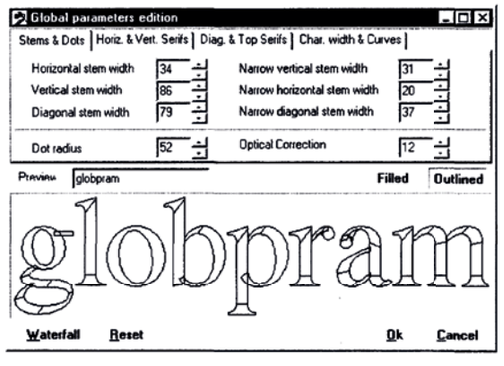
The more often you come back to this site the more you ignore the design and sometimes this makes you quite happy that there is nothing in between you and the content you like to consume. And at one point you realize it would be easier to let a machine collect the information for you and read everything in one place with the same formatting in an RSS-reader.

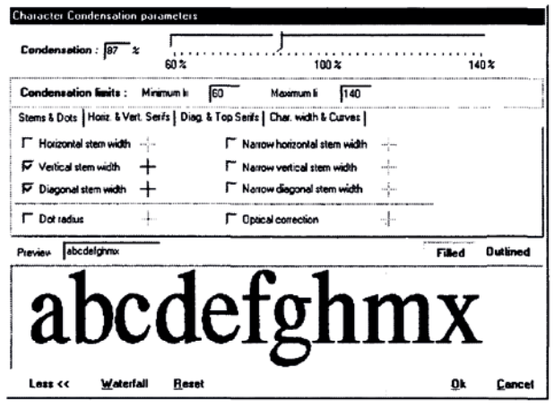
7 compare with http://help.adobe.com/en_US/aftereffects/cs/using/WS4961760E-8127-46fe-84BD-41304CA88459a.html

8 http://www.designobserver.com/observatory/entry.html?entry=5497



The Pecos development environment from infinifont. Looks very interesting but never got commercially available.





The “general global parameter editing window” and “The parameter editing window for condensation” from Changyuhan Hu’s PhD-thesis “Synthesis of Parametrisable Fonts by Shape Components”[1]. It was also never released for the public.

1 Changyuan Hu, Roger D. Hersch. Ecole Polytechnique Fédérale de Lausanne, Switzerland. Parameterizable Fonts Based on Shape Components. Page 1

To bring those people back on the page I want to do a news website that on a daily basis offers a new layout. I think content and design has to be inspiring and fresh everyday. What would be a good exercise for a font-design tool and a small layout-engine.

What makes a good font?

To make a program that generates fonts there has to be a collection of characteristics defining what a good font is.

Most legibility studies proved to be inconsistent and tested fonts against each other without taking the x-height, character and line-spacing into consideration.

And at the end they all gave different answers to what reads worse and what reads better. Some studies suggest that fonts that are people used to read better. Another one suggests that when a text is set in a font that is hard to read you will remember more of the content you have read.[9] So legibility is already a really ambiguous factor and might not be a serious one when you want to do a display type face anyway.

The French type designer Jean-François Porchez states: “The only criterion I rely on is simple: a good typeface fits the need of the subject.” “This rather ambiguous answer points to the problem: how can a type designer design a typeface when he is not in control of the subject? Does it mean that we need to have an endless library of typefaces to fit an endless number of subjects?”[10] A good question. But maybe we need a feature database rather then a type face database. So if you need for example a type face for small print, you need a list of features that help readability in small sizes.

Ilene Strizver claims that “[c]onsistent design characteristics”[11] are important. But sometimes visual inconsistency can give some interesting flavour to a type face?

Some other people claim that a type face has to have good curves.[12] So you have to find the “right” relation from the inner to the outer curve. You have to “repair” optical illusions that are caused by geometrical type design. But what is a “harmonious” curve? Is “G2-continuity” enough? No kinks? “flow, harmony and tension”. What defines a “steady rhythm”?

At the end it all comes down to a feeling. Is it a good name? Can I sell it to the client? Can I afford it? Can I download a test version? Is everybody else already using it?

Do I like it? It seems, it is all personal and you definately need your personal font.

9 Connor Diemand-Yauman, Daniel M. Oppenheimer, Erikka B. Vaughan. Fortune favors the Bold (and the Italicized): Effects of disfluency on educational outcomes. Cognition. Volume 118. Issue 1. January 2011. Pages 111-115. ISSN 0010-0277.

10 http://www.typotheque.com/articles/in_search_of_a_comprehensive_type_design_theory

11 Ilene Strizver. What Makes a Good Typeface, Part 1 of 2. http://www.itcfonts.com/Ulc/4012/GoodTextFace.htm

12 Different Authors. Sitting and looking at curves... http://typophile.com/node/69500

Outlines and possible kind of lower case “A”s generated by Frederik Berlaen’s Kalliculator.[1]

1 Berlaen, Frederik. Kalliculator (Masterthesis). Page 15 and 58. <http://www.kalliculator.com/about/researchpaper.pdf>.

More Questions to Be Answered ... Later

What is the Future of Type Design?

Nowadays we see bigger font families with more details, more weights, more family members, more glyphs, more opentype features. And in the future we are probably going to see more online collaboration, more free fonts?
And maybe more private fonts for friends? Fonts especially designed for web applications? More characters for more languages? More private “foundries”? A kind of css notation to generate typefaces in the browser? A unique typeface for every project? No fear of getting copied?

What typeface will become a classic in the future?

I recently stumbled across an article written by Fred Smeijers for the April—May 2006 (no. 70) edition of the magazine Tipográfica. He is asking the question: “Is it possible to determine what typeface of the 1990s will become a classic in the future?” And to illustrate this question he is making up a story in which the font Meta gets forgotten in 2026 and comes back as a revival in 2036. As a forgotten evergreen becomes a classic in that way.

I am thinking that in 2036 there will be still the classic typefaces but there will not be any new ones. As the total metaness and individualization of fonts has taken over. Every mainstream consumer that is just opening word will be presented the font he likes the most. As a construction of fonts his friends like, the latest clothes he bought and the car that he is driving. Welcome to the future.

Appendix

Really Short Glossary

Interpolation
In type design the approximation between two corresponding points on two corresponding letters.

Metaness
Features of a type face that are not fixed but depend on one or many parameters.

Internet
Place where one can find things that I forgot to put in the glossary.

Bibliography

Bergerhausen, Johannes, Deborah Anderson, and Ingo Preuß. **SIGNA – Beiträge Zur Signographie.** Ed. Ingo Preuß and Andreas Stötzner. 1st ed. Vol. 6. Grimma: Denkmalschmiede Höfgen, 2004. Print.

Cheng, Karen. **Anatomie der Buchstaben. Basiswissen Für Schriftgestalter (engl. Designing Type).** Mainz: Schmidt, 2006. Print.

Dwiggins, W. A., and Bruce Rogers. **WAD to RR: a Letter about Designing Type.** Cambridge, Mass.: Harvard College Library, Dept. of Print. and Graphic Arts, 1940. Print.

Frutiger, Adrian. **Buch der Schriften – Anleitungen Für Schriftenentwerfer.** Wiesbaden: Marixverlag, 2005. Print.

Geider, Dr. Thomas, Tilo Richter and Andreas Stötzner. **SIGNA – Beiträge Zur Signographie.** Ed. Dr. Uwe Andrich and Andreas Stötzner. 2nd ed. Vol. 1. Grimma: Denkmalschmiede Höfgen, 2005. Print.

Jammes, André. **La Naissance D'un Caractère: Le Grandjean.** [Paris?]: Editions Promodis, 1985. Print.

Kinross, Robin. **Modern Typography: an Essay in Critical History.** 2nd ed. London: Hyphen, 2004. Print.

Knuth, Donald Ervin. **Art of Computer Programming Volume 1: Fundamanetal Algorithms.** London: Addison-Wesley Company, 1972. Print.

Knuth, Donald Ervin. **Digital Typography.** Stanford, Calif.: CSLI Publications, 1999. Print.

Kopka, Helmut. **LATEX-Erweiterungsmöglichkeiten: Mit Einer Einführung in METAFONT.** Bonn: Addison-Wesley, 1990. Print.

Korger, Hildegard. **Schrift und Schreiben: Ein Fachbuch für alle, die mit dem Schreiben und Zeichnen von Schriften und ihrer Anwendung zu tun haben.** 4th ed. Leipzig: VEB-Fachbuchverlag, 1981. Print.

Mosley, James. **Le Romain Du Roi La Typographie Au Service De L'Etat, 1702-2002.** Lyon: Musée De L'imprimerie, 2002. Print.

Party, Ian. **Le Romain Du Roi – Calligraphie Ou Construction Géométrique**. Lausanne: Ian Party, 2010. Unpublished.

Schenk, Walter. **Die Schrift im Malerhandwerk eine Anleitung für Maler, Schrift- und Plakatmaler, Gebrauchswerber und andere schriftgestaltende Berufe.** 4th ed. Berlin: Verl. Für Bauwesen, 1962. Print.

Smeijers, Fred, and Robin Kinross. **Counterpunch: Making Type in the Sixteenth Century, Designing Typefaces Now.** London: Hyphen, 1996. Print.

Tschichold, Jan. **Meisterbuch der Schrift. Ein Lehrbuch mit vorbildlichen Schriften aus Vergangenheit und Gegenwart für Schriftenmaler, Graphiker, Bildhauer, Graveure, Lithographen, Verlagshersteller, Buchdrucker, Architekten und Kunstschulen.** 2nd ed. Ravensburg: Otto Maier Verlag, 1965. Print.

Wiescher, Gert. **Schriftdesign.** München: Systhema Verlag GmbH, 1991. Print.

Webography

Blokland, Frank E. **"Automating Type Design Processes."** Typophile. 24 Aug. 2008. Web. 18 Apr. 2010. <http://typophile.com/node/48736>.

Fabian, Nicholas. **"Type Generation Systems."** Nicholas Fabian, 2000. Web. 18 Apr. 2010. <http://web.archive.org/web/20010104101800/webcom.net/~nfhome/fontgen.htm>.

Various authors. **Various Posts.** <http://typophile.com/>

Various authors. **Various Essays.** <http://www.typotheque.com/site/articles.php?category=Essays>

Various authors. **Various Articles.** <http://www.designobserver.com/observatory/entry.html?entry=5497>

"Electronics will soon force its claims upon letterforms, and let us hope it will liberate us from the dust of the past."

Hermann Zapf, 1968[1]

1 Letterform Design Systems by Lynn Ruggles, Page 1

"Woe, if the machine wins out and the characters are shaped after its judgment! Who will then need to wonder if the emergent letter is cold and soulless?"

Hermann Zapf, 1970[1]

1 Tim Ahrens – Size-specific adjustments to type designs. Page 24