Implementing Programming Methods in the Process of Designing Typefaces:

A Designers Point of View

Elsa Jónsdóttir

Implementing Programming Methods in the Process of Designing Typefaces:

A Designers Point of View

Elsa Jónsdóttir

Final project for a degree in Bachelor of Arts in Graphic Design

Advisor: Steinar Ingi Farestveit

Graphic Design

Department of Design and Architecture
December 2013

Abstract

This thesis explores why designers use programming methods in typeface design, both for speeding up the process of the design itself and for adding an extra dimension to the typeface, more conceptually and for clearly professional purposes. During the process of writing the thesis the author interviewed six graphic designers from five different countries to get a wide range of opinions on various matters related to the purpose of parametrized type design and technologies in typeface design. The answers were used as a driving force for the research and then referenced throughout the thesis. The conclusion was that programming and various plugins within font editors are currently being widely used by professional type designers and usage of them is a good way to customize ones workflow when designing type, especially when the font family gets larger. In that context parametrized font editors are discussed resulting in conclusion that those kinds of applications, as they exist today, will not become standard — though with progressions and enhancement they might. What is unfortunate about the parametrized applications the author found is that they are not flexible enough, mainly because they all start out with a base font designed by the font developer. Therefore the outcome from such applications might become more influenced by the tool than desired. Conceptual experiments with typography, often with interdisciplinary emphasis, tend to raise questions and conclusions not directly related to typeface design, but in the authors opinion involving typeface design in conceptual experiments is an interesting way to approach subjects and additionally brings important conclusions to various matters which in can affect design and other disciplines in numerous ways.

Table of Contents

Introduction

In this thesis I will primarily explore the nature of programmed typefaces and font softwares or programming languages designers can use for enhancing the possibilities of their type designs. I am not a programmer so the thesis is aiming at viewing the subject from a designers point of view. There have been many experiments on the subject, for example where parametrized type forms are generated to speed up the type design process in projects such as Metaflop by Simon Egli, Prototyp-o byYannick Mathey and Louis-Rémi Babé and Computed Type by Cristoph Knoth, but programming has also been used for more conceptual experiments in single fonts such as FF Beowolf by Erik van Blokland and Just van Rossum. Over the years the process of designing typefaces has undergone fundamental changes in relation to technological advances and the fact is that over the span of a type designers career he will have to adopt to new and increasingly advanced methods numerous times.[1] Using mathematical equations in type design, as are used in programming based typefaces, is nothing new, in the past when typefaces were transferred to negatives for production with a digital laser photocomposer the letter shapes used were carefully calculated with a computer before hand.[2]

Interesting questions can be raised when contemplating about type design: What can programming typefaces additionally bring to the process of the design? What can it bring that conventional type design has difficulties to do? What deeper message can the typeface contain than only displaying characters and communicating lingual information? Adding extra layers to an already known form deepens the medium and experiments with such things might get us artistic but also challenging results. Those kinds of experiments could allow the discipline to grow in different, even eccentric, directions. It seems that this technical aid might play an important role in typeface design in the future and might even become a standard tool in coming years. I interviewed designers from five different countries during the process of writing the thesis, to get answers to relevant questions from designers and font application developers and the answers can be seen the extra material that comes with the thesis.

Designers approach type design and concepts in numerous different ways so in that sense, there are also numerous ways programming languages and softwares for type design

[1] Lehni, Jürg, *Typeface as Program,* ECAL: University of Art and Design, Switzerland, 2010, page 83.

[2] Hu, Changyuan, “Synthesis of Parametrisable Fonts by Shape Components”, master thesis at Nanjing University, China, 1998, page 7.

are made.[3] In this thesis I will emphasize two main aspects of programming based type design. On one hand, the more frequent kind of type design where designers use scripting and other programming methods for speeding up the design process and reducing manual labor[4] and on the other hand, programming based type design where the form is challenged and artistic or conceptual experiments are being produced. Before writing this thesis I had limited knowledge about the programming methods used in typeface design. I saw the thesis as a chance to explore this interesting section of type design and during the process I found out that there are numerous different techniques in the field and even more opinions on the matter, for all of them to be listed out there would be need for a much bigger platform. In the thesis I will list and discuss a few of the reasons designers have used programming in their typeface design and mention some projects who have been developed in the field and what concepts they aim to communicate. The research question for this thesis is:

What are the possibilities of utilizing programming languages in type design, both for better workflows and more conceptual projects, how far along are we in parametrized type design and will that field become standard in the coming years?

[3] Knoth, Cristoph, "Computed Type Design", bachelor thesis at ECAL: University of Art and Design, Switzerland, 2011, page 25.

[4] Interview 1 (31.10.2013). Written interview between the author and Niklas Ekholm.

1. Interviews

During the process of this thesis the author interviewed six designers on certain matters relating to programming related typeface design. The reason was to see if designers from different countries had different opinions on the subject and to execute a personal research on matters that could add interesting aspects to the thesis, also have the chance to ask further questions related to the answers given. The interviewees were Sigurður Ármannsson from Iceland, a designer and font developer which has interest and experience with working on various design-related technological matters, Niklas Ekholm from Finland, a graphic designer and type designer, Rafe Copeland from New Zealand, a graphic designer which has worked with programming, Cristoph Knoth from Germany, a graphic designer who built the Computed Type application and teaches various courses on graphic design and web technology and Yannick Mathey and Louis-Rémi Babé from France who are working on the Prototyp-o application which works with parametrized type design. Not all of them have utilized programming in their design, but they all have some knowledge and opinion on the matter. All of the interviews took place via email from the end of November to mid-October, 2013. All of the interviewees were asked the same questions besides from Niklas Ekholm, he was first interviewed and helped the author shape the following interviews. These are the first questions sent to Niklas Ekholm:

1. **What are your thoughts about having a programming language solely design a typeface?**
2. **In your opinion what is the purpose of using programming in type design?**
3. **Where can it lead type design?**
4. **How important is legibility vs. artistic expression?**
5. **How important is legibility vs. additional information within the type? That is: making the medium somehow deeper or serve different purposes.** *(For example the font experiments by van Blokland and van Rossum were they pushed various limits of digital type design. These extra type experiments, though never published, pushed type design thinking further in exploring new usage of the medium. The experiments contain fonts such as a virus font, that if installed on a device, it started to make your installed Helvetica font illegible and a font that when used in a text document, automatically adds typos. These experiments, though they could be thought of as unprofessional, or even criminal, to actually use them, are important in the process of investigating how deep type design can be, how it can mediate something else than languages. And how much more can live behind a programming based typeface.)*

After receiving his answers to those questions the author both discovered better phrasing for some subjects at hand and after further research decided to change some of the questions. The other interviewees received the following questions:

6. **Have you used programming in your type design?**
7. **Are you familiar in the developments in using programming as an aid in type design?** *(For example parametrized type design in programs such as Metapolator and Metaflop by Simon Egli, Prototyp-o by Byte Foundry and Computed Type by Cristoph Knoth. Also in single fonts, for example FF Beowolf by van Blokland and van Rossum)*
8. **Where can parametrized programming methods it lead type design?**
9. **Will it become a standard method in coming years?**
10. **Do you know of any more programs or single fonts that touch the field of parametrized type design** *(Or have any involvement in writing programming languages in the process of the design)***?**
11. **"The visual form of type design is inspired by the technology of each era". What are your thoughts on this statement?**
12. **What are your thoughts on using a typeface as a software? Having additional information within the type that serves some other purpose than communicating lingual information?** *(For example the font experiments by van Blokland and van Rossum were they pushed various limits of digital type design. These extra type experiments, though never published, pushed type design thinking further in exploring new usage of the medium. The experiments contain fonts such as a virus font, that if installed on a device, it started to make your installed Helvetica font illegible and a font that when used in a text document, automatically adds typos. These experiments, though they could be thought of as unprofessional, or even criminal, to actually use them, are important in the process of investigating how deep type design can be, how it can mediate something else than languages. And how much more can live behind a programming based typeface.[5])*

Type developers related to font applications were additionally asked:

13. **What were your initial thoughts (or concept) when designing this typeface/program?**
14. **What kind of programming language did you use?**
15. **Can you describe in few steps the process of setting up your typeface/program?**
16. **In your opinion what is the purpose of parametrized type design?**

[5] Blokland, Erik van and Just van Rossum, "Is Best Really Better?", in *Emigre 18,* Emigre Inc., The United States, 1991, page 27.

They all had different answers and thoughts to the questions and the results will be quoted and used throughout the thesis. Full interviews can be seen in the extra material included with the thesis.

2. Literature Review

2.1 Implementing Scripts

For some part the tools and technology used shape the outcome of projects,[6] so it seems almost equally important as mastering and exploiting the newest tools at hand, to challenge, add to and change the tools at hand to suit new ideas, new discoveries or frankly, to suit various kinds different ideas and concepts. It is not necessary to know programming to take use of different type design implementations for some of them have been programmed and can be manipulated through an already built interface alongside a font editor.

2.1.1 Using Programming to Automate Actions

When using tools such as programming there is more time for perfecting the font itself and the concept it is supposed to communicate while a lot of the usual repetitive manual labor tasks can be performed with automatic scripted commands. Normally those small repetitive tasks can take away a large sum of the time the designer has to spend on each project. In todays fast paced world, every discipline needs to keep up with technological advances, especially in the field of graphic design, as technology is a huge part of our work process and environment. In general, every discipline has to be open to new comings, and there, technology often plays a big role. On top of the constant addition of complexity on top of complexity in the world of technology, the type designers ongoing passion for letter forms is the key to the continuous quest to redefine the shapes of the alphabet and the processes that precede the outcome itself[7], and therefore new technologies are constantly being implemented in to the field of typographic design. One way to utilize scripts in the typeface design process is to familiarize oneself with the Python programming language. Numerous designers today use Python scripts to speed up the process of font designing though not all of them write scripts or program their fonts directly. The Python programming language was developed by Guido van Rossum[8] and has been used within font editors to speed up the process of adding standard things such as anchor points for the placement of accents for diacritics and for speeding up the sometimes painful process of kerning letters. All of those methods are extremely helpful in designing but do not exactly challenge the external look or concept of a typeface. One aspect of programming related

[6] Lehni, Jürg, *Typeface as Program,* ECAL: University of Art and Design, Switzerland, 2010, page 121.

[7] McMillan, Neil, *An A–Z of Type Designers,* Laurence King Publishers, United Kingdom, 2006, page 7.

[8] Middendorp, Jan, *Dutch Type,* 010 Uitgeverij, The Netherlands, 2004, page 187.

type design is the use of the UFO format. The UFO format is made by Erik van Blokland, Tal Lemming and Just van Rossum in 2004.[9] UFO stands for Unified Font Object and is a format made for outline fonts where the format stores font data. It was initially made because the formats in the font editors Fontographer and Fontlab were not as flexible as these designers wanted. The format is a good way to manipulate fonts by using Python scripting. The UFO format is human readable, making it easier for designers to integrate it into new programs than other programming language formats used with typeface design applications.[10] The same authors who built the format also built a font editing program where the format is meant to be used, called RoboFont. The program is made with their ideology as guidance; that the user should help build and shape his own tools, and the programs and plugins those authors have created are all very easily manipulated if the user is familiar with the Python language. The good thing about RoboFont is the fact that the user can tailor the application to his advantages and needs, the bad thing about it is that a good understanding of Python is needed.

Projects have also been produced where programming is applied to typefaces to communicate more than lingual information. The font FF Chartwell is one of those examples. The designer uses the font to its full extent by writing simple strings of various appropriate numbers which are then automatically generated into charts[11] (Image 1). It is a quick way to speed up the process of chart-making and prevents the designer from spending too much time on perfecting the measurements of charts and diagrams.

2.1.2 Programming knowledge not needed

It is not mandatory to be familiar with programming to utilize speed-up tools during the process of typeface design. Various plugins have been used throughout the years which the user can implement to a font editor. The Font Remix Tools are useful plugins which are meant for usage within FontLab (Image 2) but have already been released as a beta version for Glyphs. Many designers find the tools indispensable and a quite essential part of a more efficient type design process.[12] Tools such as this one are a good start for a type designer that is not a programmer himself but has interest in using some of the technology related to a more advanced way of designing typefaces. Such tools can enhance the process in

[9] *Unified Font Object*, LettError and Type Supply, 2012, accessed 14.11.2013, http://unifiedfontobject.org/team.html.

[10] Knoth, Cristoph, "Computed Type Design", bachelor thesis at ECAL: University of Art and Design, Switzerland, 2011, page 25.

[11] *How to Use FF Chartwell*, FontShop International, year missing, accessed 30.11.2013, fontfont.com/how-to-use-ff-chartwell#intro.

[12] *Font Remix Tools 1.6,* Just Another Foundry, 2012, accessed 10.11.2013, remix-tools.com.

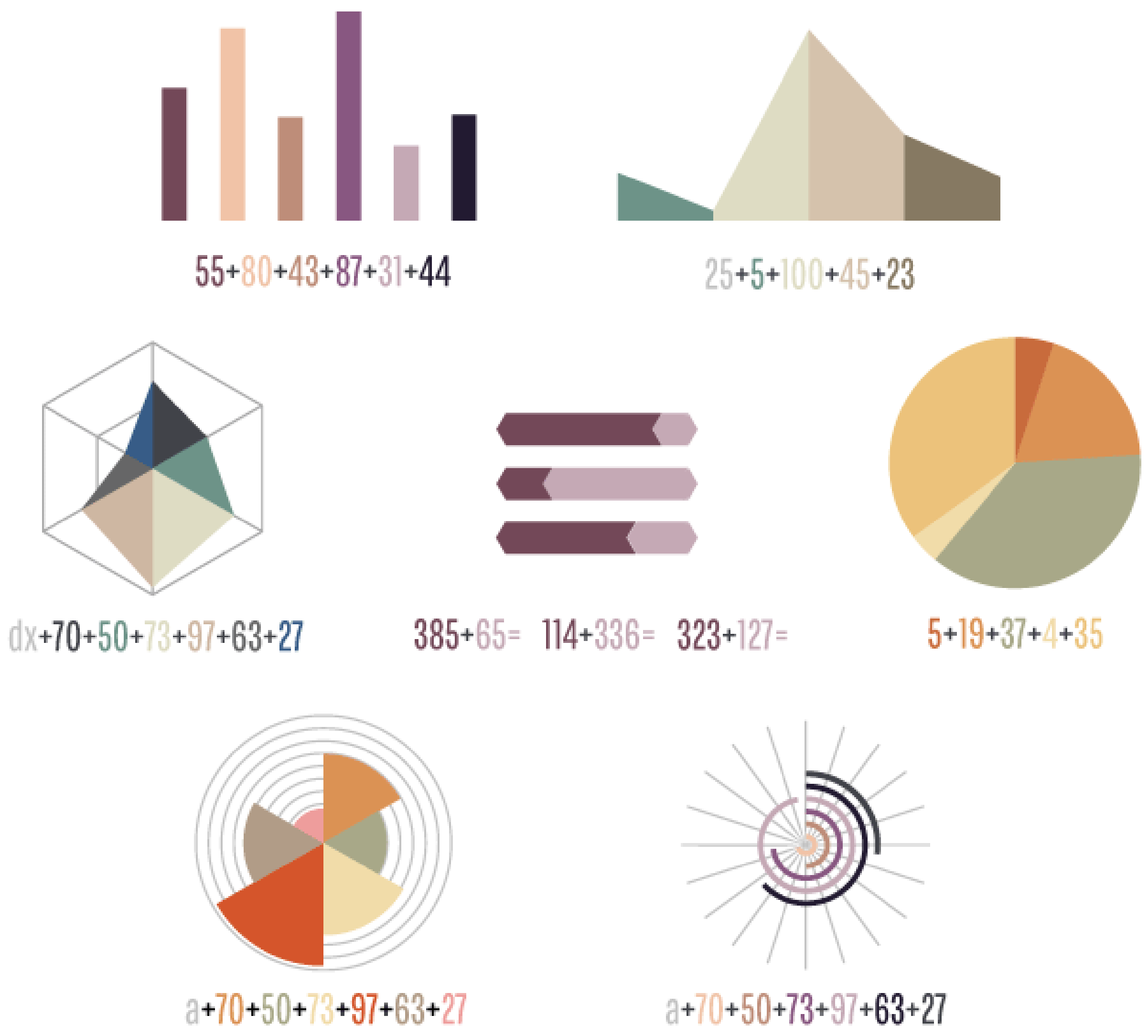
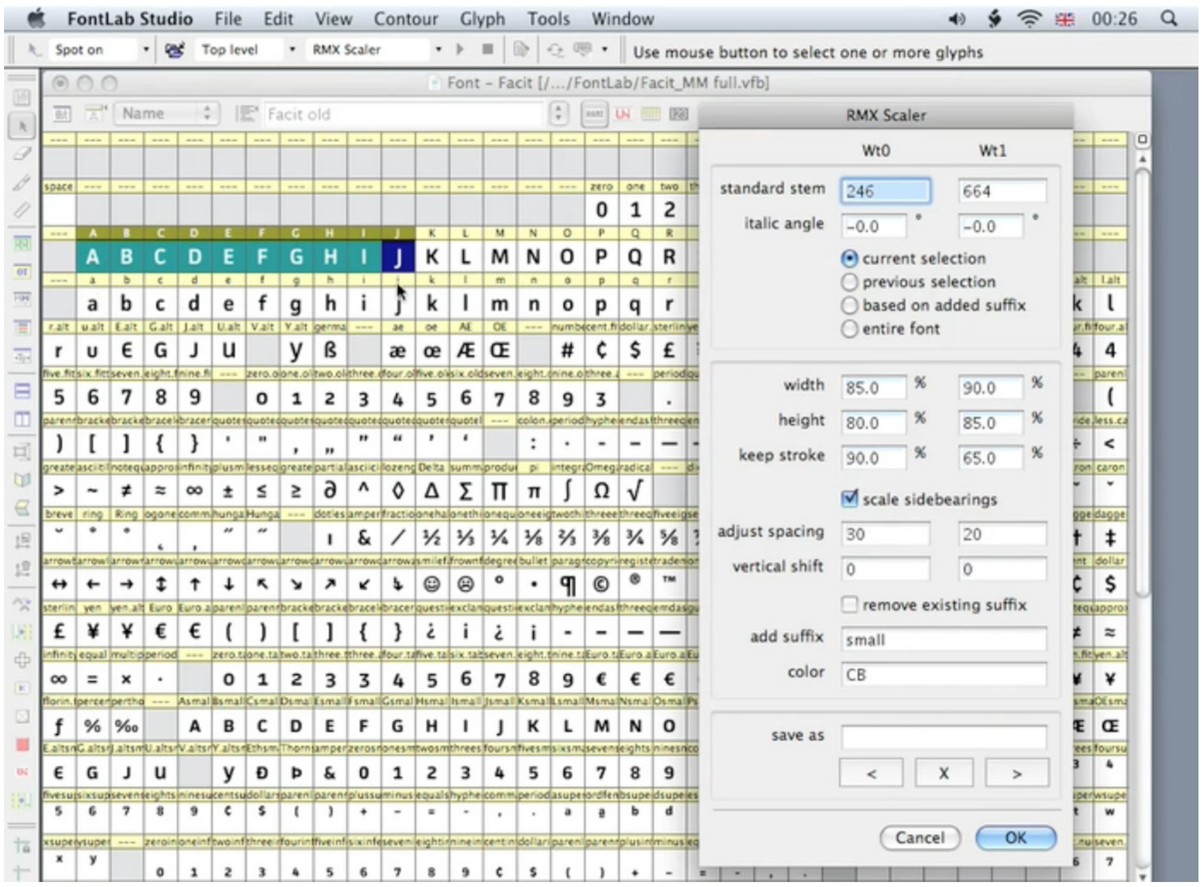


Image 1


FontLab Studio File Edit View Contour Glyph Tools Window 00:26
Spot on
Top level
RMX Scaler
Use mouse button to select one or more glyphs
Font – Facit [/…/FontLab/Facit_MM full.vfb]
Name
Facit old
RMX Scaler

	Wt0	Wt1
standard stem	246	664
italic angle	-0.0 °	-0.0 °

[✓ current selection]
previous selection
based on added suffix
entire font

	Wt0	Wt1
width	85.0 %	90.0 %
height	80.0 %	85.0 %
keep stroke	90.0 %	65.0 %

[✓ scale sidebearings]

	Wt0	Wt1
adjust spacing	30	20
vertical shift	0	0

remove existing suffix

add suffix: small

color: CB

save as:

< X > Cancel OK

Image 2

numerous ways, but many of them only serve a specific purpose. Within the Font Remix package there are numerous highly useful tools such as the RMX Harmonizer which helps the designer sync his anchor points for better curves and the RMX Scaler which enables the designer to easily adjust heights and widths without loosing the stem width of the character. The scaler tool can for example be extremely helpful for creating small caps or a condensed version from a fully designed typeface. Though the tools may be difficult to implement at first they have proven to improve the workflow of type design and are of no doubt a good start for a more convenient design flow.[13] Numerous similar help-tools have been produced, most can be related to the Python programming language.[14] To give a wider idea of what the toolkits are good for the Superpolator and MetricsMachine are worth mentioning, both of those are meant for working with the UFO format which was mentioned earlier. In Superpolator the designer can both interpolate and extrapolate fonts, which means to make different width variations[15], interpolate stands for making the font thinner or lighter and extrapolate stands for making it thicker or bolder. It is a stand alone format which means it can also be used in numerous font editing applications like FontLab and Glyphs[16]. The MetricsMachine is used for a faster flow of kerning, or adjusting space between letters, of already designed alphabets.[17]

2.2 Applications for Parametrized Type Design

In the design world there are many tools, methods and softwares designers use to make their life easier, and in type design, the same as in any part of design, there is a long tradition of designers creating their own custom tools which in some cases have become somewhat industry standards, such as the font editor Glyphs. Having access to such tools in a home computer enhances all production availability and gives anyone willing to learn, a manufacturing independence. It is still a fact that technology is no magic tool to produce work, and creative skills and general design knowledge is always needed to produce the dynamics that make letter forms harmonize with each other in words and text blocks.[18]

[13] Interview 2 (7.11.2013). Written interview between the author and Sigurður Ármannsson.

[14] Ibid.

[15] Ibid.

[16] *Working with UFOs*, Robofont, 2011, accessed 7.11.2013, http://doc.robofont.com/documentation/welcome-to-robofont/working-with-ufos.

[17] Bosler, Denise, *Mastering Type: The Essential Guide to Typography for Print and Web Design*, HOW, The United States, 2012, page 42.

[18] McMillan, Neil, *An A–Z of Type Designers,* Laurence King Publishers, United Kingdom, 2006, page 7.

Programming parameters for typefaces can be used as an art form, where parameters are defined by certain unusual standards to fit a certain project, but more often that not it is used for a faster process in designing general typefaces. It can be interesting to see what can come out of a collaboration of two disciplines, like in this case at hand, software engineering or programming and type design. Creating rules for similar letter forms reduces the work needed to complete the typeface and can additionally inspire new aesthetics in the process.[19]

There have been worries about the outcome of programming based type design, such as that parametrically designed fonts might loose its so called soul.[20] On the other hand the computer itself is completely blind to aesthetics and that can mean that having a program generate a typeface can create something blunt, that is, without the human touch which can often be influenced by some era-related trends.[21]

2.2.1 The Purpose of Parametrized Applications

As the time passes the numbers of design projects and therefore typefaces created become greater every day. Though making the process of the design easier inevitably leads to a higher number of quality-lacking work, it also continuously elevates the discipline towards developing more stimulating and innovating type work.[22] There is no such thing as a universal typeface that can fit every task at hand, so the demand for new typefaces for unique projects will never stop. That fact increases the need for good technological software which can enable designers to, for example, speed up the process of sketching new letter ideas and produce new variations of their previous work to fit a new type of project.[23] Processes have and will continue to become faster. Applications programmed, with the possibility of controlling manipulation devices which affect many similar letter forms simultaneously, seem to be one way to meet that demand. Donald Knuth is a computer scientist who, after the implementation of digital printing, was not satisfied with the quality of typefaces in his printed material comparing to his earlier work, which was printed with metal type. His answer to this problem was to design a math-based programming language, called Metafont, which made it possible for programmers to define certain parameters of similar letter shapes in a font and with those shapes generate a font.

[19] Interview 1 (31.10.2013). Written interview between the author and Niklas Ekholm.

[20] Dooley, Jeremy, *Creative characters: The faces behind the fonts*, MyFonts, 2011, accessed 11.11.2013, myfonts.com/newsletters/cc/201110.html.

[21] Interview 3 (31.10.2013). Written interview between the author and Rafe Copeland.

[22] McMillan, Neil, *An A–Z of Type Designers,* Laurence King Publishers, United Kingdom, 2006, page 7.

[23] Ibid, page 27.

The final version of Metafont was developed in 1982[24] and it is thought as the earliest serious work in the field of parametrized type design.[25] The typefaces generated in the system are not outline fonts, they are path-based, that means that everything is calculated from the centerline trajectory of the letter, therefore resulting in better quality letter forms than in other productions at that time. In this process there is no real automation for the user is in charge every step of the way. In later years many have experimented with the subject of parametrized type design in projects such as Metaflop by Simon Egli, which the Metafont base is used, Prototyp-o by Yannick Mathey and Louis-Rémi Babé and Computed Type by Cristoph Knoth. Knoth for example made his application after finding out that the long thorough process of letter sketching was not his strong side. He started to draw the letter shapes in vector programs and soon realized that the process insisted on numerous repetitive tasks.[26] He then decided to produce a simple font editor where all these nuances could be executed simultaneously by adjusting various sliders or inputting numeral values (Image 3). The program is based on the Python programming language and but also employs some factors from RoboFab, that is a plugin which was one of the reasons the RoboFont font editor was developed. The interface of Computed Type aims to be understood by users not familiar with programming. In the program the user is bound to a certain base font which he can then manipulate in various ways. Knoth says the program is good for quickly creating a custom typeface but it can also help the user understand the process of typeface design in a better way.[27]

2.2.2 Resisting plagiarism

When a young designer starts to experiment with graphic design, he will need fonts. Sadly, it is a fact that unlicensed fonts are still known to roam around and the case is that it is very hard to keep track of a font once it is distributed.[28] So a young designer might use unlicensed fonts or find free fonts, if not, the designer either buys the fonts legally or produces an easy, likely to be quality-lacking, typeface, because for a type design novice the process of designing a satisfactory typeface is not an easy job. Piracy on all sorts of

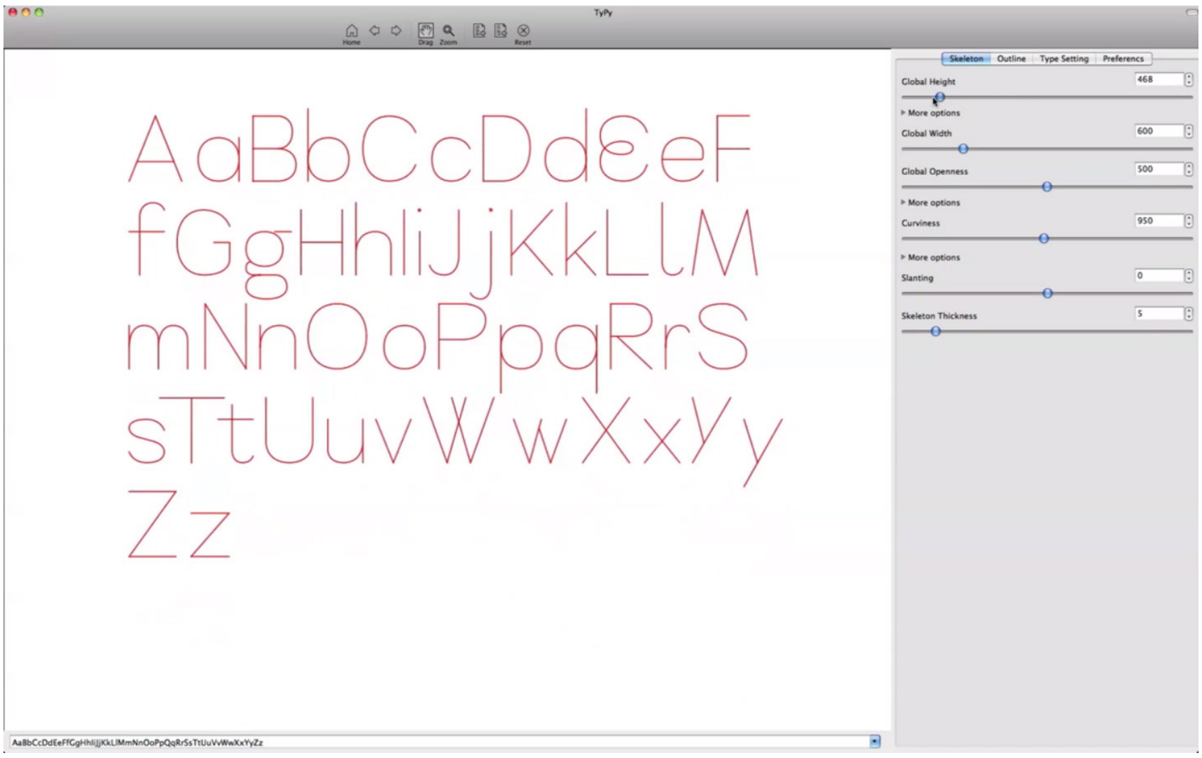
[24] Beebe, Nelson H. F., “Years of TeX and Metafont: Looking Back and Looking Forward”, thesis for University of Utah, The United States, 2003, page 9.

[25] Hu, Changyuan, “Synthesis of Parametrisable Fonts by Shape Components”, master thesis at Nanjing University, China, 1998, page 7.

[26] Interview 4 (13.11.2013). Written interview between the author and Cristoph Knoth.

[27] Ibid.

[28] McMillan, Neil, *An A–Z of Type Designers,* Laurence King Publishers, United Kingdom, 2006, page 13–14.



Global Height 468
Global Width 600
Global Openness 500
Curviness 950
Slanting 0
Skeleton Thickness 5

Image 3

creative work will never be eliminated[29] but one way of tackling this problem is making a font editor where similar coherent letter shapes, which all bow to the elementary rules of type design, can be manipulated all at once. That means, with a designed interface, editing the recurrent shapes of the typeface simultaneously.[30] This makes the process of designing or sketching a typeface increasingly easier. This fact was one of the main reason two french designers, Yannick Mathey and Louis-Rémi Babé, started making the font editor Prototyp-o (Image 4). In the start their idea revolved around the fact that there are plentiful different base rules in typography which normally are carefully studied, thought out and followed through by the designers hand, and based on that fact there must be a way for a program to learn those rules, for they are, pre-defined. Prototyp-o enables the designer to adjust various letter elements, such as serifs, thickness and curves, at the same time. Even though the process is made easier, the fact is that a computer software will never replace the eye of the designer, as it has not in the past, but what it can manage is to ease up the process by offering the opportunity to synchronize the design of similar shapes for an faster design flow.[31]

2.2.3 Outcome

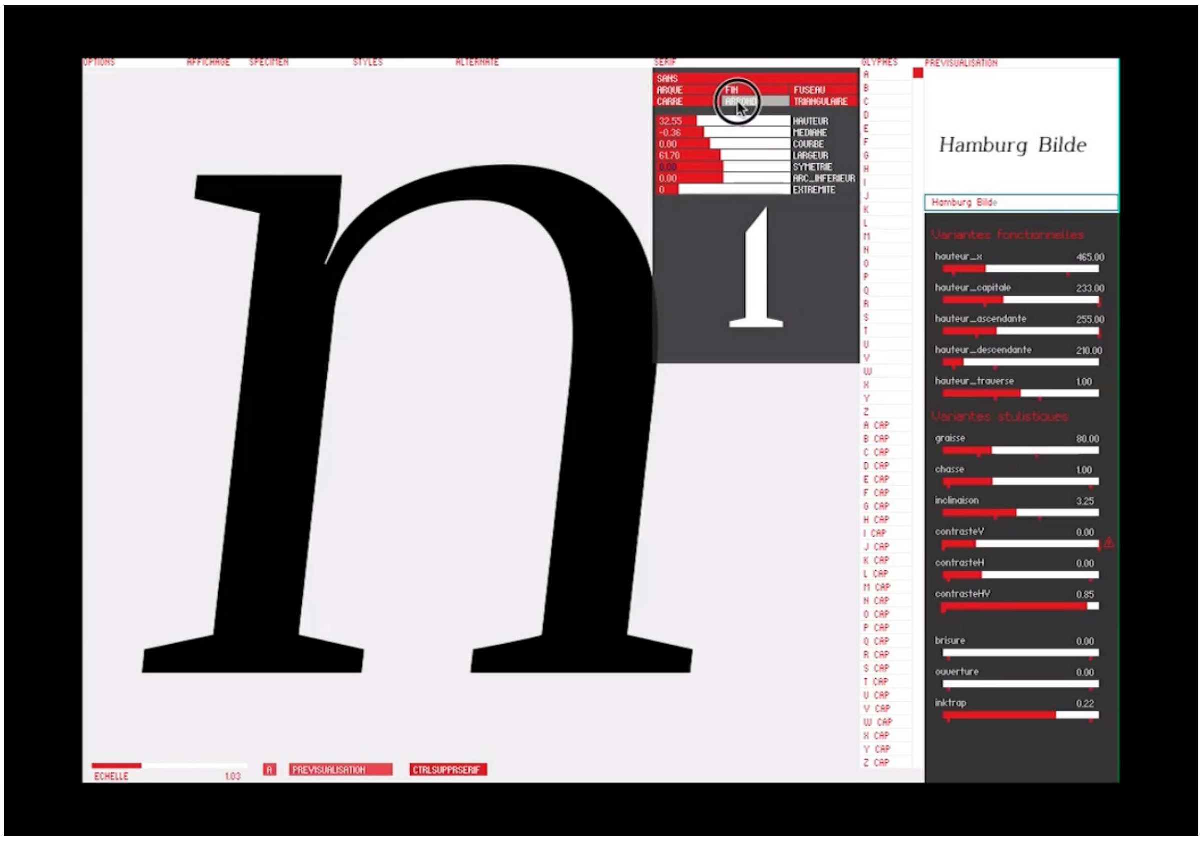
The Metafont language, mentioned earlier, has never gained much popularity within the design community because the designer would need to know programming basics and learn that particular language in order to use it. That is in itself a stopping point for the tool would likely only be used by programmers who are also interested in type design. Although some programmers have seen the value in this system and used it to build a more user friendly type design tools, one of those tools, built in 2012 by Simon Egli, Marco Müller and Alexis Reigel, is called Metaflop.[32] With Metaflop the user faces a fairly easily understood user interface, where he can play around with the different parameters of the type and affect simultaneously similar forms in the font. In continuation of that the font can be exported in an .otf file format and further developed in any supporting software. Though this gives you a quick way of generating a font, similar as in the Computed Type application, the control you have over the letter shapes is not ideal — and to begin with the designer is not working with his own design. The actual results from Prototyp-o are yet to be seen but comparing to information already gathered on the application and the

[29] McMillan, Neil, *An A–Z of Type Designers,* Laurence King Publishers, United Kingdom, 2006, page 15.

[30] Interview 5 (15.11.2013). Written interview between the author and Yannick Mathey and Louis-Rémi Babé.

[31] Ibid.

[32] *Metaflop terms,* Metaflop, 2012, accesssed 6.11.2013, metaflop.com/terms.



32.55 HAUTEUR
-0.36 MEDIANE
0.00 COURBE
61.70 LARGEUR
0.00 SYMETRIE
0.00 ARC_INFERIEUR
0 EXTREMITE

Hamburg Bilde

Hamburg Bilde

Variantes fonctionnelles

hauteur_x 465.00
hauteur_capitale 233.00
hauteur_ascendante 255.00
hauteur_descendante 210.00
hauteur_traverse 1.00

Variantes stylistiques

graisse 80.00
chasse 1.00
inclinaison 3.25
contrasteV 0.00
contrasteH 0.00
contrasteHV 0.85
brisure 0.00
ouverture 0.00
inktrap 0.22

ECHELLE 1.03

Image 4

promotional material published, the results seem to be a quite effective next step in the progressions of parametrized type design applications.

2.3 Conceptual Experiments

When designing a typeface, the design itself is never a stopping point, it means that a new tool has been created which will then be in the hands of the designers that utilizes the tool.[33] Like any other tool, it can be used well or poorly. While the font is used as a tool, it can also be thought of as a playground. During development of new type design technologies it is a fact that the computer is often thought of as a simulator for previous tools and the further possibilities of the computer itself are rarely challenged.[34]

Experimentation within type design is as free as in any other discipline that touches the subject of artistic thinking. Type design has always contained a certain element of expression, the touch of the designer is always there, and with new technologies and different ways of applying design thinking, the expression, as well as the message, in type design can be explored in numerous different ways. In the definition of music, if any sound can be called music then any item, that has the possible perception of a letter form, can be called a character.[35]

2.3.1 Reasons for experimenting with programming and type design

Design is constantly keeping up with contemporary currents and in any creative industry there is also a continuous discourse on all aspects. But everything that is, is based on everything that was, so a valuable guideline when designing is taking aim of traditional ways. A balanced or contradicting mix of the two can create something new, strong and challenging.[36] It has been argued that the design style of each era is highly influenced by the technology available at each time, new progressions in technology mean new possibilities in designing, but they also limit the user, which is working within a certain program, to execute things in a certain way. Pre decided methods, effects and work flows knowingly or unknowingly guide anyone who uses a certain tool. With that in mind it is as important, as mastering new technologies, to challenge the tools or programs at hand.[37] Both by modification and invention of different tools.

[33] Lehni, Jürg, *Typeface as Program,* ECAL: University of Art and Design, Switzerland, 2010, page 127.

[34] Ibid, page 85.

[35] Rocha, Ivan Terestschenko, “A Metaphysical and Linguistic Approach to Type Design and Typography”, master thesis at Rochester Institute of Technology, The United States, 1993, page 40.

[36] Baines, Phil, “Modernity and Tradition”, in *Eye No. 7*, Wordsearch Ltd., United Kingdom, 1992, page 6.

[37] Interview 1 (31.10.2013). Written interview between the author and Niklas Ekholm.

> "I would argue that the visual form of things is mostly inspired by technology, which presently can be seen in all the perfect euclidean basic shapes that are prominent in graphic design, or various stretched and bent forms. These tendencies are the result of the most basic tools in contemporary design programs like Adobe Illustrator."[38]

On that subject the type experiments by Erik van Blokland and Just van Rossum are a good example of testing the limits of available technology. Their main project related to the matter was a font called Beowolf, which is programmed with a randomizing algorithm that generates different outlines every time the font is used in different output applications (Image 5). That is that the letters changed once they were sent to the printer, so no change happened on the computer screen.[39] The font was not completely random for it had to be legible, so boundaries were set for the outline randomization. Additionally the user could decide to which extent the randomness goes for there are three levels of extremes available. When facing negative critique on the project Blokland and van Rossum point out an interesting fact: "There is no technical reason why characters should appear the same every time they are used".[40] These experiments push the boundaries between type design and technology rather then just serving a lingual communicative purpose. With the digitization of fonts there are both positive and negative sides, one of the negative points, interestingly pointed out by Blokland and van Rossum, is that a lot of the charming irregularities from hand-crafted methods were lost in the sterile computer environment after the developments conjured after the so called desktop revolution.[41] To some extent the randomness in Beowolf gives a human touch to the digital process, for through history it has often been asserted that a clean modern approach suits new developments, but in that process the lively touch of imperfection is clearly lost. While working on Beowolf, van Blokland and van Rossum pushed various limits of digital type design. These extra type experiments, though never published, pushed type design thinking further in exploring new usage of the medium. The experiments contain fonts such as a virus font, that if installed on a device, it started to make your installed Helvetica font illegible and a font that when used in a text document, automatically adds typos. These experiments, though it could be

[38] Interview 1 (31.10.2013). Written interview between the author and Niklas Ekholm.

[39] McMillan, Neil, *An A–Z of Type Designers,* Laurence King Publishers, United Kingdom, 2006, page 11.

[40] Blokland, Erik van and Just van Rossum, "Is Best Really Better?", in *Emigre 18,* Emigre Inc., The United States, 1991, page 26.

[41] Rocha, Ivan Terestschenko, "A Metaphysical and Linguistic Approach to Type Design and Typography", master thesis at Rochester Institute of Technology, The United States, 1993, page IV.

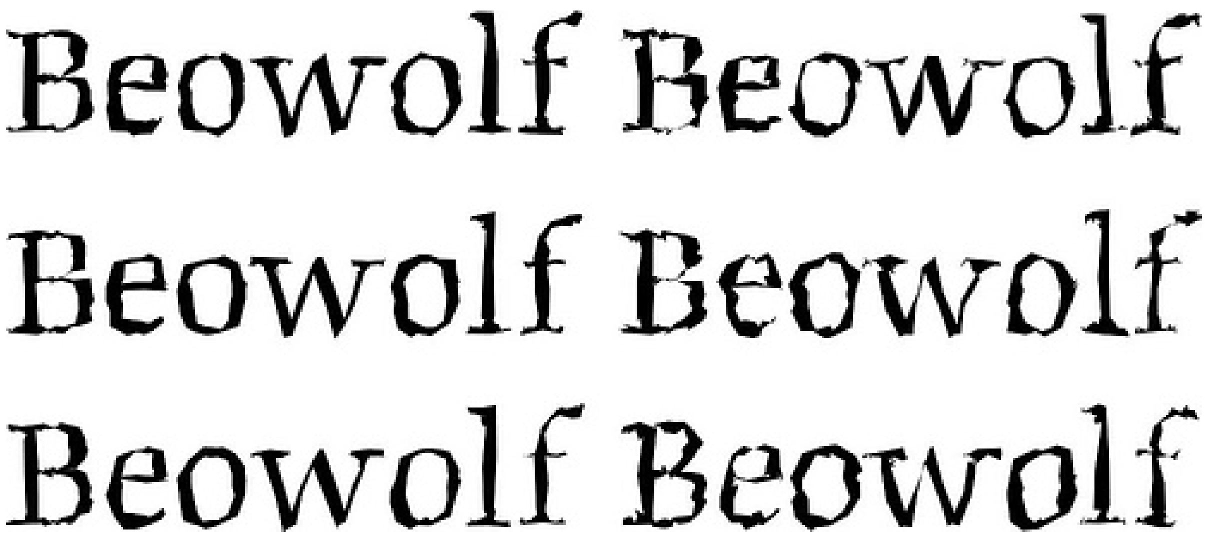


Image 5

thought of as unprofessional, or even criminal, to actually use them, are important in the process of investigating how deep type design can be, how it can mediate something else than languages and how much more can live behind a programming based typeface.[42]

2.3.2 Testing the limits of typeface communication

When designing a typeface, an important part of the process is to guard the legibility of the typeface, that it is communicative and enables the viewer to read and decipher. But in a modern world there is an enormous amount of type designers doing exactly that. What is more emphasized in some projects, is readability. Readability stands for how well the typeface is suited for the project or task at hand. When organizing heavy text for reading, in for example a book or a magazine, it is obvious that the text has to be comfortable for the eye, have a certain rhythm and a discernible flow. There, a well crafted and adequate typeface is the right choice. In some cases those kinds of typefaces do not fit the task that has been laid out. Those kinds of projects focus more on the typefaces ability to communicate some kind of identity or concept.[43] When designing typefaces with the aid of programming there is openness to a new dimension of experimentation, for the type is not designed by the eye but by a mathematical formula deemed appropriate to convey a certain message or action. In this branch of type design, legibility is only one of the aims of the design, an equally important aim is to see how the role of characters can be adjusted and applied to a digital world and with that communicate a certain concept or idea. In a discipline like Programming based type design, that has only been partly explored, there seems to be even more room for experimentation. One aspect of readability can be defined as what the viewer is used to and finds comfortable to see and understand, but living in a modern world where usualness is constantly evolving[44], all definitions also evolve, they are changed, discussed and redefined. All letters have some sort of concept lying behind the surface, even though most people would say letters merely present to us a shape rather than a concept. If the shape of small letter ‘a’ is studied across a broad selection of typefaces, it is easy to see how there is a common structure to all instances; it conforms to a prototype, but in some cases the structure is bent to an extreme extent without the communicating factor of the letter being lost.[45] In relation to these thought an highly conceptual case can be

[42] Blokland, Erik van and Just van Rossum, “Is Best Really Better?”, in *Emigre 18,* Emigre Inc., The United States, 1991, page 27.

[43] Interview 1 (31.10.2013). Written interview between the author and Niklas Ekholm.

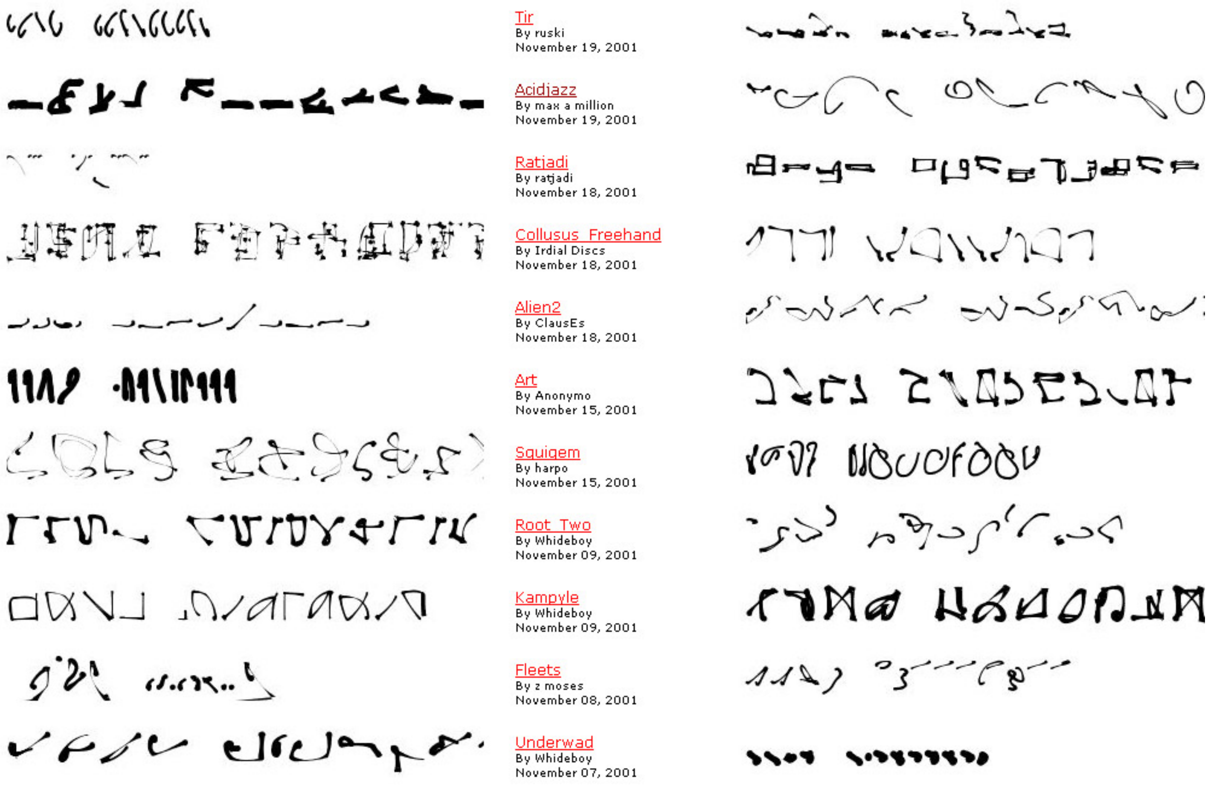
[44] Rocha, Ivan Terestschenko, “A Metaphysical and Linguistic Approach to Type Design and Typography”, master thesis at Rochester Institute of Technology, The United States, 1993, page 38.

[45] McGraw, Gary, “Emergent High-Level Perception of Gridletters Using Fluid Concepts, Part 1: The Letter Spirit Project”, doctoral thesis at Indiana University, The United States, 1995, page 4.

mentioned. That project is called the Alphabet Synthesis Machine, created by Golan Levin with Jonathan Feinberg and Cassidy Curtis. What is different about this project is that the alphabet generated with the machine is not supposed to be understood, but rather communicate a concept. The idea goes back to the authors childhood, when he first saw a foreign writing system and for he was not familiar with the letters presented, did not see words, but shapes. With the machine the user makes a single glyph which is then generated in to an alphabet system which hypothetically belongs to a new civilization (Image 6).[46] "The products of the Machine probe the liminal territories between familiarity and chaos, language and gesture."[47] The project uses the concept of an alphabet to communicate something entirely deviant from lingual information.

[46] Levin, Golan, "The Alphabet Synthesis Machine", thesis for a project at Art21 and PBS, United States, 2006, page 3.

[47] Ibid, page 3.



Tir
By ruski
November 19, 2001

Acidjazz
By max a million
November 19, 2001

Ratjadi
By ratjadi
November 18, 2001

Collusus_Freehand
By Irdial Discs
November 18, 2001

Alien2
By ClausEs
November 18, 2001

Art
By Anonymo
November 15, 2001

Squigem
By harpo
November 15, 2001

Root_Two
By Whideboy
November 09, 2001

Kampyle
By Whideboy
November 09, 2001

Fleets
By z moses
November 08, 2001

Underwad
By Whideboy
November 07, 2001

Image 6

Conclusion

The objective of this thesis was to investigate the field of parametrized and programmed typeface design and find out what has been done and the possibilities this interesting field contains. The question was: What are the possibilities of utilizing programming languages in type design, both for better workflows and more conceptual projects, how far along are we in parametrized type design and will that field become standard in the coming years?

The analysis of literature has led to the conclusion that scripting in typography seems to be generally used amongst professional type designers though it is hard to assert if such things are considered as standard. The method is already widely known amongst professional type designers and in my opinion those kinds of actions have a good chance of becoming industry standards in the near future. In relation to that I think producing numerous different plugins, that have the possibility of being implemented to a font editor, are a promising branch of typography. In that case a designer with little or no programming knowledge can take use of helpful programming based methods. The designer can utilize plugins of his own choice for specific tasks without having to learn the basics of programming or getting to know a whole new font editor. With these little scripts and plugins the process will flow faster, especially when designing big typeface families. To skip these kinds of helpful tools seems to be an old fashioned way of designing type.

The interview outcome was extremely helpful to see different opinions on the subject and for driving the research on with new thoughts and terms. They also supported the authors view that technology affects, in some way, the way we work with type and the outcome produced, though today the gap is smaller than ever because for the whole procedure often there is only one person with a computer device calling the shots. With that in mind producing new font tools can also bring on new surprising aesthetics.

I contemplated about some of the progressions so far in relation to appropriate literature. During the research I found out that the field is still young and a lot is unexplored. On the matter of parametrized font applications, starting out with a base font does not seem to be the best way to generally execute the process, for all of the users start with the same base, and therefore become more influenced than one would want to be when starting to design a typeface. It is great for experimenting in the field for single or a group of designers but for parametrized font applications to become some kind of standard they have to contain more flexibility. The programs are whatsoever interesting experiments for giving the designer surprising results, by producing pre-calculated values with various sliders the designer can take an interesting use of current technologies and try out extremes that might not be produced by type design sketching. The nearest application to becoming a good parametrized font editor is in my opinion Prototyp-o. Their reasons for creating the application and the developments so far are very promising. Also on the case of resisting plagiarism the application could be a start to a solution, at least a good gesture. In my opinion, and actually what I thought some of these projects were able to do in the beginning of the writing, the perfect situation for applications that deal with parametrized type design is if the user could upload older work and tweak the alphabet as well as each letter to suit new upcoming projects. So far all of the font editors I came across start out with different base fonts, which means the typeface is never solely designed by the user and therefore there is a good chance the software might effect the result more than desired. If the application is public the worst case scenario is that all fonts produced are just various versions of the base font designed by the creator of the application. Prototyp-o has not yet been publicly released but the ideal situation would be, though the program has a base font to begin with, that the base font is neutral enough and that the manipulation devices are so flexible that the outcome will not end up being only various alterations of the same font. I think that parametrized font applications will never become standard unless someone figures out a way to make them extremely flexible. The method of designing similar coherent glyphs at the same time sounds extremely convenient, but for such a tool to become a professional design tool there has to be a way for every user to design a font from scratch using parametrized methods, or at least implement them somewhere in the process.

Though experimenting with programming in the subject of more conceptual typefaces is an interesting take on the purpose of typefaces, their more general important role, which has greater possibility of influencing a bigger group of people, is when it is used in a clearly useful and productive manner in fonts such as FF Chartwell. For typefaces containing deviant programming bases to be professionally useful the typeface has to improve at least some aspects of the discipline, for example by speeding up the design flow. Conceptual experiments with programming and typeface design focus more of raising a single point and decreasing the gap between two disciplines, like in The Alphabet Synthesis Machine project, where psychology and typeface design are boiled together to raise an interesting point. Or in the project FF Beowolf, where my understanding was that they were using typeface design to point out simultaneously how many advantages and disadvantages new technologies give to design. Advantages being how technology can be implemented and used to its full extent in the process to give surprising results, and the disadvantages pointed out were that with the use of new technologies the human touch can be lost. Not to belittle the conceptual experiments with previous statements, because I and others think that they are extremely interesting and thought-provoking in many ways. Though they tend to raise questions and conclusions not directly related to typeface design, involving typeface design in conceptual experiments is an interesting way to approach subjects and additionally brings important conclusions to various matters which in my opinion can affect design and other disciplines in numerous ways.

Bibliography

Printed references:

Baines, Phil, “Modernity and Tradition”, in *Eye No. 7*, Wordsearch Ltd., United Kingdom, 1992.

Beebe, Nelson H. F., “Years of TeX and Metafont: Looking Back and Looking Forward”, thesis for University of Utah, The United States, 2003.

Blokland, Erik van and Just van Rossum, “Is Best Really Better?”, in *Emigre 18,* Emigre Inc., The United States, 1991.

Bosler, Denise, *Mastering Type: The Essential Guide to Typography for Print and Web Design,* HOW, The United States, 2012.

Hu, Changyuan, “Synthesis of Parametrisable Fonts by Shape Components”, master thesis at Nanjing University, China, 1998.

Knoth, Cristoph, “Computed Type Design”, bachelor thesis at ECAL: University of Art and Design, Switzerland, 2011.

Lehni, Jürg, *Typeface as Program,* ECAL: University of Art and Design, Switzerland, 2010.

Levin, Golan, “The Alphabet Synthesis Machine”, thesis for a project at Art21 and PBS, United States, 2006.

McGraw, Gary, “Emergent High-Level Perception of Gridletters Using Fluid Concepts, Part 1: The Letter Spirit Project”, doctoral thesis at Indiana University, The United States, 1995.

McMillan, Neil, *An A–Z of Type Designers,* Laurence King Publishers, United Kingdom, 2006.

Middendorp, Jan, *Dutch Type,* 010 Uitgeverij, The Netherlands, 2004.

Rocha, Ivan Terestschenko, “A Metaphysical and Linguistic Approach to Type Design and Typography”, master thesis at Rochester Institute of Technology, The United States, 1993.

Web references:

Dooley, Jeremy, *Creative characters: The faces behind the fonts*, MyFonts, 2011, accessed 11.11.2013, myfonts.com/newsletters/cc/201110.html.

How to Use FF Chartwell, FontShop International, year missing, accessed 30.11.2013, www.fontfont.com/how-to-use-ff-chartwell#intro.

Font Remix Tools 1.6, Just Another Foundry, 2012, accessed 10.11.2013, remix-tools.com.

Metaflop terms, Metaflop, 2012, accessed 6.11.2013, metaflop.com/terms.

Unified Font Object, LettError and Type Supply, 2012, accessed 14.11.2013, http://unifiedfontobject.org/team.html.

Working with UFOs, Robofont, 2011, accessed 7.11.2013, http://doc.robofont.com/documentation/welcome-to-robofont/working-with-ufos.

Interviews:

Interview 1 (31.10.2013). Written interview between the author and Niklas Ekholm.

Interview 2 (7.11.2013). Written interview between the author and Sigurður Ármannsson.

Interview 3 (31.10.2013). Written interview between the author and Rafe Copeland.

Interview 4 (13.11.2013). Written interview between the author and Cristoph Knoth.

Interview 5 (15.11.2013). Written interview between the author and Yannick Mathey and Louis-Rémi Babé.

Photography

Ahrens, Tim, "RMX Scaler", the photo is a screen shot of a video taken from *Font Remix Tools: RMX Scaler,* Font Remix Tools, year unknown, accessed 03.12.2013, http://remix-tools.com/scaler. (Image 2)

Knoth, Cristoph, "Computed Type – Cristoph Knoth", the photo is a screen shot of a video taken from *Vimeo*, Vimeo, 2013, accessed 03.12.2013, vimeo.com/60651938. (Image 3)

Photographer Unknown, "Alphabet Synthesis Machine", photo taken from *Alphabet Synthesis Machine*, Golan Levin and Collaborators, 2001, accessed 03.12.2013, flong.com/projects/alphabet. (Image 6)

Photographer unknown, "Beowolf", photo taken from *Deep in the Archives*, Eye Magazine, 2010, accessed 03.12.2013, eyemagazine.com/feature/article/deep-in-the-archives. (Image 5)

Photographer Unknown, "FF Chartwell", photo taken from *FontFont has really given FF Chartwell an amazing second life*, FontFont, 2013, accessed 03.12.2013, www.fontfont.com/news/travis-kochel-on-ff-chartwell. (Image 1)

Yannick Mathey and Louis-Rémi Babé, "Kickstarting Font Creation", the photo is a screen shot of a video taken from *Vimeo*, Vimeo, 2013, accessed 03.12.2013, vimeo.com/72371197. (Image 4)