THE FRANKEN-FONT

by
Cem Sina Çetin

Submitted to the Graduate School of Arts and Social Sciences
in partial fulfillment of
the requirements for the degree of
Master of Arts

Sabancı University
June 2014

THE FRANKEN-FONT

Cem Sina Çetin
M.A, Visual Arts and Visual Communication Design Thesis, 2014
Thesis Supervisor: Onur Fatih Yazıcıgil

Keywords: Digital Typography, Generative Arts, Automatic Font Design, Letter Generation, Hand Writing

ABSTRACT

Today, the digital font system provides a standardized infrastructure for publishing and utilizing fonts with great versatility; such as simulating ligatures and specialty fonts for custom uses. Although the current fonts are able to provide solutions for almost any use case, everything they are capable of depend on the fact that font files are large prefabricated [1] letterform libraries, governed by a set of complex rules. Therefore, it is fundamentally impossible to create fonts, which generate the letters on the go. This would provide a solution for a scenario, where each instance of the same letter should be almost the same but slightly different, while maintaining an overall visual consistency, such as the handwriting.

The Franken-font is a two-phase project. The first phase focuses on whether or not the human visual perception rejects a font with constantly changing letterforms and the second phase focuses on providing a new kind of font system, which is capable of generating letters instead of just fetching them from a library. This thesis project covers the first phase.

The hypothesis of this thesis is that as long as the letterforms are consistent, the human brain should not marginally reject the font. To test this, a custom application is developed to interpolate between Helvetica and Syntax, which creates letters carrying characteristics of both typefaces. An online survey will be executed to see the preference rate of the Franken font. Consequently, the findings of the survey should show whether or not the phase two is feasible.

THE FRANKEN-FONT

Cem Sina Çetin
Görsel Sanatlar ve Görsel İletişim Tasarımı Yüksek Lisans Programı Tezi, 2014
Tez Danışmanı: Onur Fatih Yazıcıgil

Anahtar Kelimeler: Sayısal Tipografi, Yaratımsal Sanat, Otomatik Font Tasarımı, Harf Yaratımı, El Yazısı

ÖZET

Günümüzde dijital font sistemi, fontların kullanımı ve yayınlanmasında ligatür simülasyonu ya da özel amaçlar için tasarlanmış fontlar gibi bir çok kolaylık sağlamaktadır. Mevcut fontlar neredeyse tüm kullanımlar için çözüm sunabilse dahi, kabil oldukları her şey fontların önceden tasarlanmış[1] ve harf şekilleri kapsayan, karmaşık kurallar tarafından kontrol edilen büyük birer kitaplık olmalarına dayanmaktadır. Bu nedenle de, harfleri kullanım anında üreten bir font tasarlamak sistemin doğası gereği imkansızdır. Böyle bir font olsaydı, her harfin birbiriyle neredeyse aynı fakat biraz farklı olduğu ve genel görsel tutarlılığın korunmasını gerektirecek, el yazısı taklidi gibi amaçlar için çözüm sunabilirdi.

Franken-font iki aşamalı bir projedir. İlk aşaması, insan beyninin harfleri sürekli değişen bir fontu kabul edip etmeyeceğini araştırmaya odaklanmıştır. İkinci aşaması ise, harf şekillerini kitaplıktan çekmek yerine kullanım anında üreten bir sistem tasarlamayı amaçlamaktadır. Bu tez projesi, ilk aşamayı kapsamaktadır.

Bu tez projesinin hipotezi, harfler tutarlı olduğu sürece insan beyninin fontu marjinal şekilde reddetmeyeceğidir. Bunun test edilmesi için Helvetica ve Syntax fontlarının karakterlerinin ortalamasını alarak, iki fontun da özelliklerini taşıyan harfler üreten bir uygulama tasarlanmıştır. Bir internet araştırması düzenlenerek, Franken-font'un tercih edilebilirliği ölçülecektir. Böylece araştırma sonuçları, projenin ikinci aşamasının uygulanabilir olup olmadığını gösterecektir.

ACKNOWLEDGEMENTS

I sincerely thank my thesis advisor, Onur Yazıcıgil, for his invaluable contributions to this thesis, for being an erudate mentor as well as a supportive friend.

I would also like to thank Servet Ulaş and Doruk Türkmen for the unique masters experience, during which the Franken-font was fostered.

I always was and always will be grateful to Merve Çaylı for my mind.

I finally thank my parents, Ayşe Sedef Antay and Orhan Cem Çetin, for the diverse set of talents I was lucky enough to inherit from them, both through nature and nurture.

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

Ever since the invention of the printing press, one of the premises that typeface design has been relying on is that every instance of a letter should be represented by the same glyph[1]. Even if it is not, the set of glyphs used to represent a letter should be very close variations of each other with minimal adjustments to support the text's visual coherence, such as a longer tail for the "z" letter at the end of a line. (Zapf, 2007, p.127) [2]

It would be unfair to question a type cutter's intention for creating only a couple of glyphs for each character, when the physical limitations and difficulties caused by manually carving letters from lead are considered. Yet, as the process of type design is translated to the digital medium, the means of creation and thus the nature of the product change. The glyphs are no longer defined by their physical properties, but by mathematical abstractions of the desired shapes, stored as vector image definitions in a font file.

> *"The computer is, on the face of it, an ideal device for reviving the old luxury of random variations in design at the threshold of perception. But conventional typesetting software focuses instead on the suicidal notion of absolute control - and has been hamstrung in the past by the idea of a single glyph per character. The modern computer with its practically limitless computing power is able to create different variations of a letter at each keystroke. The creation can be completely random, with controlled randomness or based on a pattern."* (Bringhurst, 2002, p.185) [3]

In order to mold lead into the desired shape, appropriate conditions should be met such as correct temperature, enough time and expertise. By contrast, when a shape is produced from a digital abstraction, the entity of the glyph - through the nature of the

[1] Through out this paper, the term glyph is used for the visual representations of letters, independent of the font that is used.

algorithmic bits that unlike their analogue counterparts, the atoms are open to infinite manipulation - is ready to be reshaped in endless ways without needing more than a couple of milliseconds. The algorithm replaces the analogue technology, the vertices replace the physical material of the typeset, the act of printing by ink is replaced by rasterizing the instances of the letterforms. Consequently, once the user is able to access the vertices and to transform them, it is no longer impractical to create thousands of variations of the same glyph in a heartbeat. Why then, should a designer be content with using only a limited number of glyphs for each letter?

Once we know what a word looks like in its written form, the human brain no longer reads the letters that create the word but instead, the word is perceived as a whole shape. Therefore, the process of reading depends on the pattern recognition ability of our brain, which is highly evolved in our species.

> *"Once we have learned the pattern of the word "window", we never again read the individual letters; the larger pattern is immediately matched as a gestalt."* (Hill, 1999 p. 13) [4]

The process of pattern recognition relies not only upon visual perception, but also the ability to intellectually understand and interpret features, visual references and concepts. In order to recognize letters and words, the brain does not run an algorithm to match the vertices of the shape and determine, which glyph it is. Instead, it refers to the glyph's features and our past experiences to understand the meaning. Bill Hill refers to this as a result of an evolutionary selection: the survival of those, who are better at seeing subtle differences in texture and form (e.g. determining whether or not a fruit is edible by looking at its texture or understanding the emotional response of another primate by reading its facial gestures).

"In a very real sense, the form of the book as we know it today was predetermined by the decision of developing humans to specialize in visual pattern recognition as a core survival skill." (Hill, 1999 p. 7) [4]

Since we have an evolutionary advantage in "seeing" things, we are also able to recognize the letters no matter how distorted or differently designed they are. If this wasn't the case, then we wouldn't be able to read our own handwriting, in which no two

letters are the same. In other words, the technique of handwriting would be nonexistent as we wouldn't be able to develop it to begin with. Therefore, it is not unimaginable to expect the human brain to easily interpret a piece of text, when it is written by a system that is capable of producing glyphs at each occurrence of a letter, which are very similar but slightly different versions of each other.

Consequently, the project Franken-font proposes a testing procedure to understand whether or not using slightly different glyphs for each letter whilst reading a long text is obstructive to the reading experience. The procedure consists of two parts, namely the typing environment and the survey.

The developed typing environment is responsible of generating letterforms for any given text and providing a vector image output of the rendered text, while the survey is tailor-made to use these outputs to gauge the user response to the generated text. The working principles of the typing environment and the survey will be explained in detail in the chapters *The Frankenstein Application* and *The Survey*, respectively.

2 Generative Art and Typography

Generative art is the practice of using an autonomous system to *generate* either the entire artwork or a part of it. Although the definition encourages to imagine the aforementioned "autonomous system" as a computer algorithm, generative art does not necessarily have to be digital. A generative artwork can be created by using simple machines, manual randomization or even biological, chemical and physical processes. An example for an artwork that utilizes physical events to generate the final spectacle is the *Condensation Cube* (1963-1965) by Hans Haacke. This work features a partially water filled, airtight plexiglass box of size 76 cm at each size. [5] The water inside the cube evaporates and condenses in an infinite loop, thus reflecting the artist's focus on the motion, light and refraction. This work is considered as generative art because it is created by the autonomous system that is the laws of physics governing the state of water inside.

The advances in computer science has an obvious impact on generative art, since the modern computer's number crunching abilities far exceed the range of manual work and it is possible to simulate both real and imaginary phenomena to create generative artworks that are otherwise impossible or very difficult to realize. The exceptional rate and speed of randomizations the computer can achieve is at the core of this thesis. Conventional typography, however, does not tolerate randomness. Therefore merging generative arts and typography is a design challenge more than an algorithmic one.

Even though digital fonts are - by definition - assets prepared for computer use, they inherit a five-century-old practice of the printing press, which begins with the production of the Gutenberg Bible in 1455. Even the terminology (e.g. "kerning", "leading" and "bearing") used for digital fonts is borrowed from its predecessor.

Consequently, digital fonts are not re-inventing type design but merely translating them into a new medium. By extension, the digital typography's objective is the same with that of conventional typography: *conveying specific and coherent ideas* (Warde, 1955, p.2) [6]

In the essay *The Crystal Goblet*, Beatrice Warde explains transparency as a necessary quality of good type design and the image of a crystal clear goblet acts as a metaphor of successful typography. Warde argues that typography should stay humble and not distractive, so that it does not interfere with the ideas it is meant to convey. Just like a perfect goblet, which doesn't alter the visual quality of the wine it holds.

> *"A public speaker is more 'audible' in that sense when he bellows. But a good speaking voice is one, which is inaudible as a voice. *** Type well used is invisible as type, just as the perfect talking voice is the unnoticed vehicle for the transmission of words, ideas. "* (Warde, 1955, p.2) [6]

So, is *transparent generative typeface* an oxymoron? Or is it possible to achieve a sense of humility and invisibility, while intruding every single building block of each letter at each occurrence?

> *"The book typographer has the job of erecting a window between the reader inside the room and that landscape which is the author's words. He may put up a stained-glass window of marvellous beauty, but a failure as a window; that is, he may use some rich superb type like text gothic that is something to be looked at, not through."* (Warde, 1955, p.3) [6]

If good typography is a window to be looked through, then good generative typography should be a system of building windows to be looked through. Eventually, the aimed success of the Franken-font does not only lie in its ability to please aesthetically, but also to do this silently. The mechanism that keeps the letterforms constantly changing must not distract the readers and they should be able to focus on the written content that the text is meant to convey.

3 Related Work

Although currently there exists no dynamic font that creates letterforms during the runtime, there have been several attempts of creating fonts with random glyphs. The lack of dynamic fonts is due to the modern and standardized operating system architecture, which uses a font file that works as a look up table to fetch related glyphs rather than a generator. Consequently, a system that provides a generation algorithm rather than a shape library cannot currently be incorporated into the general usage.

> Among some of the notable examples, the typeface Kosmik (FontShop 1993) by Erik van Blokland and Just van Rossum tries to tackle the randomization problem by providing three alternative glyphs for every letter. Not only the system is not truly dynamic (because the forms are still predefined), but also the algorithm is extremely resource consuming: *"the printer-driver leaps from font to font with each repetition of each letter. This reduces many systems to a state of nervous collapse after setting a few lines."* (Bringhurst, 2002, p.185) [3]

Another notable example is the font Beowolf (FontShop, 1990), by Erik van Blokland.

> *"The letterforms are sent to the output device through a subroutine, devised by Just van Rossum, that provokes distortions of each letter within predetermined limits in unpredetermined ways. Three degrees of randomization are available. Within the specified limits, every letter is a surprise."* (Bringhurst, 2002, p.185) [3]

The problem with the font Beowolf is that the randomizations are all based on a single seed[2] and distortions tend to render the letters illegible after a few iterations. In other

[2] Seed is the initial state of a random generation system. Seed is necessary for digital randomization and arbitrary seeds result in different, uncorrelated random numbers.

words, the manipulations tend to diverge from the starting form and therefore not feasible as a way of setting constraints for the randomization.

A more modern approach to creating hybrid forms is the font Roboto by Google Android, designed for mobile platforms. Roboto, ironically called a frankenfont as well, is a mixture of five different fonts: Helvetica, Myriad, Univers, FF DIN and Ronnia. Google describes it as *having a "dual nature. It has a mechanical skeleton and the forms are largely geometric. At the same time the font's sweeping semi-circular curves give it a cheerful demeanor."* (Coles, 2011) [7]

Roboto is not a dynamic font and available on Google Webfonts. However, the design decision and naming shows a striking similarity with the Franken-Font, in terms of trying to merge grotesque sanserif with humanist sanserif.

Last but not least, the project *Text Invader* by Onur Yazıcıgil tackles the problem of whether or not a piece of text can be enhanced by planting visual elements within the linear flow of the text. [8] To do this, image and word pairings are implanted in an existing font, which will in turn replace the words with the corresponding images. This way, unpredicted forms and meanings emerge from the rendered (or invaded) text. The randomness and generation in this workflow lies in the act of typesetting images within a body text. Yet, this approach as well does not address the problem of true generative typography as the pairings are predefined in the font, just like all the letterforms.

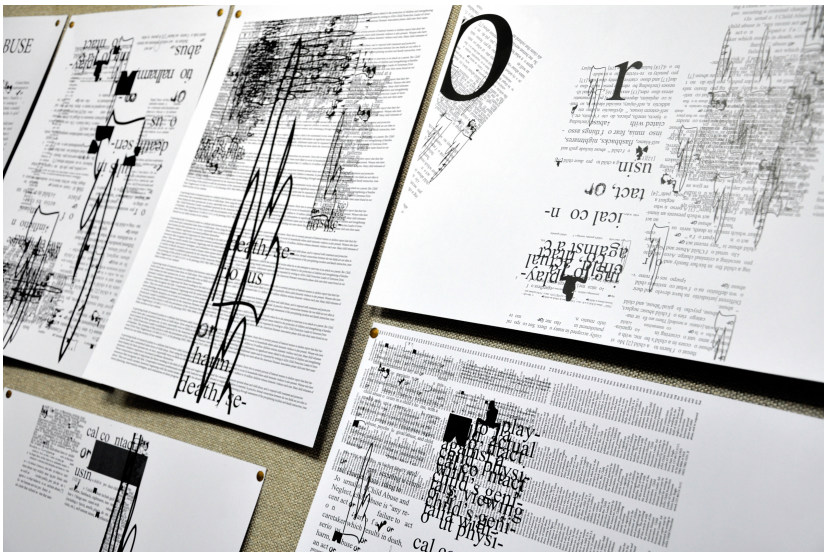


Figure 1 Bilkent University (2011), Text Invader Workshop [8]

4 Choosing Fonts for Interpolation

Since the aim of the project is to see the effects of using different glyphs for the same letter at each occurrence, it sets certain constraints for choosing the correct fonts: The fonts that will be used for interpolation should not be too close, so that there is a tangible difference between the generated glyphs. Yet, they should not be too different either, so that even though the resulting glyphs are different, their skeleton is not.

The findings of the paper Humanist vs. Grotesque Sanserif by Onur Yazıcıgil show that, when reading habits are considered - due to historical and habitual reasons - it is observed that people who are used to reading from printed media prefer humanist fonts, whereas people, who are used to reading from a digital environment such as a computer, prefer grotesque fonts. This preference however does not imply any correlation to the preference of serif or sanserif fonts.

> *"More and more people use the screen for continuous reading. This can be explained by the high use of sophisticated mobile devices, and the accessibility of such devices like, Apple's I Phone and other Internet connected multimedia devices. Consequently, findings in this paper illustrate a tendency to prefer grotesque sanserifs, which are highly used as default screen types. In such cases, habit factor may override the historical factor.*
>
> *On the other hand, respondents who chose print for often-used medium tended to prefer humanist sanserifs, which is a descendent of a long tradition of printed book types. This correlation between often-used medium and type preference may be a guide for type designers, typographers and graphic designers when designing for the intended user. "* (Yazıcıgil, 2009 p. 47) [9]

Based on this finding, for the Frankenfont project, it was decided to use a humanist and a grotesque sanserif fonts for interpolation. The survey results of Yazıcıgil clearly show that there is a transition in preference for humanist and grotesque fonts for reading long

texts. [9] Using a font from each family would generate letterforms that bear similarity to both categories, therefore supplying the missing gap between the two inclinations for fonts.

Helvetica was selected as the grotesque end of the interpolation, whereas Syntax was selected as the humanist end of the interpolation. Both bear clear characteristics of their own families, added to which is that both of these fonts are designed for long texts. The following two specimens in Figure 2 belong to Helvetica and Syntax, respectively.

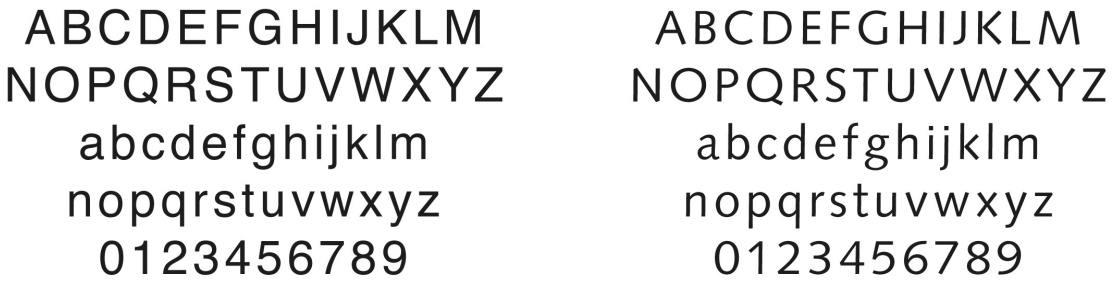


Figure 2 Helvetica & Syntax Specimens

Figure 3 Helvetica vs. Syntax

5 The Role of Typography on Subliminal Perception

Typography is, by definition, an inseparable part of written communication and therefore evolves along with the contemporary writing habits of the society. As an example, with the emergence of the Internet and user created content, there have appeared Internet-specific writing rules and ethics, such as writing in capital letters on an online platform being directly considered as an analogy of shouting.

The analogy between capital letters and shouting has become an intrinsic part of the perception of the reader only within the last couple of decades, whereas typography existed long before the Internet. If a period as short as a couple of decades can transform our understanding of capital letters, it is not a farfetched assumption that we are hard coded with type design and context associations; Certain ways of type design carries a silent message to the reader, without needing to explicitly tell what the message is. For instance, a poet does not need to tell us what he writes is a poem; the format of the body of the text already implies what it is. In other words, even though the reader does not necessarily have to consciously perceive the design decisions on a piece text, he or she certainly perceives their effects on the message.

As it is stated on paper Humanist vs. Grotesque Sanserif, people with specific reading habits tend to prefer either grotesque or humanist fonts. [9] In other words, the medium of the text and the form of the typeface has an implicit association. Therefore, once the interpolated font is used to render a piece of text, how the brain will respond is unknown. It might completely distract the reader as different letters might have different associations. On the other hand, it might not effect at all just like reading a hand written text, where every letter is a little bit different but they have over all consistency. The proposal of this project, as stated earlier, is that it won’t have a negative effect.

6 The Frankenstein Application

In order to extract the letterforms from a font file in an editable format and compile them into a readable chunk of text, a number of programmatic procedures and manual preparation are necessary. The reason for this is the aforementioned standardization of font files, which forces them to be read-only vector shape look up tables. Consequently, a tailor-made platform is necessary.

Existing commercial design systems are not possible to alter. On the other hand, open source solutions exist but they are prepared to cover a very large range of uses. Hence it is very inefficient to try to tweak an existing open source project to meet the needs of this special topic. Therefore, the easiest approach to solve this problem is to develop an environment from scratch.

An application capable of undertaking the given task, needs to be able to:

- Read vector graphics,
- Decode them into a form of information that can be manipulated,
- Render the manipulated data back into visual information.

Among many vector graphics file formats, SVG format stores the vector data in readable ASCII format (opposed to binary), which is even possible to mentally visualize a given shape by simply reading the list of plain text instructions that are stored in the file. Therefore, using SVG format reduces the "import file" routine into text file reading, so decoding and rendering this information boils down to string parsing and 2D graphics generation.

"Processing API" developed by Ben Fry, provides a programming environment for artists by simplifying the coding process. [10] Although simplified, it still supports

advanced JAVA programming, as well as coming bundled with a variety of libraries to make it easier to visualize data. The applications can effortlessly be compiled via processing for different operating systems as well. Thus, the Frankenstein Application is developed on Processing.

6.1 Glyph Generation from a Font File

A font file is a library of vector images that are associated with keystrokes from a keyboard. As the user presses a key on the keyboard while a text input area is selected, the operating system:

- Recognizes the keystroke
- Searches for the relevant vector image (the glyph) stored in the font file
- Rasterizes it on the desired location.

Most of the font files provide relevant glyphs for relevant letters, i.e. when the user presses the ‘g’ key on the keyboard, a representation of the letter ‘g’ appears on the screen. This is not necessarily the case, because there can be font files for special uses such as ornaments or musical symbols, where pressing the ‘g’ key may result in generating the musical notation G-Key, instead of the letter g. In other words, a font file is only a list of glyphs and it does not necessarily have to contain letters, let alone recognizing them as letters at all.

The process of fetching a vector image from a font file and creating a visual representation on the screen is executed on a lower level; It is hidden from the user and it should be hidden for the user friendliness of any design and/or typing environment. While this is necessary for the ease of use, it also makes it impossible to easily manipulate the font file for artistic or experimental purposes - as this project - on a commercial environment.

6.2 Interpreting Font Files as Vector Images

Since a font file is a collection of vector images that are called upon keystrokes, every text written on a vector graphics application is a collection of vector images. Even though a font file by itself is not editable, the glyphs that it fetches for letters are editable, once the text field is discarded and the letters are expanded back to vector images.

As an example, typing the letter ‘a’ on Adobe Illustrator in a text field fetches the associated glyph from the selected font. This glyph, along with the parent text field, can be transformed (i.e. scaled, rotated and translated) but its vertices are hidden. Yet, after converting it to outline[3], the internal pieces of the shape are exposed as shown in Figure 4.

Figure 4 vector outline of Helvetica 'a'

[3] Convert to outlines: Adobe Illustrator feature that extracts the editable vector image from a text area by destroying its editable text features and exposing the underlying vector image.

The shape is no longer a glyph in a text field; two disjoint paths are exposed along with the vertices that create the shape. After converting a glyph into a vector image, the manipulation process no longer relies on understanding the intricate machinery of a font file, but it is simplified down to understanding arbitrary amount of Bezier curves.

6.3 Generating Unique Glyphs

In order to generate unique glyphs that resemble each other at an enough level that they can be recognized as different instances of the same letter, the computer should need a source to create derivations.

Using a single source glyph would be enough to derivate infinite amount of new ones, but the chance of achieving visual harmony is either too low, or requires too many interventions on the process of generation, which undermines the effect randomness.

Instead of using a single source, using two source glyphs results in a more desirable outcome. Choosing two different fonts and interpolating equivalent glyphs in between, practically yields as many intermediate glyphs as the floating-point accuracy of the computer (e.g. if the smallest step between two integers a computer can calculate is 0.01, then there are 99 intermediate steps between the two arbitrary numbers).

One problem of interpolating two glyphs is that in order to interpolate, the two shapes should have matching number of vertices at matching locations on each letter. For instance, the following two shapes in figure 5 cannot be interpolated because they have 19 and 52 vertices respectively.

Figure 5 Syntax 'a' vs. Helvetica 'a' in vector form

The solution for this problem is adding the missing vertices on the shape manually to prepare the shapes for interpolation. The additional vertices are highlighted in red in Figure 6.

Figure 6 Compensating the missing vertices

Now that the two shapes are ready for interpolation, the application should be told which vertices are matching. A vector image in SVG form is defined by the absolute position of the first vertex and the relative positions of the following vertices. The starting vertex can be any vertex and if the starting vertices are at matching positions on both shapes, then the interpolation can take place smoothly. Figure 7 shows the %50 interpolation for the letter a before explicitly defining the starting vertices.

Figure 7 The Interpolation Abomination

Notice how the shape is beyond recognition. When matching vertices are interpolated, the outcome vertex is positioned at the weighted mean of the two source vertices, where the weight is the ratio of interpolation. When the vertices are not matched correctly, the two shapes start from unrelated positions and the incorrect mapping between the vertices propagate through the entire shape as seen in Fig. 7.

After the glyphs' starting vertices are manually chosen before creating the SVG file, they become fully compatible for the interpolation. Explicitly declaring the starting point of a vector shape is a software specific task and therefore not relevant to the content of this thesis.

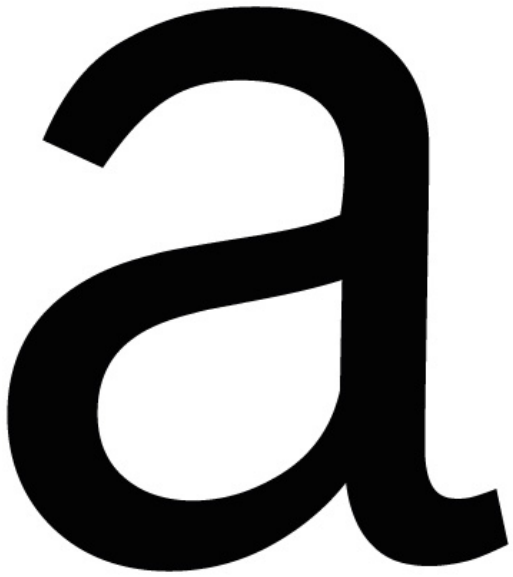


Figure 8 The Franken-a

Once the vertex shift is fixed, it is possible to generate all the intermediate steps between the two shapes.

Figure 9 Intermediate steps of the Franken-a

The following letters show a hundred random interpolations of each letter. The pure grotesque and pure humanist ends of the interpolations are represented in red and blue, respectively, while the intermediate forms are light gray. See Appendix C for 5 possible sets generated by the Franken-font.

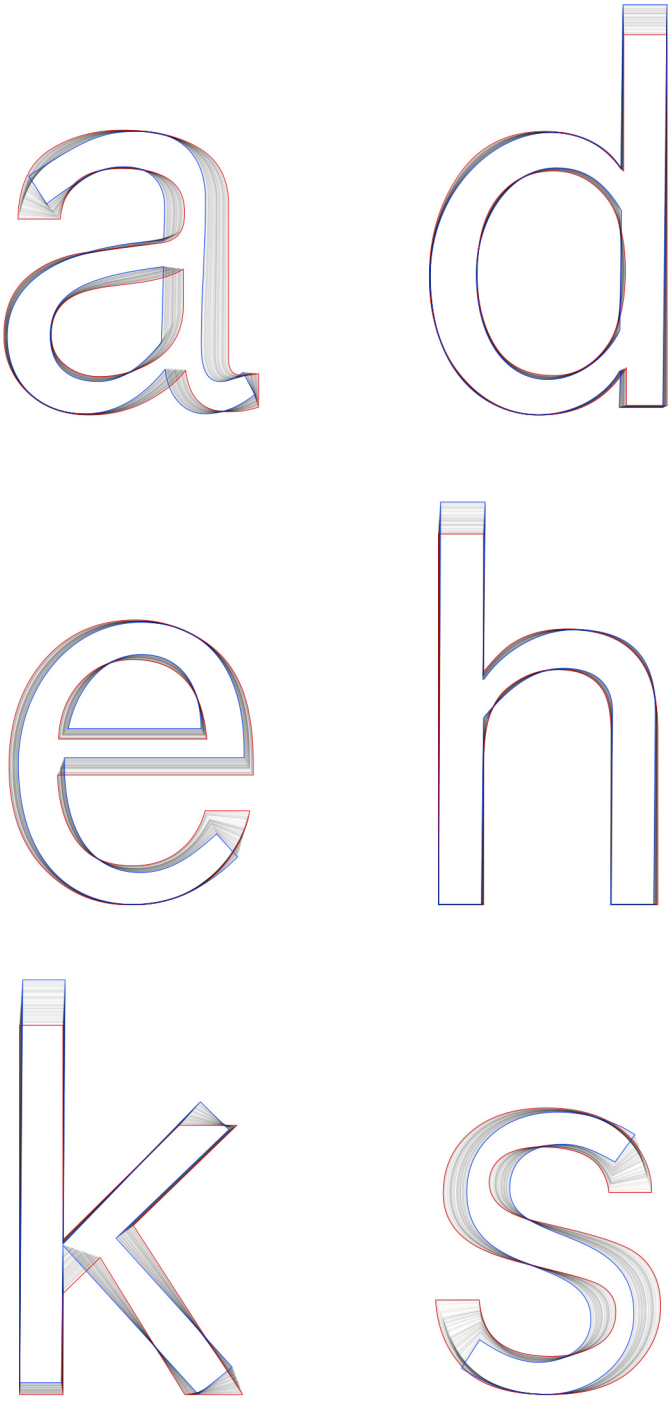


Figure 10 The intermediate forms

6.4 Typing with Generated Glyphs

As the letters are transformed into their vector image forms by creating their outlines, essential information, such as kerning is lost. Because, these information are imported from the source font file and what we have is regarded just as an image by the computer. Consequently, in order to type with the generated letters, the application should know how much space a letter spans horizontally, so that it can place the letters with appropriate spacing.

Setting a standard kerning without taking the individual letters' shapes into consideration yields a result as the following in figure 11.

Figure 11 Using static spacing

Every letter is placed in an equally separated grid, resulting in an uneven spacing between letters with extreme difference in width. As a solution, the letters are associated with predefined widths – just like the metric information of a font file – so that the separations are visually harmonious.

Figure 12 Using static per-letter spacing

Even though this method produces much pleasing results, there exist other fundamental problems. By using this approach the left bearing and the right bearing information are merged into a single variable, the kerning does not exist and the width difference between the two ends of the interpolation (i.e. the Helvetica end and the Syntax end) glyphs are not considered.

6.4.1 Interpolating the Glyph Width

The equivalent glyphs from different fonts do not necessarily have matching widths. As an example, the following two ‘s’ letters in Figure 13 are from Syntax and Helvetica, respectively.

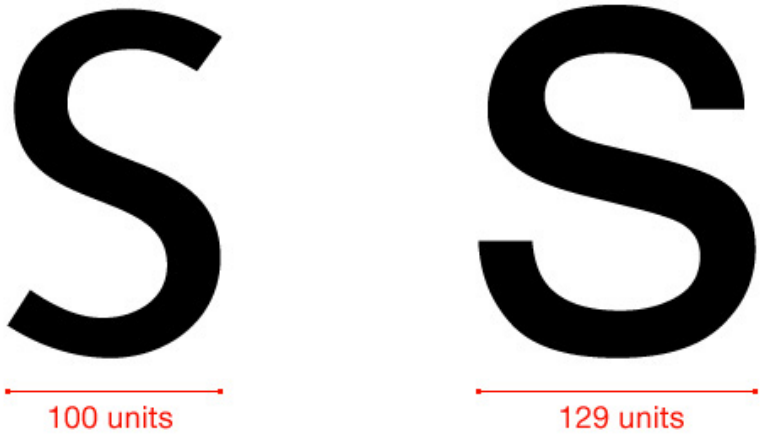


Figure 13 Width difference between Helvetica and Syntax

If the former is assumed to have a width of 100 units, then the latter’s width is 129 units. Therefore, the metric should be interpolated along with the glyphs as well.

As a solution, every glyph’s width is measured when the files are loaded, and upon matching the glyphs, their ratio is calculated and stored with the interpolator object assigned to that letter. The ratio is calculated in the form of (Helvetica / Syntax). For the example of the letter ‘s’, the ratio is 1.29.

Since Syntax and Helvetica are on each end of the interpolation, the multiplier for the metric of the glyph changes between the range, 1 and 1.29. In other words, while the interpolation is on the range of 0 percent to 100 percent, the interpolation of the metric should be mapped to the range 1 – 1.29. As an example, suppose the metric for %40 Syntax - %60 Helvetica has to be generated. So the calculation is as follows:

1.29 – 1 = 0.29	*Subtract the minimum value*
0.29 x 0.4 = 0.116	*Apply the interpolation*
0.116 + 1 = 1.116	*Add the minimum value back to find the multiplier*
100 x 1.116 = 111.6	*Apply the multiplier to the metric of the Syntax 's'*

Figure 14 Metric interpolation

6.4.2 Spacing and Kerning

The last and the most appeasing approach depends on letter width interpolation, and thus it does not use predefined spacing between the letters. Therefore, this approach also introduces the spacing information back to the application.

Without adding an extra spacing between the letters, they perfectly touch consecutive letter's bounding box as seen in Figure 15.

Quick brown fox jumps over the lazy dog.

Figure 15 Using bounding boxes for spacing

By adding an arbitrary amount of space between the letters, the positioning of the letters become legible as follows:

Figure 16 Using bounding boxes with extra spacing

Although the overall positioning works for reading, the kerning information still does not exist. In order to simulate the left and the right bearing information, each letter combination is analyzed and these combinations are assigned either a positive or a negative value of extra spacing. This extra spacing information is added to the default one to compensate the negative space between the letters, such as the extra spacing that appears between o and x in Figure 16. Notice how this o-x pair or o-v-e letters appear to fill in the extra negative space in Figure 17.

Figure 17 Taking bearings into consideration

The last approach produces the most visually pleasing result. All these examples also demonstrate how the glyphs are generated every time they are typed, as each example feature a slightly different variations of the same letters, along the intermediate steps of interpolation between Syntax and Helvetica. Although it produces acceptable results for most of the letters, punctuations and numbers, the interpolation for the letter ‘g’ yields very awkward results.

Letter ‘g’ for Helvetica and Syntax are fundamentally different as syntax has a double-story armature whereas Helvetica’s ‘g’s are single-story. As a result, the following interpolation in Figure 18 needs too many design decisions and the intermediate steps are too distracting to be characterized in any category.

Figure 18 Intermediate steps of the Franken-'g'

Therefore, as an exceptional rule, lowercase g is limited to either purely Helvetica or Syntax in order to solve this ambiguity.

6.5 Exporting Usable Text

The Frankenstein Application is an experimental environment to generate text with unique glyphs per each keystroke and not a full-fledged design environment. Therefore, the generated text should be exportable in a format that a standard design environment can interpret.

One of Processing API's pre-included libraries "PGraphicsPDF" is capable of drawing what's on the application window into a PDF[4] file in the form of a vector image.

Using this library, every letterform that is drawn on the screen is also drawn into the final pdf file. As the interpolation process is completed, the application also generates a pdf file that contains everything that had been interpolated so far.

Consequently, every piece of text that is written within the application is saved in PDF format as a vector image, which can be read and imported into all of the vector graphics applications for artistic use.

[4] Portable Document Format, PDF is an open-standard, operating system independent layout system for displaying and distributing documents and graphics.

7 The Output

Using the Frankenstein application generates the following text in Figure 19. The text, for now, cannot be easily inserted into a text file in editable text format. Due to the technical standardizations explained in previous chapters, the output text is an image file from the perspective of the computer and excludes the extra information a text field normally offers.

To give an idea, if each letter had only a hundred interpolated steps, then it would require using a hundred fonts, containing a complete set of fonts for each interpolation ratio. At each letter, the user would have to select a random font from one of the hundred interpolated fonts to roughly mimic what the Frankenstein application does within a fraction of a second.

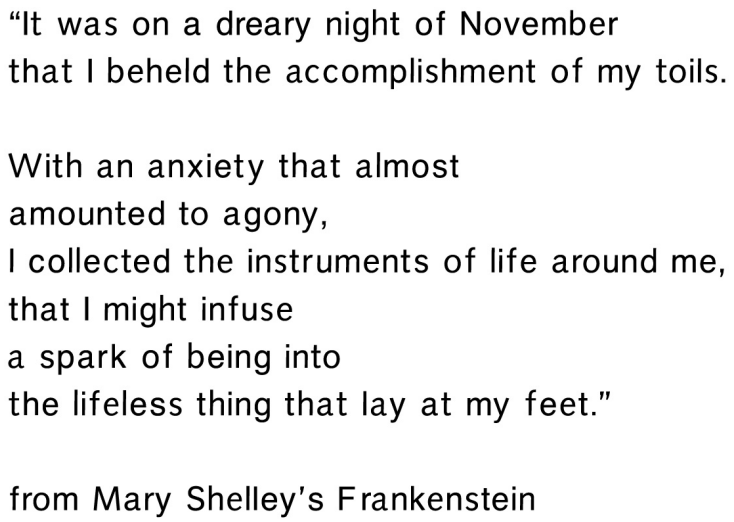


Figure 19 The Franken-text

The readability of the generated text shows that the reading process does not necessarily depend on the consistency of the letterforms, at least to the degree of an interpolation between a humanist and a grotesque font. The brain does not notice the subtle differences between the interpolated glyphs of a specific letter, possibly due to the fact that it recognizes the words as a whole, instead of registering each and every letter first.

8 The Survey

The proposed outcome of the Franken-thesis is a better understanding of the effect of varying glyphs on the legibility of a continuous text. Therefore the text generator, which is explained in detail in the previous chapters, is a necessary tool for the execution of the thesis, rather than the final product.

Consequently, the survey's aim is to see how preferable the Franken-font is, over a purely grotesque or a purely humanist font.

8.1 The Specimens

To display the properties of the fonts, the longer version of the passage in Figure 19, from Mary Shelley's Frankenstein is used.

The same piece of text is rendered in Helvetica, Syntax and the Franken-font. Since the nature of the Franken-font suggests that each instance of the same text would be different, more than one sample is used for the Franken-font specimen.

In addition, there is a special specimen of Franken-font, which uses 50% interpolation for every instance of the letters. In other words, this version of the Franken-font is static and the glyphs are halfway between grotesque and humanist. This specimen will be regarded as a separate font and it will be called Franken50 for avoiding ambiguities.

Consequently, there are four different fonts used for the specimens of the survey. Namely Helvetica, Syntax, the Franken50 and the Franken-font (see Appendix B). Among these specimens, only Franken-font has multiple specimens, since it is a dynamic font unlike others.

8.2 The Technique

The survey displays two renderings of the same text and asks the user to choose the one that looks more appealing in terms of the font used. Since the aim of the survey is to see how preferable the Franken-font is, one of the texts is always rendered in the Franken-font. In other words, the possible combinations are as follows:

- Helvetica vs. Franken-font
- Syntax vs. Franken-font
- Franken50 vs. Franken-font
- Helvetica vs. Franken50
- Syntax vs. Franken50

Before asking the participants to choose a text, the following information is asked to the test takers: how old they are, whether or not they are a designer and if they prefer reading from a book over reading from a screen or vice versa (see Appendix A). This information is later used to analyze how it affects the participants' preferences.

8.3 The Audience

The survey is executed online and it's executed globally over the Internet. The survey executed 521 times by unique participants. The results are retrieved 18 March 2013 from http://csc.im/frankensurvey2 and According to the questions asked at the beginning of the survey, the breakdown of the audience is as follows:

- The mean age is 26.94
    - The number of people younger or at the age 25 is 255
    - The number of people older than 25 is 266
- The number of designers among the participants is 217; which makes up 41.65% of the entire set.
- The number of people, who prefer to read from a book, is 315; which makes up 60.46% of the entire set. Consequently, the number of people, who prefer to read from a screen, is 206 (39.54%)

8.4 The Survey Results

The results of the survey show that, 45.9% of the 306 people, who were asked to choose between the Franken-font and Helvetica or Syntax, chose the Franken-font.

The percentage of people, who chose Franken50 over Helvetica or Syntax, is 62% of 146 people. As expected, the preference rate for the Franken-font shows that continuous text rendered in a dynamic font is not rejected, contrary to the classical notion of font design.

However, the interesting finding is that the preference rate for a static hybrid font is higher than both a pure grotesque and a pure humanist font. In addition, the Franken-font is generally more preferable to a pure humanist font, whereas a pure grotesque font is more preferable than the Franken-font.

8.4.1 Result Breakdowns

Although the results are consistent in general, analyzing the statistics depending on the age, profession and medium preference yield results with notable variations.

The following subsections focus on results divided based on the aforementioned groups of the audience. These groups are not mutually exclusive (i.e. there are overlapping participants in different groups), which makes it possible to do further analysis on finer subgroups.

8.4.1.1 Age-Based Analysis

The first of the three grouping methods is age-based division. The 521 test takers are divided into two by their age; people who are at the age 25 or younger and people who are older than 25. The reason for dividing the participants at the age 25 is that it is where these two groups have the closest number of people (i.e. 255 and 266 respectively).

As seen on the bar chart in Figure 20, the Franken-font preference rate over Helvetica is lower on both groups; 28% for people younger than 25 and 37% for people older than 25%.

The preference rate against Syntax, however, is above 50%. The preference rate is higher for older people, at 64% and 57% for young people.

When compared to its static counterpart, Franken50, the acceptance rate for the Franken-font is lower in both groups: 47% for young people and 39% for older people.

Franken50 scores better than the Franken-font in general: a 36% acceptance rate over Helvetica among younger people and 58% for older people. This rate increases even more against Syntax, with a 71% rate for younger people and 85% for older people.

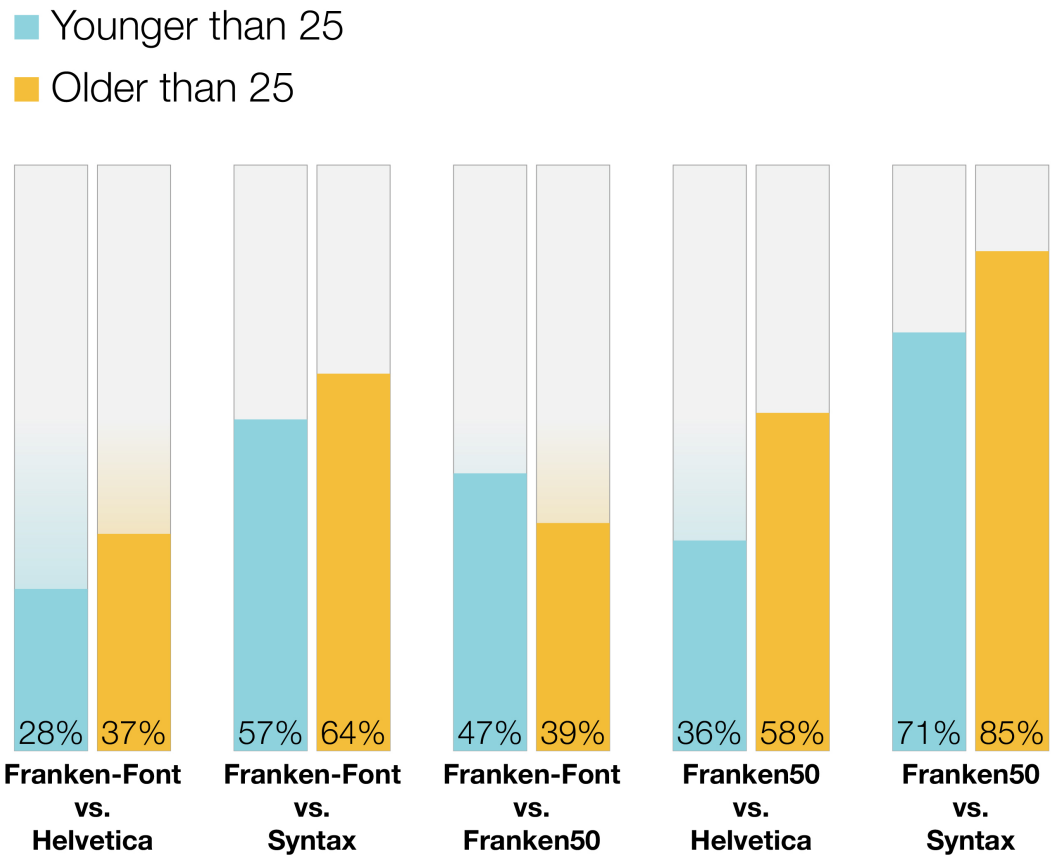


Figure 20 Age-based result analysis

8.4.1.2 Profession-based analysis

The second of the three grouping methods is profession-based division. The participants are divided into two groups, depending on whether or not they are a designer. The amount of designers among the test takers is 207, which makes up the 41 percent of the entire group, along with the other 314 non-designer test takers.

Consistent with the overall results, both groups display almost no rejection against the Franken-font, at 45% and 46% among non-designers and designers respectively.

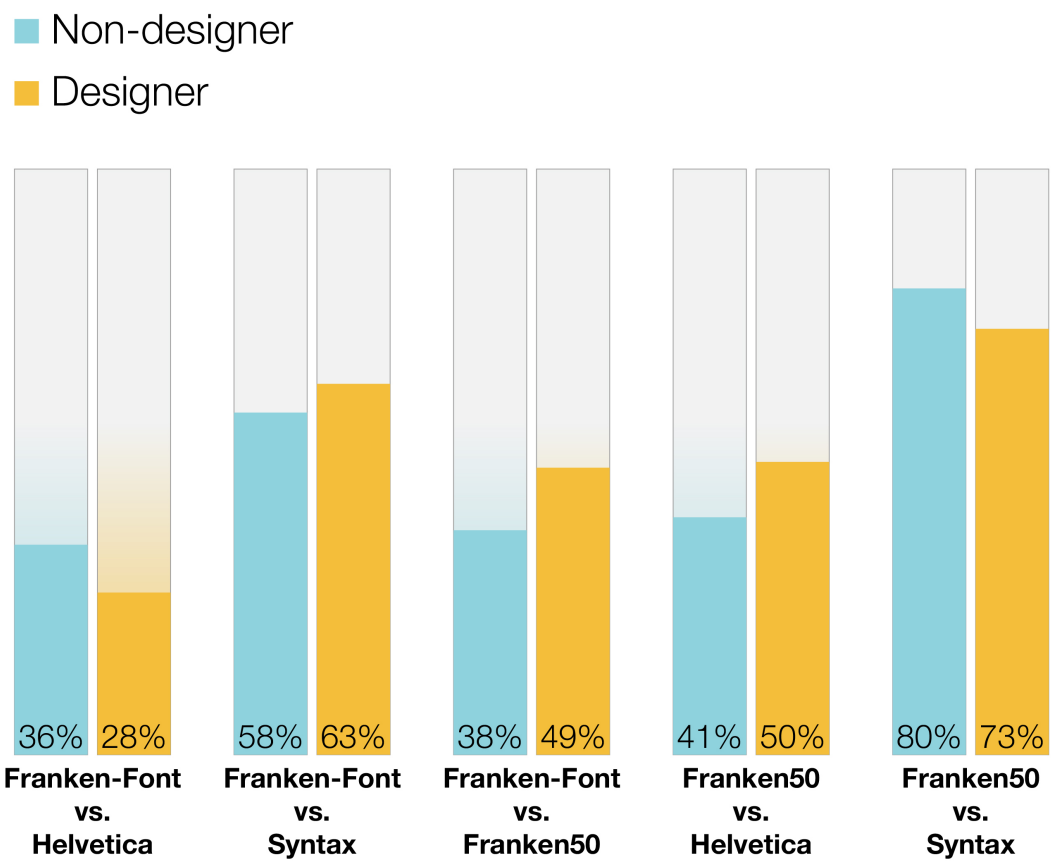


Figure 21 Profession-based result analysis

The Franken-font preference rate over Helvetica is lower on both groups: 36% and 28% for non-designers and designers respectively.

The preference rate against Syntax is again above 50%. The preference rate is higher for non-designers, at 58.4% and 63.3% for designers.
When compared to its static counterpart, Franken50, the acceptance rate for the Franken-font is lower than the average at 38.3% for non-designers and 49% for designers.

Franken50 again scores better than the Franken-font in general; with a 40.5% acceptance rate over Helvetica among non-designers and exactly 50% for designers. This rate increases even more against Syntax, with a 79.6% rate for non-designers and 72.7% for designers.

8.4.1.3 Medium of reading based analysis

The last of the three grouping methods is medium of reading based division. The participants are divided into two groups, depending on whether they prefer to read from a book or a digital screen. The amount of book readers among the test takers is 315, which makes up the 60.5 percent of the entire group, along with the other 206 screen reader test takers.

Again consistent with the overall results, both groups display almost no rejection against the Franken-font, at 44.4% and 46.7% among book readers and screen readers respectively.

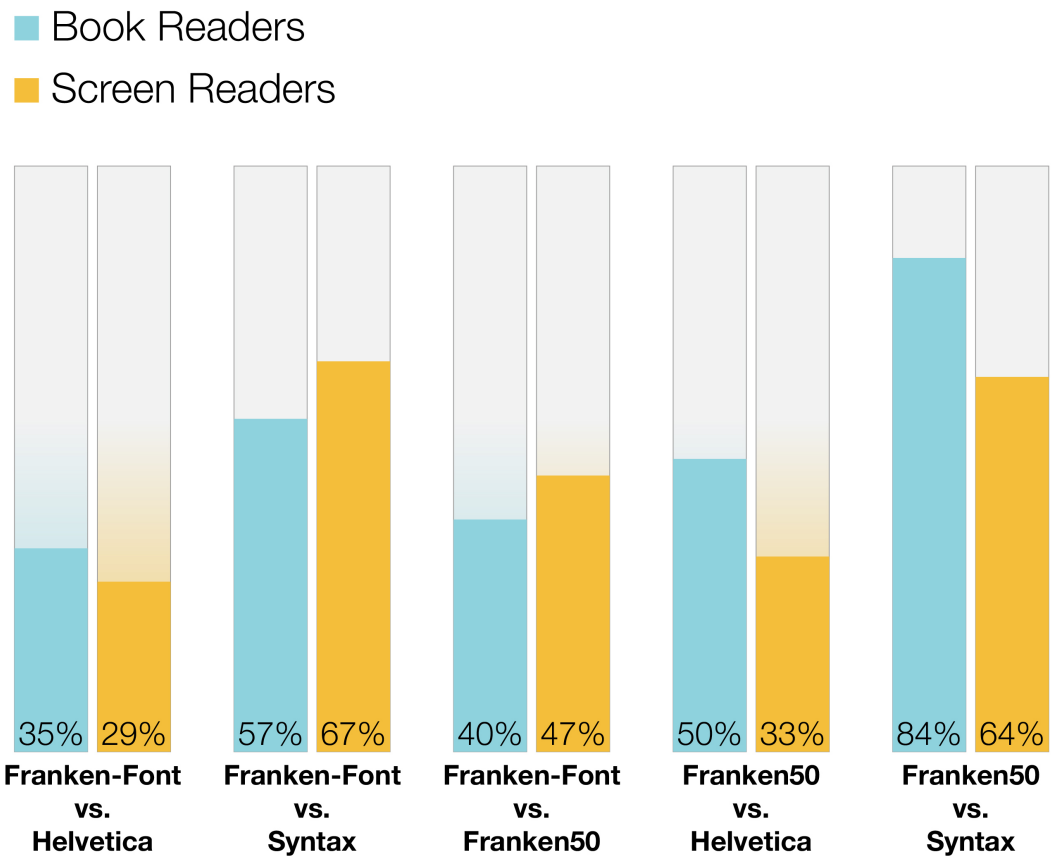


Figure 22 Medium-based result analysis

The Franken-font preference rate against Helvetica is lower on both groups; 34.7% and 29% for book readers and screen readers.

The preference rate against Syntax is again above 50%. The preference rate is higher for screen readers at 66.7% and 56.8% book readers.
When compared to its static counterpart, Franken50, the acceptance rate for the Franken-font is lower than the average at 39.7% for book readers and 47.2% for screen readers.

Franken50 scores better than the Franken-font in general: a 33.3% acceptance rate against Helvetica among screen readers and exactly 50% for book readers. This rate increases even more against Syntax, with an 84.3% rate for book readers and 64% for screen readers.

8.4.1.4 Joint analysis

Although the results in every breakdown are more or less homogeneous and predictable as seen above, some combinations are worth noticing due to their marginal values for the Franken-font and Franken50 preferences.

When compared to Helvetica, the Franken-font scores a 40.7% preference rate as a peak, for the group of book reading, non designer 27 individuals, who are older than 25.

Preference rate drops to as low as 18.8%, among the 16 book reading, designer individuals, younger than 25.

The Franken50 font scores as low as 27.3% and 37.5% for 11 screen reading non-designer individuals and 8 book reading designer individuals younger than 25, respectively; unlike its high preference rates in general.

The Franken-font's preference rate against Syntax peaks as high as a 100%, among 10 screen reading individuals, who are non-designer and younger than 25 and 82.35% for 17 book reading, designer individuals older than 25.

Franken50 also goes up to a 100% for book reading, non-designer individuals older than 25 and scores a 95.7% preference rate, both for the 23 non-designer individuals older than 25 and 23 book reading individuals older than 25.

8.5 Further analysis

The results of the survey show that:

- The readers do not display a marginal rejection or acceptation for the Franken-font. Although, the static version Franken50's acceptance rate is slightly higher than that of the dynamic version in most of the cases.

- A pure grotesque font is almost always more favorable than the Franken-font.

- A pure humanist font is almost always less favorable than the Franken-font.

- The previous two statements show that preference is not directly based on whether or not a font is dynamic or static. In other words, a dynamic font is equally favorable as a static font.

- The reason for the Franken-font to be preferred over a humanist font might be explained by an analogy of the "Uncanny Valley " theory.

  In robotics "subtle flaws in appearance and movement only seem eerie in very humanlike robots. This uncanny phenomenon may be symptomatic of entities that elicit a model of a human other but do not measure up to it. *** Uncanny robot elicits an innate fear of death and culturally-supported defenses for coping with death's inability." (MacDorman, 1). [11]

Just as the human visual perception is confused when it tries to comprehend the visual hints from what seems to be a human but in reality is not, the humanist font might possess a similar quality with a humanoid that falls within the "uncanny valley". Humanist fonts' armature imitates the wrist movement in hand writing. Yet in handwriting, the letters are never the same. Instead, they are consistent in each person's own way of writing. Therefore, the human brain might in fact expect a level of irregularity in a written text, given that the letterforms seem to be the result of a wrist movement and prefer the Franken-font to a static, humanist font.

- Helvetica is always the preferred font when compared to the Franken font derivatives (i.e. the Franken-Font and Franken50). This can be explained in the light of Yazıcıgil's survey results in *Humanist Versus Grotesque Sanserif*, which shows that people who prefer screen as the reading medium show an inclination towards a grotesque font. As the Franken-survey is executed online and therefore on screen, this can result in an eventual preference towards the grotesque specimen, Helvetica.

- When compared to the results of the Franken-font, Franken50's preference rates are distinctively higher than those of the Franken-font. This shows that although a dynamic font is not rejected, its static counterpart is still more preferable. This might be a result of a lot of factors; one being the fact that the interpolated fonts are originally static fonts, which are not expected to be dynamically created.

  The results might be radically different for root fonts, such as interpolating two different fonts from script font family or interpolating fonts that are specifically designed to be interpolated; much like an ingredient rather than a finished font (such as Helvetica and Syntax).

9 Conclusion

The results of the survey show that a font such as the Franken-font, which dynamically creates letterforms for each occurrence, is as preferable as a classical, static font such as Helvetica or Syntax. Especially when compared to Syntax (a humanist font), the Franken-font and Franken50 scores almost always above. This can be associated to a number of reasons such as the medium of the survey being digital or the fact that a dynamic font is more suitable to have humane features, because the armature of a humanist font is derived from the wrist movement and handwriting is by nature dynamic (i.e. random yet based on rules for each handwriting).

The positive outcome of this experiment suggests that a dynamic typing environment is possible both in terms of technical feasibility and user acceptance.

Although the generation is limited to a humanist and a grotesque font for this specific example, the system is virtually capable of interpolating an arbitrary number of fonts with any kind of form; including letterforms written by human hand and then reproduced in vector format. Consequently, it is possible for the user to prepare an arbitrary number of letter specimens of their own handwriting, use them as the input and create a dynamic font that generates interpolations of these specimens to imitate real handwriting.

Eventually, the next step in the Franken-font project is to design a new dynamic font that is based not on existing commercial fonts, but authentic handwriting and explore the possibilities of creating letterforms that might truly imitate the aesthetics of the owner of the input letterforms.

9.1 Possible Uses

In addition to the main purpose of imitating authentic handwriting, the underlying system does not necessarily have to be used for this purpose only.

As stated in earlier chapters, the Franken-font interpolates matching letterforms to create intermediate glyphs between Helvetica and Syntax. Yet, the fact that this interpolation takes place only once per letter or that it uses letterforms to do this interpolation are arbitrary constraints on the mechanism to better suit the needs of this project. Consequently, subtle changes in the algorithm can make it possible to cover a wider range of uses.

The first constraint to be tweaked is the frequency of the interpolations. For the purpose of creating the letterforms, the system generates an intermediate glyph once and uses it in the rendered text. This is necessary, unless the glyphs are free or even required to change their forms during the runtime[5] of the modified Franken-font application. Since the purpose of the modified executable is no longer to create the Franken-font, it will be referred just as the application from this point on.

An example for a use case that requires morphing letterforms would be an animated child e-book, where the words are animated per letter to visualize specific events or to emphasize certain ideas in the text, such as letters randomly morphing back and forth between their distorted versions to create the effect of "shivering" to emphasize a context that is focused on the abstract notion of coldness.

The next constraint that can be explored and enhanced is the pattern of animated interpolations. In order to randomly interpolate the letterforms, the application first generates a random number between 0 and 1 and then uses this value as the weighted

[5] Runtime of an application is the entire time that spans between the initial execution and the termination of a computer application.

mean to be used for the interpolation (e.g. 0.4 means 40% of the first form and 60% of the second form). Naturally, this number generation does not have to be random. It can be bound to a predefined pattern (e.g. beats of a tune) or mathematically defined (e.g. the sine curve). Binding the generation to a pattern would enhance the animation possibilities to cover time-bound events such as expanding and contracting, "breathing" or rippling letterforms. Although impossible to print, such modification would make it possible to create digital posters or titles, which utilizes this controlled animation infrastructure to emphasize the content (e.g. beating letterforms on a poster emphasizing cardiac health)

Last but not least, the modification of the application does not necessarily have to be at the algorithmic level. Changing just the input forms can drastically change the nature of the end result. The system, for instance, can be used purely for artistic purposes to render a text in a way that appropriately reflects the required aesthetics.

As a real life example, the Franken-font application is used in the artwork *İhlal* (2012) by Özlem Alkış, which is exhibited in the Commons Tense / *Commons Tijd / Müşterekler Zamanı* exhibition by Amber Platform. [12] *İhlal* visualizes the Human Rights Constitution in partially distorted letterforms, each article distorted in correspondence to how many times it is violated. The figure 23 below shows the Article 18 from the artwork.

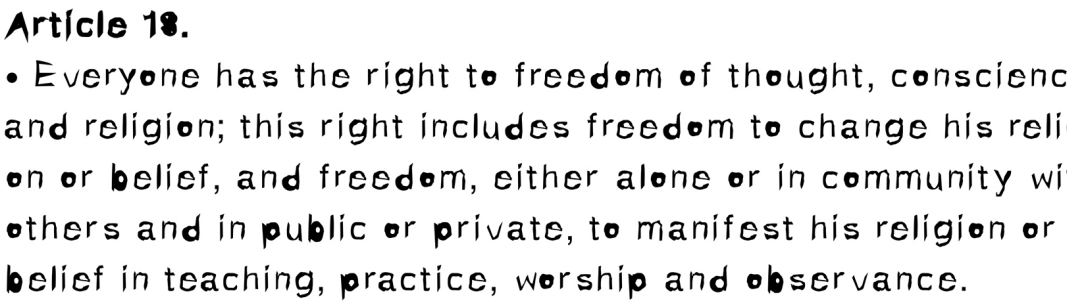


Figure 23 Article 18 from "İhlal"

Consequently, even though the existing Franken-font application is a prototype that is designed to test the usability and feasibility of a possible, dynamic handwriting font, subtle modifications on the underlying infrastructure widens the range of possible outcomes. This, by extension, makes the Franken-font a potentially flexible system that can address many previously impossible design challenges.

References

[1] Noordzij, G. (2005). *The stroke: theory of writing*. London: Hyphen Press.

[2] Zapf, H. (2007). *Alphabet stories: a chronicle of technical developments*. Rochester, N.Y.: RIT Cary Art Graphics Press.

[3] Bringhurst, R. (2002). *The Elements of Typographic Style* (v2.5). Vancouver: Hartley & Marks Publishers.

[4] Hill, B (1999). The Magic of Reading.

[5] "Haacke's Condensation Cube: The Machine in the Box and the Travails of Architecture," Thresholds 30: Microcosms (Summer 2005): 99-103.

[6] Jacob, H. ed., Beatrice Warde, The Crystal Goblet: Sixteen Essays on

[7] Roboto is a Four-headed Frankenfont. (n.d.). *Typographica RSS*. Retrieved June 12, 2014, from http://typographica.org/on-typography/roboto-typeface-is-a-four-headed-frankenstein/

[8] Yazıcıgil, O. (2014). Yazı sadece Okumak için değildir. Natama - Hayat Memat Dergisi, 6.

[9] Yazıcıgil, O. (2009). *Humanist Versus Grotesque Sanserif,* MFA Thesis. Purdue University, Indiana.

[10] Fry, B. and Reas, C. (2014) Processing (Version 2.0) [Computer program]. Available at http://processing.org

[11] Hanson et al. (2012) Upending the Uncanny Valley

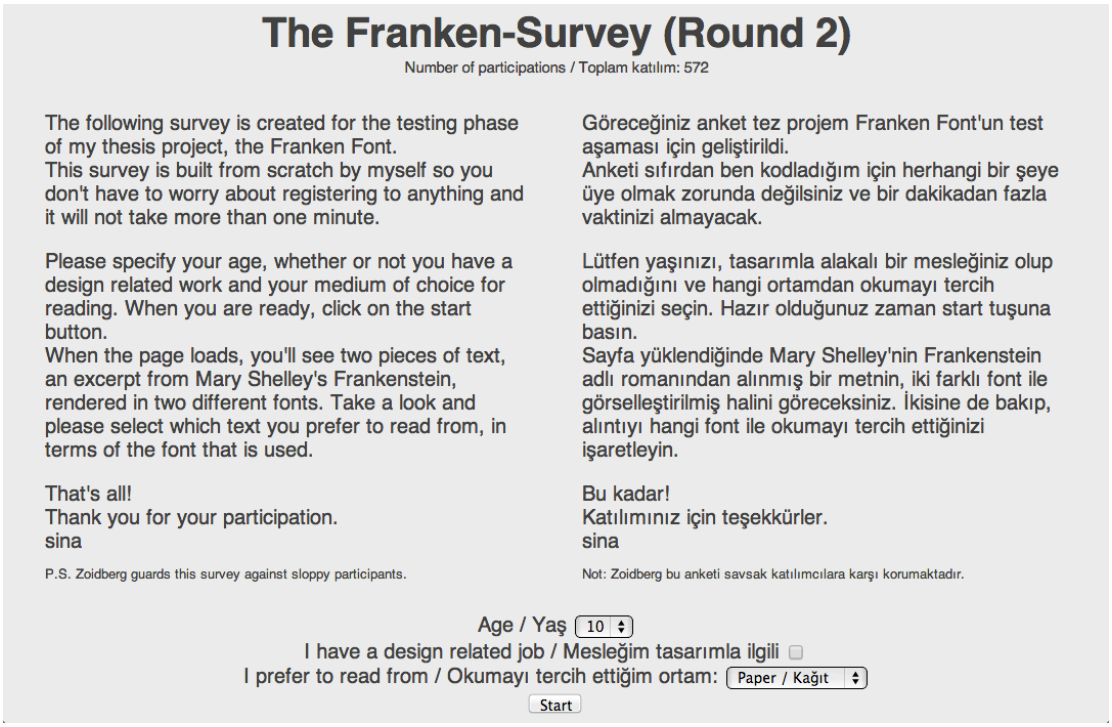
Typography, Sylvan Press, London, 1955.

[12] Commons Tense / Commons Tijd / Müşterekler Zamanı. (2012, September 15). *Amber Platform* |. Retrieved June 12, 2014, from http://www.amberplatform.org/tr/news/detail/2012/09/15/commons-tense-commons-tijd-musterekler-zaman/23

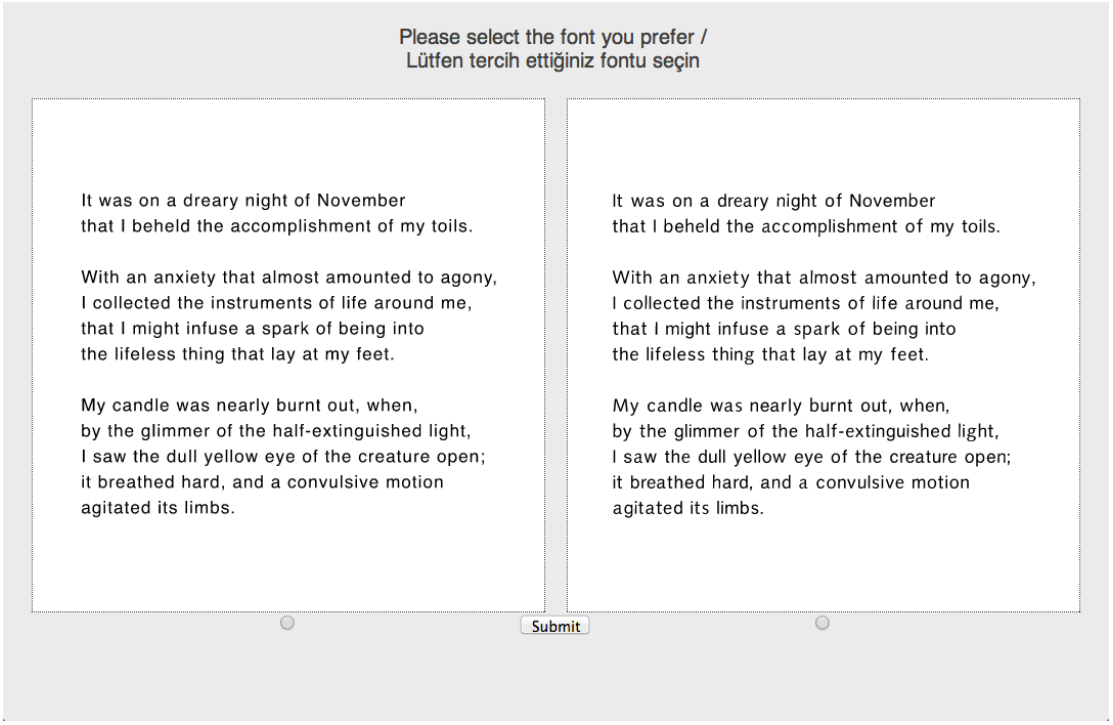
Appendices

Appendix A – Online Survey

Welcome page of the survey



Sample selection page



Appendix B – Survey Specimens

Helvetica Specimen

It was on a dreary night of November
that I beheld the accomplishment of my toils.

With an anxiety that almost amounted to agony,
I collected the instruments of life around me,
that I might infuse a spark of being into
the lifeless thing that lay at my feet.

My candle was nearly burnt out, when,
by the glimmer of the half-extinguished light,
I saw the dull yellow eye of the creature open;
it breathed hard, and a convulsive motion
agitated its limbs.

Syntax Specimen

It was on a dreary night of November
that I beheld the accomplishment of my toils.

With an anxiety that almost amounted to agony,
I collected the instruments of life around me,
that I might infuse a spark of being into
the lifeless thing that lay at my feet.

My candle was nearly burnt out, when,
by the glimmer of the half-extinguished light,
I saw the dull yellow eye of the creature open;
it breathed hard, and a convulsive motion
agitated its limbs.

Franken 50 Specimen

It was on a dreary night of November
that I beheld the accomplishment of my toils.

With an anxiety that almost amounted to agony,
I collected the instruments of life around me,
that I might infuse a spark of being into
the lifeless thing that lay at my feet.

My candle was nearly burnt out, when,
by the glimmer of the half-extinguished light,
I saw the dull yellow eye of the creature open;
it breathed hard, and a convulsive motion
agitated its limbs.

A Franken-font Specimen

It was on a dreary night of November
that I beheld the accomplishment of my toils.

With an anxiety that almost amounted to agony,
I collected the instruments of life around me,
that I might infuse a spark of being into
the lifeless thing that lay at my feet.

My candle was nearly burnt out, when,
by the glimmer of the half-extinguished light,
I saw the dull yellow eye of the creature open;
it breathed hard, and a convulsive motion
agitated its limbs.

Appendix C – Franken Font Possible Sets

The following sets are 5 consecutive, random generations of the Franken-font.

the franken-font

Aa Ee Mm Ss Rr

a b c d e f g h i j k l m

n o p q r s t u v w x y z

0 1 2 3 4 5 6 7 8 9

. , ; ? !

the franken-font

Aa Ee Mm Ss Rr

a b c d e f g h i j k l m

n o p q r s t u v w x y z

0 1 2 3 4 5 6 7 8 9

. , ; ? !

the franken-font

Aa Ee Mm Ss Rr

a b c d e f g h i j k l m

n o p q r s t u v w x y z

0 1 2 3 4 5 6 7 8 9

. , ; ? !

the franken-font

Aa Ee Mm Ss Rr

a b c d e f g h i j k l m

n o p q r s t u v w x y z

0 1 2 3 4 5 6 7 8 9

. , ; ? !