FÁBIO PEREIRA
TIAGO MARTINS
SÉRGIO REBELO
JOÃO BICKER

Web-based Tool for the Generation of Letterforms

The history of graphic design reveals a continuous reinvention of the discipline with the emergence of new technologies, providing means to new ways of designing and paving the way for new and unexpected design experimentation. For instance, computational generative processes are being employed in the development of various types of design projects, including dynamic visual identities, generative poster and cover design, and generative type design.

We present a generative web-based tool capable of automatically generating letterforms or glyphs. Each glyph is created by filling a pre-extracted typographical skeleton using an algorithmic drawing technique. Each drawing technique provides a unique visual style to the generated glyphs. The user can configure different parameters specific to each drawing technique and this way achieve a wide variety of glyphs consistent with her/his visual preferences. The presented tool, which is online at cdv.dei.uc.pt/2019/letterspecies, can generate the glyphs in real-time, allowing the fast visual feedback of any adjustment in the parameters. At any moment, the user can export a font file, which can be used in most software design tools or share a link to the editable configuration of the font.

GENERATIVE DESIGN TYPE DESIGN WEB SYSTEM TYPOGRAPHY

1. INTRODUCTION

The history of graphic design reveals a continuous reinvention of the discipline simultaneously with the emergence of new technologies. Throughout the years of technologic evolution, two important inventions defined the designers' practice: the personal computer providing computation to the masses and the Internet connecting people and information on a large scale (Armstrong, 2016).

> *"The personal computer morphed into a large networked mind through which creatives could think, make, collaborate, and distribute. Users commonly experienced content through active engagement online: pressing a button, scrolling down a page, uploading content, customizing interfaces. Interactivity took over."*
> (Armstrong, 2016, p.15)

As Helen Armstrong reports, these two tools combined provided the means to a new way of thinking and paved the way for new and unexpected explorations in the field. For example, graphic designers have begun to explore generative and algorithmic processes in the development of various types of design projects, such as dynamic visual identities, adaptive interfaces or generative typography. This type of process allowed to explore new approaches to graphic design, giving rise to new experimental and unique solutions (Armstrong, 2016).

In the context of type design, the introduction of new technologies facilitated the laborious process of creating, modifying and experimenting a font. However, Gerard Unger states that "while the technology for type design and the design process have changed fundamentally, letterforms have changed very little with the transition from tangible and analog type to digital and immaterial types". Besides, Sharon Poggenpohl remind us of the realm of possibilities brought by an interactive computer system that can execute operations such as "...fragment, distort, incline, change weight, etc." on type with simple directions. Therefore, there is space to an investigation in this field, adopting an experimental approach to the process of type design creation with an interactive system that give us a wide range of possibilities to make room for new forms and shapes for letterforms (Poggenpohl, 1983; Unger, 2018).

In this context, our work seeks to create synergy between type design and generative processes, through an algorithmic web system capable of generating glyphs from typographical skeletons and drawing techniques. The typographical skeleton consists of a structure that represents the fundamental

shape of a specific glyph. A drawing technique is a combination of one or more shapes. Each drawing technique is different and has its characteristics. In addition, each drawing technique is parameterizable. Thus, the user can configure a set of parameters that are inherent to the drawing technique and change the appearance of glyphs. In this platform, the user can visualize, in real-time, the impact of each parameter on the aspect of the glyphs. The set of these glyphs forms a font that can be later exported by the user to use in various applications.

With this work, the main contribution created is an algorithmic web system that allows the generation of glyphs with wide visual diversity in a coherent way. Other contributions include: (i) expediting the laborious process of drawing the glyphs that form a font; (ii) exploring a new approach in the type design scenario; (iii) investigation of how an algorithmically generated font affects legibility and readability; (iv) practical tool for designers and others interested, in which the generated fonts can be used in the most diverse applications.

This paper is structured into five sections. The first section, Introduction, presents the idea of this work and the main contributions achieved. In the second section, Related Work, an analysis is made on works and experiments made in the field of generative processes in type design. In the third section, System, the proposed system is described as well as its operation. The fourth section, Testing, presents an analysis of the tests performed. In the fifth and last section, the Conclusions and Future Work are presented.

2. RELATED WORK

Algorithmic processes have been explored in numerous areas to solve specific problems and eventually create innovative solutions. Consequently, these new solutions have been pushing the boundaries of the innovation and originality of algorithmic approaches. Thus, we present an analysis of web systems and works that make use of algorithmic approaches to create fonts, which may influence our work.

Ntype is a typographical web communication tool for visualizing the 4th dimension of letterforms. It was developed by Kevin Zweerink in 2015 and consists of exploiting extrusions on letterforms according to the user given parameters such as the axes of rotation, speed of rotation, trail length, draw shape only, trail only, or both. Uses WebGL and Three.js to render letterforms and opentype.js to export them as OTF type fonts. It also allows sharing the status of the rotations via Uniform Resource Locator (URL) (Zweerink, 2015a, 2015b).

Phase is a system that explores the concept of generative type design, conceptualized by Elias Hanzer and developed by Florian Zia. It consists of choosing

a module from among the many to choose, and subsequently parameterizing the associated characteristics. This system uses variable font technology and can be manipulated in real time by parameters and also by sound input, which allows an infinite number of shapes. It also allows changing the parameter values at a random way and exporting the current font as a TTF file (Hanzer, n.d.).

Metaflop is a web application for custom font creation developed by Marco Müller and Alexis Reigel. It consists on choosing one of the predefined fonts, and further parameterizing of the inherent anatomical characteristics. For this purpose, the system uses the Metafont language. It gives you access to a *layout* of the parameters as well as a preview of the chosen glyph, a graph of all glyphs and an area where you can write customizable text. It allows changing the parameter values at a random way (flop it), returning to the previous state of the values (undo), returning to the initial state (reset) and also choosing how to change the values, either by sliders or numerical input. In the end the font created can be exported as OTF file or as WOFF file and sharing it via Twitter, Facebook, email and URL (Reigel & Müller, 2012).

Prototypo is a web application that uses parametric font technology to create fonts with numerous variants. It consists on choosing one of the model fonts, and then modifying the characteristics such as x-height, font width, contrast, axis, serifs, among others with a set of sliders that represent these parameters. It provides a preview of these modifications in real time. After the basic choices to set the model font it also lets you edit glyphs individually to add more detail or correct imperfections. Also provides the possibility to create various weights and it is still possible to export the font, however you have to pay a subscription to do so (Mathey, 2009).

In the context of algorithmic approaches applied on type design there is also important explorations made throughout the emergence of new technologies. One of the most remarkable work made on this field was the radical font “RandomFont” made by Erik van Blokland and Just van Rossum in the 1990s. The designers’ duo discovered a way to modify the PostScript code of fonts, and consequently created the *Beowolf* font. This font has its particularities, when sent to print, it randomly alters the dots of each letter to eventually look deformed and consequently takes a shape that is never the same. With the original name “RandomFont”, the font caught the attention of FontFont, which ended up being marketed with the name FF Beowolf. But this technique would eventually become obsolete as printers began to ignore the deformations with the new drivers and operating systems. However, with the advent of OpenType

technology, it gave new hope, and using preprogramed randomness they managed to give new possibilities for variations to the font. In 2011, it was selected, along with 22 more projects, for the permanent collection of the Museum of Modern Art in New York and was part of the installation "Standard Deviations" (Fontshop, n.d.).

Elien is a monospace generative font created with a system developed by Tatevik Aghabayann, in Processing. It uses metaballs containing up to five levels of circles arranged together that can create transitions when the distance between them is reduced. Therefore, the metaballs are applied to the skeleton of letterforms, forming connections between the circles of the same level and consequently between the circles of neighbouring letters. This system has a set of parameters that inform the visual shape of the metaballs such as the size (which is set randomly), the space between neighbouring letters, density, the number of metaballs applied, the size difference between the metaballs and the number of circles in each meatball. The created font contain up to 39 glyphs (Aghabayan, n.d.)

In 2009, Michael Flückiger and Nicolas Kunz developed for their thesis project a dynamic font called *Laika*. The duo believed that fonts that aren't made to be printed should be dynamic to match the media in which they were used. Thus, they created a system in which a font is able to change its shape and appearance at any time by reacting in real time to a wide range of inputs such as weight, contrast, serifs and axis. The potential of this font was demonstrated in the broad set of prototypes in which it could be used, such as receiving weather data from a certain city and adjusting its shape accordingly, for the stock market price or for emotional writing (Flückiger & Kunz, n.d., 2009)

In 2013, Catarina Maçãs developed a generative font capable of representing emotional values such as aversion, happiness, fear, anger, surprise and sadness through the content of a text. This font absorbs the characteristics of the text and thus provides a representation of the kind of emotion present in the text. In this work it was given importance to the representation of emotions and the distinction between them, being objective the individual interpretation of each user according to their visual culture and experiences. This work also resulted in an application that allows user interaction with the font generation. It is possible to submit a text to which the emotional analysis is made and then a pdf is generated, with the text composed in the generated font (Maçãs, 2013).

In 2018, Jéssica Parente developed a system that explored the possibility of generating fonts from the combination of layers (anatomic parts of the letters)

of different fonts. The layers consist on the separation of the different elements of the letter anatomy. Later these layers can be modified and combined to create a new structure and subsequently a drawing method is applied that uses shapes such as circles, squares, and triangles to generate new fonts (Parente, 2018).

3. SYSTEM

The basis of the presented work consists of an algorithmic system capable of generating glyphs and consequently a font formed by the set of all glyphs created. This process is achieved using generative drawing techniques to give shape to typographical skeletons. The system receives as input a typographical skeleton and a drawing technique chosen by the user, and as output, glyphs that can be visualized and configurable in real-time by the user. Each drawing technique is different and has its characteristics. Besides, each drawing technique is configurable. Thus, the user can configure a set of parameters that are inherent to the drawing technique to change the aspect of the glyphs.

To make this system a tool to help the process of type design, we developed a web platform that can be used by a wide range of users. In this platform, we give access to all the parameters that influence the visual aspect of the glyphs. Therefore, the user can visualize, in real-time, the impact of each parameter. In the end, the fonts can be exported by the user in OpenType, Scalable Vector Graphics (SVG) and URL format. This platform can be accessed at cdv.dei.uc.pt/2019/letterspecies and a demo video can be seen at vimeo.com/372217397

To build the system we used mainly JavaScript due to its qualities for web development. Also, we designed the system in a way that will be easy to add new typographical skeletons and/or drawing techniques further developed. The following subsections explain in detail: (i) the behaviour of the system, that is, how the system generate the fonts; and (ii) the graphical user interface and its features.

3.1. Behaviour

As stated previously, the system receives two inputs. The first input is the typographical skeleton and this is because it provides the main structure to work around as well as the legibility needed to generate the glyphs. The typographical skeleton can be seen as a structure that represents the fundamental shape of a specific glyph. It contains the essential typographical characteristics such as the anatomical elements (e. g., ascenders, descenders, diagonals, vertical and horizontal lines) and metrics (e. g., x-height, baseline, total height, and width).

In the programmatic domain, it consists in a sequence of two-dimensional points that define the lines that form the centre of a specific glyph shape. All the typographical skeletons are extracted from open-source fonts such as Google Fonts and the Velvetyne Type Foundry.

To calculate these skeletons, we built our algorithmic process based on the work made by Parente et al. (Parente, 2018; Parente, Martins, & Bicker, 2018). Thus, we applied the *Zhang-Suen* thinning algorithm to extract the fundamental lines from a binary image of a glyph by removing every pixel redundant (the farthest pixels to the centre of the shape) to the recognition of the glyph (Zhang & Suen, 1984). Therefore, we can achieve the typographical skeleton when no more unnecessary pixels can be removed. This process is performed for each glyph contained in an array of the most used glyphs. For each point of the skeleton, we have the x and y coordinates and a value that corresponds to the radius (distance to the closest point outside the shape) of a circle at the corresponding point (see Figure 1).

The process of extraction ends with a large number of points in the skeleton which, in some cases, provokes some imperfections when later we apply the drawing technique. To remove the redundant points (e.g., colinear points), we apply the *Ramer-Douglas-Peucker* algorithm that decomposes a given curve made by line segments into a simplified one, which means fewer points but structurally identical. The degree of simplification is measured by a value (typically named epsilon) that defines the maximum distance between the original curve and the simplified one. Therefore, this functionality increases on a large scale the variety of the visual aspect of the generated glyphs by changing the value of epsilon (Wu & Marquez, 2003).

FIG.1
Extraction of the typographical skeleton from the glyph “a”. From left to right: original glyph; calculation of the skeleton points; skeleton points; glyph recreation with the circles obtained at each point.

The system has a set of pre-calculated typographical skeletons to apply different drawing techniques. Whereas the skeleton is the basic structure of a glyph, a drawing technique can be seen as the “flesh” that gives shape to the skeleton. It provides the stroke style that forms the glyph. In visual terms, each drawing technique is different due to a combination of one or more shapes that can be geometric or abstract. In technical terms, consists of a programming

method that receives the data from the typographical skeleton selected and the values of the parameters related to the drawing technique selected. Also, each one has its parameters that influence the visual aspect of the glyph. This means that the user can customize the set of parameters at its will to achieve the result that like the most. Additionally, in the future, more drawing techniques can be easily added.

In Figure 2, it's possible to observe the glyph "a" rendered with different drawing techniques implemented on the web system. Some of the drawing techniques were developed to replicate particular visual characteristics. For example, the first "a" was generated using the drawing technique made to simulate calligraphy. In the case of the first "a" of the second row, the intention was to explore the digital environment and for that purpose, we use pixelated shapes to achieve that visual style. On other drawing techniques, we explore more experimental approaches to study the boundaries of legibility in type design and achieve new letterforms shapes.

FIG.2
Glyph "a" generated using different drawing techniques of the system on the same typographical skeleton.

3.2.1 Graphical User Interface

As mentioned before, we incorporated the system on a web platform to make it a usable tool for type design. To do so, we give to the user the controls over the configuration parameters that influence the visual of the glyphs, as well as the options to export the font generated. Figure 3 displays the overview interface of the web system. Besides the main bar, which has only functionalities that don't affect the generation of glyphs, the web platform consists of three main sections: the configurations panel on the left side, the preview area on the middle and the export panel on the right. These features will be described in detail on the subsections below.

3.2.2 Configuration

On the configuration panel, the system has all the controls that directly influence the glyph generation as well as the preview options. The first options menu has a set of typographical skeletons pre-calculated that the user can choose from.

FIG.3
Screenshot of the web system interface.

Then the user has the drawing techniques set which contain the previous drawing techniques referred and each one of the options has its parameters that are presented as input sliders. Therefore, the user has the freedom to explore, without the fear of making mistakes, the several variations that the glyphs can take for the visual aspect. In order to help the user to explore the possibilities for these variations we implemented three functionalities: (i) the "Default Values" button which reset all parameters to its default values; (ii) the "Random Values" button that changes all the parameters values in a random way within the respective scale with the purpose of getting surprising results; and (iii) the Audio Input button turns on a functionality which replaces the normal values with sound input values and consequently open space to large range of shapes for the glyphs (see Figure 4).

FIG.4
Screenshot of the web system interface with the configurations panel opened and the sound input on.

3.2.2. Preview

The purpose of the preview area is to provide a fluid interaction to the user by generating and rendering in real-time the glyphs with the parameter's values

defined by the user. This way, the user has feedback on the impact that each parameter has on the glyph visual aspect and gains enough knowledge to modify the values to find the best result for him. Also, this area gives the user the possibility to customize the rendered text. Besides that, the text it's adjustable by changing the size and the tracking values. The system also allows visualizing the typographical skeleton selected so the user can see the relation between the structure of the glyph and the drawing technique applied (see Figure 5).

FIG.5
Screenshot of the web system interface with the typographical skeleton on.

3.2.3. Export

The system has on the export panel the options to export the font in OpenType Font (OTF) format, share the font via URL, export an SVG file of the current text rendered and can add the font to a gallery (see Figure 6). To render the glyphs created, the system uses SVG which helps the process of integration with the opentype.js library to generate the font formed by all the glyphs in an OTF file (see Figure 7). The font created is open-source thus the user can use it on any project by installing it on any computer system. If the user finds necessity it can make refinements to the resulting font in any type designing program such as Glyphs. Also, the SVG file can be used in any project or artwork by using any program for vector editing. And last but not least, the functionality of adding the resulting font to a gallery performs to demonstrate the diversity of fonts that can be created with the web system. The font is displayed along with a button "Edit" that redirects to a new webpage with the font configuration (see Figure 8).

FIG.6
Screenshot of the web system interface with the export panel opened.

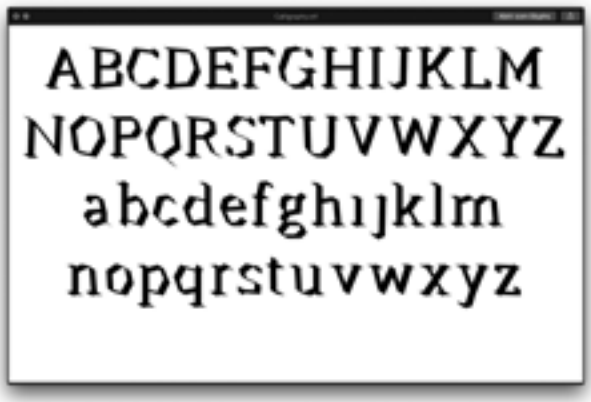

FIG.7
Screenshot the font generated by the system.

FIG.8
Screenshot of the web system interface with the gallery menu opened.

4. TESTING

After we implemented the system, we decided to test it with users to collect feedback that might help us to improve the web system. The tests were conducted taking into account the following goals: (i) the user understands how the system works; (ii) interface visually appealing; (iii) easy use of the system;

(iv) observation of visual diversity of the generated fonts; and (v) quality of the generated fonts. Therefore, we set up a stable version and test it with 15 users. The majority of the users were students and teachers of design and multimedia.

In each test we did a brief explanation of its purpose and what tasks the user had to complete. The list of tasks is presented in Table 1. These tasks were defined in a specific order so the user could interact with the system functionalities that we were evaluating. Also, at the end of the test, we asked the users to answer a short questionnaire about the tasks completed and if they had any suggestion/comment. Each answer was comprised within a range of 1 to 5 where 1 corresponds to "strongly disagree" and 5 correspond to "strongly agree". The questions list is presented in Table 2 as long as the average and the median of the answers to the questions. We calculate these both measures so we can analyse and compare the tendency with the actual results.

Task	Task Description
Task 1	Seek information about the system and read it
Task 2	Hide system information
Task 3	Change the color theme and keep it as you like
Task 4	Find the font settings
Task 5	Choose a typographical skeleton until you find one you like
Task 6	Choose a drawing technique until you find one you like
Task 7	Modify the parameters relative to the drawing technique chosen until the visual aspect of the font suits you (you can use random values and default values)
Task 8	Modify the text being displayed
Task 9	Visualize the lines of the typographical skeleton over the letters
Task 10	Change the lettering tracking
Task 11	Change the lettering size
Task 12	Export the font as an OTF file

TABLE 1
Table with the tasks list for the tests.

Question	Average	Median
I understood the operation of the system	4.8	5
I understood what a typographical skeleton is	5	5
I understood the way that the typographical skeleton is shaped	4.46	5
I liked the visual aspect of the interface	4.8	5
I found the interface easy to use	4.73	5
I found that the letterings generated are visually diverse	4.86	5
I found that the variation of the typographical skeleton (keeping the drawing technique) results in visually diverse letterings	4.26	4

I found that the variation of the drawing technique (keeping the typographical skeleton) results in visually diverse letterings	4.66	5
I liked the visual aspect of the generated letterings	4.8	5
It was easy to read the text in the lettering	4	4

TABLE 2
Table with the questions list and the corresponding average and median calculated from the answers.

Analysing the results, we can say that the goals defined before the test have been achieved. In what concerns the system functionality, the majority of the users found it easy to use and visually appealing. In terms of quality of the results, i.e., readability and visually diverse, the users with an average of 4.86 found the glyphs/fonts generated visually diverse. However, with the lowest average of 4, the users find some difficulties in reading the letterings with the generated glyphs. This may be caused by the fact that some configurations of the drawing technique parameters may generate glyphs with quite unusual shapes which makes more difficult to recognize the glyphs.

The feedback that we received from the suggestion/comment section helped us to improve the system and correct some minor errors. For example, the majority of the comments were related to the dropdown menus to choose the typographical skeleton and the drawing technique. This is because when the dropdown was opened there was no visual feedback indicating which option was selected which caused some frustration on the users. Other comments were related to: (i) colour inconsistency in the info menu compared to the rest of the website; (ii) imperfections on some drawing techniques related to horizontal and vertical strokes; and (iii) the preview area should allow multiple lines. We identified these as the most relevant comments to implement on the web system to improve it. Also, in an overall observation, we notice that most users were excited when interacting with the system and happy with the final font generated.

5. CONCLUSIONS AND FUTURE WORK

We have presented and tested a web system for the generation of glyphs by applying different drawing techniques to multiple typographical skeletons. The tests were made with 15 users in order to explore the system and improve it based on the feedback from the users.

From our point of view, the presented web system can be seen has a tool that supports type designers as well as graphic designers because it can generate

unexpected shapes making room for new explorations in the creative process. Also, it enables to represent data from specific contexts enabling a dynamic way of communication. For example, the system is capable of generating reactive fonts to sound by mapping the sound input to the drawing technique parameters. This proves the ability of the system to generate adaptive fonts that can be used in the most diverse media including new opportunities created by new media.

With the presented work we highlight the following contributions: (i) a system capable of generating fonts with coherent glyphs which have a wide range of visual diversity; (ii) a system that facilitates the process of creating all the glyphs needed to form a font which can be further refined; (iii) a practical web tool that enable the users to create costume fonts that can be used in specific projects or more experimental projects; and (iv) how an algorithmic approach in the type design field influence legibility and readability of the generated fonts. Future work will focus on: (i) improve the ability to compose the rendered text in multiple lines; (ii) implement more drawing techniques and correct possible imperfections; (iii) create a functionality that enables to export the rendered text in JPEG/PNG format; and (iv) implement the variable font technology in order to make possible to control the parameters values of the generated font on the design software's that support variable font technology.

ACKNOWLEDGMENTS

We would like to express our gratitude to all the students and teachers who have tested the presented system with excitement and willingness to help us to improve it.

This work is partially supported by national funds through the Foundation for Science and Technology (FCT), Portugal, within the scope of the project UID/CEC/00326/2019. The third author is funded by FCT under the grant SFRH/BD/132728/2017.

REFERENCES

Aghabayan, T. (n.d.). Gestalten mit Code, FH Mainz. Retrieved January 18, 2019, from *http://generative-typografie.de/generativetypografie/elien/*

Armstrong, H. (2016). Digital Design Theory: Readings From The Field. Princeton Architectural Press.

Flückiger, M., & Kunz, N. (n.d.). LAIKA – a dynamic typeface. Retrieved January 18, 2019, from *http://www.laikafont.ch/*

Flückiger, M., & Kunz, N. (2009). LAIKA. Retrieved January 18, 2019, from *https://vimeo.com/6993808*

Fontshop. (n.d.). FF Beowolf. Retrieved January 17, 2019, from *https://www.fontshop.com/families/ff-beowolf*

Hanzer, E. (n.d.). Phase. Retrieved January 15, 2019, from *https://www.eliashanzer.com/phase/*

Maçãs, C. (2013). Comportamentos da Tipografia Generativa: Uma Proposta para um Tipo Generativo. Universidade de Coimbra.

Mathey, Y (2009). Prototypo. Retrieved September 20, 2001, from *https://www.prototypo.io/*

Parente, J. (2018). Desenho Generativo de Tipos de Letra.

Parente, J., Martins, T., & Bicker, J. (2018). Generative Type Design. Proceedings of the Ninth Typography Meeting (9ET).

Poggenpohl. (1983). Creativity and Technology. STA Design Journal, 14–15.

Reigel, A., & Müller, M. (2012). Metaflop. Retrieved January 16, 2019, from *https://www.metaflop.com/*

Unger, G. (2018). Theory of Type Design. nai010 publishers.

Wu, S.-., & marquez, m. r. g. (2003). A non-self-intersection Douglas-Peucker algorithm. In 16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003) (pp. 60–66). *https://doi.org/10.1109/SIBGRA.2003.1240992*

Zhang, T. Y., & Suen, C. Y. (1984). A Fast Parallel Algorithm for Thinning Digital Patterns. Commun. ACM, 27(3), 236–239. *https://doi.org/10.1145/357994.358023*

Zweerink, K. (2015a). NType. Retrieved December 9, 2018, from *https://github.com/kevinzweerink/ntype*

Zweerink, K. (2015b). NType. Retrieved December 9, 2018, from *https://experiments.withgoogle.com/ntype*