TypeAdviser: a type design aiding-tool

João M. Cunha, Tiago Martins, Pedro Martins, João Bicker, Penousal Machado

CISUC, Department of Informatics Engineering, University of Coimbra
{jmacunha,tiagofm,pjmm,bicker,machado}@dei.uc.pt

Abstract. The number of people who design typefaces has drastically increased in the last twenty years. However, not all typefaces work as they should, i.e., as a group of characters with shared attributes. We present a tool for helping type designers in their creative process, which explores the anatomic relations among characters of a typeface. Having computer-aided design as a goal, our tool helps in the early stages of designing a typeface by using semi-automatic letter-part sharing and allowing the users to compare their design with existing typefaces.

Keywords: Computational Design, Computer-Aided Design, Type Design

1 Introduction

In the last century there was a massive proliferation of typefaces, being difficult to know how many there are today [3]. The accessible type design tools and the ease of designing and releasing a typeface led to a huge number of typefaces, whose quality is quite diverse.

A typeface has a key role in terms of communication as can even change the intent of its message. However, its importance is often overlooked and the impact of the communication is negatively affected by choosing an unsuitable or even poorly designed typeface. As changing users' habits is something not feasible, we believe that we should instead focus on trying to improve the way users design typefaces. By doing this, we aim at increasing the overall likelihood of the production of better quality designs by both experienced type designers and users without a design background. One of the criteria to evaluate a typeface's quality is its *consistency* – the way typeforms match each other, i.e. how the characters are designed into a set of diversified and yet unified forms [2].

Having computer-aided design as a goal, we focus on helping the designer in the early stages of the design process by combining type design principles, derived during the research stage, with an exploration of the Creativity-Support Tools domain, using mixed-initiative approaches [15].

We present a prototype of a type design aiding-tool which is currently composed of two different components: *the Part-sharing* – based on the consistency criterion – and *the Adviser* – which uses Self-Organizing Maps to suggest similar designs to the user.

2 Related work

The development of type related digital applications is nothing new and the current approaches can be divided into different categories according to their authors/users or their end purposes. There are three main types of authors: designers, engineers and artists. This difference on background knowledge has a huge effect on the type of application: some are purely artistic; others are technical and often dependent on too rigid rules or structure, thus resulting sometimes in visually non-appealing outputs; and others are within the scope of software specifically developed for type design – none of which explores the domain of computational creativity to its advantage in a significant way. Being more interested in type design, we will avoid addressing applications which have a visual artifact as output and focus on those that have a font as final product.

CarveLCD [1] generates letters using a grid which defines the location of their outline-points. This grid is controlled by the user and its modification is applied to every character, maintaining some coherence among them. It often results in abstract shapes which are not easily read.

Other applications allow the user to change the value of pre-defined parameters which control a given characteristic in order to generate fonts (e.g. *Metaflop* [5], *Modular typographic generator* [8], *Typeconstructor* [10] and *Prototypo* [13]).

Schmitz [14] developed a tool which uses Genetic Recombination as a metaphor and allows the user to recombine a limited number of fonts generating hybrids. This generation is done by using their genes which store information about three features: font skeleton, line width and serif shape.

There are also some systems that use Evolutionary Computation for designing type. Some require user interaction (e.g. [11]) and others focus on being autonomous (e.g. *Evotype* [12] which generates glyph designs from scratch).

In spite of the existence of such applications and systems, not only none of them addresses all the issues we have previously identified but also their output is considered the final product – we act on the initial stage of the design process. Concerning the followed approach, most of the aforementioned applications rely on predefined typefaces or skeletons, whereas we aim to allow the user to input its own designs. Additionally, the majority of them presents as final output something that – as a text font – has low quality both in terms of legibility and coherence among characters.

3 Aiding type design: *Part-sharing*

Our tool aims at helping type designers by combining co-creativity with type design principles. The developed prototype is currently composed of two different components: the Part-sharing and the Adviser. The Part-sharing component is strongly based on type design principles and uses them to both help designer reduce time-consuming tasks and stimulate creativity.

1. Looking for consistency – By focusing on the principle of consistency, the first stage of the development of the Part-sharing component has three main

goals: identify the characteristics that make a typeface work as a whole; understand the reason for the existence of relationships among letters; and identify the possible patterns in their construction.

Our research approach consisted not only in bibliographic research but also in the conduction of interviews with type designers. These interviews allowed us to understand their way of working and the details they pay attention to when drawing a typeface. In addition, we also analyzed the different letters, in order to identify patterns in their construction and infer possible rules followed in the design of a text typeface. The collected material and reached conclusions were then used as a support and applied in the development of the application.

As observed in some of the projects presented in Section 2, it is possible to make a metaphor between typefaces and biology: different typefaces can be seen as different species as they have a different structure. On the other hand, as living creatures within the same species share a given karyotype, within a typeface the different letters also share a set of characteristics.

According to Meseguer [7], there are three distinct ways a designer can use to create a set of characters: (1) *step by step* – in which each character is divided in parts according to the calligraphic drawing and stays that way until the last stage of the drawing process; (2) *modular* – which is based on the repetition of shapes throughout the set of the characters (e.g. the stem of *i* is repeated on the *n* or *m*); (3) *shape derivation* – characters are drawn in a sequence and shapes are derived from others.

Both the repetition of modules and shape derivation have key roles in terms of giving coherence and maintaining the harmony among characters.

By drawing the first letters, it is already possible to foresee how the others will be as making a decision for one letter often affects the rest of the letters. The designer is then faced with the question of which elements of the drawn letters can be used in the new ones and which ones must be adjusted or modified.

In addition, it is possible to separate the characters into several groups based on their morphological similarity (e.g. in upper case the round shapes group is *O, Q C, G* and *S*; in lower case is *o, c* and *e*). This categorization is of great importance since similar structures can and should be designed as related forms [3]. Moreover, by firstly drawing one character of each group, it is possible to design the remaining characters with less effort and sparing time.

The process of designing in sequence was mentioned by all the interviewed designers when describing their way of working. They not only share the preference of drawing the lowercase first – upper case is much more limited in terms of creativity and differentiation – but they also have a set of favorite first letters.

Most designers start with the key letters which define the proportions and personality of the type, good starting points are the lowercase letters *a,e,g,n* and *o*. The letter *n* is normally one of the first, due to the fact that it is easy to obtain parts of the letters *h, m, u* and *r* from it. And with it, it is possible to define a lot of the rhythm and proportion of the typeface.

2. Our approach – Our initial ideia was to establish rules that would, in some way, allow to generate all the characters of a typeface based only on

some given by the user. The user would input some previously drawn vectorial glyphs and then identify their parts. After identifying the letter-parts, it would be possible to generate the rest of the letters. This generation would be based on the predefined rules and would use the details about the introduced glyph, given by the user. After the generation, the user would be able to change the generated shapes, in order to correct any imperfection, and afterwards export the results. The goal of this concept was to generate a font nearly ready to be used. The end version of the application works similar to what has already been described but its main goal is completely different.

As explained in Section 3 (*Looking for consistency*), a system with too rigid rules, would lead to something greatly repetitive and would probably result in something with a modular appearance. Such a product would go against one of our main goals: to have something serious in terms of typographic quality and useful as a text typeface. According to Goudy [6], the characters should be coherent among each other but at the same time each of them should have a quality of completeness and not seem to be made of pieces put together. Obviously, a modular-looking result would fall short of this idea. In addition, there is often the need of doing optic compensations before considering a typeface as final. These were not considered in the initial concept.

Another important issue is related to the introduced characters. As mentioned in Section 3 (*Looking for consistency*), there are some letters that allow to foresee how the rest of the font might look like (e.g. the *n* or *o*) – by partially defining the proportions, contrast, etc.. However, there are others that do not and are much more difficult to extract useful information from (e.g. *k*, *x* or *z*), consequently increasing the difficulty of generating the rest of the characters. Moreover, just a few letters are not enough to generate the all of the remaining characters, which denotes a clear problem in our initial concept.

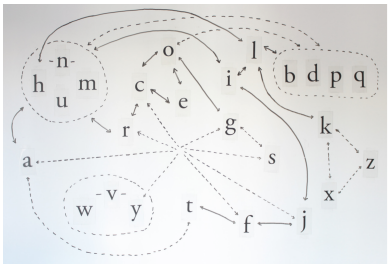

Fig. 1. Left side: Relationships among lowercase letters. The letters inside round shapes are greatly related; the lines symbolize a relationship between two letters; the dashed lines are less relevant relationships. Right side: Some of the skeleton characters.

For already mentioned reasons, the goal of the application was changed: instead of being considered as an application able to generate a complete and ready to use font, it should be seen as an aid-tool to the type designer, giving

support to the design task and stimulating creativity. It helps to reduce the time spent by the designers in the initial stage of the design process.

The way of working is highly similar to what has already been described. The user is able, at any time, to add new previously drawn glyphs and identify their parts. The shape-repetition is done automatically by the application. When the user is satisfied with the characters drawn using the tool, he can continue the design process with another software specially developed for type design, using the output of our application as a draft and an initial version of the typeface.

The current version is a prototype, not having all its functionalities fully implemented. In the part-sharing component, we have chosen to begin with the lowercase because it has a greater importance and usage – as already mentioned. Not only that but also all the interviewed type designers start the majority of their typefaces with the lowercase letters. The implemented rule system was based on the identified shape-sharing relationships among letters (see Fig. 1).

3. User input – The current version of the application was implemented using *Processing* and the library *Geomerative*. This library was used to read the introduced *svg* files and access the points and Bézier curves of the vectorial shapes. By introducing a vectorial file, the user is first asked to identify the letter it corresponds to. The user is then able to identify the several parts of the introduced glyphs. These parts are predefined in the application, i.e. the user has a list of possible parts to identify (based on the relationships, see left side of Fig. 1). This identification is done through point selection of all the points belonging to the part being identified. For example: having introduced a serifed glyph, the user can select all the outline points which are part of the serif and choose "serif" to save it as such. After identifying a letter-part, the application automatically checks which letters might use it in their construction and assigns it to them (see Fig. 2). The input is not limited: the user is able to input vectorial files for as many letters as he/she wishes and even replace a previously given one. This allows the user to work step-by-step in an incremental way.

4. Letter-part attribution and positioning – The attribution of the letter parts is done according to two criteria: *need* – does the letter need that part or does it already have one specially drawn for it? – and *suitability degree* – is there any other already defined letter-part which is more suitable or should it be replaced? For example, the top part of the l stem can be used to make an i stem but the top part of the j might be more adequate.

One of the issues that had to be solved was related to the positioning of letter-parts – i.e. how should a copied shape be positioned? The solution was to use a letter skeleton with the several letter-parts identified (see right side of Fig. 1). Contrary to some of the applications mentioned in Section 2, the skeleton is only used for letter-part positioning purposes. This is not an optimal solution as it does not account for every style/proportion. Despite this, we do not consider it a problem, as the user is able to move the letter-parts.

The positioning according to the skeleton is also not trivial, as the positioning method is not the same in all letter components. An example of this can be seen in the letters n and l. The left stem of the letter n can be positioned by aligning

it to the left with the corresponding skeleton part. Differently, the stem of the *l* is centered in relation to its skeleton. This makes it necessary to previously define the positioning method for each letter-part.

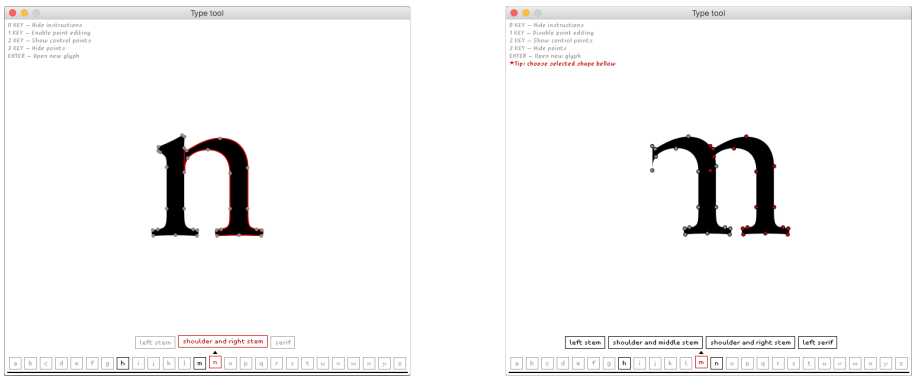

Fig. 2. Left side: Selecting letter-part. Right side: Letter *m* built using part-sharing

5. Editing shapes – In spite of not having a final typeface as a goal output nor aiming at developing an application to compete with software specialized in type design, we decided that the user should be able to edit the letter shapes. The importance of this functionality is even more evident when considering positioning. Therefore, the user is able to both move shapes, by selecting all their points, move single points or even their respective control points. This makes it possible to change the generated shapes in order to correct any imperfection and make slight adjustments (e.g. optical compensations).

4 Aiding type design: *Adviser*

The second component is used as a guide and its purpose is to present the user with possible letter designs, based on a user given glyph. The suggested designs are from already released typefaces that are similar in style to the one introduced by the user. This functionality is particularly helpful for non-experienced users and has two main possible uses: (i) it allows the user to see how the characters of similar-looking typefaces were designed, thus being useful to guide the user when designing a typeface of a specific style; on the other hand, (ii) it helps the user in distancing the typeface being designed from others, thus avoiding an excess of similarity of the output.

To implement the Adviser, we use a Self-Organizing Map (SOM) [9], which is a technique normally employed in data visualization, as it reveals relationships between vast amounts of data. It consists of self-organizing neural networks which require no external supervision and are able to represent multidimensional data in a space of a lower number of dimensions. The output is a map which groups similar data items together, displaying similarities.

A SOM is used to produce a map of different character styles for each letter, organized in terms of similarity. This makes it possible to group similar typefaces and find the ones visually near to a given one. Moreover, it also allows a gradual distancing from a particular style – without getting too far from what is established by tradition for that specific style of typeface. It is important to bear in mind that breaking with tradition when designing a typeface is often not a good choice as it might have a negative effect on legibility and easiness of reading [4]. By using SOMs, a user can start by drawing one letter and then see different possibilities for other characters, ordered according to similarity degree.

1. Producing SOMs – Differently from the component described in Section 3, in the current version of the Adviser component we only used uppercase letters. This was mainly due to fact that, the upper case is much more limited in terms of differentiation and creative freedom when compared to the lowercase [7]. Given that the Adviser deals with variation of style, we considered that the uppercase was a better choice for a first assessment of the validity of using SOMs, avoiding the greater amount of character style variation of the lowercase.

We produced a SOM for each letter with a dataset of 4034 typefaces. The currently used SOMs followed the following experimental parameters: *number of iterations*: 800; *lattice size*: 25×25; *samples per character*: 250.

Following the production of the SOMs, we calculated the correspondence between the dataset typefaces and the best match SOMs node for each character, i.e. each SOM (e.g. SOM of the character *A*, see left side of Fig. 3) has a list of best match node for each typeface (e.g. the node that is most similar-looking to the glyph *A* of a given typeface). This establishes proximity relationships among the typefaces from the dataset.

2. Finding look-alikes – The Adviser component uses the produced SOMs to suggest the most similar-looking typefaces, based on the glyph given by the user. This process consists in finding the node of the character's SOM that has the highest degree of similarity and then provide the user with a list of typefaces. The similarity comparison is done using the Euclidean distance – also used in the SOMs production – which compares the input glyph with each node. The list of best matched typefaces is then produced by gathering the typefaces which were attributed to the best matched node and the nodes that are close to it. This results in a list ordered by similarity degree.

3. Guiding the user – The interface is the same as the one developed for the first component, as they are part of the same tool. When using the functionalities of the *Adviser component*, the user is able to click on every letter button, even though some of the letters do not have a defined glyph yet. This is due to the components way of working: it does not need a glyph to show a suggestion of a possible design.

The suggestion is shown with a shadow-like effect, being only a visual clue to the user (see right side of Fig. 3). There are two different modes of suggestion: single suggestion – only a glyph typeface is shown – and multiple suggestion – several are shown by overlaying them. Despite being totally functional, the

interface still needs some improvements in order to make both components fit well together in terms of usability and way of working.

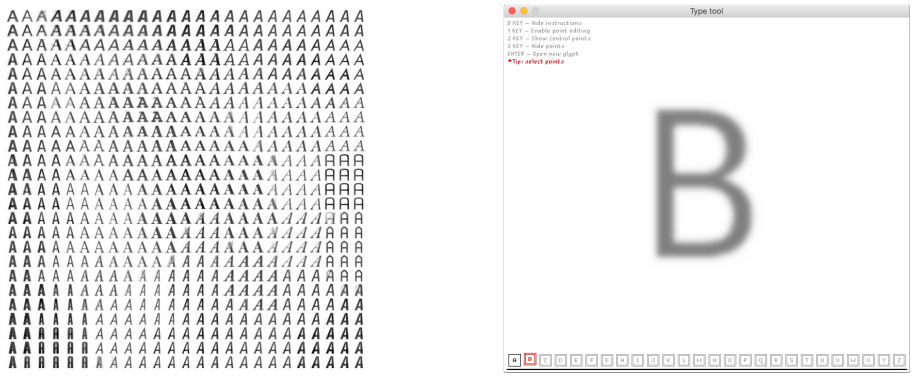

Fig. 3. Left side: SOM of the character *A*. Right side: Suggestion of *B* based on glyph *A* introduced by the user.

5 Conclusion & future work

We have presented and described the functionalities of a type design aiding-tool which has as main purpose to make the design process easier by reducing time-consuming tasks and stimulating creativity. It follows a mixed-initiative approach and is currently composed of two different components: *the Part-sharing* – strongly based on type design principles – and *the Adviser* – which uses Self-Organizing Maps to suggest similar designs to the user. The main goal is not to create completely finished and ready to use typeface but instead to allow the user to design a first draft by using the tool as a support. It is to be used in the early stage of the design process as the user will be able to export the typeface draft and continue the process in an application specially developed for type design, thus allowing the user to afterwards improve details and correct drawing problems such as those related to optical compensations.

Concerning the *Advisor component*, there are also issues to be solved. We consider that the use of SOMs fullfils its purpose as it allows similar typefaces to be found. Notwithstanding, the used dataset has some clear problems which need to be corrected: (1) Some of the typefaces used are too similar; (2) It contains different weights – this is pointless, as we are only designing the regular weight.

In addition, as every SOM is different and finds different similarities among the data, more attention should be given to both SOM production and SOM quality assessment as well as to the used experimental setup.

Regarding the process of finding similar typefaces using SOMs, we aim to further develop its way of working by focusing on multiple matching. The current version only finds similar typefaces based on a single glyph; with multiple matching it would be based on several glyphs, consequently aiming to find typefaces that match the maximum number of glyphs.

References

[1] Balzien, A.: CarveLCD. In: Pape, P., Jenett, F. (eds.) Gestalten mit Code, FH Mainz. http://generative-typografie.de (last accessed in June 2016)
[2] Celso, A.L.: Rhythm in type design (2005)
[3] Cheng, K.: Designing type. Yale University Press (2005)
[4] Cunha, J.M.: Dissertation on anatomical relationships among characters of a typeface (Dissertação sobre relações anatómicas entre caracteres de um tipo de letra). Master's thesis, University of Coimbra (2013)
[5] Eglie, S., M.M., Reigel, A.: Metaflop. http://www.metaflop.com/ (last accessed in June 2016)
[6] Goudy, F.W.: Typologia: studies in type design & type making, with comments on the invention of typography, the first types, legibility, and fine printing. Univ of California Press (1977)
[7] Henestrosa, C., Meseguer, L., Scaglione, J.: Cómo crear tipografías: del boceto a la pantalla. Tipo E (2012)
[8] Kaniowski, A.: Generative stuff. http://generativestuff.com (last accessed in January 2013)
[9] Kohonen, T., Schroeder, M.R., Huang, T.S. (eds.): Self-Organizing Maps. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 3rd edn. (2001)
[10] Koomen, J.: Haagse letters. http://haagseletters.nl (last accessed in June 2016)
[11] Lund, A.: Evolving the shape of things to come: A comparison of direct manipulation and interactive evolutionary design. Proceedings of Generative Art 2000 (2000)
[12] Martins, T., Correia, J., Costa, E., Machado, P.: Evotype: Evolutionary type design. In: Evolutionary and Biologically Inspired Music, Sound, Art and Design, pp. 136–147. Springer (2015)
[13] Mathey, Y., Babé, L.R., et al.: Prototypo. https://www.prototypo.io (last accessed in June 2016)
[14] Schmitz, M.: genotyp, an experiment about genetic typography. Proceedings of Generative Art 2004 (2004)
[15] Yannakakis, G.N., Liapis, A., Alexopoulos, C.: Mixed-initiative co-creativity. In: Proceedings of the 9th Conference on the Foundations of Digital Games (2014)