Building Typefaces as Programs: A node-based approach for modular type design

Tiago Martins
CISUC, UNIVERSITY OF COIMBRA
PORTUGAL

Sérgio Rebelo
CISUC, UNIVERSITY OF COIMBRA
PORTUGAL

João Bicker
CISUC, UNIVERSITY OF COIMBRA
PORTUGAL

Penousal Machado
CISUC, UNIVERSITY OF COIMBRA
PORTUGAL

THEMATIC AREAS
TYPE DESIGN; TYPOGRAPHY AND GRAPHIC DESIGN; TYPOGRAPHY AND MULTIMEDIA DESIGN

KEYWORDS
GENERATIVE DESIGN; MODULAR TYPEFACE; PROTOTYPING DESIGN TOOL; TYPE DESIGN; VISUAL PROGRAMMING LANGUAGE

▶ Designing a modular typeface typically involves the creation of geometric relationships between shapes that are repeated in the same letterform and/or among different letterforms. One can see these relationships as workflows that follow a logic of input, processing, and output of shapes. Therefore, designing a typeface involves a step-by-step set of operations, or instructions, that enables the designer to create a typeface in an algorithmic way.

This paper presents a system that allows one to design modular typefaces. The system allows the user to design a typeface by formalising a "recipe" that transforms a set of input shapes into glyphs throughout a node-based approach. The user can input a set of shapes that through a set of geometric operations are transformed and recombined. There are three types of operation nodes: move, rotate, and scale. The relationships between nodes are established by links that connect them. The output of one node is passed as an input to another creating a flow of shapes from node to node. This way, any modification in a node is immediately propagated to the following nodes and consequently to the final glyphs. To analyse the possibilities and limitations of this approach, we tested the proposed system with fifteen graphic designers. The outcomes and feedback made by the users unveil the value and relevance of the system and point out future directions for this work.

▶ Desenhar um tipo de letra modular tipicamente implica definir um conjunto de relações geométricas entre as formas que são repetidas dentro de um glifo e/ou ao longo de todo os glifos de um tipo de letra. Podemos olhar para estes relacionamentos como um conjunto de fluxos de trabalho que seguem uma lógica de entrada, processamento e saída. Portanto, desenhar um tipo de letra envolve um conjunto de operações, ou instruções, passo a passo que permitem o designer criar tipos de letra de uma forma algorítmica.

Este artigo apresenta um sistema que permite desenhar tipos de letra modulares. O sistema permite que o utilizador desenhe um tipo de letra através da formalização de uma «receita» que transforma um conjunto de formas de entrada em glifos através de uma abordagem baseada em nós. O utilizador pode inserir um conjunto de formas que através de um conjunto de operações geométricas são transformadas e recombinadas. O sistema permite três tipos de nós de operações: mover, rodar e redimensionar. As relações entre os nós são estabelecidas pelas ligações entre eles. O resultando de um nó é passada como entrada para outro nó criando assim um fluxo de formas de nó para nó. Desta forma, qualquer mudança em um nó é imediatamente propagada para os nós seguintes e consequentemente para os glifos finais. Para analisar as possibilidades e as limitações desta abordagem, nós testamos o sistema proposto com quinze designers gráficos. Os resultados e os comentários dos utilizadores revelam o valor e a relevância do sistema e apontam as futuras directrizes para o este projecto.

Desenho Algorítmico de Tipos de Letra: Abordagem baseada em nós para o desenvolvimento de tipos de letra modulares

Tiago Martins
CISUC, UNIVERSIDADE DE COIMBRA
PORTUGAL

Sérgio Rebelo
CISUC, UNIVERSIDADE DE COIMBRA
PORTUGAL

João Bicker
CISUC, UNIVERSIDADE DE COIMBRA
PORTUGAL

Penousal Machado
CISUC, UNIVERSIDADE DE COIMBRA
PORTUGAL

ÁREA CIENTÍFICA
DESIGN DE TIPOS; TIPOGRAFIA E DESIGN GRÁFICO; TIPOGRAFIA E DESIGN MULTIMÉDIA.

PALAVRAS-CHAVE
DESENHO DE TIPOS DE LETRA, DESENHO GENERATIVO, FERRAMENTA DE DESENHO, LINGUAGEM DE PROGRAMAÇÃO VISUAL, TIPO DE LETRA MODULAR

Introduction

Letterforms are part of the history of visual communication since the invention of writing five thousand years ago (Carter, Meggs and Day, 2011). Until the 15th century, books were only accessible to the wealthier classes of society. The bookmaking process was slow and expensive. A simple two-hundred-page book required months of labour and its value was similar to the value of a farm or a vineyard (Meggs and Purvis, 2011). However, with the introduction of the typographic printing process in the West, by Johannes Gutenberg, in the mid-15th century, books became cheaper and print-houses spread rapidly across Europe. Nevertheless, a similar shift occurred with the technological revolution (Flake, 1994; Meggs and Purvis, 2011). Until then, designers' typograpy choices have been restricted by expensive foundries and typesetters. Although the earlier type design systems (e.g. the *Ikarus*) was costly and inaccessible, with the democratisation of the personal computer emerged computer-aided font design software (e.g. *Fontographer*) that enabled designers to easily develop digital types and to sell them in web-based foundries such as *Emigre Fonts* or *Adobe Systems* (Meggs and Purvis, 2011). Consequently, we entered in one of the most creative times in the history of typography, wherein classical concepts were revived and some of the most disruptive and experimental type designs were developed, either in its shape or in its technology (Blokland, van and J. van Rossum, 1990; Miller and Lupton, 2006).

The design of modular typefaces benefited greatly from this revolution. This type of typefaces is defined by the repetition of a set of basic shapes, i.e. modules (Bringhurst, 2004; Lupton, 2014). Accordingly, it makes easiest anyone to design glyphs (Willen and Strals, 2009).

The concept of modularity was always inherent to typography. Traditional typography is composed using modular movable types and digital typefaces are, often, developed based on a modular grid (Lupton and Phillips, 2015; Meggs and Purvis, 2011). Nevertheless, the first modular systems only appear in the early decades of the 20th century. Typefaces such as the *Patrona Grotesk* (V. Kánský, 1928), the *Fregio Mecano* (unknown author, c. 1920), the *Super Tipo Veloz* (Joan Torichut, 1942) or the experiments developed by modernist artists such as Theo Van Doesburg or Josef Albers were notable at the time (Cunha, Bicker and Machado, 2013; Meggs and Purvis, 2011).

Nowadays, modular design is a very popular and common way of designing. Generally, during this process, designers establish a set of geometric relationships between the elements. These relationships can be defined as workflows or step-by-step actions that generate a result. Perhaps, without knowing it, designers are working somehow algorithmically and programmatically, following a logic of input, process, and output.

In this work, we propose an interactive computational system for the creation of modular typefaces. The proposed system follows a programmatic approach because our main goal is to enable the user to build a typeface by creating a program, or a recipe, rather than designing its static shape. This design approach is inspired by the concept of embryogenesis, the process by which form grows in nature (Bentley and Kumar, 1999; Kumar and Bentley, 2003), in the sense that the designer creates a typeface by encoding its design process into a system of rules, or a program.

The proposed system provides a design paradigm based on: (i) shape grammars (Stiny and Gips, 1971), wherein the input is a set of geometric shapes, the translation process is defined by sequences of geometric operations, and the output is a set of letterforms; and (ii) visual programming, which allows the user to assemble programs graphically in a node-based drag-and-drop fashion rather than writing code. The result is a design process in which the designer models the design of a typeface by creating a network of geometric operations, including translation, rotation, and scaling. These operations are organised hierarchically and are intended to transform and combine a set of input shapes in order to construct glyphs for a typeface. The output from one node, i.e. the shapes, is passed as input to another node, creating a flow of shapes from node to node. This node-based approach enables an interactive creation and manipulation of form and this way it develops a modular typeface in a dynamic manner.

This work was initiated in 2014 by the first author as an academic work in a course of his doctoral program. Back then, a proof of concept was developed. However, it was never assessed and disseminated properly. That proof of concept remained as a work in progress, presenting functional limitations and technical issues. Nevertheless, back then, the system already allowed the user to build typefaces.

The remainder of this paper is organised as follows: Section 2 summarises related work focusing on computational approaches that employ hierarchical methods to create typefaces; Section 3 overviews the proposed system; Section 4 describes how the system was tested and analyses the experimental results; finally, Section 5 presents conclusions and directions for future work.

Related Work

The development of type design systems that use a set of basis shapes (i.e. modules) to design letterforms, and consequently typefaces, still is a poorly unexplored field. Since the early times of digital type design, computer-aided typeface design systems create letterforms though the definition of the anatomical parts of the glyphs (e.g. stems, serifs, spines or terminals). These parts were then transformed using parametric approaches (e.g. *ITSYLF* (Mergler and Vargo, 1968), *CSD* (Coueignoux, 1975) or *Metafont* (Knuth, 1982)). However, back then, these systems enabled only a limited range of modification, were difficult to use, and/or did not enable the edition of outcomes directly. Consequently, designers preferred visual direct manipulation font design tools in prejudice of these programmable/parametric systems (Morris, 1989; Shamir and Rappoport, 1998).

Although not working directly with shape modules, Schneider's *DaType* (Schneider, 1998a) presents an interesting case in of exploration of the modular features in type design. This system explores a hierarchical composition approach using object-oriented concepts, such as instantiation, inheritance and overloading (Schneider, 1998b). It enables the reuse of "stroke elements" that share "style attributes," maintaining the consistency between all letterforms in a typeface.

Shamir and Rappoport (Shamir and Rappoport, 1998) proposed a feature-based approach to type design. Their system uses glyphs parts

and sets global constraints to design letterforms. It enables users to change the appearance of a glyph part and modify all the similar parts, in the project, preserving the coherency and the harmony.

Hu and Hersch (Hu, 1998; Hu and Hersch, 2001) developed a component-based font description for synthesising typographic glyphs' shapes. In their system, a glyph is described by its structural elements (i.e. stems, bars, serifs, etc.) and by the implementation of these elements either typeface-category-dependent (e.g. the junction types) or the global font-dependent metrics (e.g. the location of reference lines, the width of stems, etc.). This approach allows a parametric change of the shapes and, consequently, the change of all similar parts in the typeface.

Antoni Kaniowski (Kaniowski, 2011) experimented the possibility of creating a typeface using dynamically defined modules. His *Modular* typographic generator divides the glyphs into modules (that can be dynamically defined) and designs a typeface.

Bastard, developed by Tobias Tschese, is an application (Tschese, 2008) that generates new typefaces through the combination of glyph parts producing of different glyphs. The glyphs parts inserted in the system during the development.

Yoshida et al. (Yoshida, Nakagawa and Köppen, 2010) developed the *Personal Adapted LETTEr* (PALLETE), a system capable of detecting similar glyph parts in a typeface and design new letterform reusing these parts. Furthermore, the system creates new typefaces (modifying these glyph parts) throughout an interactive evolutionary computation approach.

Phan *et al.* (Phan, Fu and Chan, 2015) developed a framework that from a set of letters, inputted by the user, produces complete typefaces. The system decomposes the inputted information into a sustainable representation and, from here, it extracts and synthesis the typeface's "style" to infer/predict the glyphs' composition rules. Using this style information, the system is able to generate new glyphs. The style consistency of these glyphs is evaluated by the similarities between the glyph parts. In this way, the "style" given by the user is preserved. Beyond that, the authors designed an interface that allows designers to create interactively a typeface from scratch.

Martins *et al.* are developing *Evotype* (Martins *et al.*, 2015, 2016, 2018), a system for type design that employs evolutionary computation and machine learning techniques to automatically generate glyphs. In one of the project's iterations (Martins *et al.*, 2016) the system explores the idea of assembling a set of basic visual shapes like modules to create glyphs. The modules are given to the system as input, by the user, through a vector file. The system employs a Genetic Algorithm to generate the letterforms evaluating the merit of each shape calculating the similarity between the result and a well-designed glyph and/or using a Convolutional Neural Network to character recognition.

Cunha *et al.* (Cunha, Bicker and Machado, 2013; Cunha *et al.*, 2016) also explored the modularity of letterforms in *TypeAdviser*. This system allows the creation of a typeface from a user input vector shapes and an initial definition of glyph parts in shapes. The anatomic relationships between characters are used by the system to generate the missing characters of the typeface.

In this system, the anatomic relationships between characters are used in order to generate the missing characters of the typeface. Besides

that, when the user makes changes in any glyph, all the glyphs with the corresponding part are affected.

Stefan Ellmer created *The Pyte Foundry* (Stefan, 2016) that released (during 1 year) one typeface every week. Each typeface was available for free download only during a week and it is a new design or a novel interpretation of an existing design. This only was possible because he developed a component-based system where the same shapes are flipped, rotated, scaled and nested with other shapes to faster create novel letterforms. In several releases, Johannes Lang helped him with code-based transformations (Griesshammer, 2017).

Each of these systems presents an interesting case of the exploration of the modular and hierarchical characteristics in the type design projects. Nevertheless, *Evotype* (Martins *et al.*, 2016) is the only one that permits the user to define, a priori, the basis modular shapes. (On the tools developed for *The Pyte Foundry* (Stefan, 2016), a similar method appears to be employed; however, we did not find enough information to clarify the production method of these tools.) In most of cases, the systems recognise the modular parts of a glyph by automatic methods of features extraction (e.g. (Phan, Fu and Chan, 2015) or (Yoshida, Nakagawa and Köppen, 2010)). Moreover, some systems request the user to define these glyph parts when the design process starts (e.g. (Cunha *et al.*, 2016)). In other systems, this information is already defined (e.g. (Kaniowski, 2011) or (Tschese, 2008)).

In all these projects, the user cannot define the workflow and how the modules are employed to create the typeface. Several examples use interactive evolutionary computation to conduct the system to the user preferences (e.g. (Yoshida, Nakagawa and Köppen, 2010)) or use parametric approaches (e.g. (Hu and Hersch, 2001) or (Kaniowski, 2011)). However, the system has already defined (or have its autonomous way to define) the way that the modules should be arranged to generate a typeface. Furthermore, most of the systems are not concerned with the interface and user experience. Even though, these systems may be powerful tools to type design, especially during the earliest and exploratory stages of project.

Approach

The design process consists of the creation of flows of shapes throughout a network of nodes. Each node has one in port and one out port that are intended to receive and pass shapes from/to other nodes. A flow of shapes from node A to node B is established by creating a link from the out port of A to the in port of B. This flow copies the output shapes of node A to node B. The system prevents the user to link nodes in a way that originates cycles, or loops, in the flow of shapes. For instance, the user is not able to connect node A to node B when B is already connected to A.

There are five types of nodes: three transformation nodes (move, rotate, and scale) and two shape nodes (input and output). Transformation nodes offer three geometric operations that act on the shapes that they receive. Input nodes are intended to contain shapes inserted by the user that will be the building blocks of the typeface. Output nodes are intended to contain compositions of shapes that are

collected from other nodes. When the user exports the font to file, the system considers each output node as one final glyph. In a typical scenario, a flow of shapes begins with an input node, goes through a series of transformation nodes, and ends with an output node.

Each transformation node has its set of parameters. A move node has two parameters: units to move horizontally and units to move vertically. A rotation node has three parameters: rotation angle, x-coordinate of the rotation anchor, and y-coordinate of the rotation anchor. A scale node has four parameters: horizontal scaling, vertical scaling, x-coordinate of the scaling anchor, and y-coordinate of the scaling anchor.

The graphical interface of the system has two main areas: (i) the area of the network of nodes, on the left, where the user sets the nodes and their links in a drag-and-drop fashion, and (ii) a panel with different options, on the right. The user can create a new project, load an existing project from file, and save the current project to file. The user can export the font to OTF or a specimen of it to a vector file, which can be used in any other design software for further refinement. Figure 1 shows a screenshot of the system. A demo video can be seen at cdv.dei.uc.pt/2018/9et/building-typefaces.mov.

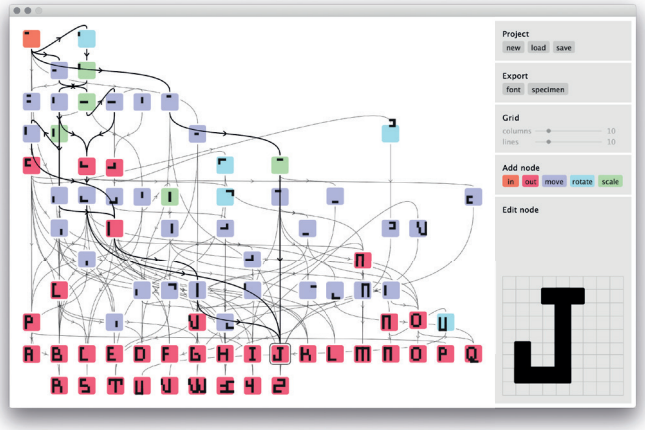


Figure 1 – Screenshot of the system. A demo video can be seen at cdv.dei.uc.pt/2018/9et/building-typefaces.mov

The different types of nodes are visually distinguished with colours: warm colours for shape nodes and cool colours for transformation nodes. Each node shows a preview of its content, i.e. shapes that are passed to the next nodes in the flow. When the user selects a node, the entire flow of shapes to it and from it is highlighted. This way, the user can easily visualise the flow of shapes throughout the nodes. Also, the positioning of each node is constrained by a hidden rectangular grid to better organise them and simplify the links between them.

The coordinates of the vertexes that define the shapes are constrained by a rectangular grid. The user can configure this grid, namely the number of columns and lines, when a new project is created or at any other moment when no shapes exist. The rationale behind this is the fact that a change made in the grid could result in the misplacement

or distortion of existing shapes. A representation of this grid is presented at the bottom of the right panel and it can be used for two purposes depending on the type of the node that is selected. When the user selects an input node, the grid can be used to edit the shapes contained in it. When the user selects a non-input node, the grid is used to preview the transformed shapes.

When an input node is selected, the grid can be used to draw and remove shapes, and to edit the stroke thickness of a shape. The user draws a new shape by defining the sequence of vertexes that define it. A vertex is added by clicking over or close to one point of the grid. If the last vertex coincides with the first one, the shape is considered closed and therefore it will be drawn with fill. To select an existing shape, the user clicks on it. With a shape selected, the user has two options: (i) edit the value of its stroke thickness or (ii) remove it by pressing the backspace key on the keyboard.

When a non-input node is selected, the shapes contained in it are shown on the grid. In the case of transformation nodes, we added two features to the grid to help the user understanding the impact of their geometric operations on the shapes that pass through them. First, when a transformation node is selected, the shapes that enter in them, i.e. before being transformed, are shown on the grid, in background with opacity. Second, when a transformation node that employs an anchor point (rotation or scale node) is selected, the coordinates of the anchor point are represented in the grid with one vertical and one horizontal white line.

Testing

This section overviews the testing of the proposed system. We used a classical task-based usability test method (Rubin and Chisnell, 2016) with some adjustments made according to the nature and goal of this project. First, we explain how the system was tested. Then, we present and analyse the experimental results, and discuss opportunities created with the system.

Setup

We conducted these tests with the goal of gathering data to identify the opportunities and limitations of the proposed system. We consider that this system is useful for graphic designers, who often use modular typefaces in the design of visual identities or titling. Therefore, we considered graphic designers as our representative group of users and asked fifteen graphic designers to test the system.

The testing sessions started with an introduction to the system and its context. Afterwards, we conducted a brief demonstration of the system functionalities explaining what each node does.

Users were asked to perform nine tasks using the system (see Table 1). In the first four tasks (T1 to T4), users experimented with the different nodes. In the three following tasks (T5 to T7), users created glyphs for the letters 'B', 'C', and 'E'. In the next task (T8), users created glyphs for the remaining uppercase letters. In the last task (T9), using the same network of nodes created in T8, users created another typeface by only modifying the input shapes.

We supervised the tests in order to assist the users, measure the duration time of each task, and take some notes related to the usability of the system (e.g. difficulties, comments or compliments). In the end, a short discussion session occurred, where the opportunities, advantages, disadvantages, and user comments are discussed and reported.

Table 1 – Table presenting the task plan for the testing sessions. In this table, we present the tasks performed by the user (from T1 to T9), the description of each task and the result of a successful completion of each task.

Task		Description of the task	Description of a successful completion of the task
T1	Create shape	Create an input node with a shape designed using the system.	Implementation of an input node
T2	Move shape	Implement a move node to move a shape.	Implementation of a move node
T3	Rotate shape	Implement a rotate node to rotate a shape.	Implementation of a rotate node
T4	Scale shape	Implement a scale node to scale a shape.	Implementation of a scale node
T5	Design 'B' glyph	Connect one or more transformation nodes to design a 'B' glyph.	Implementation of an output node similar to a 'B' glyph
T6	Design 'C' glyph	Create and/or reuse the necessary nodes to design a 'C' glyph.	Implementation of an output node similar to a 'C' glyph
T7	Design 'E' glyph	Create and/or reuse the necessary nodes to design an 'E' glyph.	Implementation of an output node similar to an 'E' glyph
T8	Create typeface	Create and/or reuse the necessary nodes to design an uppercase typeface.	Implementation of 26 output nodes with the content similar to the uppercase glyphs of the Roman alphabet
T9	Redefine module	Perform one or more alterations in one or more input nodes.	Change shapes of one or more input nodes

Results

Table 2 shows a selection of the typefaces created during the tests. More experimental results can be visualised in the video at cdv.dei.uc.pt/2018/9et/building-typefaces.mov.

Table 2 – Typical typefaces created in task 8 (left) and task 9 (right). Each horizontal pair of typefaces was created by the same user, share the same network of nodes, but use different input shapes. One can visualise more experimental results in the video at *cdv.dei.uc.pt/2018/9et/building-typefaces.mov*

User	Typeface created in task 8	Typeface created in task 9
1		
2		
3		
4		
5		

Looking at the experimental results, one can observe that (i) the glyphs are legible, i.e. easy to be recognised; and (ii) there is visual diversity among typefaces created by different users.

Based on the observation of the drawing process during the tests, we observe that the users defined one or two modules to develop the requested first tasks (see Table 1). These modules are, after, used to develop other glyphs, until the user is not capable to create a specific glyph. Therefore, the user creates new modules. However, each designer works in a different way, some designers start to design sequentially by the first characters of the alphabet, others by the last characters and another designed without rules.

Most of the users were able to envision the glyphs employed with their knowledge on typography anatomy. They subdivide a letter into smaller and simpler parts (i.e. input shapes) and configure these blocks to build the typeface. Regarding the input shapes drawn by the users, we can observe several levels of complexity, as we can see in the developed typefaces (see Table 3 and 4).

Analysing the networks of nodes created by different users (see Table 4), it is visible different levels of complexity in terms of network topology. At the same time, it is also noticed some similarities. This aspect is aligned with the different typography styles which we can identify over the time and in the nowadays typographic scenario.

During the second part of the tests, based on the observations of the drawing process, users were able to play with the shapes in input nodes to create variations of the typographies initial developed. Table 2 shows typical fonts created by five users. where each pair of fonts have the same structure (network of nodes) but are built with different sets of shapes. This proves that the system is a dynamic environment that provides the user, without effort, automatically propagate a change to the entire typeface in a coherent manner. In other words, any change in an inputted glyph part is reflected in all letters that use this part. In this sense, the user is able to change the visual style of the font while maintaining its structure.

The other way around, i.e. changing the structure while maintaining the style, is also possible. The user just has to change the network of the nodes without changing the shapes of the input nodes. This approach facilitates the generation of variations based on an initial font.

Table 3 – Two sets of shapes used to build two fonts by user 1. The set of shapes on the left builds the font A and set of shapes on the right builds the font B (see Table 2).

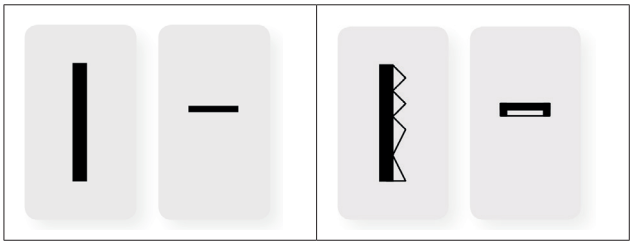
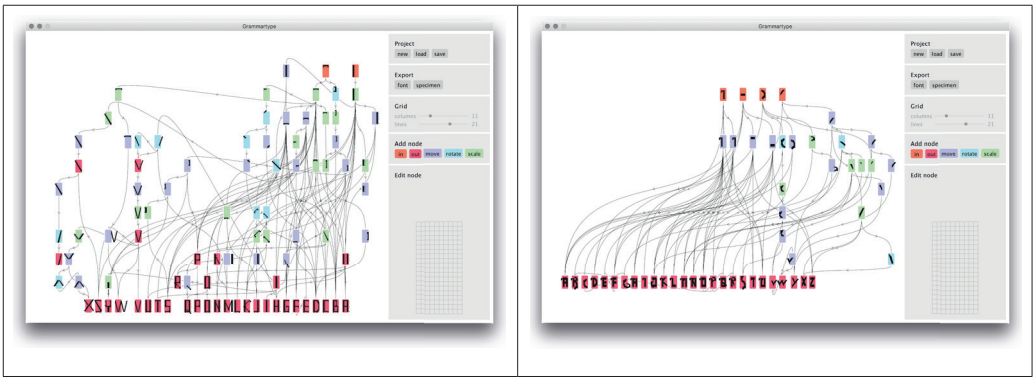


Table 4 – Screenshots of two networks of nodes created by different users. The network on the left generates the typeface A by user 1. and the network on the right generates the typeface B by user 4 (see Table 2).



Based on comments given by users we identify the demand for two type of nodes: a node to group a set of shapes and the mirror transformation node. Furthermore, several users highlighted that the system should allow altering the shapes' construction points after constructed. Nevertheless, most users did not have much difficulty in using the flow system and employing the node-based geometric transformation to create a typeface considering the difficulties derives from technical issues, resulting from the system still yet be in a beta version.

Discussion

Although the presented system consists of a prototype, we consider that it demonstrates great creative potential and enhances the type design process by (i) supporting visual interactive manipulation of type designs at various representational levels, and (ii) supporting generation and exploration of potential alternative letterforms for the same character.

Contemporary computational tools for type design offer great support for detailed design tasks, providing visual editors that allow the creation and manipulation of letterforms in a precise manner. However, we consider that these tools provide limited support to explore alternative letterforms and poor capabilities to make global adjustments in the whole typeface. A local adjustment in a specific glyph must be manually executed in the other ones. With the presented system, designers can interactively manipulate design data and develop a typeface in a hierarchical manner. This approach creates many design opportunities. For instance, it is possible to change a glyph element and propagate the effect to all glyphs that use that same element. Accordingly, it is easy to modify the input shapes, and consequently the style of the typeface, while preserving the coherence among the resulting glyphs. Also, after creating a typeface with the system, one is able to create a typeface family that has a continuous range of weights by simply changing, e.g., the stroke thickness of the input shapes.

Some users who tested the system mentioned on multiple occasions the educational potential it provides. The visual development environment provided by the system may help design students to study the anatomy of typefaces and their design process.

Conclusion and Future Work

In this paper, we have described and tested a node-based system to build modular typefaces. The proposed system is developed to replicate the traditional process of creating a modular typeface, where designers employ a set of geometric operations to transform and combine a set of initial shapes in order to design glyphs that form a typeface. The system allows the user to perform sequences of geometrical transformations (scale, rotate and move) in an initial set of shapes. These sequences of transformations are set by flows of nodes, i.e. nodes connected by links. The output of one node is passed as input to another creating a flow of shapes from node to node. This way, the user is able to create glyphs that can be manipulated in an interactive and dynamic manner.

Although the presented system remains a work in progress, presenting some functional limitations and technical issues, it is already able to create typefaces. In order to demonstrate this, we tested the system with a group of fifteen graphic designers. These tests enabled the assessment of the approach employed in the system, as well as the identification of limitations and opportunities that will be considered in future work.

In future work, we will focus on different paths and possibilities. For instance, we intend to experiment with the proposed system in other design tasks, where the reuse of graphic modules is essential, e.g. the design of signs.

For instance, we intend to enable users to import their own vector shapes as input shapes in order to expand the visual possibilities of the typefaces created with the system. It would also be interesting to provide the user with a library of typefaces created by other users. This way, users could use any typeface (created with the system) and adapt it to their own concepts and requirements. This environment of typefaces created by different people could benefit from a web version of the system, which would allow anyone to use the system easily.

We are also should implement methods to allow the user to tag each node with metadata (e.g. title or keywords). This data would not only enable users to search and filter nodes during the design process but also provide a valuable layer of information that could be used to generate knowledge related to each type design process (e.g. meaning of each node, anatomy of each glyph, and how glyphs relate to each other).

In future iterations of the system, we expect to integrate it with an evolutionary algorithm to: (i) evolve the input shapes while using a pre-designed network of nodes; (ii) evolve the network topology while using a set of pre-designed input shapes; and/or (iii) evolve the input shapes and the network topology, simultaneously. The evolutionary process could be (i) semi-automatic, with evaluation provided by the user, or (ii) automatic, with evaluation calculated using, e.g. a machine learning mechanism such as a classifier of characters. We believe this research path may enhance the explorative capabilities of the proposed system, enabling the automatic generation of novel and unforeseen letterforms with little effort.

Acknowledgments

The first and second authors are funded by *Fundação para a Ciência e Tecnologia* (FCT), Portugal, under the grants SFRH/BD/105506/2014 and SFRH/BD/132728/2017, respectively. We would also like to express our gratitude to the graphic designers who tested with great enthusiasm the system developed in this work.

References

BENTLEY, Peter; KUMAR, Sanjeev - ***Three Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem. In Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*** GECCO'99. . San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1999

BLOKLAND, Erik VAN; ROSSUM, Just VAN - ***Is Best Really Better***. Emigre. 18 (1990).

BRINGHURST, Robert - ***The elements of typographic style.*** 3.º ed. Vanouver, Canada : Hartley & Marks, Publishers, 2004. ISBN 9780881792065.

CARTER, Rob; MEGGS, Philip B.; DAY, Ben - ***Typographic design: Form and communication***. Hoboken, New Jersey, United States : John Wiley & Sons, 2011

COUEIGNOUX, Philippe Jean-Marie - ***Generation of roman printed fonts.*** Massachusetts Institute of Technology, 1975

CUNHA, João; BICKER, João; MACHADO, Penousal - ***Dissertation on anatomical relations among characters of a typeface*** (Dissertação sobre relações anatómicas entre caracteres de um tipo de letra). University of Coimbra, 2013

CUNHA, João M. et al. - ***TypeAdviser: A type design aiding-tool***. In Ceur Workshop Proceedings

FLAKE, Günther - ***Font Production in past and present***. Em Font Technology. Springer, 1994. p. 59–76.

GRIESSHAMMER, Frank - ***The Pyte Foundry*** [Online], atual. 2017. [Retrieve in 1 August. 2018]. Available in WWW:<URL:https://typographica.org/typeface-reviews/the-pyte-foundry/>.

HU, Changyuan - ***Synthesis of parametrisable fonts by shape components***. EPFL, 1998

HU, Changyuan; HERSCH, Roger D. - ***Parameterizable fonts based on shape components***. IEEE Comput. Graph. Appl. 21:3 (2001) 70–85.

KANIOWSKI, Antoni - ***Modular Typographic Generator*** [online]. 2011. [Retrieve in 1 August 2018]. Available in WWW:<URL:https://www.behance.net/gallery/1824431/Modular-Typographic-Generator>.

KNUTH, Donald E. - ***The concept of a meta-font***. Visible language. 16:1 (1982) 3–27.

KUMAR, Sanjeev; BENTLEY, Peter J. - C***omputational embryology: past, present and future***. In Advances in evolutionary computing. Springer, 2003. p. 461–477.

LUPTON, Ellen - ***Thinking with Type: A Critical Guide for Designers, Writers, Editors, & Students. 2nd. ed***. New Princeton Architectural Press, 2014

LUPTON, Ellen; PHILLIPS, Jennifer Cole - ***Graphic Design: The New Basics: Revised and Expanded. 2nd. ed***. Princeton Architectural Press, 2015

MARTINS, Tiago et al. - ***Evotype: Evolutionary Type Design***. In JOHNSON, COLIN; CARBALLAL, ADRIÁN; CORREIA, JOÃO (Eds.) - Evolutionary and Biologically Inspired Music, Sound, Art and Design – 4th International Conference, EvoMUSART 2015, Copenhagen, Denmark, April 8-10, 2015, ProceedingsLecture Notes in Computer Science. [Online]. Springer, 2015 Available in WWW:<URL:http://dx.doi.org/10.1007/978-3-319-16498-4_13>.

MARTINS, Tiago et al. - ***Evotype: From Shapes to Glyphs***. In Proceedings of the Genetic and Evolutionary Computation Conference 2016GECCO '16. . New York, NY, USA : ACM, 2016. ISBN 978-1-4503-4206-3

MARTINS, Tiago et al. - ***Evotype: Towards the Evolution of Type Stencils***. In LIAPIS, ANTONIOS et al. (Eds.) - Computational Intelligence in Music, Sound, Art and Design. Cham : Springer International Publishing, 2018

MEGGS, Philip B.; PURVIS, Alston W. - ***Meggs' history of graphic design. 5th. ed.*** John Wiley & Sons, 2011

MERGLER, H. W.; VARGO, P. M. - ***One approach to computer assisted letter design***. Visible Language. 2:4 (1968) 299–322.

MILLER, J. Abbott; LUPTON, E. - ***Design/Writing/Research: Writing on Graphic Design***. London, UK : Phaidon Press, 2006

MORRIS, Robert A. - ***Rendering digital type: a historical and economic view of technology***. The Computer Journal. 32:6 (1989) 524–532.

PHAN, Quoc Huy; FU, Hongbo; CHAN, Antoni B. - ***FlexyFont: Learning Transferring Rules for Flexible Typeface Synthesis***. Computer Graphics Forum. 34:7 (2015) 245–256.

RUBIN, Jeffrey; CHISNELL, Dana - ***Handbook of usability testing: how to plan, design and conduct effective tests***. Hoboken, NJ, USA : John Wiley & Sons, 2016

SCHNEIDER, Uwe - ***DaType: a stroke-based typeface design system***. **Computers and Graphics**. 22:4 (1998a) 515–526.

SCHNEIDER, Uwe - ***An object-oriented model for the hierarchical composition of letterforms in computer-aided typeface design. In Electronic Publishing, Artistic Imaging, and Digital Typography***. Springer, 1998b. p. 109–125.

SHAMIR, Ariel; RAPPOPORT, Ari - ***Feature-based design of fonts using constraints***. In Electronic Publishing, Artistic Imaging, and Digital Typography. Springer, 1998. p. 93–108.

STEFAN, Ellmer - ***The Pyte Foundry | About*** [Online], atual. 2016. [Retrieve in 1 August 2018]. Available in WWW:<URL:http://www.thepytefoundry.net/about.html>.

STINY, George; GIPS, James - ***Shape grammars and the generative specification of painting and sculpture***. In **IFIP Congress (2)**

TSCHESE, Tobias - ***Bastard : Gestalten mit Code***. 2008).

WILLEN, Bruce; STRALS, Nolen - ***Lettering & Type: Creating Letters and Designing Typefaces.*** Princeton Architectural Press, 2009

YOSHIDA, Kaori; NAKAGAWA, Yuta; KÖPPEN, Mario - ***Interactive genetic algorithm for font generation system***. In World Automation Congress (WAC), 2010