xCoAx 2020
Computation,
Communication,
Aesthetics & X
2020.xCoAx.org
Graz, Austria

Ȧdea: Evolving Glyphs for Aiding Creativity in the Graphic Design Workflow

Daniel Lopes
dfl@dei.uc.pt
University of Coimbra, Centre for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering, Portugal

João Correia
jncor@dei.uc.pt
University of Coimbra, Centre for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering, Portugal

Penousal Machado
machado@dei.uc.pt
University of Coimbra, Centre for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering, Portugal

Keywords: Generative Design, Generative Typography, Computational Creativity, Genetic Algorithm.

In creative fields such as Graphic Design, it is often difficult to break away from the past and find novel solutions. We present an evolutionary system for generating typographical glyphs along with a 3-stage workflow for using it as a graphic design tool. We evolve SVG using both interactive and automatic fitness assignments and comparing traditional operators with topological ones. The results suggest that the implemented topological operators are less destructive than conventional ones. Advantages to both interactive and automatic evaluation methods are addressed. Finally, we refer to a set of design artefacts developed out of the generated glyphs, demonstrating the relevance of including the system in the designers' workflow.

1. Introduction

Novelty is one of the fundamental characteristics of describing creativity (Boden 1996). Though, making novel advances in creative fields such as art or design often is a protracted process. Novelty may result from stochastic events, for example, exploited by experimentation (trial and error). For instance, even the interpolation of existent ideas may carry a stochastic factor, once it may be necessary to experiment with several different ways of interpolating ideas. For machines to be creative, these must overcome the same challenges as human beings do (Veale and Cardoso 2019). Thus, as Computational Creativity (CC) algorithms step forward in many creative fields, it is noticeable that many systems, mainly the ones based on Machine Learning (ML), often end up creating *pastiche*—imitations of existent styles (Toivonen and Gross 2015). Evolutionary Computation (EC), inspired by Darwin's theory of natural evolution, has potential to find novelty due to their similarity to the search processes of human designers—search the unexplored space of possibilities, often with a specific conceptual target limiting the possibilities. The main difficulty of applying EC for generating aesthetics is developing appropriate fitness functions. Nevertheless, ML and EC may complement each other for getting more capable CC systems. For instance, using EC for generating and ML for evaluating individuals. As long as there is not a perfect solution for finding novel designs, we believe that the most successful contemporary solution resides in the collaboration Human-Computer. In that sense, we propose a collaboration between humans, EC and ML (Romero 2007).

In this paper we are presenting Ȧdea, an online EC system for aiding designers during the creation of typographical glyphs, offering initial sketches for given characters. Human designers may then use the glyphs for creating design artefacts such as logotypes, typefaces or artworks. We opted to evolve SVG glyphs rather than raster ones for easing its editing and usage—it enables (i) direct vectorial manipulation, (ii) endless resizing and (iii) direct usage in typeface development software.

In the following sections, we present related work regarding ML and EC for aiding humans in the creative process. Then, we present our approach and we showcase some final artefacts designed using the generated glyphs. We conclude by reflecting on the current state of the system, along with future work.

2. Related work

Ȧdea uses EC to explore a set of parameters (gene values) and find novel typographic glyphs that help designers to construct new designs. Similarly, other systems have been developed using Artificial Intelligence (EC or ML) for aiding graphic designers' creativity.

A common ML approach is training models with existing examples and then handling the latent space to generate interpolations of these. However, as already referred, such approaches often produce *pastiche* results (Toivonen and Gross 2015). EC approaches also have their own shortcomings once it may be difficult to write conditions for automatically evaluating aesthetics. Yet, as well as it was employed in our system, ML may be used to endorse automatic fitness assignment, permitting extensive exploration. Correia et al. (2013) demonstrate the feasibility of such by evolving figurative images using classifiers trained to recognize objects.

Most of the EC systems for creative purposes employ interactive evaluation (manually managed by the user). This process is often slower yet more controllable, being useful in many creative contexts (Cunha et al. 2019). Along with ML and interactive aesthetics evaluation, Ȧdea regards EC for typography and vector formats. A related system is Heijer and Eiben's (2011), which allows the automatic evolution of computational-art out of vectorizations of photographs. The system also relates to ours by allowing one to start evolving from meaningful images.

Respecting both vector graphics and typography, Unemi and Soda (2003) proposed an interactive system for generating Japanese typefaces. The stroke of vectorial skeletons was evolved. Although there is no reference to it, it is likely that this approach could be applied to Roman alphabets. From the reviewed systems, Schmitz's (2004) is one of the most related to Ȧdea, by allowing to interactively evolve typefaces out of existing ones. The phenotype is encoded by three arrays of vectorial points describing the skeleton, line strength and serif shape (if available). The greater shortcoming of this approach is the need for a non-standard font format, which makes it difficult to create initial populations, leading to little variety. Yoshida et al. (2010) developed an interactive system for evolving typefaces out of existing ones. The anatomical parts of the initial glyphs had to be previously defined using complementary software. These requirements still obligate a protracted creation process, which may lead to the aforementioned shortcomings again. Levin's et al. (2006) permits to automatically evolve abstract yet congruent typefaces (represented by Bezier curves), starting from a single glyph and a set of parameters defined by the user. This approach could be highly complementary to Ȧdea. Though, there is no evidence that the system is able to generate glyphs for real alphabets. A more recent EC system for aiding type design is the one by Martins et al. (2016). The user may define a set of starting SVG modules and the system automatically evolves a style-congruent and camera-ready typeface out of them. By altering the starting modules, it is possible to create typefaces that are conceptually suited for a wide variety of purposes.

3. Approach

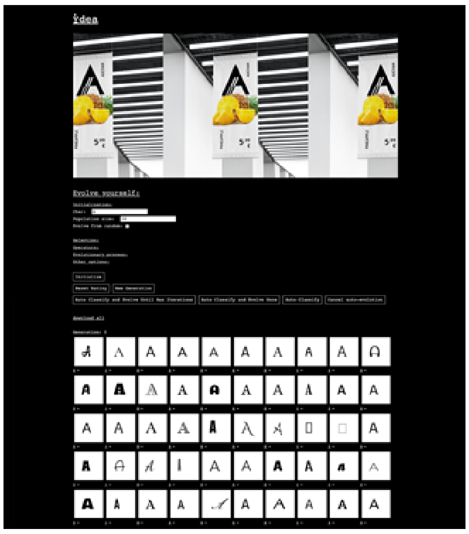

Fig. 1. Ȧdea's Interface
https://student.dei.uc.pt/~dfl/Adea

As a Human-Computer collaboration system, Ȧdea needs to be easily accessible to users (designers) and for that reason it was developed as a web page. The evolutionary engine runs natively on JavaScript (JS) on the client-side. The user interface (HTML, CSS, JS) (see Fig. 1) allows the definition of several different algorithm parameters, start and stop the evolutionary process, toggle between manual and automatic evaluation, and download single individuals or entire populations. After downloading the intended individuals, the user may manipulate them through external resources (for example, vector-edition software) or use them straight for creating designs. For trying the latest version of the system, please visit: https://student.dei.uc.pt/~dfl/Adea.

3.1. Evolutionary Engine

In our system, a genotype consists of a set of coordinates (relative to the canvas' origin) and their respective type of point—"line" (L) or "move to" (M). As a whole, those define SVG paths embedded in a 200x200 pixels view box.

Is it possible to start evolving from any population, whether it is a set of abstract individuals (randomly generated points, for example) or a set of meaningful glyphs (for example, the character "A"). It is also possible to evolve towards a given target, stop the process and then start from the already evolved population towards a different target.

At this stage, the interface does not allow the users to evolve from their own initial glyphs. The system starts by randomly choosing a number (population size) of typefaces out of a dataset of 977 *Google* fonts. Then, it automatically converts the characters into SVG paths using *opentype.js*.

The points of the paths are controlled using our one JS library, which allows point remotion, translation and type shifting (L or M). As the class cannot handle all the types of points, we convert them all into L and M types. This does not represent a problem because most of the paths can preserve their topology. Also, we do not seek to perfectly render the typefaces, but get a diverse set of initial meaningful glyphs. For starting from random initial populations, the points of each individual are randomized within the SVG canvas.

There are two different types of crossover available—2-point crossover and topological crossover. Variation operations involve two individuals at a time, each one with a 50% chance to be selected as predominant—first father. We refer to the latest as i1 and the non-predominant one as i2.

The user may set a probability for the variation operators to perform, as well as a maximum percentage of points to crossover in a single interaction (MPC). We set the latest as a percentage once the number of points in the genotype may vary in the order of hundreds. By doing that, we ensure that we are crossing over parts with relatively similar sizes.

The 2-point crossover operator takes a random slice of consecutive points from i1 and swaps it with a random slice of consecutive points from i2. The size of the slides is equal or smaller than the defined MPC. Also, the slices may not be the same size nor aligned.

The topological crossover operator aims to better keep the topology of the initial population, yet still maintain variety. For that to happen, a point from i1 is more likely to crossover with its closer points (in the Cartesian's plane) from i2. The crossover chances decrease exponentially as the points are further away.

We implemented five mutation methods which run by the following order: (i) point deletion; (ii) conventional/topological point translation; (iii) point type shifting; and (iv) point creation. The user may (i) choose between conventional or topological point translation; (ii) set a probability for a mutation to perform; (iii) set individual probabilities for each mutation operator to run; (iv) set a maximum percentage of points to be mutated in a single operation. The deletion method deletes one random point out of an individual's genotype. The conventional translation method translates one random point within a maximum radius (MR) defined by the user. The topological translation method translates a random array of consecutive points (>=1) according to a single random vector whose maximum magnitude is MR. The type shifting method toggles the type of one random point between L and M. The creation method picks a random point from i1 and adds a new point around it within MR; the type of the new point may be either M or L according to a probability defined by the user.

During the evolutionary process, the user may alternate between interactive and automatic evaluation. The fitness value is assigned from 0 to 1, being 1 the best hypothesis. An individual is considered properly fitted if its fitness value is greater than 1 minus a set satisfactory distance. Interactive evaluation is performed by clicking over individuals from better to worst fit. The not clicked individuals are assigned with the fitness of 0. Automatic evaluation is computed using a pre-trained neural network from *Tesseract.js* and it takes into account two considerations: (i) does *Tesseract* recognize the character; and (ii) with how much confidence it recognizes the character. If a glyph is not recognized or it is smaller than a minimum size defined by the user, the fitness value is set to 0. Otherwise, the fitness value is the distance between the confidence value returned by *Tesseract* and the target confidence set by the user. Ideally, a novel glyph for a given character must be far enough from *Tesseract*'s training examples (the existing glyphs), but close enough for the glyphs to be representative of the character. Thus, an optimal confidence target must not be 100%, but a lower value (80%, for example).

Regarding selection, the user may opt between tournament or elitist methods, and also set a tournament/elite size. The evolutionary process may be finished manually or by the completion of one of the following conditions: (i) a given maximum number of generations was run; (ii) a given percentage of the population is properly fitted.

4. Experimental Setup and Results

We conducted experiments to assess: (i) whether the system was able to generate suitable glyphs for a given character; (ii) whether we could find new ideas for aiding the design of novel glyphs. For assessing full detail on the settings used in the experiments, please visit: https://cdv.dei.uc.pt/adea.

We started by comparing conventional operators (Fig. 2.a and 2.b) and topological operators (Fig. 2.c and 2.d) for a population of *Google* "A"s evolved without evaluating individuals (random selection). As a result, we noticed that topological operators better keep the topology of the glyphs. We also noticed that having no evaluation is substantially destructive to the phenotypes, and the problem tends to increase as we step up generations (see Figure 2.b and 2.c for 50 generations and Figure 2.b and 2.d for 100 generations).

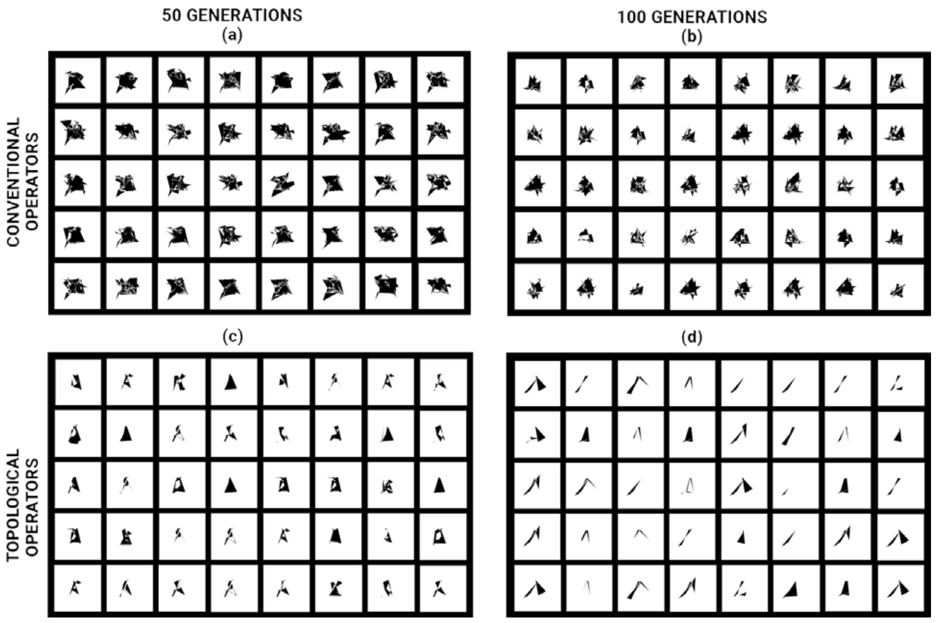

Fig. 2. Populations of Google "A"s evolved using random selection;
(a) 50 generations, conventional operators;
(b) 100 generations,conventional operators;
(c) 50 generations, topological operators;
(d) 100 generations, topological operators.

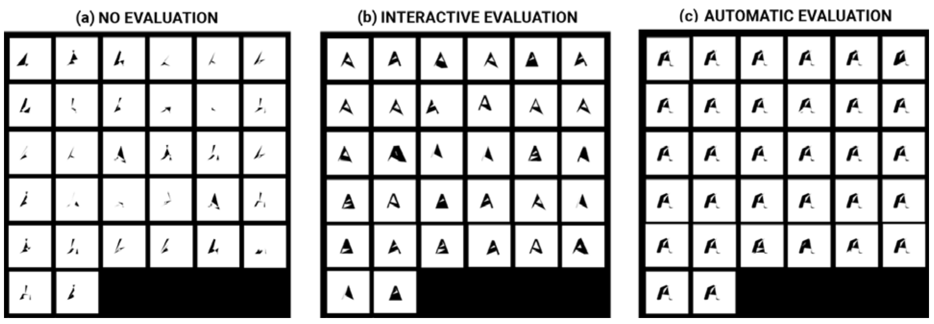

Fig. 3. 50th generation of Google "A"s evolved using topological operators; (a) using random selection; (b) using interactive evaluation and elitist selection; (c) using automatic evaluation and elitist selection.

Therefore, we compared random, interactive and automatic evaluation methods, using topological operators. As expected, both interactive (Fig. 3.b) and automatic (Fig. 3.c) evaluation turned out to be less destructive to the phenotypes than using no evaluation (Fig. 3.a).

By comparing Fig. 3.b and Fig. 3.c, we noticed that interactive evaluation may allow better control on the process, in this case, aiding the generation of less noisy glyphs. Moreover, these may be fully camera-ready and usable in graphic applications already. Its shortcoming may be the binding protracted process which may frustrate users after some seeds.

The automatic evaluation may be more agile and efficient once it has been successful in finding unexpected solutions without requiring major human effort. For instance, by searching for glyphs of 80% confidence, which may have higher chances to be novel due to their distance to the training examples of the network.

To address whether the aforementioned insights could be generalized, we evolved glyphs for all the uppercase characters in the Latin alphabet, using elitist selection and topological operators. Figure 4.a and 4.b showcases glyphs regarding interactive and automatic evaluation, respectively. Their similar graphisms suggest that both methods may lead to similar results, so the automatic one may stand out by being able to fasten the process, allowing the generation of a higher number of ideas per time period.

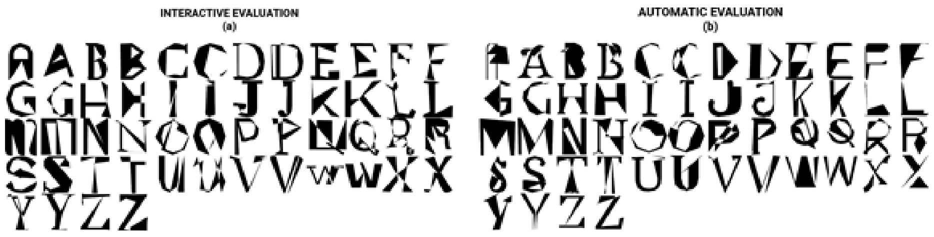

Fig. 4. Different characters regarding different seeds and generations, using elitist selection and topological operators; (a) regarding interactive evaluation; (b) regarding automatic evaluation.

As the generated glyphs may not be camera-ready (they may only trigger off new ideas), we recommend a 3-stage workflow (see Figures 5 and 6) for designers to use the system in its full potential: (i) generate and download glyphs straight out of Ȧdea; (ii) Use a third party software (vector-editing software, for example) for post-editing the glyphs or create new elements out of them (for example, a typeface or a logotype); (iii) Create artefacts using the previously designed elements.

For illustrating such, we present glyphs generated using varied operators and varied evaluation methods. Figure 5 presents a comparison between non-edited and post-edited glyphs, exemplifying possible manual fixes. We picked glyphs we considered noteworthy, yet different users may find different glyphs interesting. Figure 6 showcases artefacts designed using the same post-edited glyphs, demonstrating the potential of including Ȧdea in graphic designers' workflow.

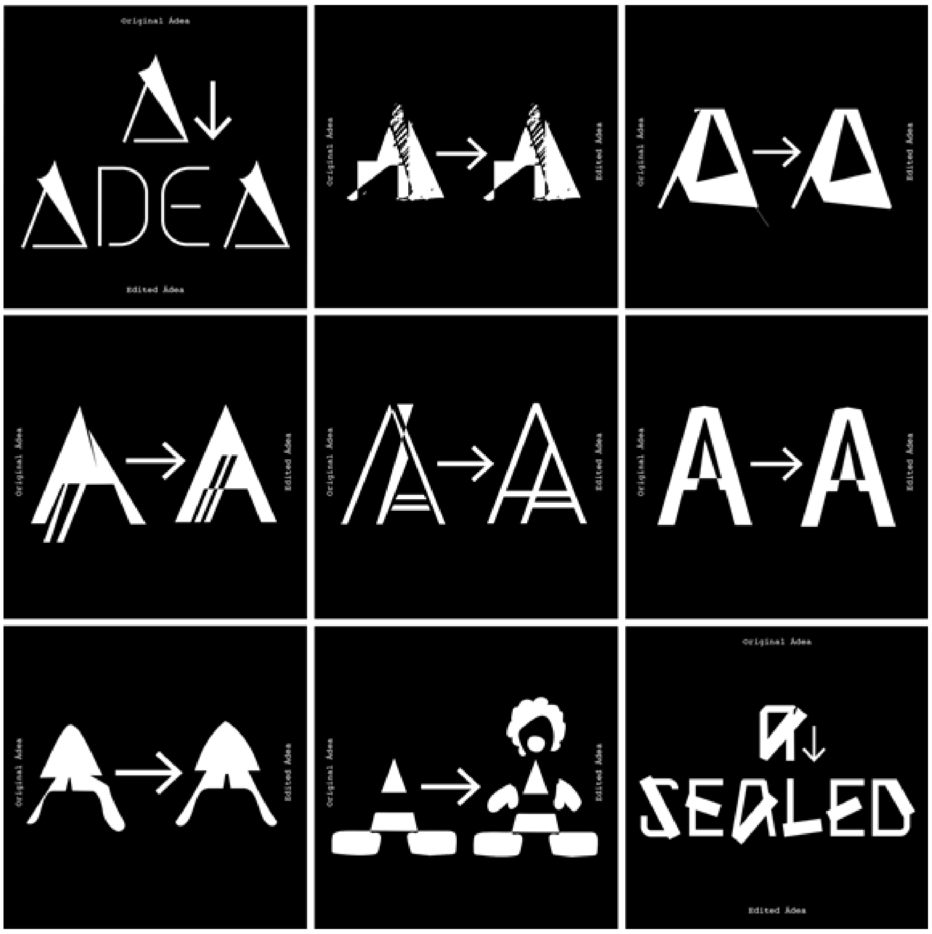

Fig. 5. Not-edited glyphs *vs* post-edited glyphs/ typefaces/ logos.

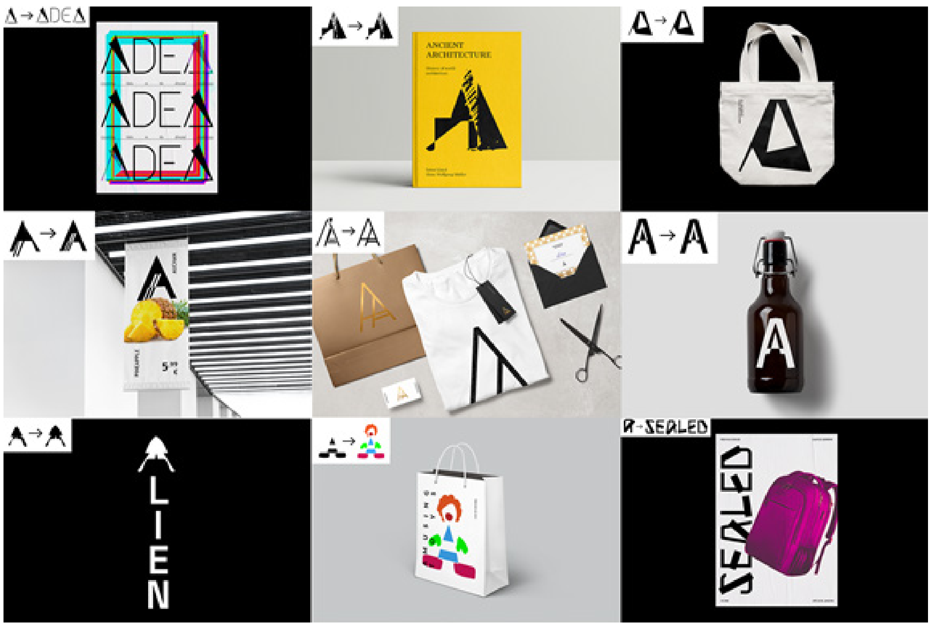

Fig. 6. Final artefacts designed using the manually manipulated glyphs/ typefaces/ logos.

5. Conclusion and Future Work

We have presented Ȧdea, an evolutionary system for aiding designers to find novel glyphs by offering starting points to conceptualize, construct and explore new design spaces. Instead of starting to evolve from randomness, we used glyphs from *Google fonts* for constructing initial populations—a starting point that we know to be closer from the intended results. These allowed us to use topological crossover and mutation operators which revealed to better keep the topology of the initial populations.

We tested both interactive and automatic fitness assignments (performed by a pre-trained neural network—Tesseract.js). We assessed that interactive evaluation may allow better control of the results, so it is easier to drive the process into camera-ready glyphs. Nevertheless, noteworthy results were also produced using the automatic fitness assignment. Thus we may pinpoint some advantages in using the latest, such as fastening the exploration process and allowing the generation of a higher number of ideas per time period.

Looking at the visual results presented in this paper, we consider that we have been successful in evolving suitable new glyphs for several different characters (see Fig. 4). Also, from our experience in GD and sustaining our conclusion in the artefacts designed out Ȧdea's glyphs (see Fig. 6), we consider that the system is capable of fostering creativity by offering ideas for GD applications. Future work will focus on (i) supporting the aforementioned statement by a user survey; (ii) finding other metrics for automatic evaluation (for example, pixel-to-pixel distance to existing glyphs); (iii) using figurative images in initial populations; (iv) using more mutation operators; (v) inviting different designers to test the system and use it to develop design artefacts; (vi) generate whole typefaces out of the generated glyphs.

Acknowledgements: This work is funded by national funds through the FCT - Foundation for Science and Technology, I.P., within the scope of the project CISUC - UID/CEC/00326/2020 and by European Social Fund, through the Regional Operational Program Centro 2020 and is partially supported by Fundação para a Ciência e Tecnologia, under the grant SFRH/BD/143553/2019.

References

Boden, M. A.
1996. Creativity. In Artificial intelligence. Elsevier. 267–291.

Correia, J.; Machado, P.; Romero, J.; and Carballal, A.
2013. Evolving figurative images using expression-based evolutionary art. In Proceedings of the Fourth International Conference on Computational Creativity, 24–31.

Cunha, J. M.; Lourenço, N.; Correia, J.; Martins, P.; and Machado, P.
2019. Emojinating: Evolving emoji blends. In Ekárt, A.; Liapis, A.; and Castro Pena, M. L., eds., Computational Intelligence in Music, Sound, Art and Design, 110–126. Cham: Springer International Publishing.

den Heijer, E., and Eiben, A. E.
2011. Evolving art with scalable vector graphics. In Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11, 427. New York, New York, USA: ACM Press.

Levin, G.; Feinberg, J.; and Curtis, C.
2006. Alphabet synthesis machine.

Martins, T.; Correia, J.; Costa, E.; and Machado, P.
2016. *Evotype*: from shapes to glyphs. In Proceedings of the Genetic and Evolutionary Computation Conference 2016, 261–268. ACM.

Romero J., Machado P., Santos A., Cardoso A.
2003. On the Development of Critics in Evolutionary Computation Artists. In: Cagnoni S. et al. (eds) Applications of Evolutionary Computing. EvoWorkshops 2003. Lecture Notes in Computer Science, vol 2611. Springer, Berlin, Heidelberg

Schmitz, M.
2004. GenoType.

Toivonen, H., and Gross, O.
2015. Data mining and machine learning in computational creativity. Wiley Int. Rev. Data Min. and Knowl. Disc. 5(6):265–275.

Unemi, T., and Soda, M.
2003. An IEC-based support system for font design. In SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483), volume 1, 968–973. IEEE.

Veale, T., and Cardoso, F. A., eds.
2019. Computational Creativity - The Philosophy and Engineering of Autonomously Creative Systems. Springer.

Yoshida, K.; Nakagawa, Y.; and Koppen, M.
2010. Interactive genetic algorithm for font generation system. In 2010 World Automation Congress, 1–6. IEEE.