AutoVyaz: Automating the Formation of Slavic Calligraphy Ligatures

Alexey Tikhonov
altsoph@gmail.com
Inworld.AI
Berlin, Germany

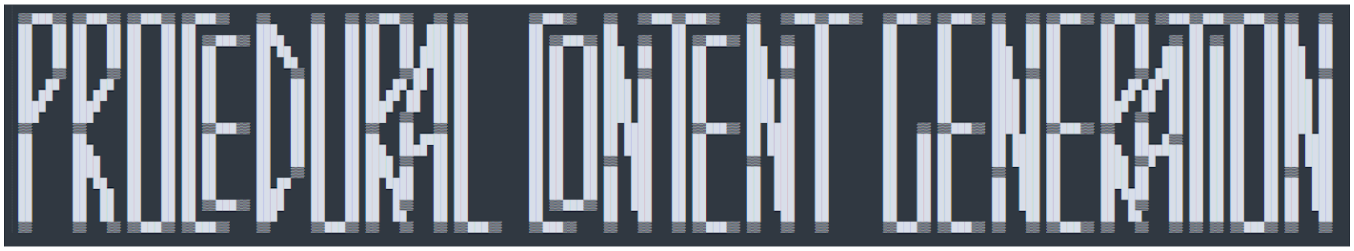

Figure 1: Procedural Vyaz generation

ABSTRACT

This paper introduces a procedural technique for the automated generation of Slavic Vyaz, a traditional Cyrillic calligraphy style known for its ornamental complexity and cultural significance. By developing a specialized description of font geometry that includes anchor points and edges to define character shapes, our approach facilitates the creation of ligatures and character deformations that maintain the aesthetic and rhythmic qualities of Vyaz. We propose several heuristics for arranging text layouts that enable the generation of calligraphic patterns, which can be customized and adapted to various design needs. Our technique's capability to simulate traditional calligraphy is demonstrated through comparative analyses and examples of preliminary results.

CCS CONCEPTS

• **Applied computing** → **Fine arts**; • **Computing methodologies** → *Graphics systems and interfaces.*

KEYWORDS

Slavic Calligraphy, Vyaz, automated calligraphy generation, digital typography, cultural heritage preservation, procedural content generation.

ACM Reference Format:

Alexey Tikhonov. 2024. AutoVyaz: Automating the Formation of Slavic Calligraphy Ligatures. In *Proceedings of the 19th International Conference on the Foundations of Digital Games (FDG 2024), May 21–24, 2024, Worcester, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3649921.3659845

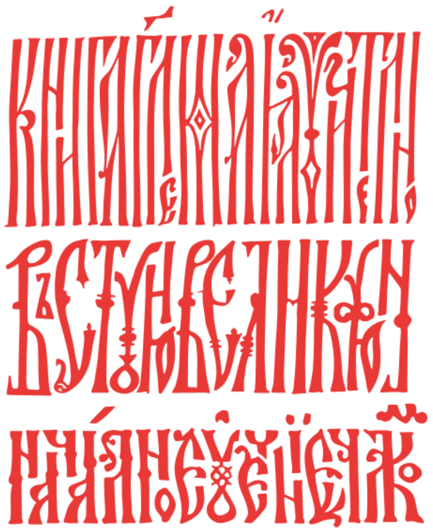

Figure 2: Cyrillic calligraphy sample, via Wikimedia Commons. (https://w.wiki/9RYy).

1 INTRODUCTION

Vyaz[1] (from Slavic 'to bind, to tie'), an ancient Cyrillic calligraphy style known for its decorative features, started in the 13th century within South Slavic monuments. By the 15th century, it spread to East Slavic and Wallachian regions, becoming a significant part of Slavic cultural heritage. This style experienced its golden age under Ivan the Terrible but eventually fell into disuse. Examples

[1]https://w.wiki/9Rdc

of this calligraphy can be seen in Figure 2. Despite sharing common features with other calligraphic families like Chinese [15, 17], Japanese [14], and various forms of Arabic calligraphy [4], Vyaz is distinctively different. Studies indicate that the graphical structures of calligraphic writing and the semantic structures of the text interact strongly, affecting the viewer's perception and providing an additional communication channel [1].

Slavic Vyaz is known for its variations—simple, complex, and patterned—which show the style's versatility and the creative techniques used, including ligature, diminution of letters, and placing smaller letters within larger ones. Ligatures, merging two or more characters into one, along with other local deformations, offer a distinctive way to create calligraphy [14]. However, these deformations are context-dependent and cannot always be explicitly described. Instead, end-to-end models [8, 9, 16, 18] or procedural methods [14] are often used for generating combinations.

As an advantage, procedural methods offer complete control over the process by parameterizing the interactions between elements. An example is the METAFONT language by Donald Knuth, which allows for the description of individual character outlines and ligature principles [7]. A common drawback of procedural methods is their predictability and monotony, which can be mitigated by character randomization[2], introducing variations naturally found in handwritten text to enhance uniqueness and reduce artificiality.

In contrast, end-to-end methods are convenient in their ability to replicate the original author's style provided with enough number of samples [10] without any additional specification. They can also use font distortions to convey emotions related to the text's emotional tone [2]. Nonetheless, end-to-end methods may lack controllability and interpretability in their results. Combined approaches also have gained popularity lately, offering the generation of content using machine learning models trained on existing content [11].

This paper presents a procedural technique for generating Slavic Vyaz, combining parameterized character descriptions with various ligature formation strategies, including randomized approaches. Inspired by Vyaz's ornamental style and complex letter intertwining, this project aims to digitally replicate these patterns, preserving and revitalizing interest in this unique cultural heritage for contemporary exploration and appreciation. We have made the code[3] for generating Vyaz, as well as the geometric descriptions used for both Latin and Cyrillic fonts, publicly available. As work in progress, the proposed technique is not yet thoroughly validated or evaluated; so we provide a set of illustrative examples that demonstrates the approach.

2 APPROACH

This article proposes an approach to the automated generation of patterns characteristic of Slavic Vyaz, including ligatures that preserve the rhythm of vertical strokes in character writing, as well as the deformation of individual characters and their integration into the spaces of larger ones.

The authors propose a Domain-Specific Language (DSL) to enrich the procedural generation of Slavic Vyaz. Similar to how DSLs have been effectively utilized in automated game design and procedural content generation (PCG) [13], this DSL is crafted to express the intricate patterns of Slavic calligraphy, focusing specifically on the formation of ligatures. The DSL for Character Descriptions enables detailed specification of how characters are deformed and intertwined, resembling systems such as L-Systems[12], Ludoscope[5], and Wave Function Collapse[6]. These systems are renowned for their use of patterns and tile replacement techniques, employing rewrite rules that allow the transformation of basic elements into complex patterns dynamically.

This DSL approach enhances our method by allowing the explicit definition of calligraphic rules, which are crucial for replicating the traditional aspects of Vyaz. By specifying rules that dictate how characters merge and interact, the DSL not only preserves the stylistic nuances of Vyaz but also facilitates the generation of new variations that maintain the essence of this ancient art form.

2.1 Character Description

The geometry of each character is described by the following formalism. The description consists of a list of anchors and a list of edges connecting them. Each anchor has a name and coordinates, using the vertical number (verticals are numbered with integers starting from 0) as the horizontal coordinate and a vertical position label (one of top, upper, middle, down, bottom). Edges are described by the names of two connected anchors, and a label for their mutual alignment, taking one of the following values: "<", "<=", "=", ">=", ">", determining the permissible vertical positioning of connected anchors.

For example, the character shown in Figure 3 consists of 11 anchors and 10 edges, described as follows:

anchors:

```
- lb:  [0, b]
- ld:  [0, d]
- lu:  [0, u]
- lt:  [0, t]
- mb:  [1, b]
- mm:  [1, m]
- mt:  [1, t]
- rb:  [2, b]
- rd:  [2, d]
- ru:  [2, u]
- rt:  [2, t]
```

edges:

```

- [lb, ld, '<']
- [lu, lt, '<']
- [rb, rd, '<']
- [ru, rt, '<']
- [mb, mm, '<']
- [mm, mt, '<']
- [lu, mm, '>']
- [ld, mm, '<']
- [ru, mm, '>']
- [rd, mm, '<']
```

2.2 Formation of Joints and Ligatures

To form joints and ligatures, we propose several heuristics that create a vector layout of the text, taking into account the geometry of characters and the general rules of articulation in Slavic Vyaz:

- The simplest method, 'plain', does not form joints but simply places characters on sequential verticals, one after another.
- Other methods attempt to form joints using the following meta-rules, where symbols can be combined if they can be shifted without overlap:
  - Without distorting the characters themselves (see Figure 4a).

[2]www.calligraphr.com/en/docs/faq/#faq-random
[3]https://github.com/altsoph/asciivyaz

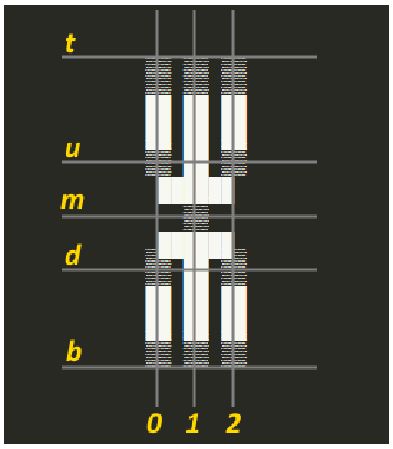

Figure 3: Character grid definition

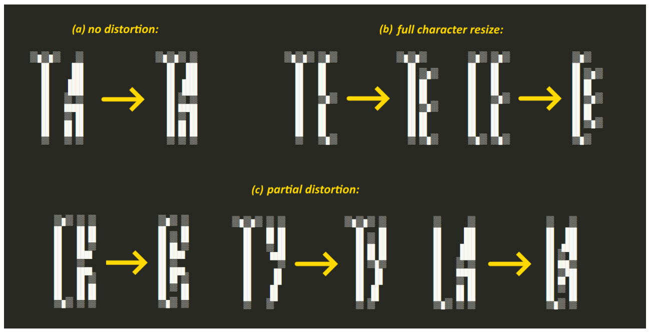

Figure 4: Formation of Joints

- By changing the vertical scale of one of the characters (see Figure 4b), where the permissible scale range is a font parameter.
- By shifting one or more of their anchors, while keeping the inequalities defined on edges valid (see Figure 4c), where the permissible distortion range is a font parameter.

These rules set constraints within which the algorithm seeks an optimal layout – minimizing both the final width of the text and the average distortion of characters. Given the potentially vast search space, we propose the use of several heuristics – greedy optimization, where we sequentially add one character and try to compress the current text, and random search methods in the parameter space.

The parameters being explored in the random search include the vertical positioning (high and low points) and distortion factors for each character. The algorithm attempts to find a combination that allows characters to be placed next to each other without overlap while minimizing vertical distortion and maintaining the aesthetic integrity of the text.

The utility function evaluates each configuration by considering the width (how much horizontal space the text occupies) and the distortion (how much the characters deviate from their original form); aggregation of distortion factors for individual characters is done by multiplication of their factors to get a combined measure

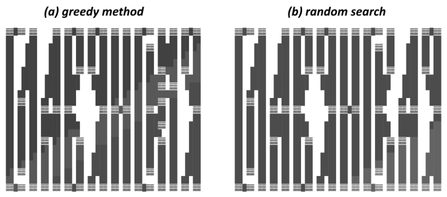

Figure 5: Greedy vs random seach.

of how much the overall text layout deviates from the ideal or original forms due to necessary adjustments. The score is based on a combination of these factors, often prioritizing configurations that achieve a compact layout with minimal character distortion.

In the greedy approach, the text compression technique chosen involves sequentially adding characters to the layout and trying to compress the existing text to fit the new character as close as possible. This method iteratively adjusts the positions of characters to reduce the overall width of the text block. Each new character is added based on its potential to fit into the current layout with minimal overlap and distortion, adjusting the position of previous characters as necessary to make space.

Figure 5 shows how random search sometimes gives aesthetically better results.

2.3 Font Description

The global font configuration thus consists of:

- Descriptions of individual characters.
- Global geometry parameters, including height (the pixel distance between top and bottom labels), the distance between verticals, and the desired (but not guaranteed) positioning of vertical anchor labels.
- Ligature formation algorithm settings – permissible distortions, the number of random search iterations, etc.

2.4 Rasterization

We used Bresenham's line algorithm [3] to convert the final vector layout of the inscription into a raster. The rasterization parameters are also algorithm parameters, but by default, we produce the result in the form of Unicode art, similar to utilities like figlet, cowsay, etc., allowing the result to be decorated with utilities like lolcat.

3 DISCUSSION

In this paper, we presented a procedural method for generating Slavic Vyaz, focusing on the creation of ligatures and the integration of character deformations to preserve the unique rhythm and vertical stroke dynamics characteristic of this calligraphy style. Our approach leverages a detailed font description, incorporating anchor points and edges to define the geometry of each character, thus enabling a flexible yet controlled way of forming connections and ligatures.

The implementation of various heuristics for text layout, ranging from simple sequential character placement to more complex strategies involving character scaling and anchor adjustments, demonstrates the potential of our method to generate calligraphy that

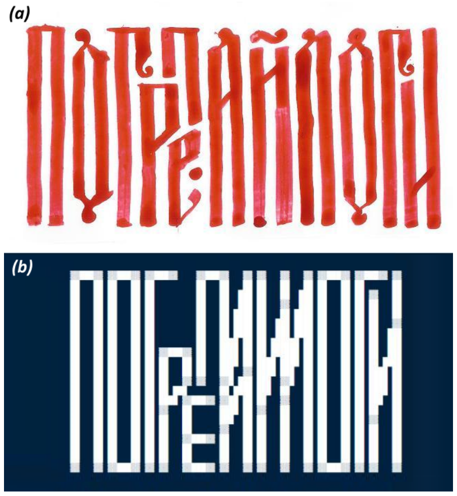

Figure 6: (a) the art work of Anna Shishlyakova; (b) the result of a random search for optimal layout.

respects traditional Slavic Vyaz patterns while allowing for creative variations.

The proposed technique is still under development and hasn't been thoroughly validated or evaluated yet. To illustrate our method, we refer to Figure 6. The top part, Figure 6a, showcases Anna Shishlyakova's work, while the bottom part, Figure 6b, shows an example from a random search for optimal layout. Both examples use similar joint strategies, but our goal with automated calligraphy isn't to replace human creativity. Instead, we want to enhance it by making traditional art forms more accessible and encouraging artistic experimentation. Figure 7 shows more examples of Vyaz texts generated by our tool.

Looking ahead, we plan to improve our algorithm to better mimic the detailed variations of hand-written calligraphy and to expand it to include Arabic and Western scripts. This will help us cater to a wider range of artistic styles. We also aim to make our tool easier for those without a background in digital typography or programming to use. Furthermore, we hope to work with contemporary artists to see how our tool can contribute to modern art projects, combining traditional calligraphy with new media to showcase Vyaz's adaptability in modern art.

ACKNOWLEDGMENTS

This project owes an inspiration to the calligraphy work of Anna Shishlyakova, that sparked the initial curiosity and subsequent exploration into the automation of Slavic Vyaz ligatures.

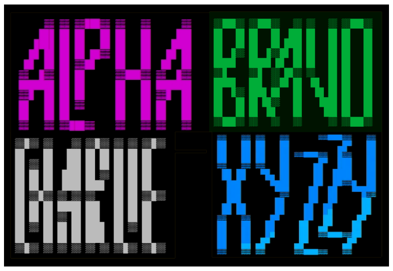

Figure 7: A few more examples of generated Vyaz.

REFERENCES

[1] Kaddour Abdallah-Tani and Abdallah K. 2022. Visual semiotics in the structure of Kufic calligraphy. *International Journal of Visual and Performing Arts* Vol 3 (02 2022), 110–116. https://doi.org/10.31763/viperarts.v3i2.516

[2] Yana Agafonova, Alexey Tikhonov, and Ivan P. Yamshchikov. 2020. Paranoid Transformer: Reading Narrative of Madness as Computational Approach to Creativity. *Future Internet* 12, 11 (2020). https://doi.org/10.3390/fi12110182

[3] J. E. Bresenham. 1965. Algorithm for computer control of a digital plotter. *IBM Systems Journal* 4, 1 (1965), 25–30. https://doi.org/10.1147/sj.41.0025

[4] Zineb Kaoudja, Mohammed Lamine Kherfi, and Belal Khaldi. 2021. A New Computational Method for Arabic Calligraphy Style Representation and Classification. *Applied Sciences* 11, 11 (2021). https://doi.org/10.3390/app11114852

[5] Daniel Karavolos, Anders J. Bouwer, and Rafael Bidarra. 2015. Mixed-Initiative Design of Game Levels: Integrating Mission and Space into Level Generation. In *International Conference on Foundations of Digital Games*. https://api.semanticscholar.org/CorpusID:7864753

[6] Isaac Karth and Adam M. Smith. 2017. WaveFunctionCollapse is constraint solving in the wild. In *Proceedings of the 12th International Conference on the Foundations of Digital Games* (Hyannis, Massachusetts) *(FDG '17)*. Association for Computing Machinery, New York, NY, USA, Article 68, 10 pages. https://doi.org/10.1145/3102071.3110566

[7] Donald E. Knuth. 1989. *The Metafont book*. Addison-Wesley Longman Publishing Co., Inc., USA.

[8] Qisheng Liao, Zhinuo Wang, Muhammad Abdul-Mageed, and Gus Xia. 2023. CalliPaint: Chinese Calligraphy Inpainting with Diffusion Model. arXiv:2312.01536 [cs.CV]

[9] Gan Lin, Zhihua Guo, Fei Chao, Longzhi Yang, Xiang Chang, Chih-Min Lin, Changle Zhou, V. Vijayakumar, and Changjing Shang. 2021. Automatic stroke generation for style-oriented robotic Chinese calligraphy. *Future Generation Computer Systems* 119 (2021), 20–30. https://doi.org/10.1016/j.future.2021.01.029

[10] Vittorio Pippi, Silvia Cascianelli, and Rita Cucchiara. 2023. Handwritten Text Generation from Visual Archetypes. arXiv:2303.15269 [cs.CV]

[11] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K. Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural Content Generation via Machine Learning (PCGML). arXiv:1702.00539 [cs.AI]

[12] Julian Togelius, Noor Shaker, and Joris Dormans. 2016. *Grammars and L-systems with applications to vegetation and levels.* 73–98. https://doi.org/10.1007/978-3-319-42716-4_5

[13] Riemer van Rozen. 2021. Languages of Games and Play: A Systematic Mapping Study. *ACM Comput. Surv.* 53, 6, Article 123 (dec 2021), 37 pages. https://doi.org/10.1145/3412843

[14] Lisong Wang, Tsuyoshi Nakamura, Minkai Wang, Hirohisa Seki, and Hidenori Itoh. 1997. A method of generating calligraphy of Japanese character using deformable contours. In *Proceedings of the Fifteenth International Joint Conference on Artifical Intelligence - Volume 2* (Nagoya, Japan) *(IJCAI'97)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1050–1055.

[15] Yuanbo Wen and Juan Alberto Sigüenza. 2020. Chinese Calligraphy: Character Style Recognition based on Full-page Document. In *Proceedings of the 2019 8th International Conference on Computing and Pattern Recognition* (Beijing, China) *(ICCPR '19)*. Association for Computing Machinery, New York, NY, USA, 390–394. https://doi.org/10.1145/3373509.3373512

[16] Ruiqi Wu, Changle Zhou, Fei Chao, Longzhi Yang, Chih-Min Lin, and Changjing Shang. 2020. GANCCRobot: Generative adversarial nets based chinese calligraphy robot. *Information Sciences* 516 (2020), 474–490. https://doi.org/10.1016/j.ins.2019.12.079

[17] Songhua Xu, F.C.M. Lau, W.K. Cheung, and Yunhe Pan. 2005. Automatic generation of artistic chinese calligraphy. *IEEE Intelligent Systems* 20, 3 (2005), 32–39. https://doi.org/10.1109/MIS.2005.41

[18] Peichi Zhou, Zipeng Zhao, Kang Zhang, Chen Li, and Changbo Wang. 2021. An end-to-end model for chinese calligraphy generation. *Multimedia Tools and Applications* 80, 5 (01 Feb 2021), 6737–6754. https://doi.org/10.1007/s11042-020-09709-5