Typography and Storytelling

http://www.typoday.in

Creative Coding in Devanagari Typography

Tejas Nerlekar, Ektype, nerlekarstejas@gmail.com

Abstract: Creative coding—the use of code to generate designs—has expanded the possibilities of computer graphics and typography. While widely explored in Latin-based scripts, its potential in Devanagari remains largely untapped. This paper documents a personal exploration of creative coding in Devanagari typography, focusing on hands-on experiments rather than a technical or theoretical study. Through a series of computational experiments in typographic compositions, letter drawing, and interactivity, it examines the unique opportunities and challenges of integrating coding into the design process. By sharing these explorations, the paper aims to contribute to the ongoing conversation around digital tools in Devanagari typography and inspire new approaches to dynamic and generative letterform design.

Key words: Creative Coding, Computational Typography, Devanagari Typography, Tool Exploration.

1. Introduction

Creative coding is a practice where code is used to generate creative works. It has transformed computer graphics, enabling designers to move beyond the limitations of traditional software and to create dynamic, innovative visuals. Typography is a prominent theme in creative coding, influencing posters, motion graphics, and brand identities. However, its potential for exploring the Devanagari script remains largely untapped.

This paper presents the author's explorations with creative coding and Devanagari typography. It examines why creative coding was chosen as a medium, what outcomes emerged from these experiments, what insights were gained, and what possibilities lie ahead. It is a case study, of explorations of creative coding as a tool focusing specifically on Devanagari typography. While coding-based typographic experiments are often discussed in the context of the Latin script, Devanagari has its own unique structure and aesthetic. This paper does not seek to compare Devanagari with Latin or other scripts. Instead, it explores how creative coding can serve as a tool for typographic expression, working within Devanagari's distinct visual and structural logic.

2. Learning To Code

Typographic designs created through creative coding have a distinct look and feel. They often appear dynamic, with letters transformed using colour, motion, and shape in ways that make them visually striking. Achieving such designs with conventional design software can be difficult, and in some cases, not even possible. This novelty is what initially attracted the author to creative coding. It sparked curiosity about how a set of coded instructions could generate complex visual compositions. Additionally, the ability to create designs beyond the constraints of existing software encouraged an exploration of new ways to experiment with letterforms.

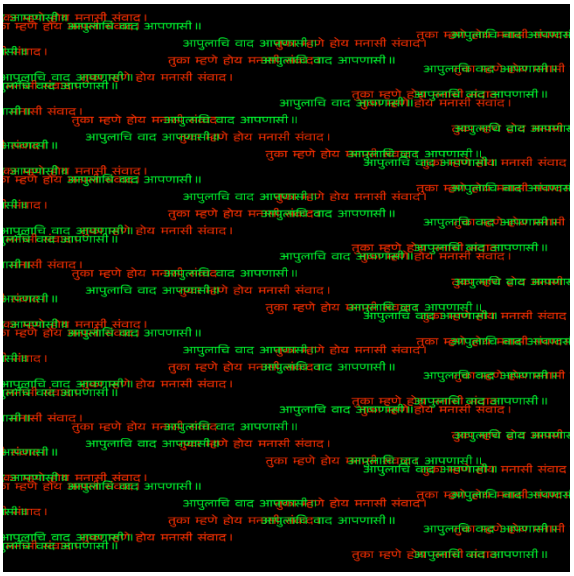
Driven by this curiosity, the author began exploring creative coding in the context of Devanagari typography. These explorations were carried out using p5.js[1], a JavaScript library designed for creative expression. There was no predefined methodology or rigid objective—rather, the focus was on experimenting with the tool itself. As a result, the outcomes took the form of iterative sketches rather than structured projects. Over time, these sketches could be broadly categorized into three groups based on their nature and approach.

[1] P5.js. https://p5js.org/

3. Exploring Creative Coding with Devanagari

3.1 Text Position and Text Style

In typographic composition, key parameters such as position, colour, size, rotation, and stroke play a crucial role in shaping the overall visual outcome. This group of sketches explores these parameters using creative coding, demonstrating how computational techniques can influence typography.



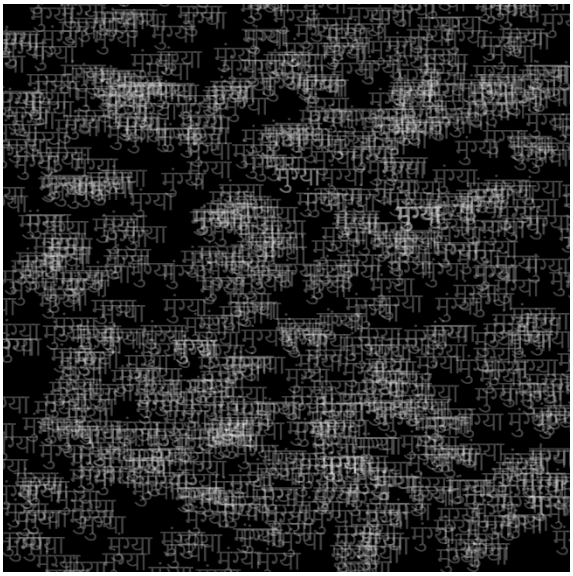
sketch.01: tukā mhane

sketch.02: matadāna

Note: All sketches shown in this paper are static frames from the videos.

At first glance, text position and style may seem like obvious parameters. However, when approached through coding, they reveal unexpected possibilities. The use of repetition, randomness, and oscillating waves is prominent in these sketches, allowing motion to be incorporated effortlessly. For example, in *sketch.01 and sketch.02,* repetition and wave functions manipulate the position and size of the text, creating structured and rhythmic compositions. In contrast, *sketch.03, sketch.04, sketch.05 and sketch.06* generate visuals that feel uncontrolled due to randomness, where parameters shift dynamically based on mathematical expressions.

sketch.03: gumtā



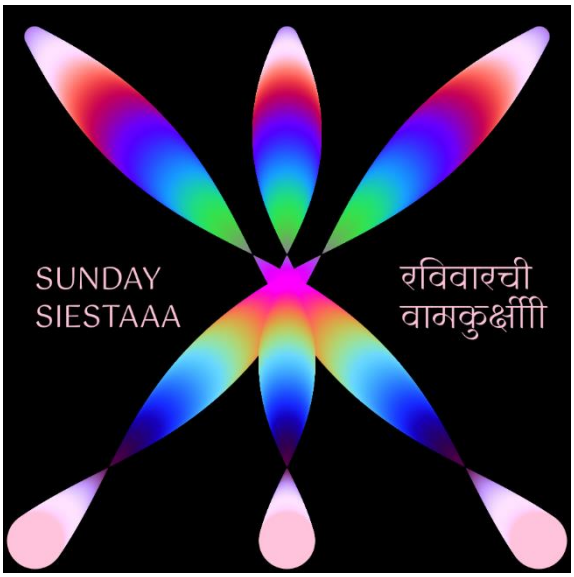
sketch.04: mungyā

sketch.05: sitārā 01

sketch.06: sitārā 02

These mathematical expressions can be applied in various ways. While some sketches use waves to alter text position (e.g., *sketch.02: matadāna*), others, such as *sketch.07 and sketch.08*, use waves to control colour, producing moving gradients and fluid transitions. Coding not only enables a deeper understanding of these individual parameters but also allows for precise control either in isolation or through interconnected relationships. Moreover, these sketches are not just standalone compositions; they represent adaptable

concepts. With minor adjustments to the code without introducing new instructions multiple variations of the same idea can be generated. This flexibility is one of the key strengths of creative coding as a tool for typographic exploration.



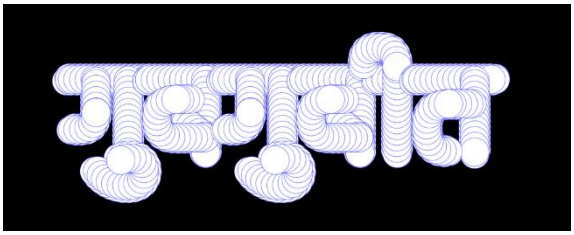
sketch.07: vāmakuksī

sketch.08: deśī videśī

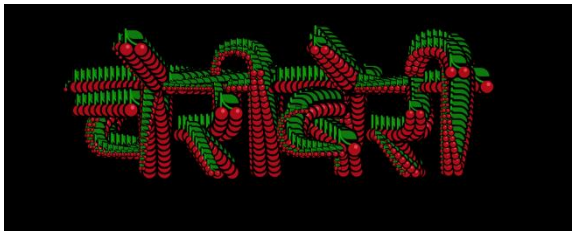
When the author began the exploration, he anticipated text rendering issues with Devanagari in creative coding. However, in most cases, Devanagari rendered correctly. Complex features like substitution, positioning, anchoring worked perfectly. Some specific scenarios like splitting a text string or parsing a font file, lead to problems. But there are temporary solutions available for such scenarios. One of these is mentioned later in the paper. But largely the text shaping for Devanagari works fine.

3.2. Letter Drawing

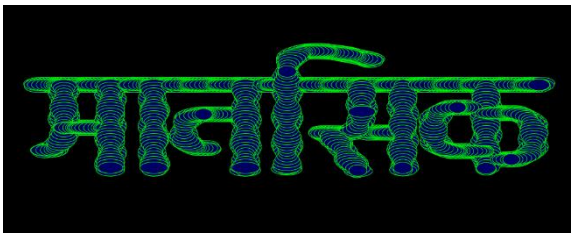
While the previous group of sketches focused on composition without significantly altering letterforms, this set explores the process of drawing letters through code. It examines how letter shapes can be constructed and modified computationally.



sketch.09: gubgubit



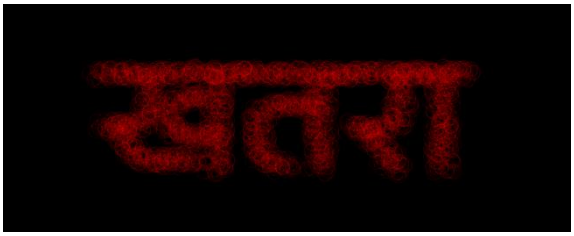
sketch.10: cherry berry



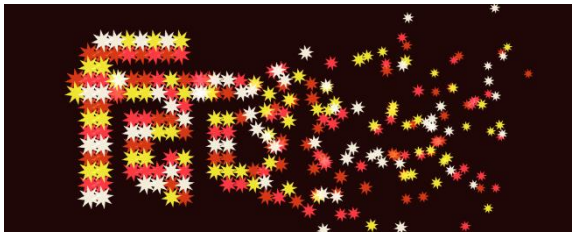
sketch.11: mānāsika

sketch.12: khuśbū

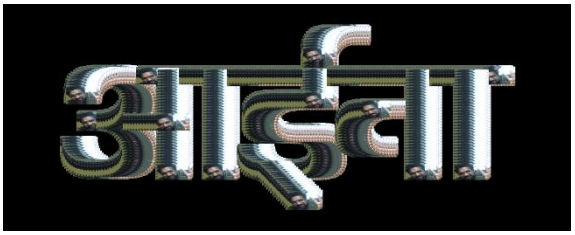
At first glance, these sketches introduce new ways of drawing letterforms. They are created by placing various elements along the path or outline of the text. In some cases, these elements repeat in a structured manner, creating a brush-like effect, as seen in *sketch.09* and *sketch.10*. In contrast, sketches like *sketch.12*, *sketch.13, sketch.14* feature irregular repetitions, resulting in more organic and unpredictable forms. Since these elements are generated through code, they remain dynamic and adaptable.



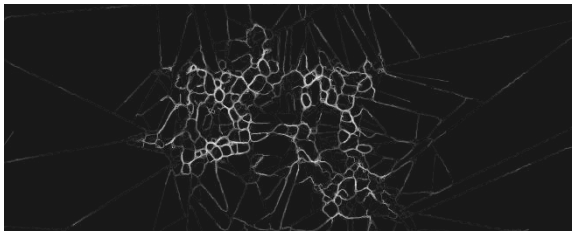
sketch.13: khatarā



sketch.14: Diwali



sketch.15: āīna



sketch.16: **jantū**

One particularly unique example is *sketch.15: āīna*, which incorporates live webcam input to create a mirrored effect within the text. The interplay of repetition and distortion makes the result feel abstract yet engaging. Another notable sketch, *sketch.16: jantū*, takes a different approach. It has an organic quality, utilizing a particle system that grows dynamically, forming a web-like structure. This effect is achieved by writing instructions that control the behaviour of elements and looping them to generate complex visual patterns. These explorations highlight the vast possibilities of coding as a tool for letter drawing, where instructions and iterative processes can produce unexpected and compelling typographic results.

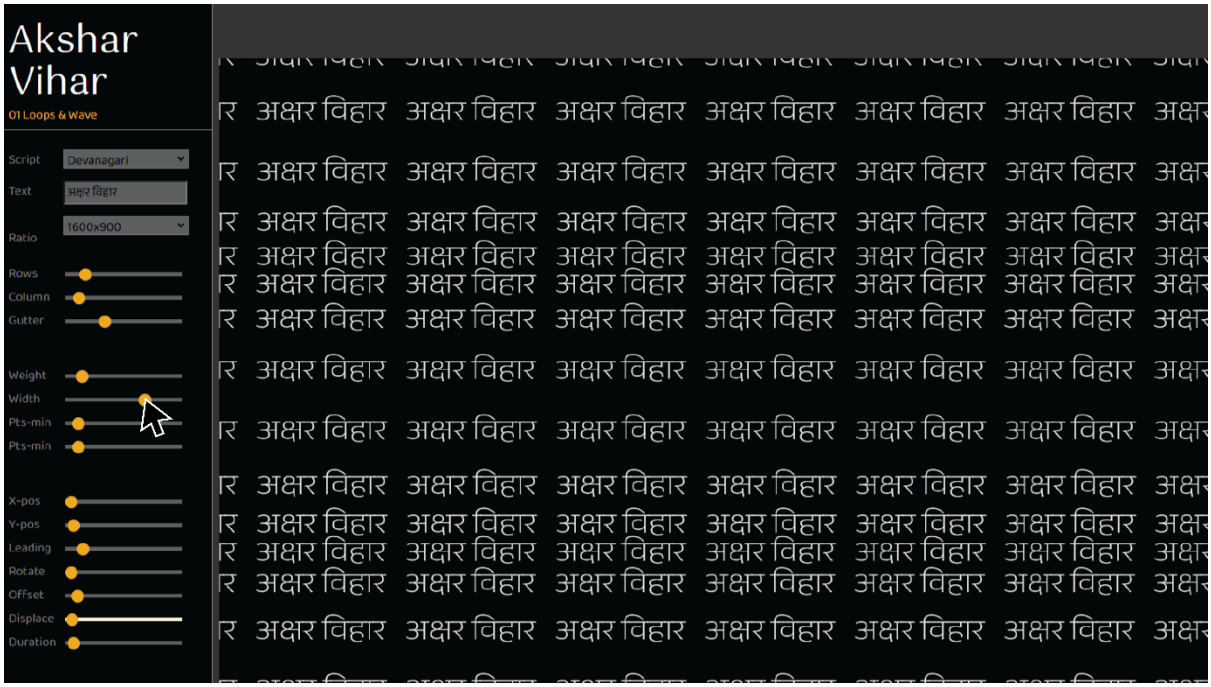
The sketches shown in this section were created in p5.js using a function called textToPoints()[2]. This function parses[3] the font file, converts the outlines of the glyphs into paths, and generates points along those paths. It relies on the opentype.js[4] library for font parsing. However, opentype.js does not support the features required for correct shaping of Indian scripts, creating challenges for Devanagari.

To address this, the author used font design software to create full words as single glyphs and loaded them as a font file. This ensured that textToPoints() processed entire words rather than individual letters. However, this is not a scalable solution. Another approach could be to access the pixel data of the rendered text instead of relying on outline paths, but this does not offer precise control. This remains a challenge for Devanagari typography, requiring further research to develop a more reliable method for parsing fonts and handling Devanagari text in path-based rendering.

[2] *TextToPoints*. (n.d.). https://p5js.org/reference/p5.Font/textToPoints/
[3] Font Parsing: Parsing a font refers to the process of analyzing and interpreting the font file's data structure to extract information such as glyph outlines, kerning, OpenType features, and character mappings.
[4] *opentype.js – JavaScript parser/writer for OpenType and TrueType fonts.* (n.d.). https://opentype.js.org/

3.3. Responsiveness & Interactivity

Beyond static compositions and letter drawing, creative coding introduces the possibility of interactivity. This allows the audience to engage with typography in real time rather than simply observing it.



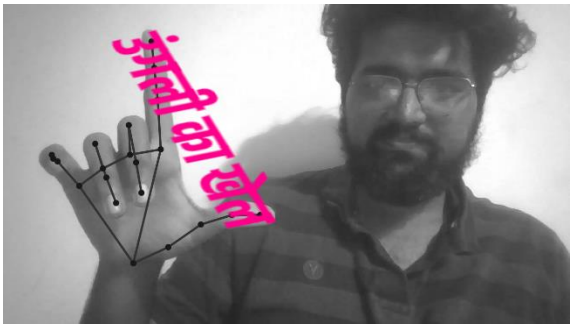
sketch.17: Akshar Vihar

Akshar Vihar[5], *sketch.17*, is a multiscript typography tool made by the author. He was inspired by the space-type generator[6] by Kiel Mutschelknaus[7] and wanted to make such a tool in Devanagari. In Akshar Vihar viewers can manipulate typography using sliders, enabling them to create their own compositions. The tool presently supports ten Indian scripts. It is based on Anek font family[8] by Ektype and supports variable axis as well. It not just allows the viewer to make their own compositions, but also export animations of the compositions in various aspect ratios as a GIF. This is an example of how the interactivity

[5] Akshar Vihar. https://tejasnerlekar.github.io/Akshar-Vihar/
[6] *SPACE TYPE GENERATOR*. (n.d.). https://spacetypegenerator.com/
[7] https://www.kielm.com/
[8] *Anek family*. (n.d.). https://ektype.in/anek-family.html

enabled through coding turns a simple typographic sketch into a tool.

sketch.18: namaste



sketch.19: uṅgalī kā khel

However, interactivity in creative coding is not limited to sliders—it can also respond to motion, sound, or other inputs. In *sketch.18 and sketch.19*, the author integrates hand tracking with typographic parameters, resulting in responsive compositions where letters move dynamically in sync with hand gestures. These sketches are done by using open sources frameworks like Media-pipe[9] and ml5.js[10]. This level of interactivity shifts the viewer's role from passive observer to an active participant, creating a dialogue between the audience and the design. Interactivity is an important part of the experience of communication. It makes the communication more life-like.

Such interactions are possible because computers interpret different media—audio, video, text, etc.—as structured data. Through code, these media can be accessed, manipulated, and interconnected, opening up new possibilities for transmedia design and dynamic typographic expression.

These are the three main approaches the designer explored. Throughout these explorations, the author created several sketches—some worked well, while others did not. He referred to resources from GitHub, YouTube, and creative coding communities. Each sketch contributed to learning new coding techniques, deepening his understanding of how code handles visuals and text. These explorations also revealed how Devanagari letters behave in such a visual environment.

[9] Hand landmarks detection guide. (n.d.). Google AI for Developers. https://ai.google.dev/edge/mediapipe/solutions/vision/hand_landmarker
[10] ml5 - A friendly machine learning library for the web. (n.d.). https://ml5js.org/

4. Insights from Explorations

Coding is a vast and powerful tool for typographic exploration. The author's journey into creative coding stemmed from a curiosity about how a set of instructions can generate visual designs and what new possibilities coding opens up for working with letterforms. This journey of exploration led to insights about coding with Devanagari typography.

4.1 Coding To Make Visual Designs

When working in traditional design software, a designer adjusts parameters through a graphical user interface (GUI), and the software's prewritten code generates the visuals accordingly. However, in creative coding, the designer must not only visualize the artwork but also construct the underlying logic that instructs the computer to create it. This shift offers greater control over the structure of the code and the relationships between different design elements. Through coding, a designer can experiment with the interconnections between parameters, apply mathematical expressions, and integrate various media and data sources into the design process. This approach provides a deeper understanding of letterforms as data—how they can be manipulated, how computers interpret them, and what constraints and possibilities exist within different parameters. In simple words coding helps the designer to go beyond the limitations of the software. It gives freedom and increases ability of the designer's creative process.

4.2. New Possibilities In Devanagari Typography

The current explorations demonstrate how coding can be used to create typographic compositions, but they represent only a fraction of the possibilities. Many of these sketches focus on single words, phrases, or repetitive patterns, but typography extends far beyond that—it encompasses everything from individual letters to long-form text. Future explorations should move beyond isolated compositions to investigate how coding can influence typography at different scales, such as paragraphs, books, websites or even identity. Each of these contexts presents unique constraints and opportunities for creative expression.

Additionally, deeper explorations into letter drawing could benefit from an understanding of data structures—specifically, how fonts are stored and manipulated as data. This knowledge would enable designers to edit fonts through code and get access to Bezier curves open type features, variable axes, etc directly from the code. This can unlock new ways to modify letterforms and create unexpected variations. Enhancing technical proficiency in

this area could open new practices in typography and type design, making it a valuable direction for further study. The potential of coding in typography is vast, but for now, these two key areas—expanding typographic exploration beyond single word compositions and studying data structures to manipulate letterforms—are the main directions the author intends to pursue in his future work.

4.3. Influence Of Tool On Thinking

A tool is not merely an instrument for execution—it influences the designer's thought process. When one practices calligraphy for some time, letters begin to appear as a series of strokes rather than just shapes. This shift in perception helps in ideation, execution, and analysis of a letterform. Similarly, engaging with coding as a design tool reshapes how one approaches typography. It helps the designer to develop a computational way of thinking.

This mindset helps one to break down complex systems into smaller components, identify patterns, and write code that can systematically generate them. It also shifts one's perspective on computers—from being seen as magical, all-powerful machines to understanding them as tools that follow structured instructions.

In an era where digital technologies shape communication, this computational thinking can be very valuable. It can enable contemporary, dynamic approaches to typography, allowing designers to create visuals that resonate with local content and context.

5. Conclusion

This paper contributes to the conversation around modern tools and Devanagari typography by offering an approachable introduction through a case study. It is a documentation of personal explorations—an attempt to understand and experiment with creative coding in the context of Devanagari. While these experiments may not always be groundbreaking, they reflect a designer's process of inquiry, trial, and discovery. By sharing these explorations, the paper aims to open up discussions on how creative coding can intersect with Devanagari typography and inspire further investigation.

However, this is only a starting point. The intersection of creative coding and Devanagari typography remains largely unexplored, and there is a need for deeper and more systematic investigations. Future explorations could expand beyond individual

compositions to consider long-form text, responsive typography, and computational type design. By continuing to experiment with code, designers can push the boundaries of Devanagari typography in ways that traditional tools may not allow, unlocking new possibilities for digital and interactive design.

References

P5.js. https://p5js.org/.

Conrad, D., Rob, V.L. and Héritier, D. (2021) Graphic Design in the post-digital age : a survey of practices fuelled by creative coding. https://arodes.hes-so.ch/record/10049?v=pdf.

Thottingal, S. and Swathanthra Malayalam Computing (2023) 'Modernizing Parametric type design: A case study of Nupuram Malayalam typeface,' *Typoday 2023* [Preprint]. https://www.typoday.in/2023/spk_papers/Santhosh_Thottingal_Typoday 2023.pdf.

Tim (2025) *Omid Nemalhabib explores the intersection of Creative Coding and Perso-Arabic Typography*. https://timrodenbroeker.de/omid-nemalhabib/.

Tejas Nerlekar. (n.d.-a). *GitHub - tejasnerlekar/Akshar-Vihar: Akshar Vihar, a creative typography tool built with p5.js, designed to explore Indian scripts in motion and grid layouts*. GitHub. https://github.com/tejasnerlekar/Akshar-Vihar