CODE AS A CREATIVE MEDIUM FOR VARIABLE FORMATION IN GRAPHIC DESIGN.

EL CÓDIGO COMO MEDIO CREATIVO PARA LA FORMACIÓN VARIABLE EN DISEÑO GRÁFICO.

Dr. Kyuha Shim
shim.kyuha@gmail.com
ORCID: 0000-0002-8006-5693
Carnegie Mellon University
EE.UU.
Autor para la correspondencia

RESUMEN

Hoy en día, gran parte del diseño gráfico que encontramos se crea, distribuye y experimenta a través de medios informáticos. Los diseñadores pueden usar computadoras más allá del software estándar, particularmente para escribir su propio software. Sin embargo, la investigación sobre el código, un medio computacional en bruto para dar formas generativas, aún está ausente en gran medida en el contexto del diseño gráfico (Bartlett y Tatum 2020; Shim 2020; Shim 2021). El ensayo de este profesional incluye una revisión de la literatura y un examen del trabajo de diseño gráfico relacionado para establecer los antecedentes contextuales de los enfoques computacionales en el diseño gráfico, en particular aquellos que definen la programación como material y proceso para diseñar sistemas generativos. Los ejemplos de tales sistemas generativos en el trabajo de este profesional producen variaciones visuales manipuladas algorítmica y paramétricamente que son dinámicas para la entrada del usuario, las fuentes de datos externas y la aleatoriedad. La clave para adoptar un enfoque computacional en el diseño gráfico y generar "formaciones variables", o el proceso de construcción de sistemas generativos que emergen formas dinámicas utilizando código y datos, es comprender y aplicar el código como un medio creativo. Como tal, este ensayo reflexivo demuestra cómo se utiliza la computación como medio de diseño gráfico para crear variabilidad en la apariencia y el comportamiento de las formas visuales.

ABSTRACT

Today, much of the graphic design we encounter is created, distributed, and experienced through computational media. Designers can use computers beyond standard software, particularly for writing their own software. However, research on code a raw computational medium for generative form-giving is yet largely absent within the context of graphic design (Bartlett and Tatum 2020; Shim 2020; Shim 2021). This practitioner's essay includes a literature review and an examination of related graphic design work to establish the contextual background of computational approaches in graphic design, particularly those that define programming as both material and process for designing generative systems. Examples of such generative systems in this practitioner's work produce algorithmically and parametrically manipulated visual variations that are dynamic to user input, external data sources and randomness. Key to taking a computational approach in graphic design and generating variable formations, or the process of building generative systems that emerge dynamic forms using code and data, is understanding and applying code as a creative medium. As such, this reflective essay demonstrates how computation is used as a graphic design medium to create variability in the appearance and behavior of visual forms.

Palabras claves:

Diseño Computacional,
Formación Computacional,
Diseño de Programas,
Diseño Gráfico,
Formación Variable.

***Keywords**:*

Computational Design,
Computational Formation,
Designing Programs,
Graphic Design,
Variable Formation.

Fecha Recibido:
31 / 03 / 2022
Fecha Aceptación:
03 / 05 / 2022
Fecha Publicación:
12 / 07 / 2022

INTRODUCTION

In my design practice, I take computational approaches by using code as a creative medium to generate systems for computational visual formation that yield dynamic graphic artifacts.[1] In taking this approach, I am exploring (1) *how forms emerge* and (2) *the interactions / relations between form and data*. The systems convert data into form according to the rules written in code. In this way, the artifacts not only become diverse in their appearances but also acquire continuously transformative behaviors.

THE CURRENT OF COMPUTATIONAL DESIGN WITHIN GRAPHIC DESIGN

For many years, creative coding environments have enabled artists and designers to create their own software through programming languages and the use of the computer as a 'meta-medium' rather than a tool (Manovich, 2013). The existing research in graphic design on the practice of "creating in the raw computational medium" (Maeda, 1999), since its explication two decades ago, is limited to computationally produced work categorized as digital design or future practice without particular focus on the medium. The understanding that computation is a design medium was sparked by graphic designer Muriel Cooper and her students at the MIT Media Lab. Cooper's work was concentrated on designing interfaces that enable users to navigate information in an interactive way. Cooper's Information Landscape (1994) utilized time and interactivity beyond graphic design to produce a dynamic representation of information through texts and images in three-dimensional virtual space (Small, Ishizaki and Cooper 1994). Although the context was user centric, Cooper's research explored how designers can manipulate form and present information not only in new ways on screen but also through various printing methods. Instead of designing computer aided tools for other users, the next generation delves into the proposition where designers use creative coding as a way of design.

Succeeding the legacy of Cooper, Maeda and his students at the Aesthetics Computation Group (1996–2003) at MIT's Media Lab made significant contributions on the subject of using code to build custom software for creative and expressive purposes. A notable example is Maeda's *Reactive Books* (1994–1999) which demonstrated unconventional relationships between various device inputs and visual outputs. In *Morisawa 10* (1996), a poster series using Morisawa's logo, he presented hyper complex and mathematical aesthetics that were unachievable without computation. His students demonstrated that exploring with computation is essential for the designer who wishes to free themself from the reinforced paradigms of existing digital tools and discover creative possibilities in design. They produced dynamic and interactive work that ranged from typography, book and installation design, to information design.[2] As both designers and programmers, they paved new ways of integrating computational methods into the design process and encouraging more designers to write code for building their own tools and creative aesthetics. Moreover, in 2004, the birth of Casey Reas and Ben Fry's *Processing*, an open-source programming environment, encouraged more designers to create their own tools for creative aesthetics. As the creative coding movement grew, graphic designers slowly began to see opportunities from code-driven design practice.

In the early 2000s, it is recognized that the process of using programming languages (or writing code) yielded new aesthetics in graphic design based on a very small number of designers using code (Küsters and King, 2001). Prominent research such as Hillner's *Virtual typography* (2009) and Brownie's *Transforming type* (2015) included few figures of the work computationally made when discussing flexibility and fluidity of kinetic forms, and yet published research on the computational medium was still limited. As an increasing number of designers incorporate the computational approach in their work, building one's own program was identified as a characteristic of the "new generation" of graphic designers' work (Blauvelt, 2011). At the time, a number of graphic design practitioners speculated that writing code would yield new design opportunities, such as generative typography that self-transforms upon real-time changes in data (Giampietro, Jen, Krishnamurthy, and Van Blokland, 2013). It is similar to how the variations in data shapes the graphs in data visualization. Designers can devise computational algorithms using data as input parameters to continuously vary visual parameters. Today, computational design has stirring implications for graphic design, particularly in designing systems that generate variable artifacts (Shim, 2020). It requires a combination of historical and practice-led investigative approaches to characterize and define the medium.

The practice of authoring software programs is based on crafting algorithms and parameters for giving forms. Designers write algorithms that instruct a computer to generate forms, rather than craft the forms themselves. The first conceptualization of programming for graphic design was described as making selections, including making components into parameters, specifying new components, and combining them (Gerstner, 2007[1968]). Most significantly, creative processes through programming transformed the focus of graphic design, from 'solutions for problems' to 'program[s] for solutions' that can encompass 'a group of solutions [selected ...] under certain conditions' (Gerstner, 2007). This highlighted that building programs can encompass various solutions for each different condition, as opposed to designing one solution at a time. The process of programming for design entails using parametrization of decisions as well as rules that translate ideas into generalized patterns, comparable to formulating an equation. More importantly, it is aimed to systematically shape the patterns among a variety of solutions generated by programs. While this conceptualization integrates the design process with

[1] This approach has transformed how I use and interact with tools. See, Shim, K. "A tool beyond a tool: computational graphic designer's means, media and methods." (working paper).

[2] For examples of work in each category, see, respectively, (Cho 1999), (Small 1999), and (Fry 2004).

programming, it came before a time when designers began to write code that instructs a computer to produce design artifacts.

Like Gerstner's proposal, a program can yield various outputs based on the selection of parameters with different values. Today, in a time of automation and exponentially increasing computing powers, programming can be used as a graphic design medium for generative systems that can produce artifacts in response to dynamically changing parameters. Unlike an efficient and optimized operation reserved for computer scientists, constructing algorithms and writing code to generate dynamic visual variations can become a creative pursuit in exploring and building computational formations. Instead of manually determining and applying values to make the changes in their designs, designers can write code to make computers vary the values automatically. The computational approach brings in absolute accuracy and considerably rapid execution. Based on automation, designers can create their own tools/systems with which they can control and manipulate visuals dynamically and create interactive visual experiences. This essay further investigates computational variable visual formation based on three different types of input parameters including pseudo-randomness, device input, data each of which affords new characteristics infinite, responsive, and data-driven.

THREE TYPES OF COMPUTATIONAL FORMATIONS

The first type of input in computational formations is pseudo-randomness, which enables autonomous generation of myriad visuals. It can uniformly produce random numbers in a range endlessly. In my work, it has allowed the manifestation of perpetually transformative visual identities. For example, in Creative Mind (2011), I aimed to communicate the idea of creativity as a fluid and evasive inspiration by showing numerous permutations of visual forms that are variant yet cohesive, building one uniform formation (Figure 1). I determined some of the variable elements in the sketch of a static form, then wrote code to algorithmically manipulate the individual visual elements pseudo-randomly in Processing. To yield systematic and coherent changes, I constructed rules that determine each part's movement in ranges. Every few seconds, all the visual elements (i.e., two dots, "creative," "mind") were unexpectedly and playfully repositioned back and forth within a few predefined positions with randomized speeds (Figure 2). I used randomness to change and choose a value within the predefined range. It was like flipping a coin or dice, but the computer was able to simulate it perpetually. The total number of static variations was 32 (160 inclusive of colors); during the program's runtime (over time while the program is executed), however, the resulting visual identity had endless flow due to autonomous and uncertain transitions (Figure 3). Unlike playing a video file containing pre-designed motions, the project demonstrates that computational formation with randomness can yield infinite flow of the movements.

In another example, Work-in-Progress (2014), I aimed to represent the idea of coexisting moments of completion and incompletion seamlessly and ceaselessly, without any determined states of beginning or end (Figure 4). To manifest the idea, I conceived a formation that generates continuously fluid outcomes rather than one that repetitively shuffles within finite options. I initially built algorithms that distribute points along the typographic path, then randomly alter the points to generate myriad variations of kinetic lines. Instead of dividing each path into a few lines mathematically, I constructed each letter through the overlay of multiple lines, each of which randomly varies on its own over time (Figure 5). The length, number, and speed of the lines were varied programmatically, which allowed me to explore and discover the scope of legible variations by tweaking the numbers. Instead of many but easily distinguishable variations (like Creative Mind), the formation in Work-in-Progress presented one continuously variable and seamless outcome throughout (Figure 6). These projects demonstrate that computational formation with pseudo-randomness enables ever-shifting and highly unexpected variations. Depending on the formations for which it is designed, the patterns of transformation created may appear repetitive or even seamless and unrecognizable.

The second type of computational formation takes human input (data or signal) from hardware input devices and uses them to generate variations. Unlike the first input type, where the results were autonomously and solely made by machines, this formation generates outcomes reactive to human input data. In my work, it has enabled the exploration of poetic visual responses in experimental generative typography. For example, in Spoken Words (2017), I aimed to express the human quality in speech beyond the written text generated by speech recognition API. To demonstrate the concept of liveness, I created an algorithm that stores the amplitude and frequency data of a person's speech input and generates sound wave-like kinetic letterforms by duplicating letters based on the loudness and varying the opacity of letters based on the pitch (Figure 7). The formation could result in incredibly various gestures in typography reflective of a person's speech. The speech was recognized and visualized after a brief pause. Due to the pause, participants tended to view the machine's interpretation of their inputs rather than control it by changing their inputs while seeing visual responses.

In another example, Performa (2014), I explored the formation that embodies the human performance of typing in typography through an experimental typewriter that exposes certain tensions and speed of the writing hand (Figure 8). Beyond displaying pressed keys on the page, Performa altered the width and weight of the letterform depending on how the key was pressed. The width of a typed letterform was extended in response to the time elapsed between keyPressed() and keyReleased() events (Figure 9). Its weight was expanded upon the measured pressure of the key press. However, speed and pressure are not existing parameters; therefore, I had to craft them by storing and calculating data whenever a key is typed. For example, I came up with a parameter "speed" by measuring the elapsed time between the events, and for "pressure," I interpreted the vector (x, y, z) of the gyro sensor built-in an Apple Macbook, which did not provide as precise of a measurement as a pressure-sensitive keyboard, but still provided relevant data for strength (Figure 10). The projects demonstrate that bespoke parameters can be conceived and used as inputs of the computational visual formation. Depending on the gap between the input and the generated form, the interaction can be like a tool with which humans use and control or a generative display that renders a machine's response upon human actions.

The third type of computational formations takes external data sources from APIs and/or databases to create variations. It can drive a large amount of and/or a range of data into systems. In my work, I create digital installations that display the results driven by data. For example, in G50 Wall (2014), I aimed to create a mesmerizing generative landscape filled with 50 years of portfolio data from RCA's School of Communication graduates and to represent a continuous re-gathering. I wrote an algorithm that continuously loaded and updated portfolio images from RCA's online database every ten minutes and displayed them in the forms of kinetic particles (Figure 11). Instead of informative representation of data (Figure 12), it was focused on manifesting a series of landscapes by letting the visual data fluidly rearrange (Figure 13). Like pigments in a painting, portfolio data was used as ingredients for creating expressive and performative variations of form. Except for the alumni who contributed their work to the database, it was impossible to notice the data-driven nature of the project, as the connection to the database was only visible in the code.

In another digital installation, One Remix (2015), I aimed to create spontaneous algorithmic renderings using the imagery data driven from Google Street View and Google Maps APIs (Figure 14). Using the data APIs, I automatically navigated different areas and retrieved data by altering the geographic coordinates in the queries. Although loaded and visualized instantaneously, the data used in the project was collected and updated annually. Therefore, to make the work performative and reflective of more live contexts, I needed additional data that changed in real-time. I chose weather data then remixed it with the Street View imagery data by transforming them (Figure 15). The wind speed and direction dictated the parameters governing the scales and angles in the computational objects. As a result, it yielded the rendition of abstract atmospheric parameters at each runtime while displaying the visible atmosphere of the street. My process of designing formations focused on selecting data and interpreting them when relating them to forms.

DISCUSSION

Code is my primary medium to design systems for formation over time or upon interaction. Beyond programmatic approaches from a conceptual standpoint proposed by Gerstner, which provided a 'rational' rather than 'instinctive' (2007) mindset and actions, my design process utilizes a computational medium to explore creative expressions by constructing formations. My design process, therefore, is grounded in experiments on the variable visual formations in generative systems that transform in response to external inputs.

Writing code, the "raw computational medium," is an essential way of designing through computation to construct variable systems that create artifacts in response to dynamic inputs. Yet, beyond perceiving the process as simply commanding computers to execute predetermined tasks through programming (implement functions), using computation can be a creative process that forms an integral part of design. By observing and reflecting on how the formations/systems perform over time and upon interaction, I am constantly modifying and iterating upon the rules and inputs. Computer scientist and pioneer of computer art Frieder Nake described how the computer simulates 'imagination, variations, and series formation' (Gerstner 2007), whereas computation embraces an imaginative aspect when designers write instructions for creating variable visual formations. A computer program can rapidly generate a variety of forms, but the complexity of variations can easily go beyond a practitioner's mind.

In designing computational formations through pseudo-randomness, I constantly imagined how my system would perform. Over a process of trial and error, I was able to find a balance between conditional statements and a range of parameters that resulted in compelling variable artifacts. For example, I achieved autonomous behavior by adding randomness when changing multiple variables in *Creative Mind*. Yet, autonomous behavior was initially not a goal, rather something that emerged naturally while iteratively crafting the formation. In *Work-in-Progress*, although I designed the formation, the scope of permutations and transformations in the visual identities exceeded my initial anticipations. I was able to describe the visual behaviors of those formations verbally; however, it could not fully encompass the variability of visual outcomes, as the results were not simply transformative but also countless, endless and seamless.

As data and signals are received, computational formations become a more exploratory and playful process. For example in *Performa*, I constantly experimented with how letters transform their appearances upon typing and over time. It was heavily based on the pairing of inputs and visual parameters (i.e. data and typographic attributes in this context) and only perceivable while participating in the work. Similarly, in *Spoken Words*, I could only tell whether the interaction (i.e. visual expressions corresponding to speech) was appropriate while seeing the series of dynamic visual responses as I spoke. The process may seem largely mathematical due to the means (i.e. code); yet, the decisions on translating parameters across different modalities (i.e. relating typing speaking to typography) were based on intuitive and impromptu interpretations.

Driving data from external sources and using them as ingredients in computational formation shifts design consideration from communicating content to context. For example, in G50 *Wall* and *One Remix*, data were used as pigments to render the archive/database abstractly. Because the data was not represented in a clear and precise manner, there always was always an element of chance or uncertainty in the production of designs with computational formation, which I embraced as its inherent characteristic. I became a spectator of my own work, full of curiosity and expectations.

In this way, code as a creative medium can play a more extensive role in the design process. I sometimes had concrete ideas before writing code but they evolved when determining constants and variants, or were revised while trying out new ideas. Programming can be understood as sketching with code, an exploratory medium for open-ended experiments without a solid plan, just as a pencil can be used for doodling and scribbling. The practice of 'creative coding' is similar in this regard: computation is used for exploring new and serendipitous possibilities. Contrary to Gerstner's perspective, which regarded the practice of designing programs as a method of making rational and non-instinctive choices, my computational perspective for variable formation also relies on intuitive and playful experiments.

Overall, my design process for building variable formations relies upon experimentation and intuition and the computer's capability of quickly generating numerous possible options as my ideas evolve. The process is satisfyingly both creative and reflective, following a "see—move—see" pattern (Schön and Wiggins 1992), in which design opportunities can be continuously identified through numerous trials of actions and reflections. The process enables me to not only conceive visuals through code and data but also to perceive visuals as generated outcomes of code and data. Computation encourages new perspectives to understanding form as a variable object, produced with code and data that generate cohesive appearances and responsive behaviors.

CONCLUSION

In my practice, computation is an appropriate, often favorable, medium for designing variable formations because I am able to design and program the way variations of form emerge in relation to dynamic changes in data. In this regard, code can be a creative medium that enables a designer to craft relational patterns between data and form. As my medium of choice for designing variable formations in graphic design, computation points to a shift in emphasis from individual form to the patterns between input and visual parameters or across variations. Using code for variable formation, the designer becomes an orchestrator who creates cohesive and relational patterns between and among data and forms. Ultimately, using computation requires a new way of perceiving and creating in graphic design; this new way involves looking at forms as variable objects generated with code and data, constructing static and kinetic patterns with variability through algorithms and parameters, and making viewers active participants in the emergence of variable forms.

REFERENCES

Bartlett B, and Tatum R (2020) The Generative Typographic Brand. *Hoffmitz Milken Center for Typography: DETI*. Available at: https://www.youtube.com/watch?v=MAlhBMqi-QU. (accessed 28 February 2022)

Blauvelt A (2011) Tool (Or, Post-production for the Graphic Designer). In: E. Lupton and A.

Brownie B (2015) Transforming Type. London and New York: Bloomsbury Academic

Cho P (1999) Computational models for expressive dimensional typography. MS thesis, Massachusetts Institute of Technology, Program in Media Arts & Sciences.

Fry B (2004) Computational Information Design. PhD dissertation, Massachusetts Institute of Technology, Program in Media Arts & Sciences.

Gerstner K (2007) Designing Programmes: Programme as Typeface, Typography, Picture, Method. Baden: Lars Müller Publishers.

Hillner M (2007) Basics Typography 01: Virtual Typography. Lausanne: AVA Publishing.

Giampietro R, Jen N, Krishnamurthy P, and Van Blokland P (2013) Type is Never the Thing Itself. The Type Directors Club: Archive. Available at: https://archive.tdc.org/news/type-is-never-the-thing-itself-generative-typography-video/ (accessed 28 February 2022)

Küsters C and King E (2001). Restart: New Systems in Graphic Design. London:

Thames & Hudson.

Maeda J (1999) Design by Numbers. Cambridge: The MIT Press.

Manovich L (2013). Software Takes Command. London and New York: Bloomsbury

Academic.

MuirMcNeil (2016) Interview: MuirMcNeil. In: K. Shim (ed.) GRAPHIC, 37, pp.121–132.

Schön DA and Wiggins G (1992) Kinds of seeing and their functions in designing. Design

Studies 13(2): pp. 135–156.

Shim (2020) Computational Approach to Graphic Design, The International Journal of Visual Design, 14(1). Chicago: Common Ground Research Networks.

Shim (2021) The Future of Typography through the Lens of Computational Design, in Kim, M. (ed.), Typography in the Digital Age. Seoul: Hongdesign.

Small D (1999) Rethinking the Book. PhD dissertation, Massachusetts Institute of Technology, Program in Media Arts & Sciences.

Small D, Ishizaki S, and Cooper M (1994) Typographic Space. Conference Companion on Human Factors in Computing Systems. New York: ACM, pp.437–438.