Vocabularies of Computation: An Interview with Kyuha (Q) Shim

Kyuha Shim, *Deconstruction, no. 1* (2015), Digital C Print, London Design Festival 2016, London, UK. © Kyuha Shim

Historically, computation has played a role in design practices to varying degrees. To some, its role has been integral in that it functions as both the outcome and the tool. For Kyuha Shim, or Q, its ability to be flexible allows him to maintain a practice that allows for improvisation through what he calls "the orchestration of logic." His interests are diverse, working through various materials and subject matter, and though his approach is consistent, his practice is difficult to define, as an effort to do so would be limiting. It is the seamless integration of a contemporary visual vernacular and the command line that suggests that new models are en route.

Q is currently an assistant professor in the school of design at Carnegie Mellon University, the director of Type Lab, and cofounder of Punct (a collaborative practice he maintains with Taery Kim, PhD). In the following conversation we discuss the role of computation in design, automation, and AI; his publication *GRAPHIC #37* (2016); time spent at the Jan Van Eyck Academie; and what it means for an algorithm to have style.

BY
Jasio Stefanski

FILED TO
Design

DATE
Nov 8, 2018

PART OF SERIES

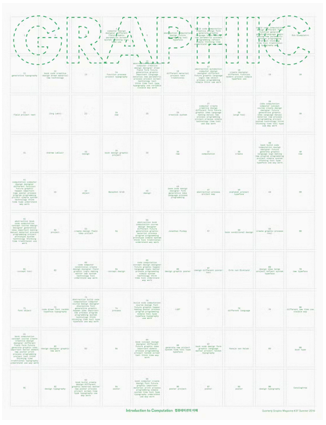
JAS STEFANSKI (JS)

You recently designed and edited an issue of *GRAPHIC*, "Introduction to Computation." What was the impetus to address the role of computation in design practices at that time?

KYUHA SHIM (Q)

Using computation and computational tools to address design concerns is highly relevant, if not integral, to the history and practices of graphic design. This is because computation can be another, albeit forceful, way for designers to create rules and parameters for the building of programmes. No longer bound by existing software tools (e.g. Adobe's Creative Cloud), graphic designers have the ability to create their own tools, their own creative language, with computation. Having said that, this is not always so clear-cut. When I first started creative coding, I was often asked, "Are you a coder or designer?" I was young, and offended, because I knew that not everyone who can code can design. I taught courses on using computation for graphic design and found that learning the syntax of programming languages did not equal learning abstraction of visuals—that is, the ability to construct visual systems using computational concepts (e.g. conditionals, loops) . Yet many people were focused on learning how to code, rather than asking, "Why code for design?" or "What is its impact on design practice?" To seriously reflect on this myself, I decided to pursue a PhD. During this time, I was able to confirm that while there are several how-to books introducing programming languages for graphic design, publications discussing the impact of computation on graphic design practices are largely absent.

While investigating the principles of computation in graphic design practices, I wanted to talk with peers and experts working in the intersection of graphic design and computation to see how they define computation in design, what vocabularies they use to describe the integration, and how they see the impact of computation in their own practices. I enjoyed curating *GRAPHIC* #37 and received a number of requests for copies from people studying in Europe and the US and writing dissertations on computation x GD who didn't get a chance to buy a copy when it was still on the shelves. Either the issue was sold out or not sold in their countries. I am happy to share that I have been working on a forthcoming publication, *Computation in Graphic Design Practices*, containing a series of expert interviews and essays, to be published late 2018 or early 2019.


GRAPHIC #37, "Introduction to Computation," Kyuha Shim (ed.), 2016, Seoul: Propaganda

1/4

GRAPHIC #37, "Introduction to Computation," Kyuha Shim (ed.), 2016, Seoul: Propaganda

JS

How would you describe the role of computation in your own practice?

Q

Computation is, first and foremost, my primary medium. It is a lens with and through which I take an analytical perspective to see things as generative pattern written in algorithms. Not everything is made out of rules, but I love to decode design decisions encrypted in artifacts. At the same time, computation is a material with which I build systems that can automatically repeat tasks, which yields continuity and actually feels like real-time. It enables me to play with contingent, complex, data-driven, and improvisational aspects of formation.

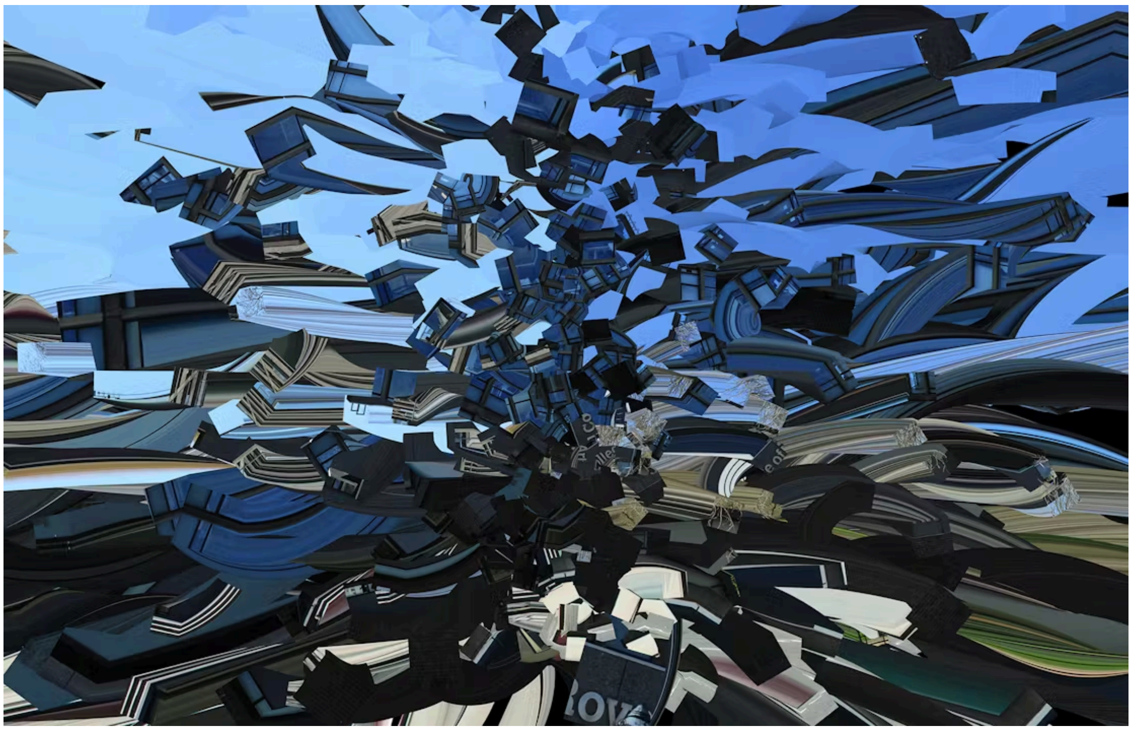
One Remix (2015), Generative Video Projection Typojanchi 2015, Seoul, KR. © Kyuha Shim

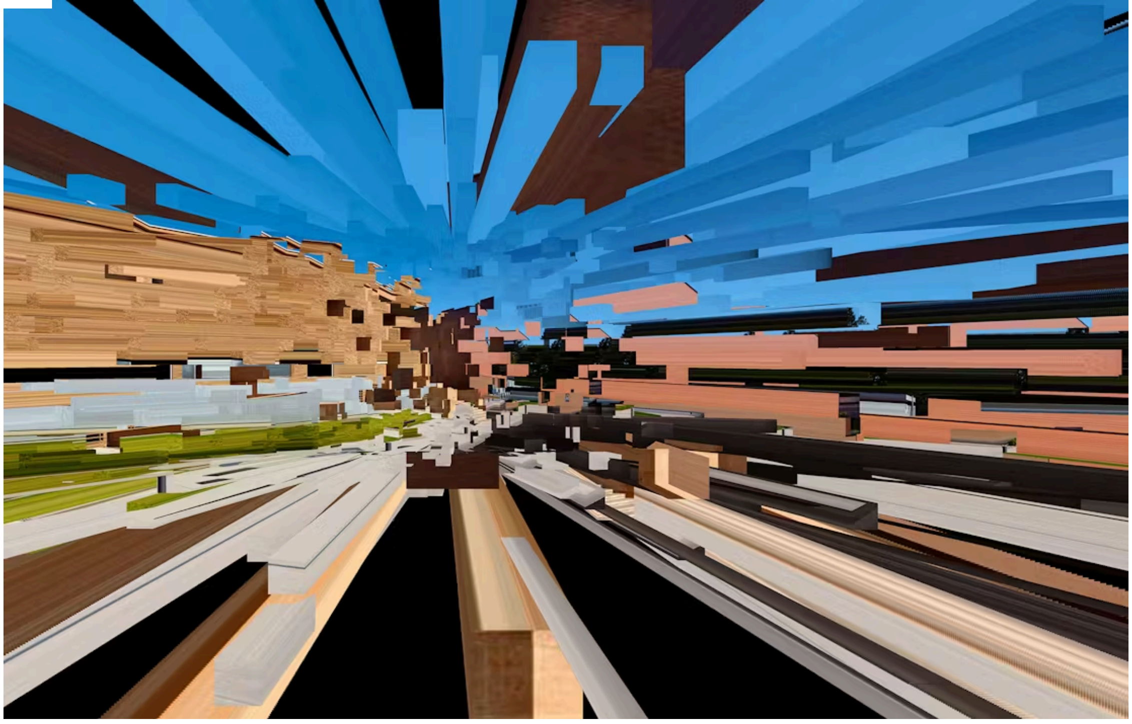
JS

What is the relationship between computational tools and computational strategies? Is it important to make a distinction?

Q

I am not so concerned with making a distinction, as I am for placing more emphasis on understanding computation as a process for building systems in graphic design practice rather than as a means for executing functional tasks. It is in this light that I ask in my research: what are the contributions of computation for building programmes in graphic design practice? I said before that knowing the syntax of code does not equal having the ability to do abstraction, which requires generalising and translating static and kinetic patterns into algorithms. I am not suggesting that everyone stops using Creative Cloud and produces everything with code. However, in comparison to being solely dependent on software tools (i.e., standard tools and filters built in Adobe's Photoshop), using computation can be liberating. It certainly makes designers more independent of existing tools because they are able to create new ones, at least new functions, new "words." But, then again, this is not always good because more freedom can also mean more time and effort required. To illustrate this: suppose you need a certain piece to complete your puzzle, and with computation you can define your own unique piece through custom rule sets and parameters. What if it takes a very long time to make one? Instead of cutting a shape out of a wooden panel or paper stock (sometimes you may just need to declare a specific type of material), would you do it? While I believe that designers use computation for creative coding, not as a way to increase efficiency for problem solving through optimizations, it is still difficult not to consider how much labor each project or task within the project would take. After all, there are many ways to draw an outlined circle.


Kyuha Shim, *Deconstruction, no. 2* (2015), digital -print, London Design Festival 2016, London, UK. © Kyuha Shim


Kyuha Shim, *Deconstruction, no. 3* (2015), digital c-print, London Design Festival 2016, London, UK. © Kyuha Shim

JS

Do you see AI playing a larger role in design output in the future? How does that relate to this statement: "Designers have become more independent with the use of computation"?

Q

Certainly. AI's strength in recognizing and replicating patterns will soon be implemented in our software tools, such as Adobe's Sensei. This will enable us to rapidly simulate and compare a myriad of solutions, like what Autodesk presents through Generative Design. The advancement of tools with AI might actually benefit everyday persons more than it will designers because the ability to borrow design decisions from the analysis of existing artifacts will become possible. When non-designers produce fair quality outcomes, it will shift our understanding of expertise and labor in design production.

On the other hand, will designers create their own AI model and employ it in their practice? Considering the amount of time and effort for collecting, organizing data, and training a model, I don't think so, unless someone releases a game-changing platform that eases the process. To be honest, I remain curious about the role AI in the future of design production even though the reality might be that AI intervening in design process would reduce much of designers' roles in production. Perhaps there would be efforts made to prevent this, such as employing random shapes or irregular elements in design, so as to make it difficult for the machine to recognize patterns.

JS

One of the main hurdles in terms of integrating computation into design practice or education is the emphasis on syntax vs. logic [in regards to coding languages]. Do you see these aspects as being at odds?

Q

I don't see the relation between syntax and logic as being in conflict. They are both essential for the process of abstraction in graphic design or visual communication, which requires the generalisation of our ideas into a set of rules that are executable by a computer. To do that, designers need to know the syntax to convert their logic into code, which I would describe as the abstraction of form into algorithms. Designers can find interesting, often unforeseen, possibilities during the abstraction process. In my case, I sketch/prototype using code with only a vague blueprint, and I often find interesting expressions or relationships between variations through trial and errors.

Formation (2016), Generative Video AGI Open Special Project Exhibition, Seoul, KR. © Kyuha Shim

JS

You make a distinction between "crafting forms“ and "designing systems." What does the difference look like?

Q

Well, I'd like to answer this question from another perspective. We don't have to talk about the difference when we're considering using computation to create systems that generate form. In designing systems, designers may not necessarily be authoring or crafting form by tweaking vertices of a shape, but rather they are authoring formation by writing a set of rules that can encompass sophisticated design decisions. This does not mean that form is not in consideration. Quite the opposite, in fact. In my interview with LUST for *GRAPHIC* #37, Jeroen Barendse points out that many people do not realize how much thought designers put into random elements in their work. I echo his claim and would like to add that setting the range (minimum and maximum values) of random variables necessitates designers to think about variations of generated outcomes over time or in response to input. Designing systems that generate form can be about crafting form, in a way, in that the nonlinear process of iteration includes adding/removing parameters, setting/changing values, building/refining rules, rendering outcomes, and testing performance of systems multiple times.

JS

Can an algorithm have style?

Q

An algorithm is a set of instructions and, usually, there are several of them in a programme. If you are asking whether an algorithm can encompass certain aesthetics, yes. Formation, or the process of generating form and its behaviours (e.g., transformation over time, upon interaction), can be described in algorithms. For example, Conditional Design's work demonstrates how a set of instructions can allow participants to produce a variety of drawings within certain patterns that share visual characteristics. Daniel Shiffman's Flocking simulates the movement of birds and has countless variations based on a cohesive visual style and movement. Mathematical algorithms yield distinctive visual patterns in variations as well, for example in Jonathan Puckey's Delauny Raster (2008) using Delauny's Triangulation and Karsten Schmidt's Type & Form (2008) using Conway's Cellular Automation. My work at Frans Masereel Centrum in 2017 was about decoding the patterns of existing artifacts by Van Doesburg, Martens, Schrofer and recoding algorithms that generate outcomes structurally similar to them.

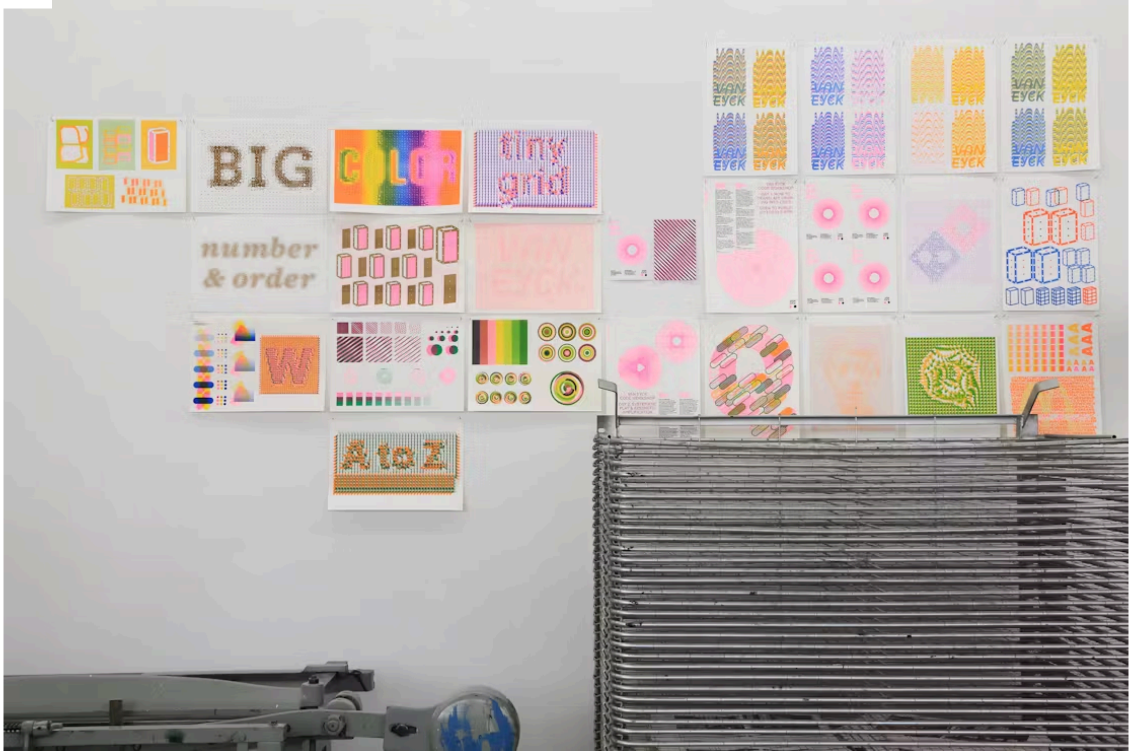
Kyuha Shim, *Raster* (2017), Video Research Residency at Frans Masereel Centrum, Kasterlee, BE. © Kyuha Shim

JS

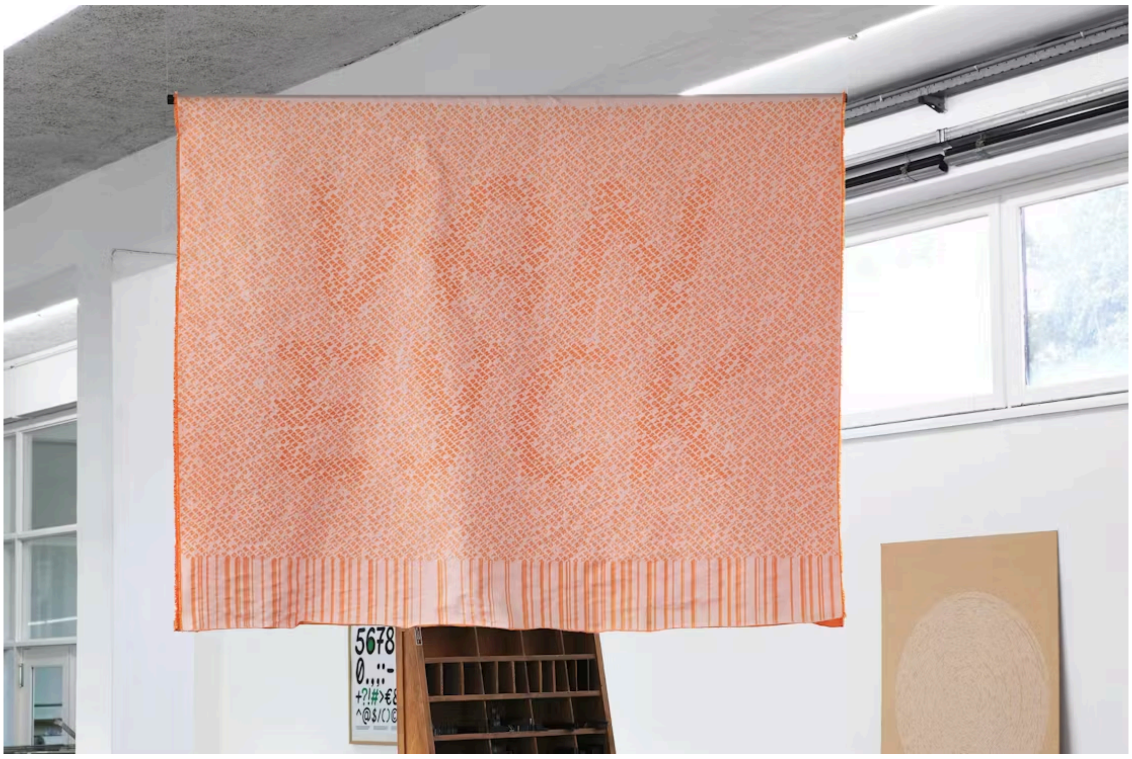
You attended the Jan Van Eyck Academie as a research fellow from 2013 to 2014. Much of the work you produced there was printed via duplicator/Risograph. What initially drew you to the historically-analogue modes of production at JVE?

Q

At that time, I was coming from a completely different environment—MIT's Senseable CityLab—where I worked as a research fellow/data visualization specialist, mainly creating various visualization systems using architectural and urban data. It was aligned with my MFA thesis on air pollution data visceralization at RISD, but I often felt that my role as designer was being limited somehow. I wanted to apply the techniques that I used for data viz to graphic design in a broader sense. I wanted to put myself into a place where I can draw inspirations from traditional graphic design production when integrating design and computation. During my time at JVE, I was working at Charles Nypels Lab (CNL) with Jo Frenken, Head of CNL, who exposed me to the Risograph (RISO) world. I was intrigued by RISO, mainly because I liked the experience of physically being a part of the production process, especially when changing drums and manipulating the offsets of paper through buttons on the duplicator. RISO comes with several screening options (fascinating, but still restrictive), so I was interested in exploring custom rasterization.

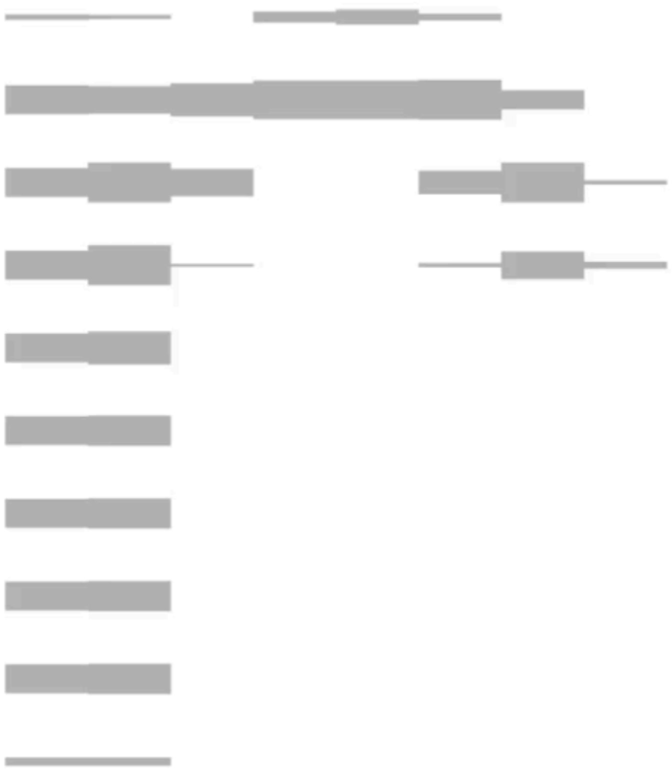


Kyuha Shim, *Miscellaneous* (2013), Risograph print, Jan van Eyck Academie Maastricht, NL. © Kyuha Shim



Kyuha Shim, *Miscellaneous* (2013), Risograph print, Jan van Eyck Academie Maastricht, NL. © Kyuha Shim

Kyuha Shim, *Miscellaneous* (2013), Risograph print, Jan van Eyck Academie Maastricht, NL. © Kyuha Shim



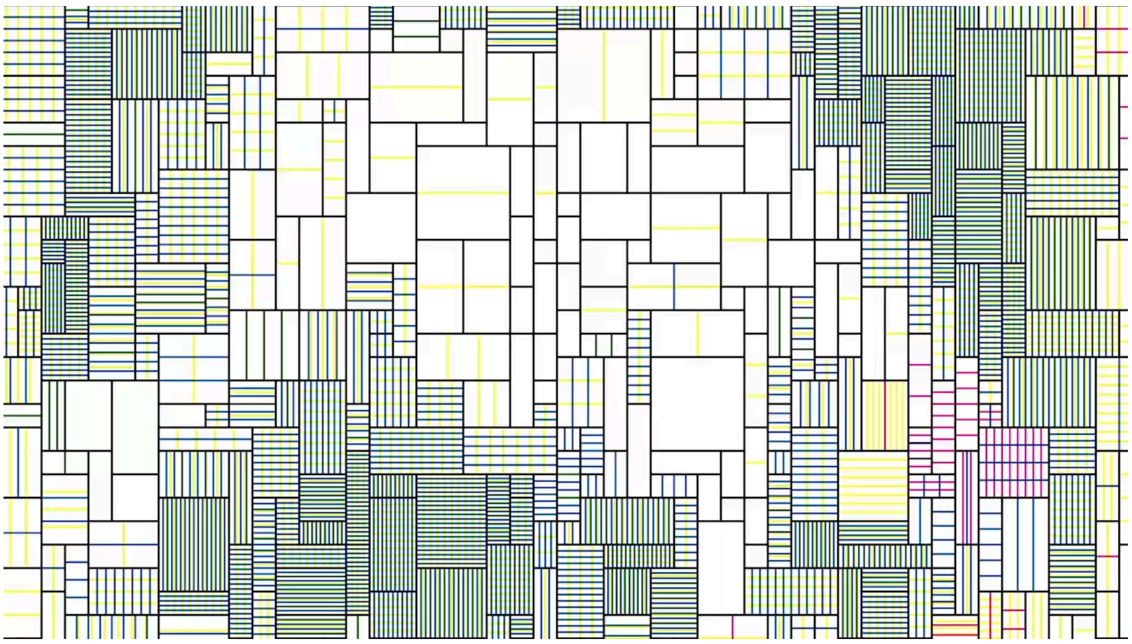
JS

Can you discuss how you approached rasterization programmatically?

Q

Sure. We can essentially view any image as an array of colors, and each color as an array of numbers in channels (e.g., rgba, hsba). Technically, rasterization process entails a repetition of reading colors from an image, converting them from RGB to CMYK, then assigning a shape, image, or text with transformations (scale, position, rotation) that correspond to the values of each channel. Compared to simplifying an image through “halftone” or “mosaic” filters in Adobe Photoshop, rasterization allows designers to add an extra layer in the process of abstraction. The module(s) can be used either continuously (gradually larger or smaller, thinner or thicker) or in a discrete manner (multiple modules swapping) in response to the values.

At the same time, I have designerly inquiries that guide my exploration of the interactions between colors and shapes. The size of each unit and the resolution of my rasterization are defined by questions, like: How abstract should the outcome be? If an input image entails letters, how legible should it be? The larger the unit, the more abstract the image, and the outcome of rasterization shows very minute details. On the other hand, the smaller the unit, the more the outcome looks like the input image, and it becomes harder to see the module itself. In terms of building a system, I typically ask questions such as: How utilitarian should it be? Should it work with many different inputs? Am I creating it for others to use in their work or purely for myself?



Kyuha Shim, *Raster* (2017), Video Research Residency at Frans Masereel Centrum, Kasterlee, BE. © Kyuha Shim