
Systema

Systema

Evan Andrew Merkel

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Fine Arts *in* Creative Technologies

Meaghan Dee, Chair
Dane Webster
Kathleen Meaney

18 December 2017
Blacksburg, Virginia

Keywords: generative design, creative code, typography

Abstract

Evan Andrew Merkel

This thesis is a three-part creative coding exploration of generative typography and pixel-based image manipulation. Systema is composed of three distinct projects named Lyra, Mensa, and Vela, respectively, that investigate and demonstrate the advantages and drawbacks of generative graphic design.

Acknowledgements

This thesis would never have been possible without the contributions, insights, and critiques of my committee. Dane, your support and guidance always pushed me to excellence. From my first creative code class to my defense, you offered encouragement, inspired confidence, and incited eternal curiousity. Meaghan, without your help this thesis would just be bland exposition; your advice and direction gave this document personality and character. Katie, your typographic taste will forever be superior to my own. Your observations and feedback polished these projects and brought them to life. Finally, thank you *magister* Yowell for completely changing how I think forever with one simple phrase.

To each and all of you, thank you.

Contents

Abstract ii
Acknowledgements iii
Figures v
Preface vii
Introduction viii
Project One: Lyra 10
Project Two: Mensa 22
Project Three: Vela 37
Conclusion 51
Future 56
References 59

Figures

Figure 02 | 10 - A photograph of Lyra in action.
Figure 02 | 13 - Lyra's generated result for ASCII 64 - "@".
Figure 02 | 14 - Early iterations of Lyra produced star-like patterns that inspired the nomenclature for this thesis.
Figure 02 | 15 - Helvetica Neue Medium after Lyra has processed for 60 seconds.
Figure 02 | 16 - Helvetica Neue Medium after Lyra has processed for 120 seconds.
Figure 02 | 17 - Helvetica Neue Medium after Lyra has processed for 180 seconds.
Figure 02 | 18 - Helvetica Neue Medium after Lyra has processed for 180 seconds using an inverted red comparison formula.
Figure 02 | 19 - A typical progression of a single character within Lyra.
Figure 02 | 20 - A zero-offset portrait of Albert Einstein, generated with Lyra.
Figure 02 | 21 - Close up of Figure 02 | 20.
Figure 03 | 22 - A photograph of Mensa in action.
Figure 03 | 26a - "Self-Portrait" by Picasso, drawn in 1900.
Figure 03 | 26b - The same self-portrait run through Mensa using a sample library of Picasso's analytical cubism genre. The result is a portrait composed of Picasso's own work.
Figure 03 | 27 - The proposed pipeline for a web-based generative portraiture studio. The important trait of this pipeline is the lack of interaction between designer and user; the only link between the two is the tool in use.
Figure 03 | 28 - Close up of Figure 03 | 26b.
Figure 03 | 29 - Close up of Figure 03 | 26b.
Figure 03 | 30a - Demonstration of Mensa's ability to vertically fit tiles under a curve.
Figure 03 | 30b - Demonstration of Mensa's ability to horizontally fit tiles under a curve.
Figure 03 | 31 - Photograph outside my apartment window, 24 January 2016.
Figure 03 | 32 - Mensa's interpretation of Figure 03 | 31 with a sample library of scans from my sketchbook.
Figure 03 | 33 - Close up of Figure 03 | 32 showing individual sample tiles. Notice the gestures and glimpses of lettering.
Figure 03 | 34 - The product of an early stress test on Mensa's primary algorithm. The original master image is 2800 by 2800 pixels and matched over 7 million tiles to produce the end result.
Figure 03 | 35 - Close up of Figure 03 | 34.
Figure 03 | 36 - Another close up of Figure 03 | 34.
Figure 03 | 37a, 37b, 37c - Three examples of typography dissected with Mensa using a sample library of images from other, unrelated creative code projects.

Figures
(continued)

Figure 04 | 38 - A photograph of Vela in action.
Figure 04 | 42 - “Participant Zero,” the first result from my generative portraiture studio. The process involved having the participant answer a series of questions, converting those answers a sample library, and processing multiple passes across the master image. The result is entirely typographic.
Figure 04 | 43 - Close up of Figure 04 | 42.
Figure 04 | 44 - A demonstration of the effect of tile size on the sampling and reconstruction algorithms. From top left to bottom right, the tile size begins very coarse and shrinks to produce a more legible, recognizable result.
Figure 04 | 45a, 45b, 45c, 45d - Examples of single-character, single-pass dissection. Each charecter is Caslon, reconstructed from every capital letter except itself.
Figure 04 | 46a, 46b, 46c - Examples of single-character, multi-pass, recursive dissection. The user can specificy recursion depth from the interface, which substitutes the previous iteration’s result for a new, undissected master image. This creates exponential decay of form and legibility.
Figure 04 | 47 - The New York Times front page from 29 July 1914 dissected using Vela. I used newspaper headlines as platforms to test word, sentence, and paragraph-level dissections due to their preformatted hierarchies and layouts.
Figure 04 | 48 - Close up of Figure 04 | 47.
Figure 04 | 49 - Example of representing data visualization using Vela.
Figure 04 | 50 - Not a failure, but an unexpected result. Accidents like this occured quite often during all stages of development.
Figure 05 | 53 - A screenshot of code.
Figure 05 | 55 - Harmut Bohnacker’s diagram of the generative design process.
Figure 05 | 56 - My modification of Bohnacker’s process, representing the function and workflow of the program after it has been delivered.
Figure 05 | 57 - Proof-of-concept for Figure 05 | 54. I developed a generative branding system using six programs written for the Virginia Bioinformatics Institute, which they still use to create assets and drive their identity.
Figure 05 | 58 - A screenshot of code.
Figure 05 | 60 - Returning full circle to the quote that started this all.

Preface

Systema is just a word. In Latin, it means a system, a collection of elements that relate to each other in one or many ways. In English, it has no meaning. The question Systema asks is not how to achieve absolute simplicity but how to communicate clearly and intentionally with the fewest possible elements.

Meaning is a spectrum, not a lightswitch. The three projects documented herein - Lyra, Mensa, and Vela - seek to quantify thresholds along this spectrum using some of the simplest digital elements: point, line, and pixel.

Introduction

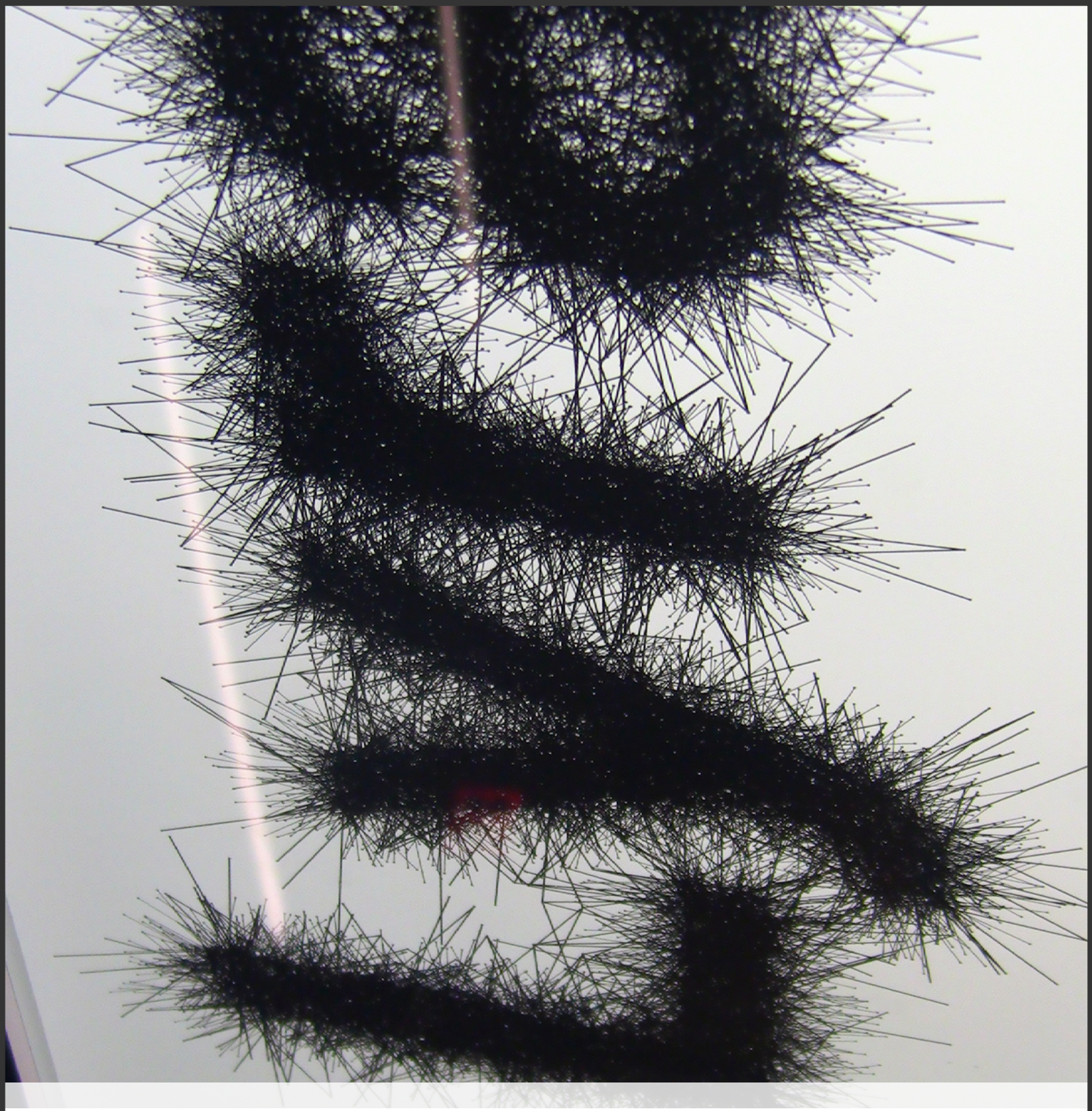
The most complex ideas can grow from the simplest seeds. The idea that inspired my graduate work comes from 2007 while I sat in my first high school class: Latin 1. In an analogy to explain communication theory to a group of freshmen, my teacher drew three lines in the shape of a capital letter “I” on the blackboard. “What is this?” he asked. The students pondered: capital “I”, a letter, a symbol, a word, a mark, a railroad tie, a ladder rung; all wrong. After five minutes of incorrect guesses, my teacher looked back at the class and said “It’s just chalk dust on slate,” (Yowell).

The purpose of Systema is to find the threshold of meaning where significance emerges from seemingly arbitrary elements. I refer to this concept throughout this document as the “event horizon of complexity,” or the point of no return when a system coalesces into a whole greater than its individual parts. Throughout these projects, I travel in both directions across this horizon. To acknowledge this horizon is to acknowledge the possibility that meaning can be disassembled, reassembled, extracted, and even manipulated. While this is no secret to art or design, the exact quantum moment when “chalk dust on slate” becomes a symbol that conveys sound, intent, and meaning remains unquantified.

Introduction
(continued)

Casey Reas' "Process" work is my primary inspiration behind the idea of constructing complexity from simple elements ("Process Compendium"). For example, Reas defines Process 1 as the application of four "Behaviors" to a single "Element" — in this case, a simple circle. Visually, this resolves to a system of circles that move according to four rules. However, I find more value in Reas' abstraction of the specified behavior into text. This abstraction was inspired by Sol LeWitt's instructional drawings that depend on a specific set of instructions that are left open to interpretation by exhibition artists ("{Software} Structures"). The separation of process from product and the important role of instruction are what interested me the most, as these were issues I faced while researching my thesis. The algorithm became the deliverable while the output acted as proof that the algorithm worked to specifications.

During the five semesters I have been enrolled in graduate school, I built and refined a series of three distinct projects exploring the concept that simple elements, when combined or arranged in specific ways, become endowed with elevated significance as a product of their relationship to one another. This thesis talks about each project, its process, workflow, execution, as well as its role within Systema.



Project One

Lyra

02|10

Lyra

Lyra is a program built with Processing that creates passes of unintelligent agents that, in turn, follow behaviors that form symbols and glyphs. Early results from the program inspired its namesake: simple, concentric patterns of point and line resembled patterns and figures of stars, so I took to naming each of my projects after a different constellation. Lyra began as a completely separate project that was designed to visualize a cellular reproductive process; however, as I developed the program I caught momentary glimpses of more meaningful patterns as the agents moved across the screen. Near-moments of quasi-meaning beckoned for their own avenue of research. It occurred to me that although these glimpses were accidental examples of pattern, there was pattern nonetheless. Lyra seeks to capture these accidents as they happen.

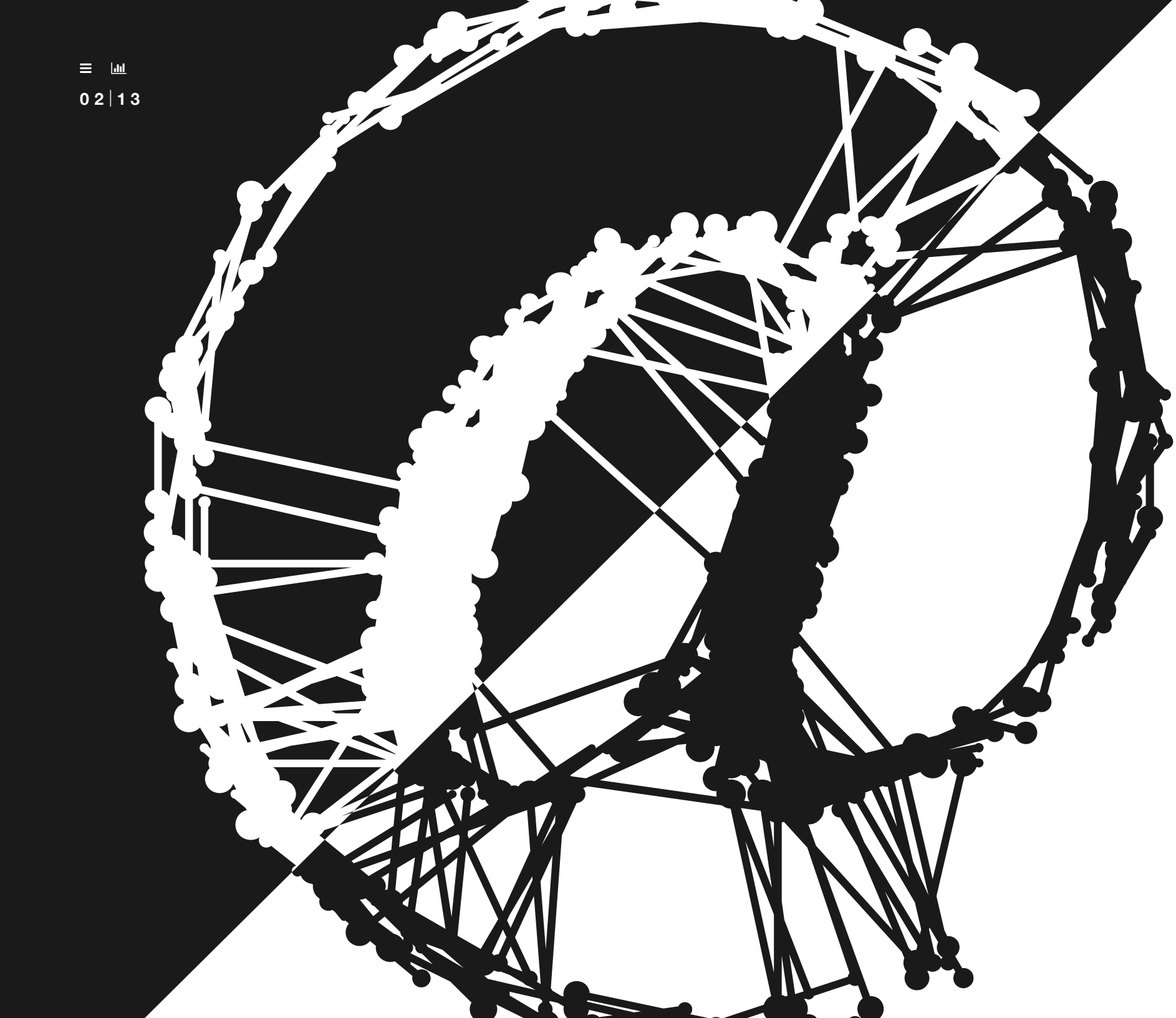
My initial use for Lyra was a generative typography machine; I started with individual letters and keyboard symbols, building my own typeface from scratch. Early iterations taught me a valuable lesson about my process, one that would play a role in the development of Systema through the rest of my graduate education: any editing that happened post-process defiled the purely generative nature of the output. Extra modifications to the results of one of my programs meant that the instructions were incomplete. Rather than tune the output of the program, I tuned the algorithm itself so that its logic aligned with my vision. I knew that the end goal was to create a self-sustainable, standalone system capable of containing every step of the generative design process. Anything less means that the algorithm is flawed.

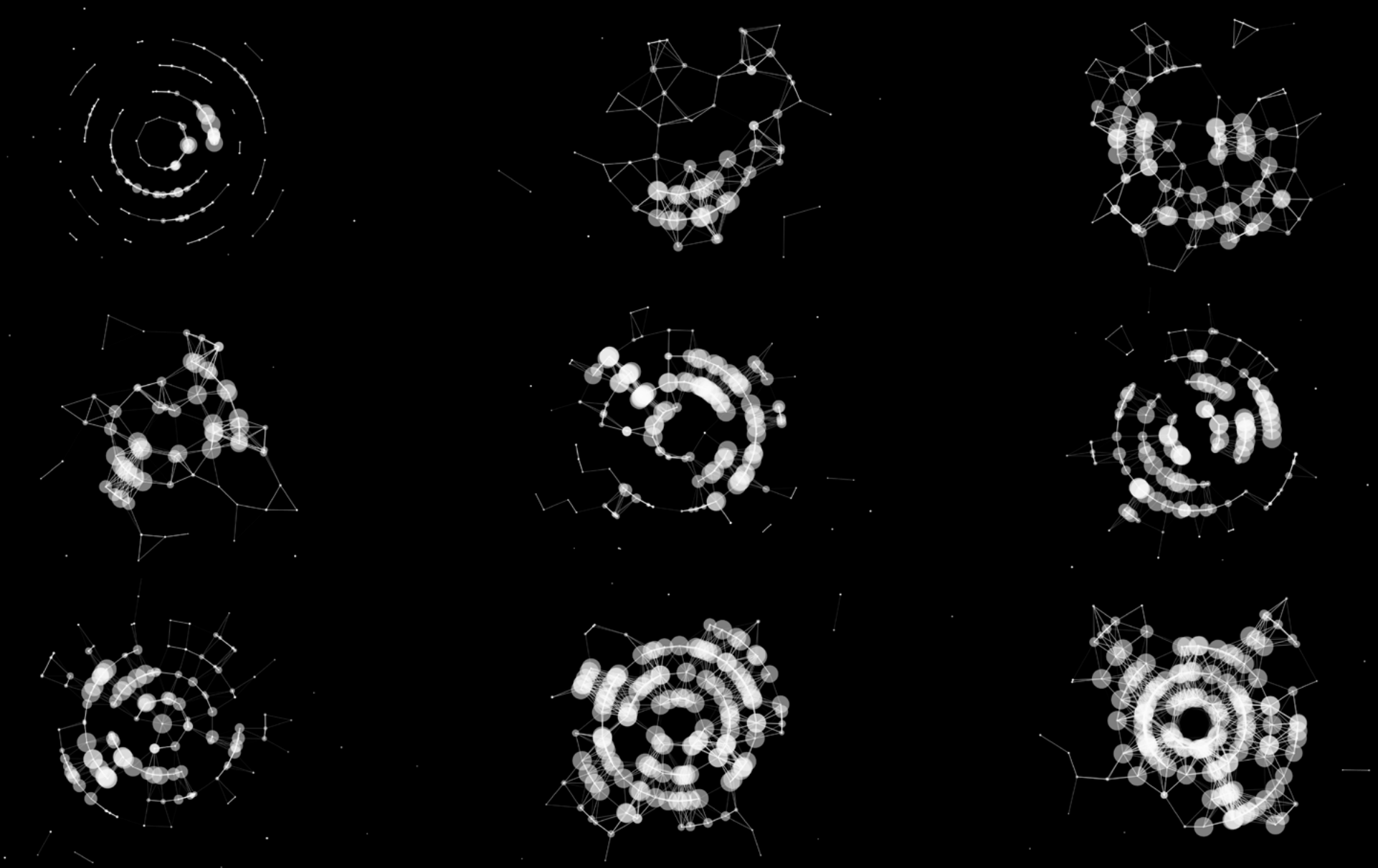
What fascinated me about Lyra was the ability to “cook” typefaces for variable amounts of time. The longer Lyra has to create and apply agents the more developed, and often more legible, the result will be. Although legibility is typically a benchmark of good typography, the unrecognizable results held the most interest and value for Systema. Successfully abstracting a letterform into a foreign shape pushes it across the event horizon of complexity by disassembling a familiar form into its component parts.

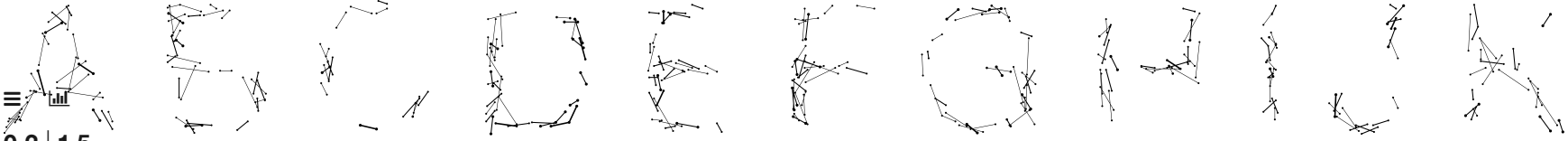
Lyra lacks depth but it is a simple program with a consistent result. I realized that the only two variables in the program that had a significant effect on the outcome were the source image and the amount of time an image was left to “cook”. The effect of time on the output was already apparent, so the only remaining opportunity for growth was to experiment with the input image.

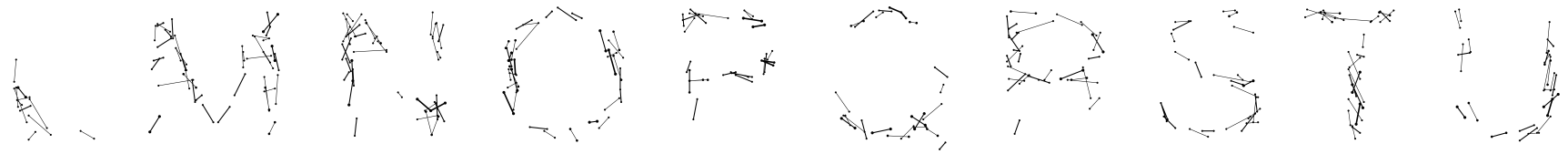
I turned to portraits for inspiration. In keeping with Systema’s theme of building complexity from simple elements, I wanted to reverse the process and reduce a complex image to simple visual patterns. I only created a single portrait with Lyra yet it marked the transition to my next project, based entirely on generative portraiture.

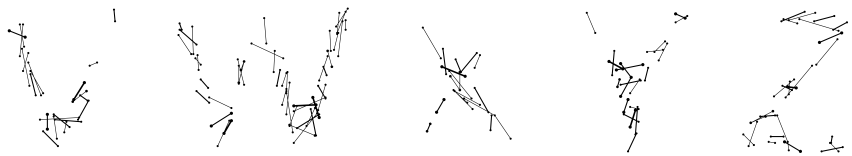
lyra.mp4

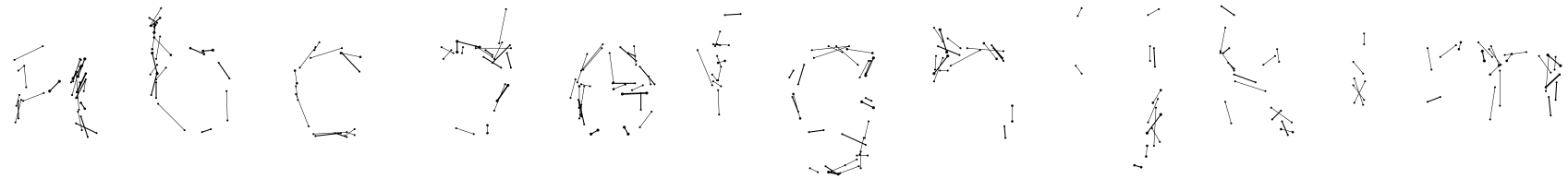


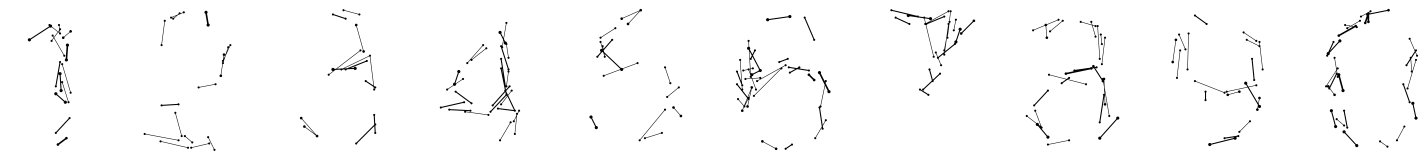


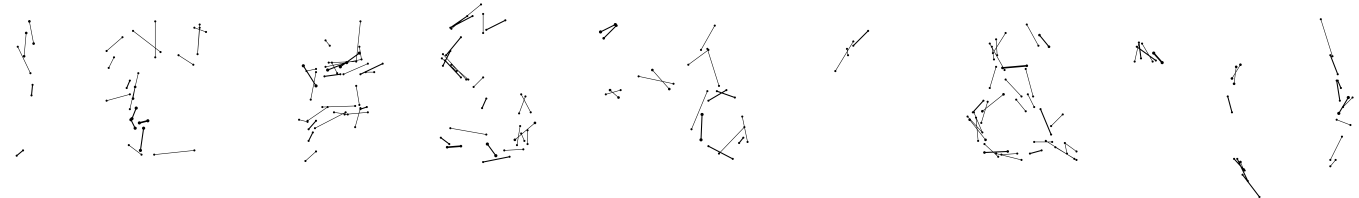


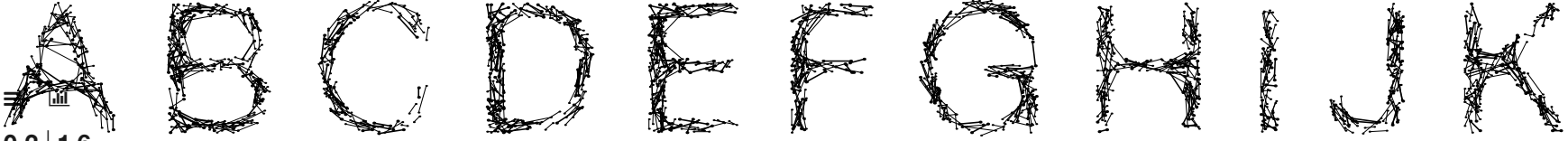


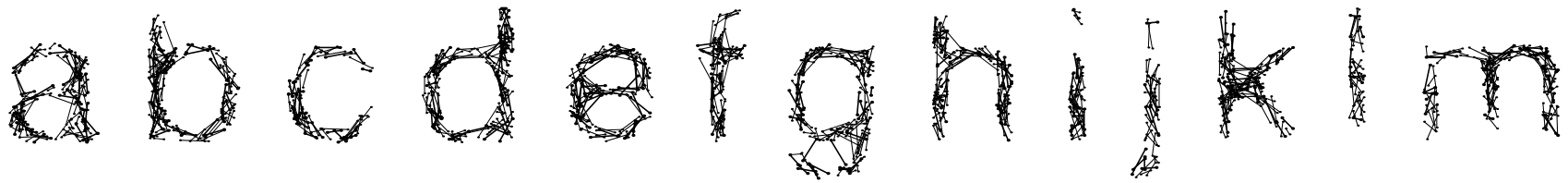


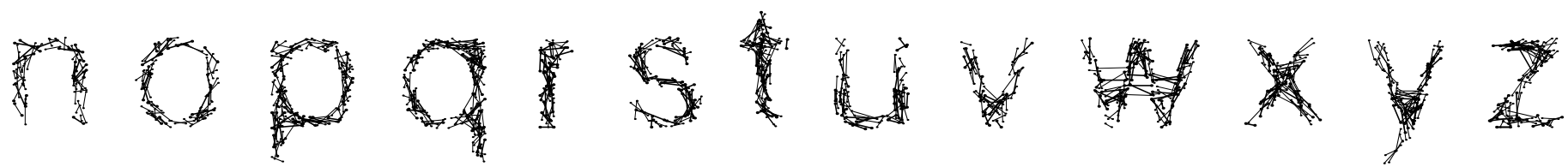


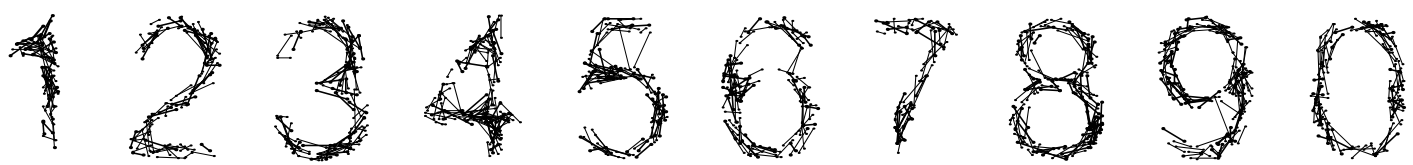


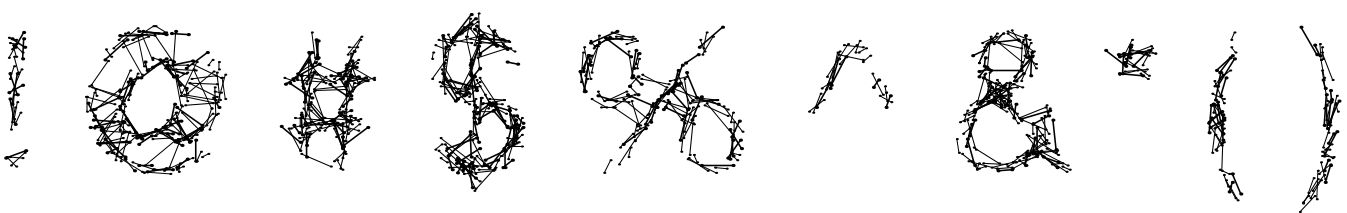




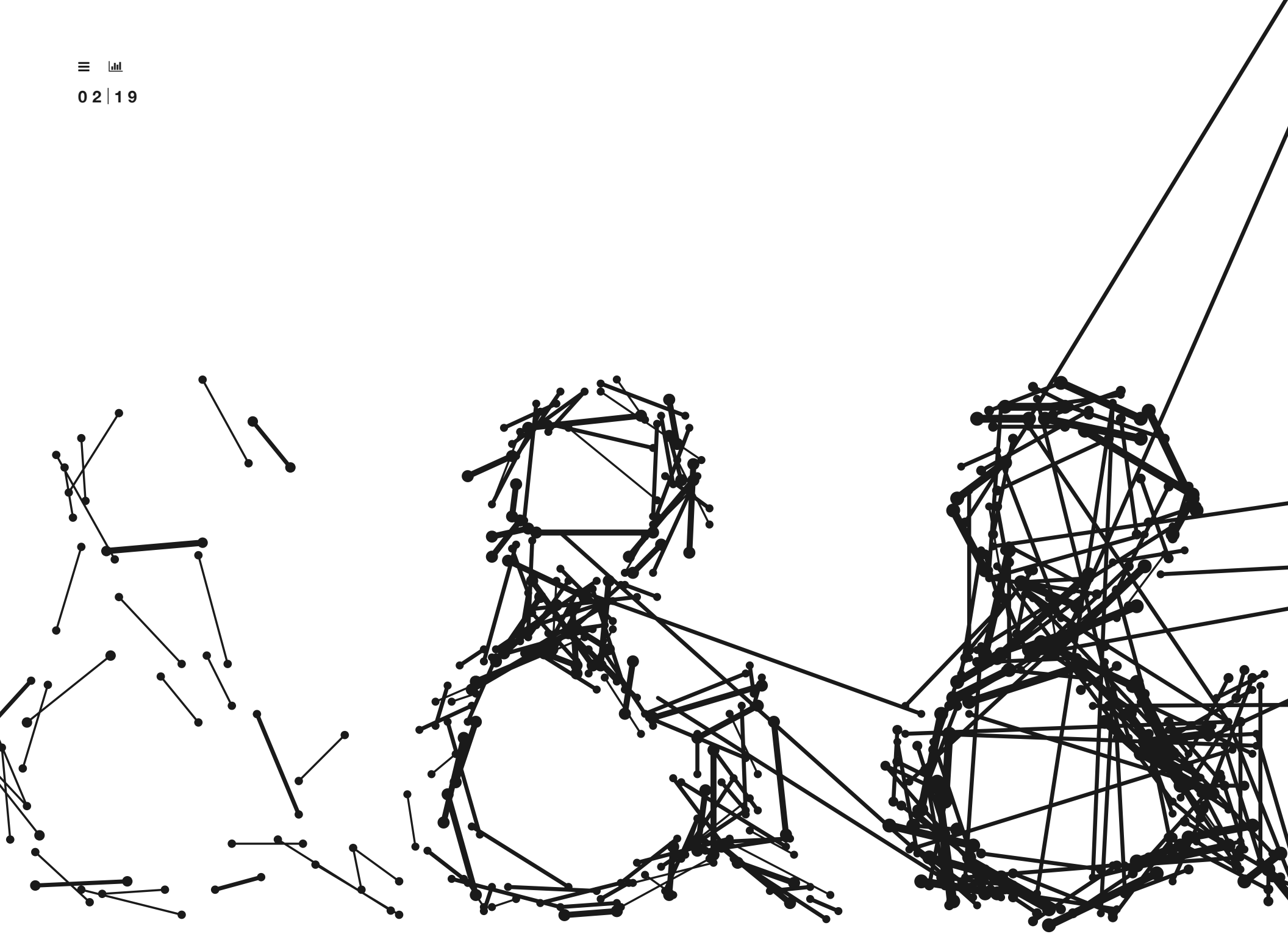


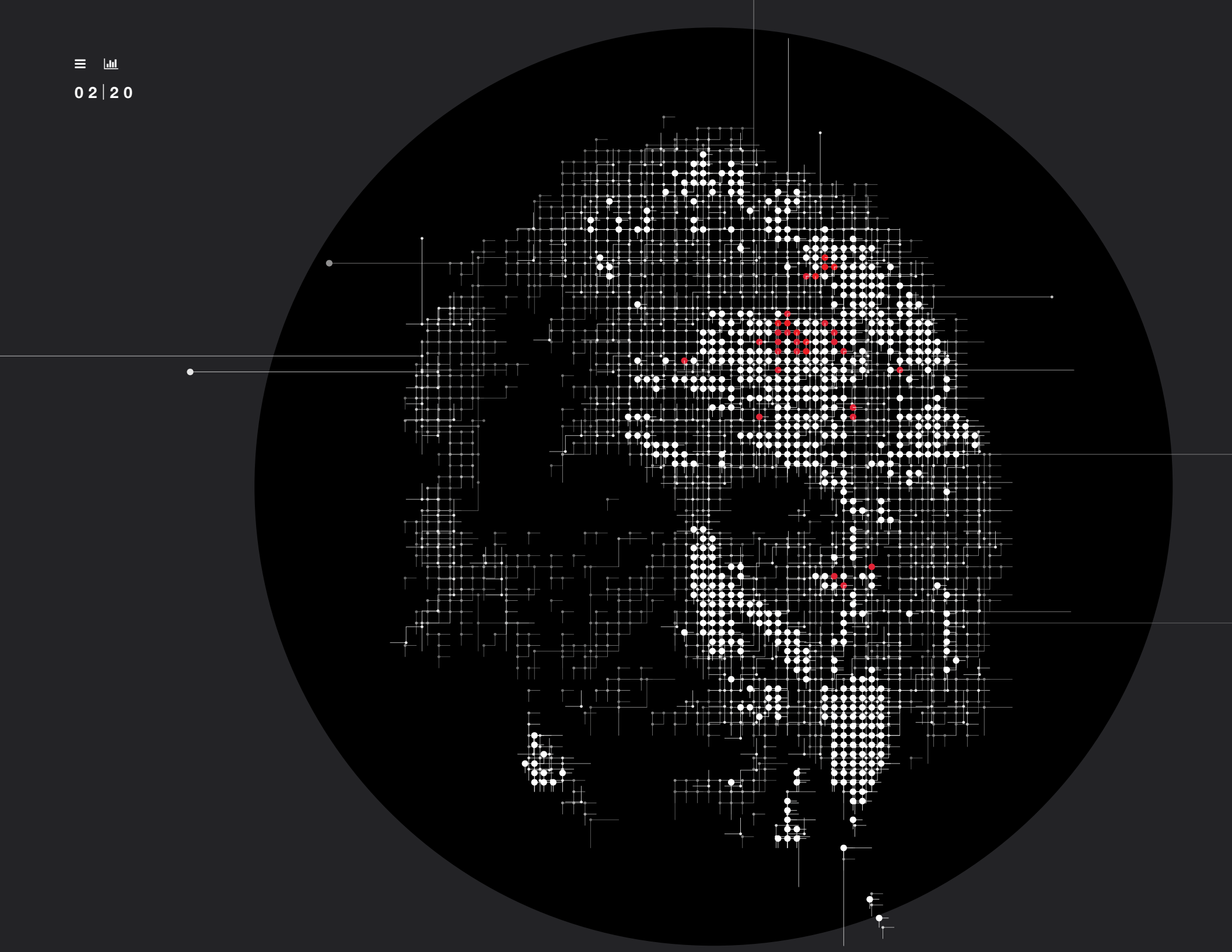


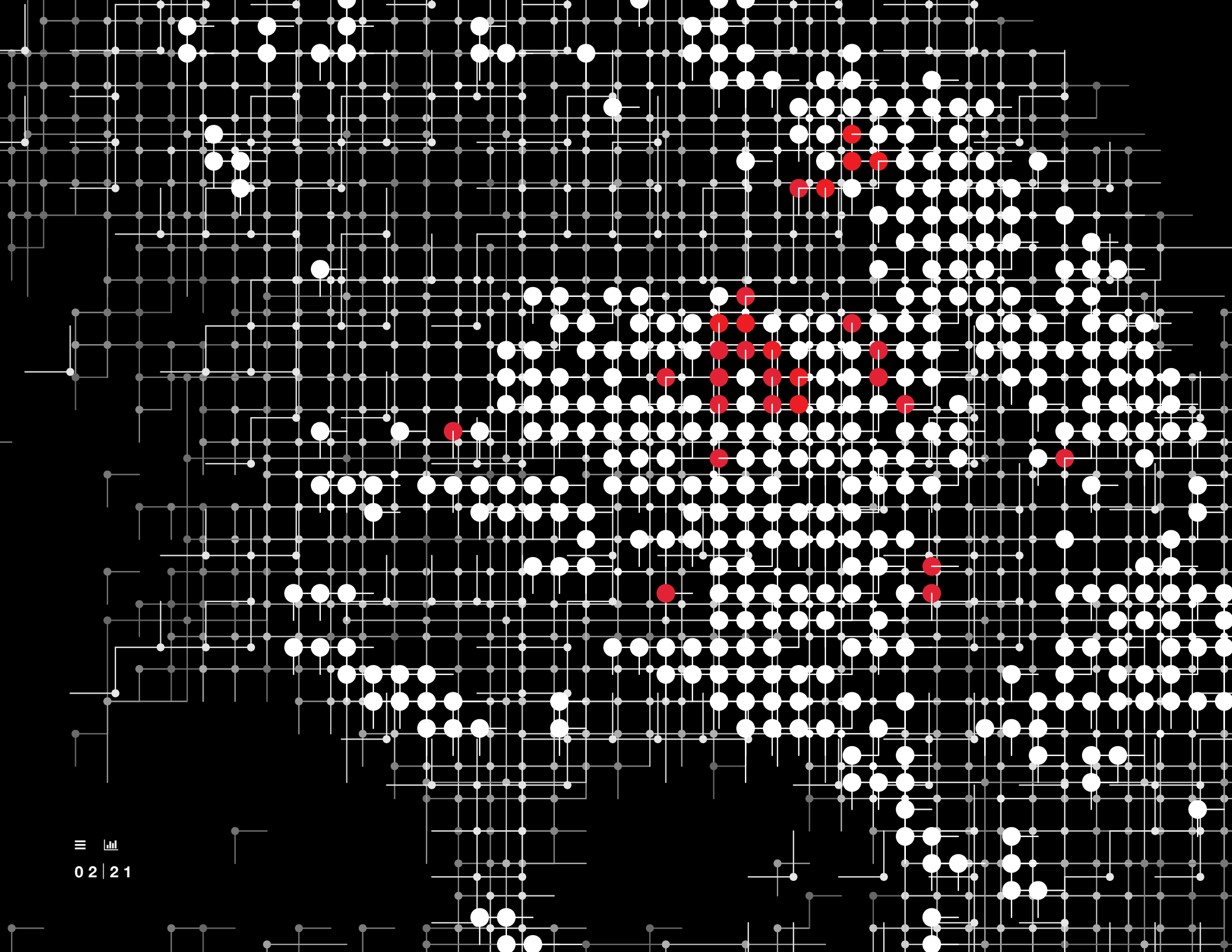





ABCDEFGHIJK
LMNOPQRSTU
VWXYZ
abcdefghijklm
nopqrstuvwxyz
1234567890
!@#$%^&*()







Project Two

Mensa

03 | 22

Mensa

Mensa, the second project in Systema, is built around the idea that identity and significance are isolatable traits. Mensa is a program written in Processing and Java that reconstructs a “master” image based on a set of selected “sample” images according to parameters and variables specified by the user. The process is similar to creating a photomosaic, except Mensa dissects the sample images into same-size chunks before reassembly.

Mensa marks the first, fully-formed concept for what my final thesis would become: generative portraiture. If Lyra is about allowing users to form symbols, Mensa’s goal is to allow a user to create a portrait of themselves to their own specifications. The assumption with this project is that by examining the resulting portraits, I was hoping to identify themes about how simple stories and events contribute to building a creature as complex as a human.

My proposed process was straightforward: create a portrait studio “booth” consisting of a computer running Mensa, a webcam, and a series of questions for each user to answer. The booth would have been designed for a single user at a time, ensuring necessary privacy that would hopefully foster intimacy. The user would type their answers to each question, saying as much or as little as they chose, take a picture using the webcam, and select additional stylistic parameters for the portrait. The user’s answers became the source language for the typographic portrait while the parameters dictated how that language is aesthetically arranged. Each user would have created their own portrait and the final product would have been a gallery of all the various results from each participant.

Unfortunately, I was not the only generative artist pursuing portraiture. Selgio Albiac, a generative artist whose work has inspired my own, released his project “I Am.” in June 2016. His project and process were nearly identical to mine on the surface, relying on similar mechanisms to abstract type and symbol into image. Without describing his project in detail here, Mensa shared too many qualities with Albiac’s project for me to continue to working in that direction. I abandoned the idea of making the portrait “booth” and shifted my focus.

Mensa was still a unique, functional program, but I was having difficulty finding a role that highlighted its successes. The thought occurred to me that Mensa was still relatively new, only about a semester old. I had not explored its limits, computational or conceptual. Data would push Mensa to its full potential. I incorporated a debugging mode into the program which pulled metrics from every part of the process of creating a portrait. Every single tile of every single image has core attributes that are critical to the

Mensa
(continued)

success of the algorithm, including grid size (in both horizontal and vertical directions), number of sampled images, dimensions of the master image and each sample image, color matching mode, levels of recursion (if any), tolerance threshold, as well as all pixel-specific data such as the red, green, blue, hue, saturation, and lightness values and colorspace coordinates in CIELAB space. Each time Mensa is run, the debugger logs all of this information in a spreadsheet, like a data snapshot of every operation.

After enough dissections, certain patterns emerged in the data that allowed me to optimize Mensa further. For example, the tolerance threshold of the color-matching algorithm is the key to the entire program running quickly, a fact that became apparent after observing that a typical portrait creation takes almost 29 million matching operations to complete. This number decreases inversely proportional to the tolerance value: as the tolerance drops, quality increases at the cost of speed. I was able to refactor the way tolerance affected the speed of the operation and decrease the amount of time each dissection took by a factor of two.

Despite the multiple iterations and optimizations to Mensa's algorithm, there was one issue that persisted beyond the code: poor quality input yielded poor quality output. Large images have more pixels in them from which to sample, which equates to more opportunities for the algorithm to find a good match. At this point, the content of the source material was not important to me, only metrics that indicated image quality. For example, I began experimenting with landscape photography because taking large, clear pictures of my immediate environment is easy, not because landscapes are relevant to Systema. I would snap a handful of quick pictures every time I took the bus to campus, and eventually I had a sizable library of high-quality imagery with which to work.

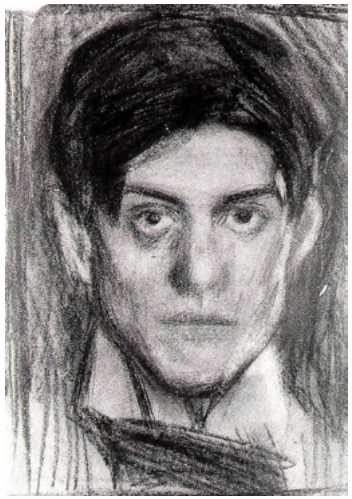
I soon found that reassembling images of Blacksburg hillsides with other images of Blacksburg hillsides leads to uninteresting results. I realized another important factor in maximizing the effectiveness of Mensa was the difference in subject matter between the master image and the sample images. In other words, recreating an image of a galaxy with other pictures of galaxies or celestial bodies does not produce visually interesting results. Only when the master image is reassembled using unexpected, dissimilar images does the algorithm output a meaningful result. Motivated by this realization, I combined my collection of found type - one of my long-running, personal hobbies - with Systema.

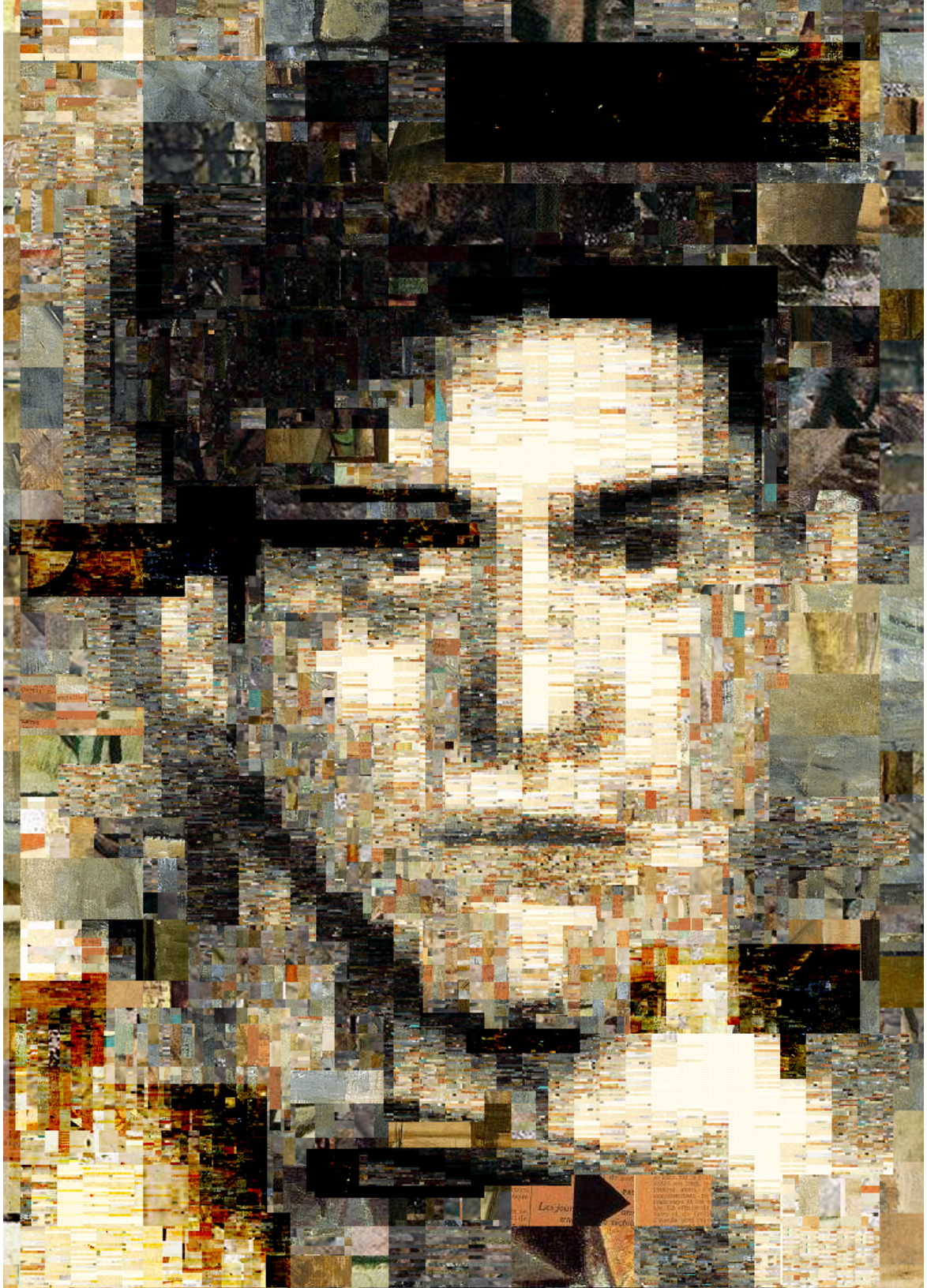
Mensa
(continued)

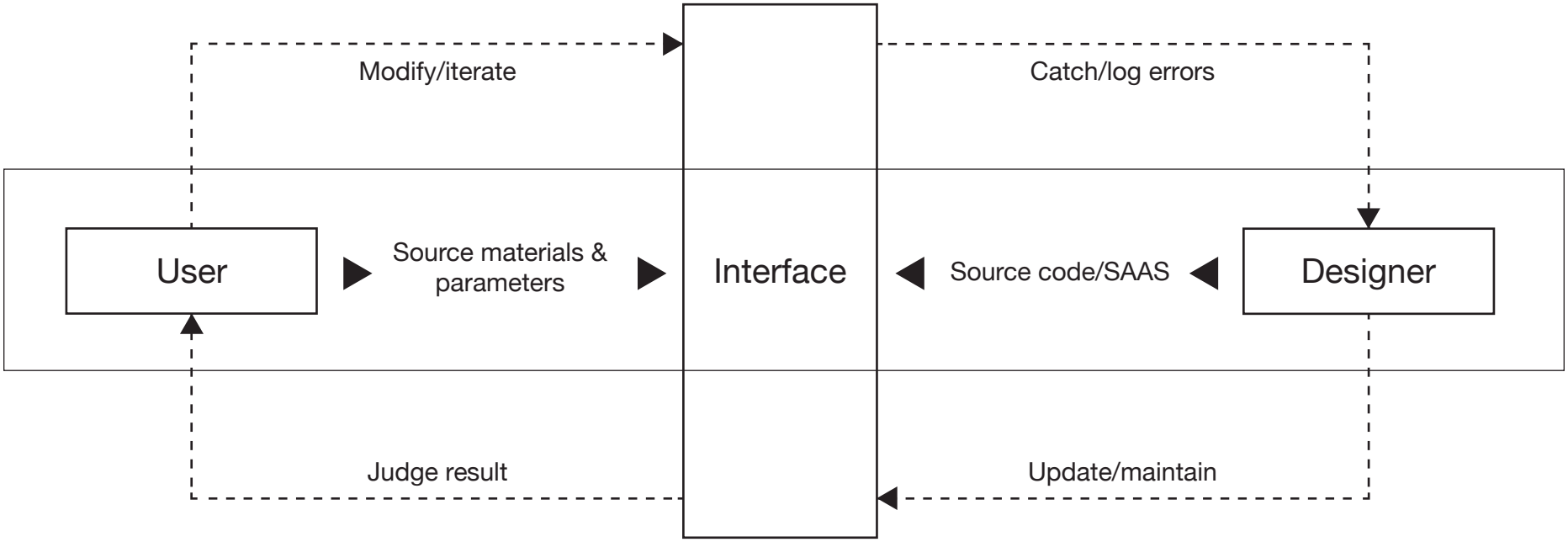
My collection of found type began with travel. I would save receipts from restaurants I liked so I could remember the occasion years in the future. It wasn't long before my collection expanded to any scrap of paper with language on it, be it a ticket stub, a wrapper, a take-out menu, even hotel keys. I scanned in each fragment, categorized the scans, and accumulated them into a single folder for use with Mensa.

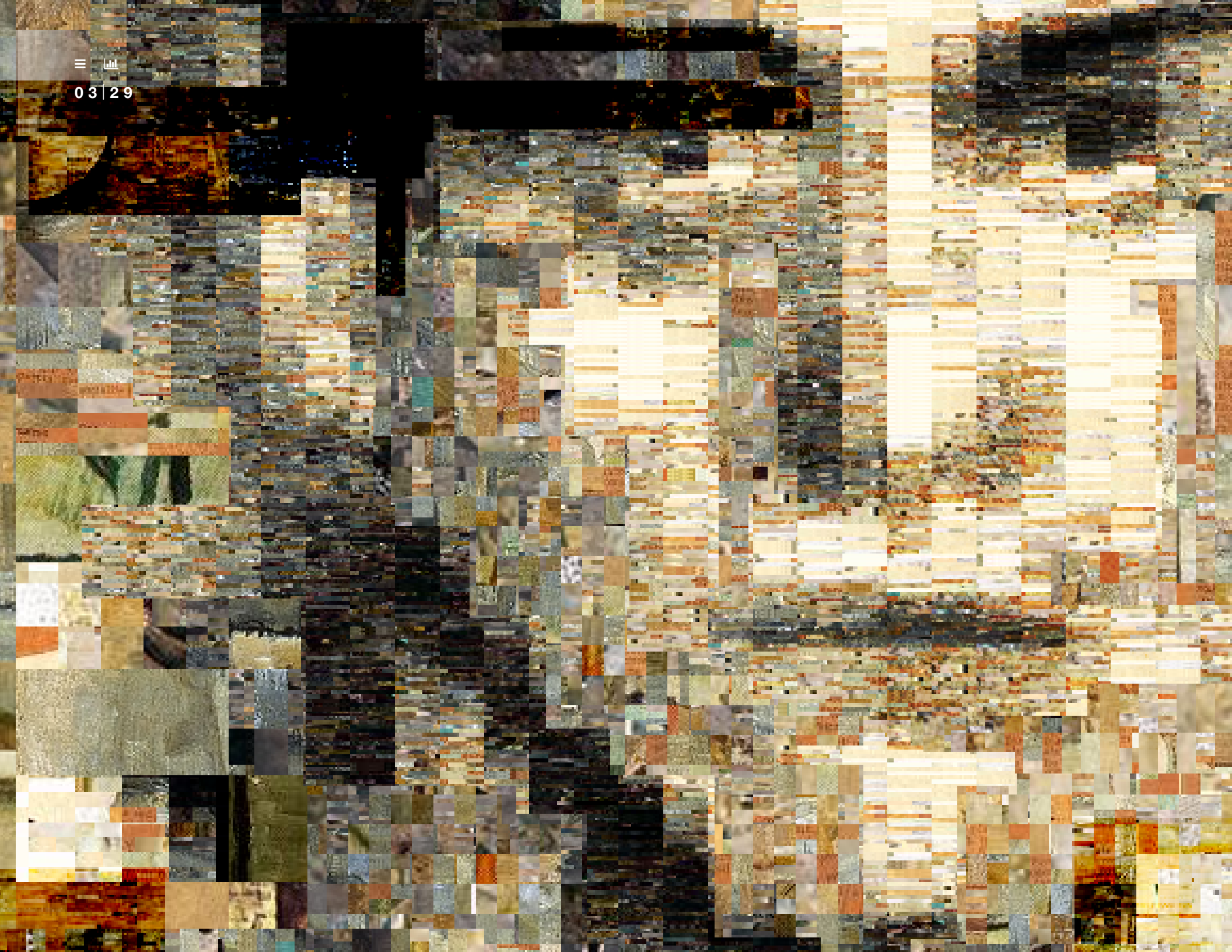
The improvement was immediate and significant. I created landscapes composed singularly of my own handwriting. Moments I had chosen to preserve through lettering became a new form of media, themselves abstracted even further by the algorithm. The horizontal dissection grid acts as a clothesline, stringing together moments of meaning into quasi-sentences. Descenders and ascenders, counters and loops, pieces of letters that, on their own, convey no meaning. However, evening aligning these pieces associates them in a recognizable way.

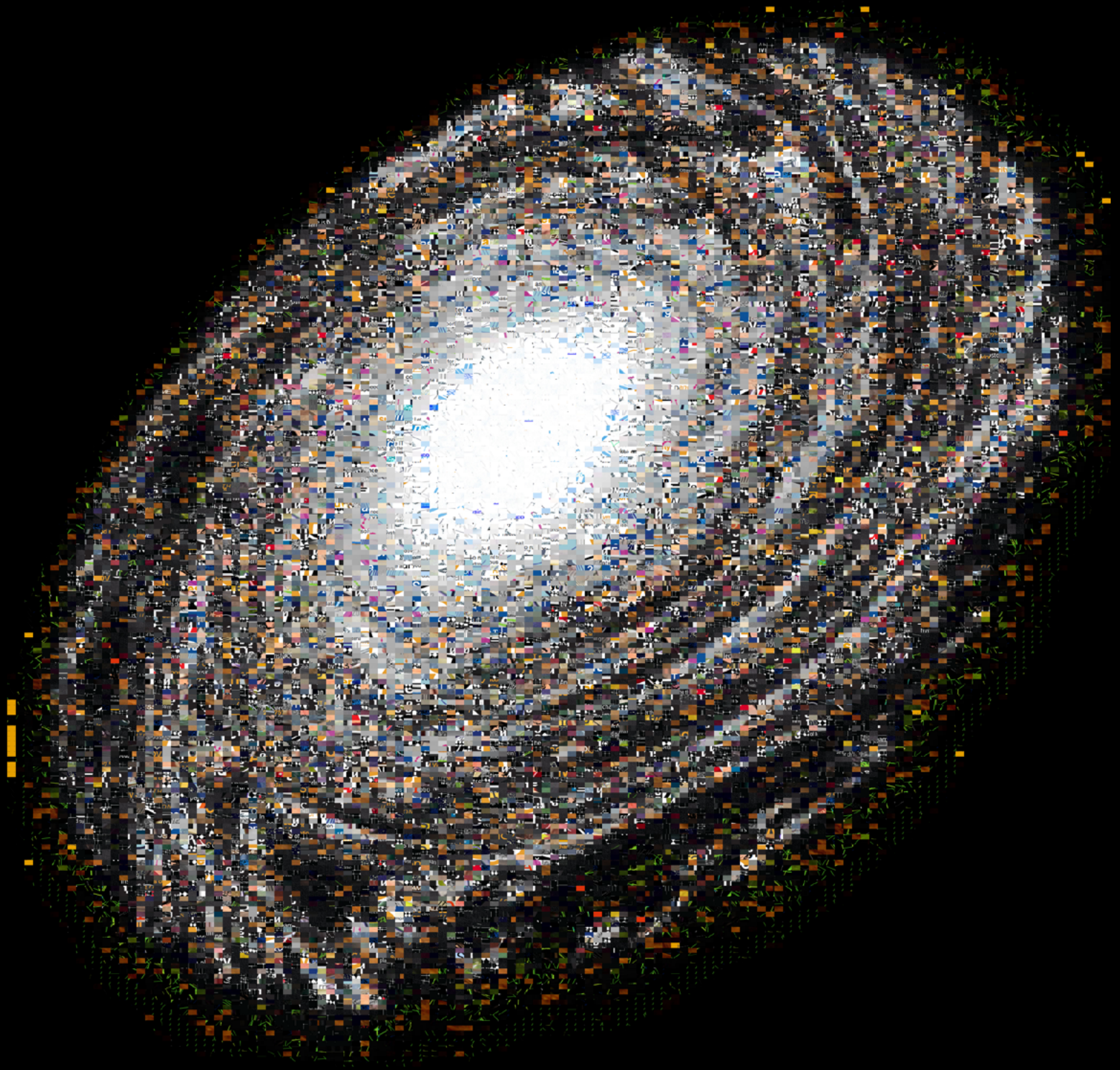
Up until this point I had not yet conducted any typographic experimentation within Mensa. Unfortunately, the algorithm was not designed to work with pure white or black pixel values, which meant that only the anti-aliased parts were targeted for matching. As a result, only letterforms with curves or diagonal lines produced meaningful output. In adjusting and tuning Mensa to handle photographic and typographic materials simultaneously, I created a wholly new version of the program that warranted its own line of thorough experimentation.

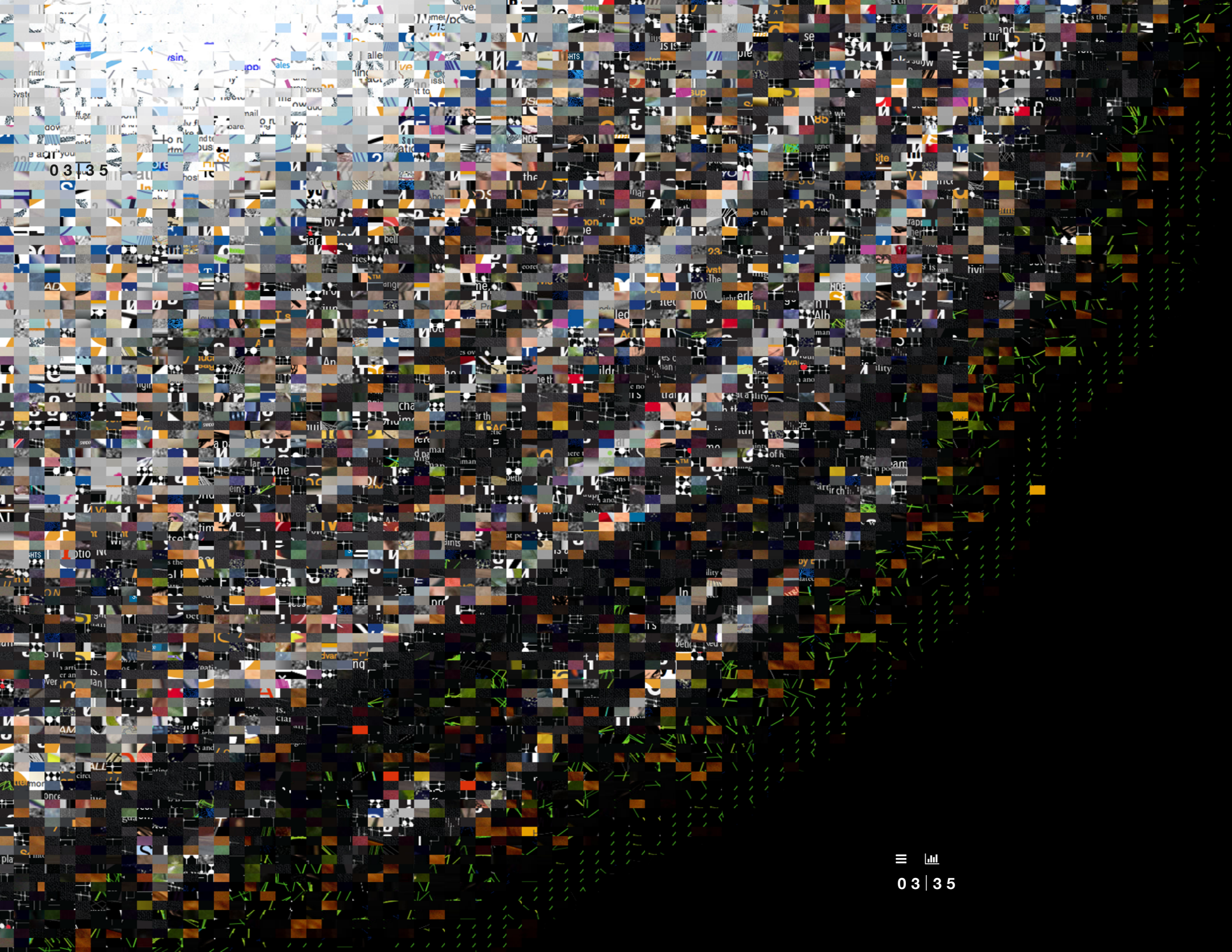


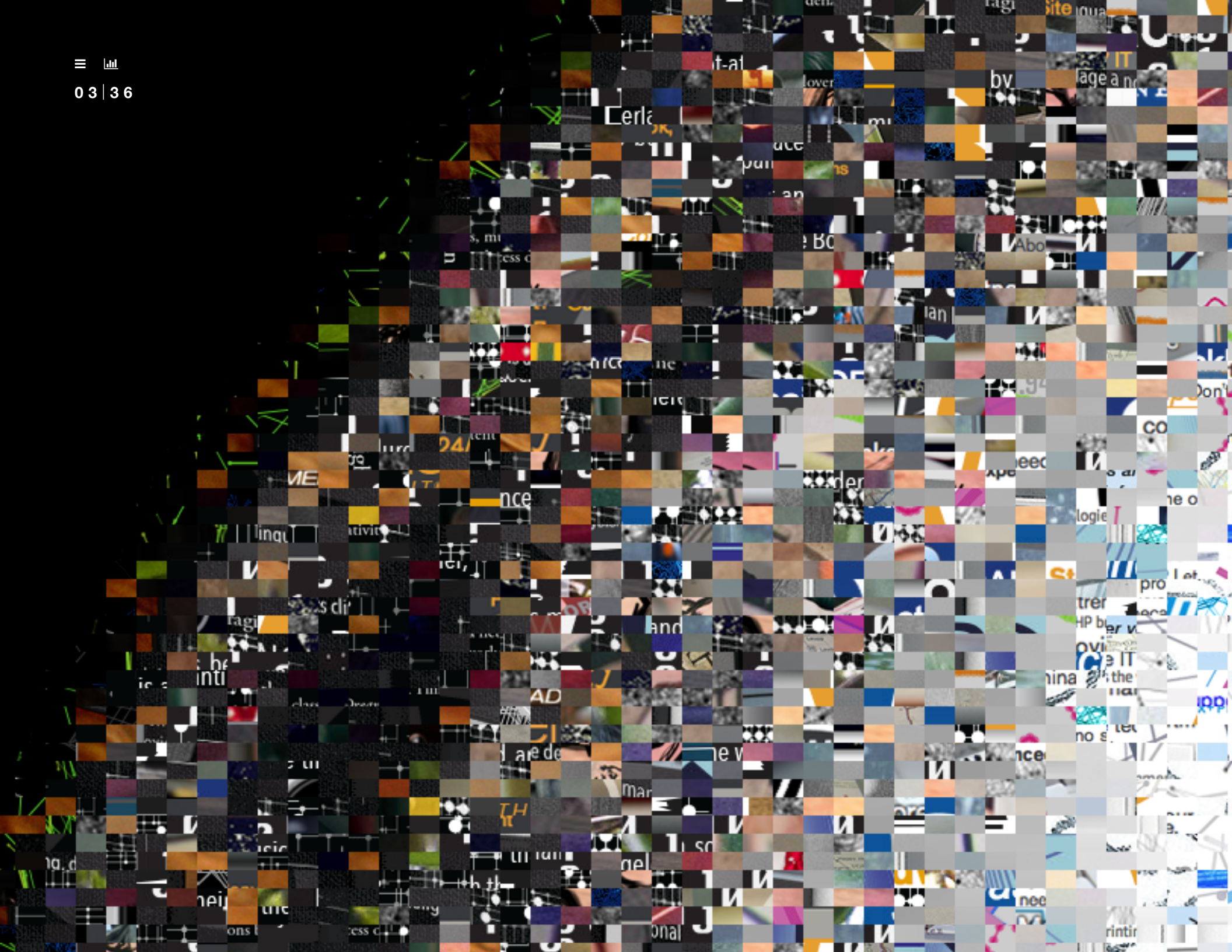


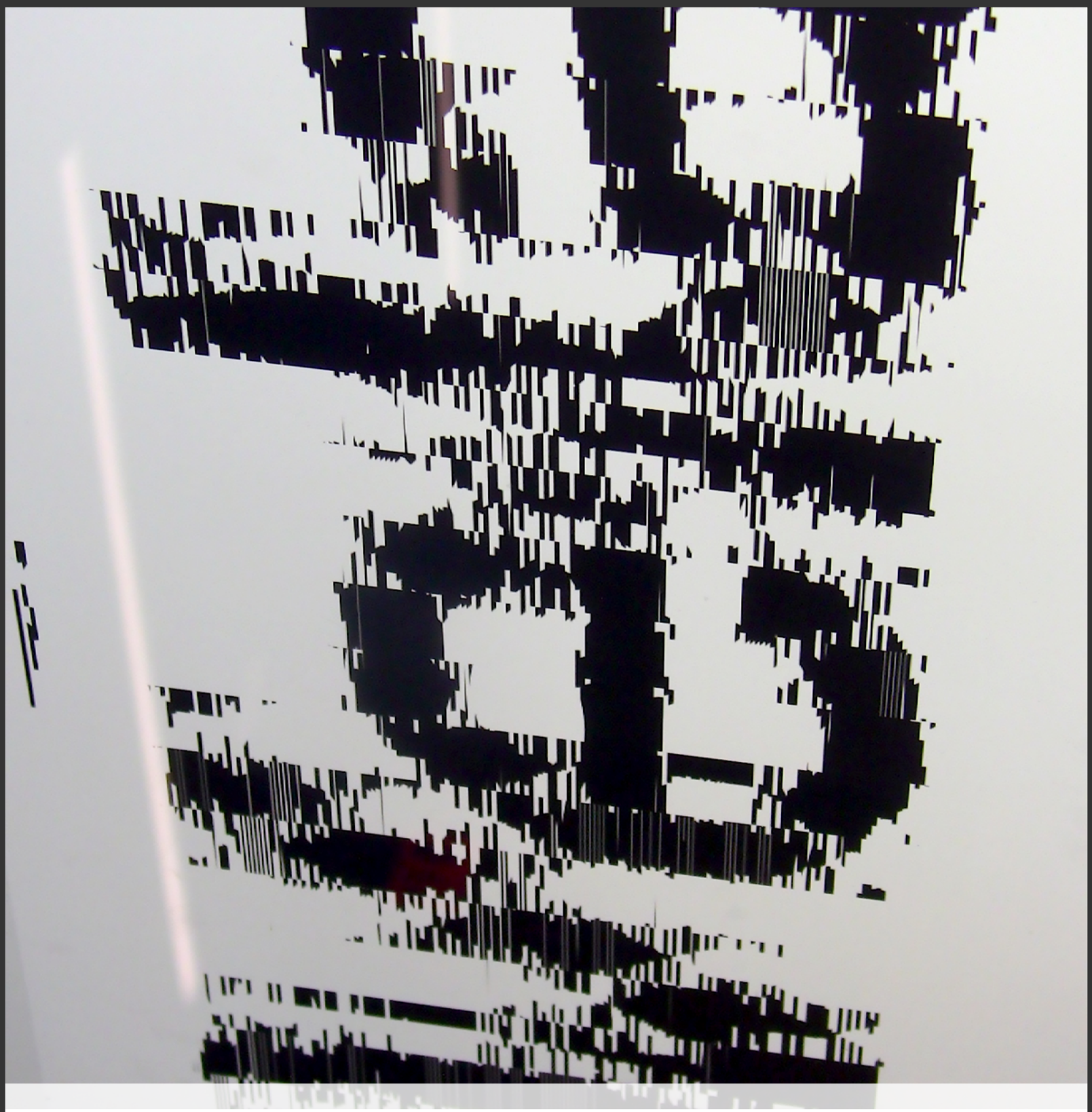

Modify/iterate
Catch/log errors
User
Source materials & parameters
Interface
Source code/SAAS
Designer
Judge result
Update/maintain






03|35
03|35





Project Three

Vela

04 | 38

Vela

Vela is the main focus of Systema and the culmination of my graduate work. Using the same framework as Mensa, this program samples from vector graphics only, specifically type and different letterforms. The reason for the distinction between two very similar projects lies in the output: Vela samples type to make type, while Mensa has no specific sample or source material.

Like the previous two projects, Vela is software. It is an image manipulation machine designed to exist alongside other creative software. Using symbols as media, Vela aims to construct infinite complexity while drawing from a finite, pre-established library of simple elements: the conventional keyboard alphabet, more specifically the ASCII set of characters from 33 to 126, inclusive. The process is the same as Mensa since the two projects share the same underlying structure: the user selects a "master" or source image that acts as the template, then selects a folder of "sample" images that are the materials from which the algorithm dissects and matches.

Following closely on the themes of Mensa, I experimented with portraiture first to test the new algorithm. The results exceeded expectations, handling vectors and type spectacularly. At this point, I also began hard-coding extra features into Vela to reduce processing time and increase general user experience. I moved the computationally-dense matching algorithm out of the animation thread which allowed multiple dissections to run simultaneously, as well as include conversions from RGB to CIELAB colorspace to allow for perceptually-uniform color matching.

Vela relies heavily on grid size to determine legibility of the output. Larger grid sizes yield coarser results. In this example, moving from left to right decreases the width of the grid while height stays uniform. Moving from top to bottom decreases the height of the grid while width stays uniform. The top left corner uses a grid size of one hundred pixels wide by one hundred pixels tall, shrinking to four pixels by four pixels in the bottom right corner.

Most of the experimentation revolves around single letters and symbols. A non-recursive dissection samples every letter except the one being dissected. For example, dissecting a capital letter "S" samples from every letter in the alphabet except capital and lowercase "S." This is to prevent the algorithm from "self-sampling" and returning the input image as the output image. Additionally, almost all experiments with Vela exclusively sample letters from the typeface Caslon; this is because serifs contribute to more visually interesting results.

Vela
(continued)

Recursive dissection more closely represents the goals of Systema: deconstructing complex forms into their components. The recursive process runs a single dissection initially, but saves the result to a buffer which is then recycled back into the algorithm as the master image for the next iteration. This allows for extreme abstraction after only a few generations and often yields unrecognizable results.

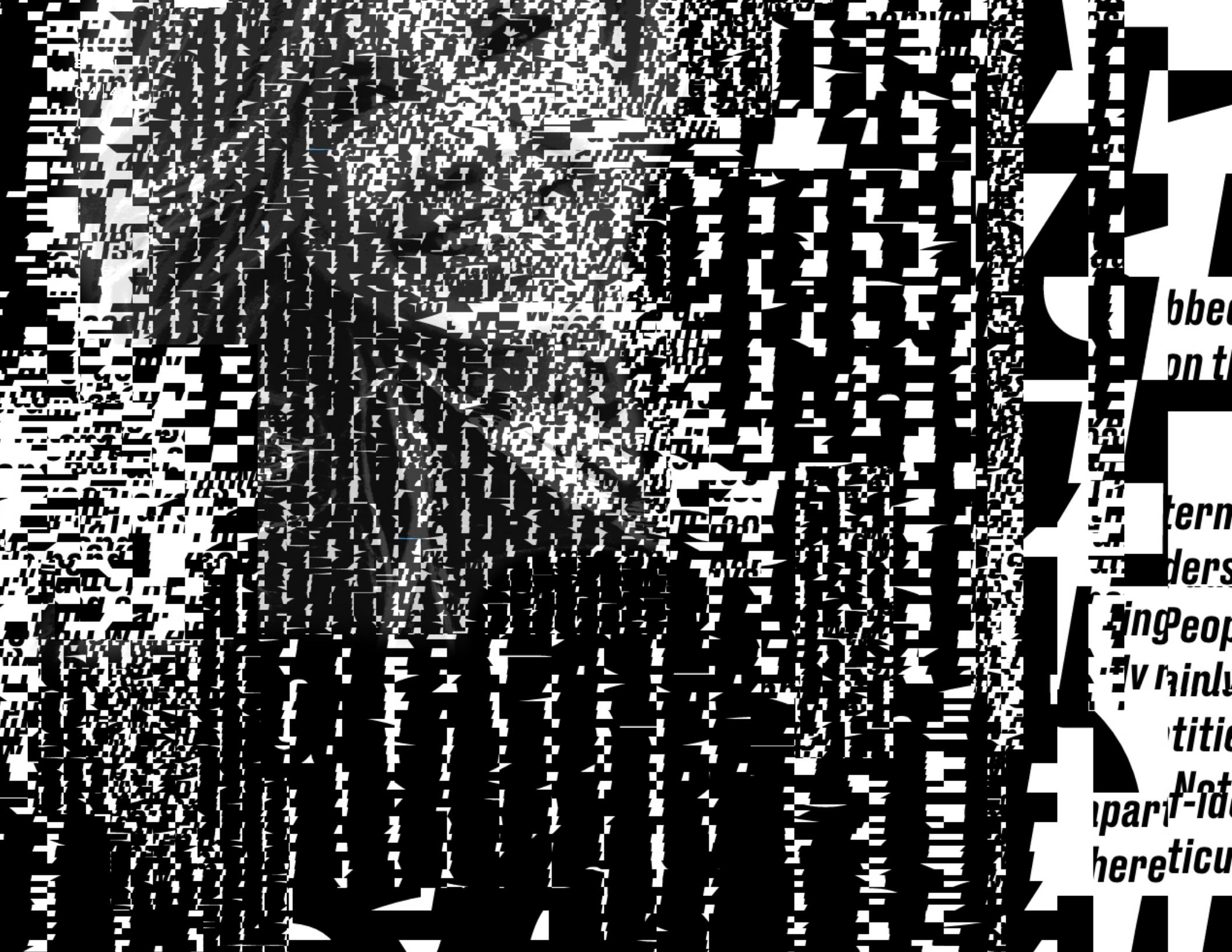
I attempted to elevate my investigations from single letters to single words. Whereas letters are the building blocks for language, words communicate a full idea. Analogously, I wanted to find out if dissolving a word into its components would dissolve the idea into its components. Words dissect and dissolve much like letters, although they require a few less recursive iterations to obfuscate since the algorithm samples nearby letters as well.

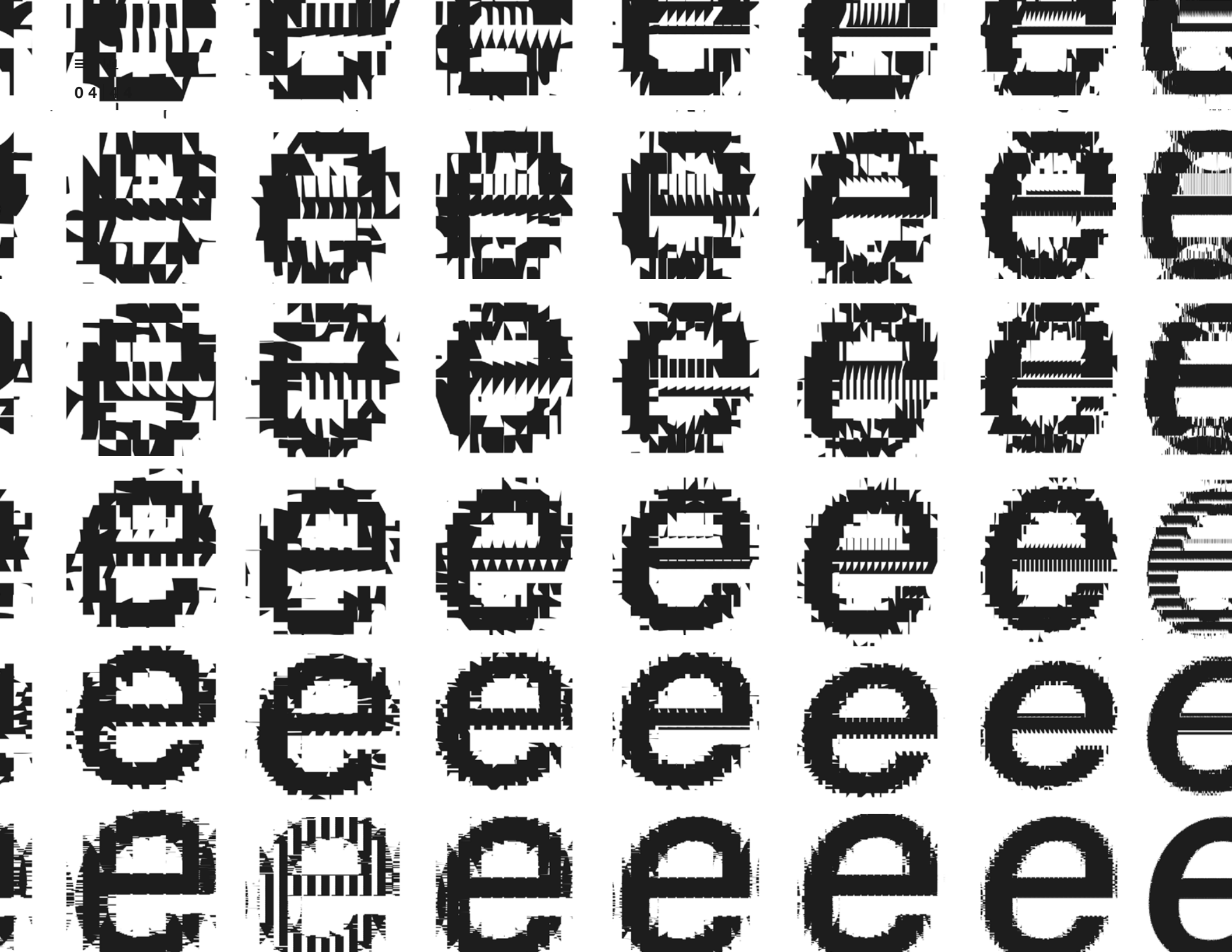
Despite its successes, Vela produced hundreds of unexpected results. I refrain from labelling these as “failures” because the program always followed my instructions; it was the instructions themselves that were incomplete. In many ways the unexpected results became more valuable than producing something intentional as they indicated a gap in the logic of the algorithm. They represented edge cases and opportunities for improvement.

I made a few crude animations from single-frame dissections that provided visual proof of the “event horizon of complexity.” The viewer can watch frame-by-frame as a familiar letterform dissolves into ambiguity only to be reborn as a completely different letter. Moreover, animating the grid size to loop from extreme height to extreme width pushes and pulls the form through different levels of abstraction.

vela.mp4

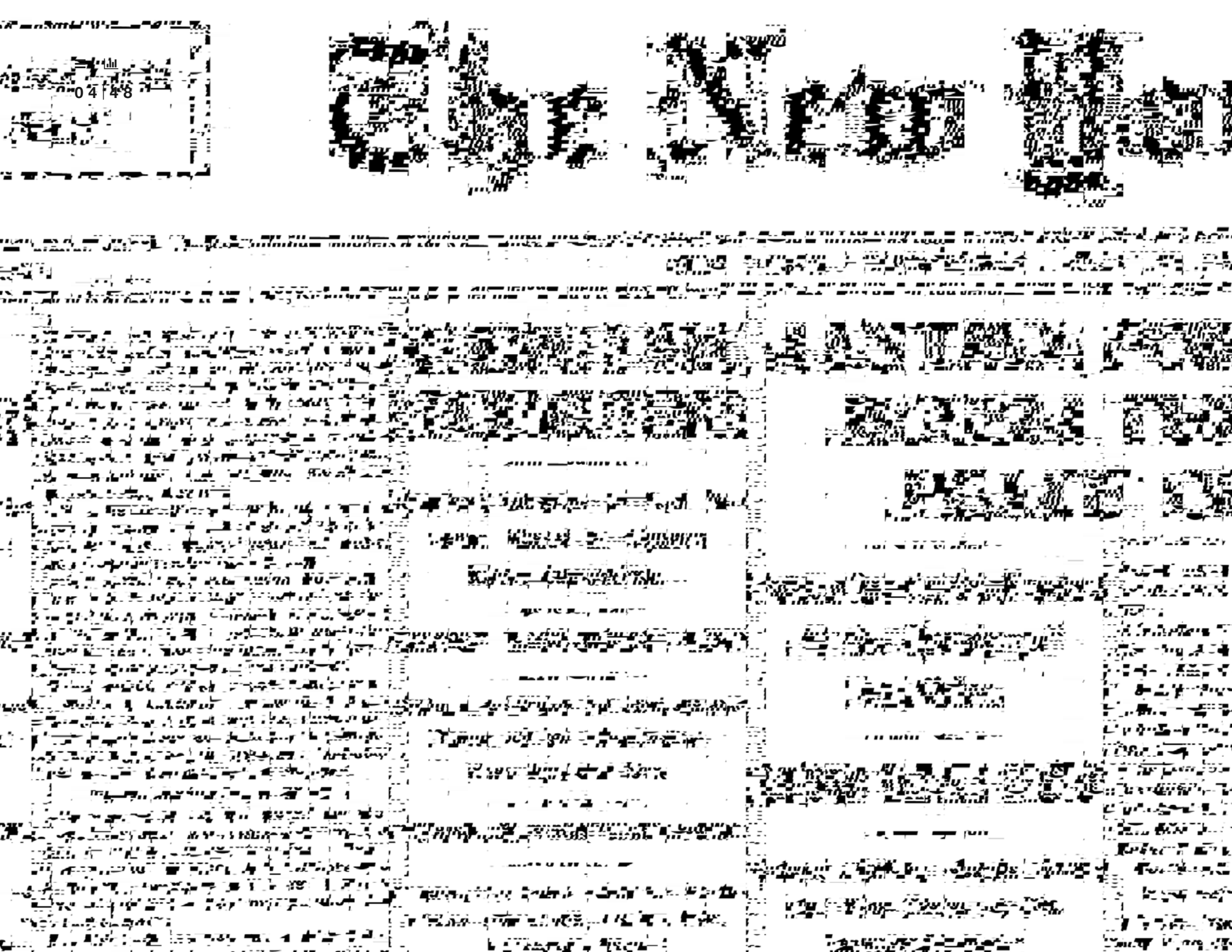

We are all just a car crash,
a diagnosis, an unexpected phone call,

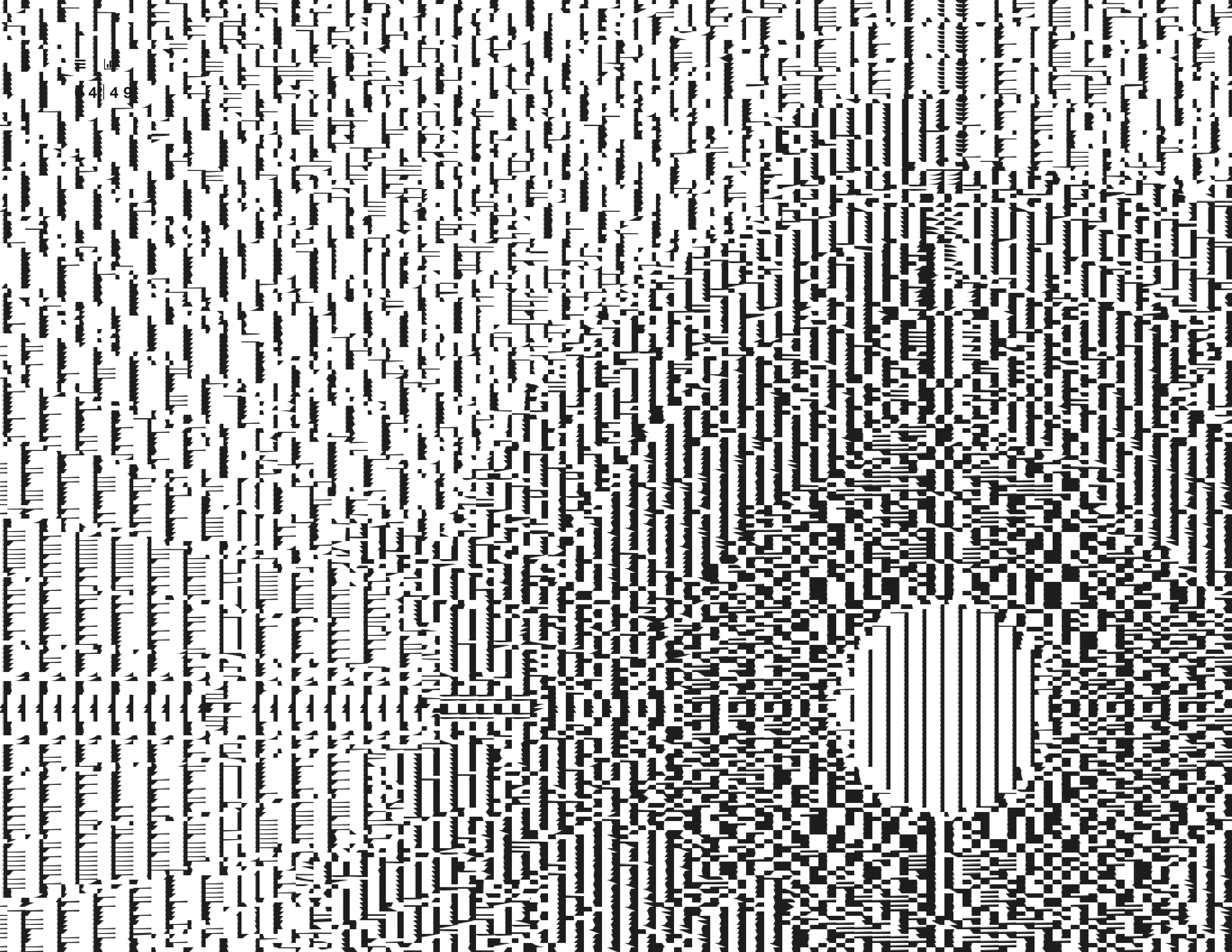


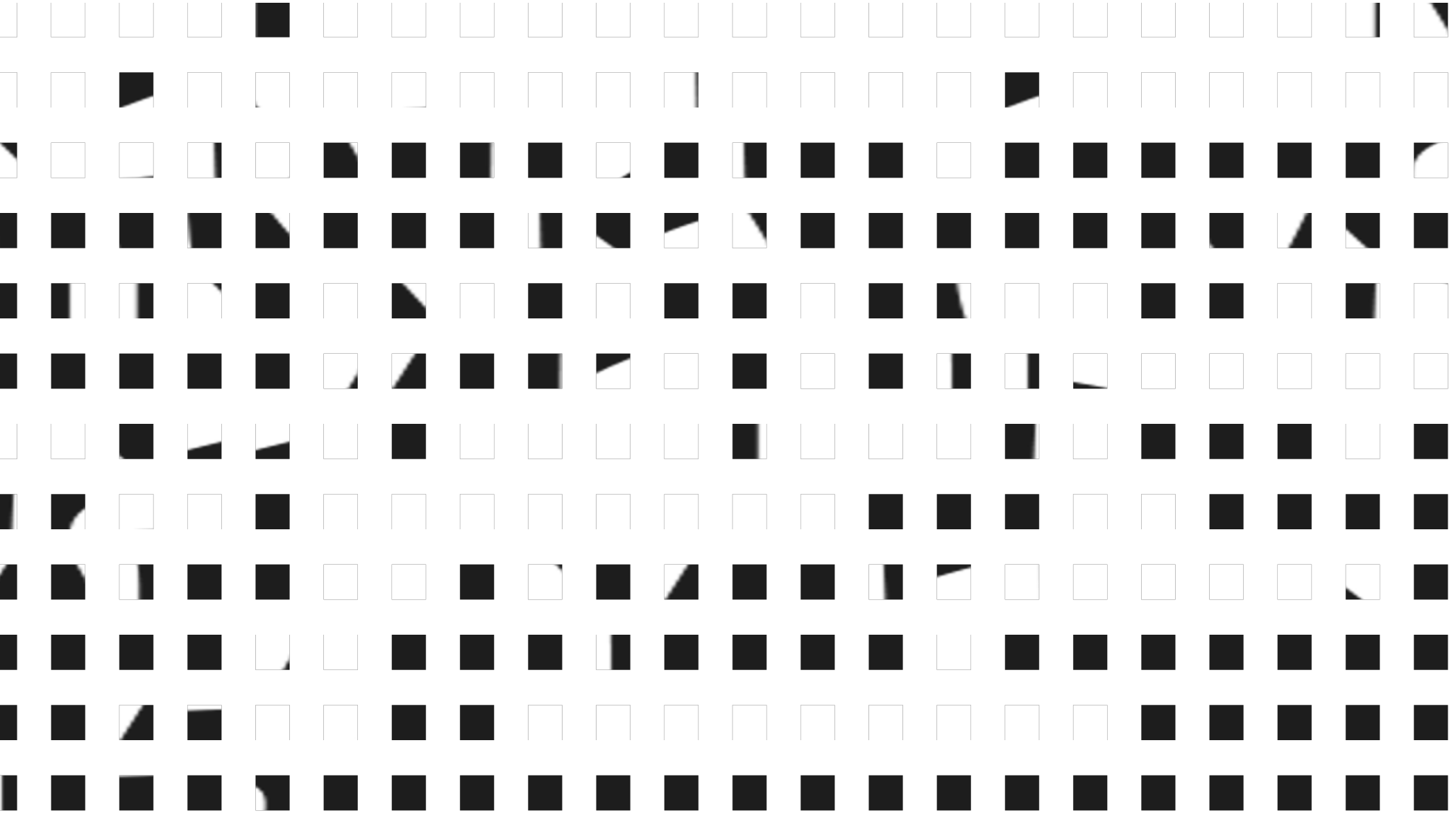


0


The New York Times



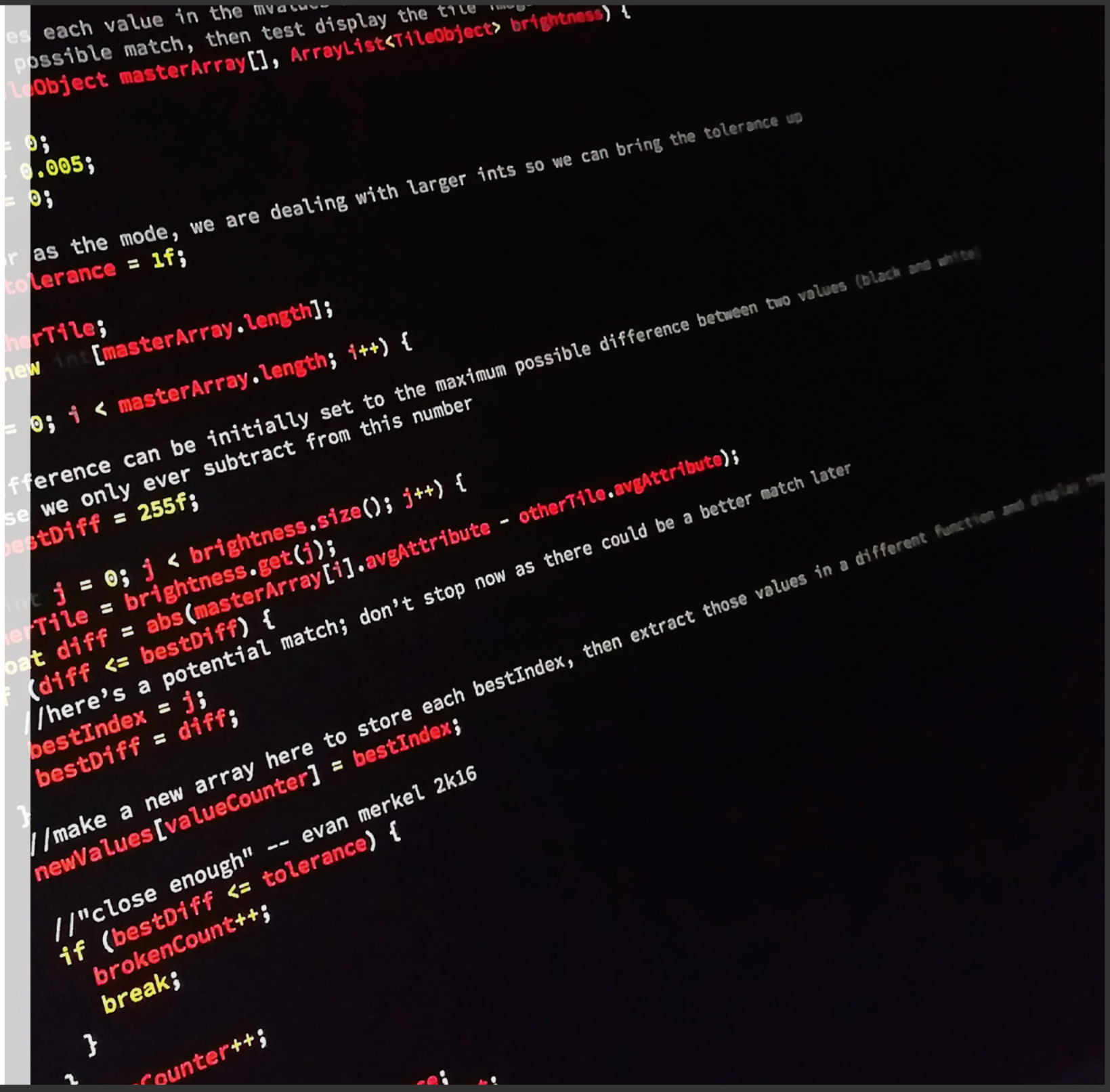




q-to-b.gif

m-vtoh.gif

Conclusion


each value in the mvalue
possible match, then test display the tile
leObject masterArray[], ArrayList<TileObject> brightness) {
= 0;
0.005;
= 0;
as the mode, we are dealing with larger ints so we can bring the tolerance up
olerance = 1f;
herTile;
new int[masterArray.length];
= 0; i < masterArray.length; i++) {
fference can be initially set to the maximum possible difference between two values (black and white)
we only ever subtract from this number
estDiff = 255f;
j = 0; j < brightness.size(); j++) {
herTile = brightness.get(j);
oat diff = abs(masterArray[i].avgAttribute - otherTile.avgAttribute);
f (diff <= bestDiff) {
//here's a potential match; don't stop now as there could be a better match later
bestIndex = j;
bestDiff = diff;
}
//make a new array here to store each bestIndex, then extract those values in a different function and display
newValues[valueCounter] = bestIndex;
//"close enough" -- evan merkel 2k16
if (bestDiff <= tolerance) {
brokenCount++;
break;
}
counter++;

Conclusion

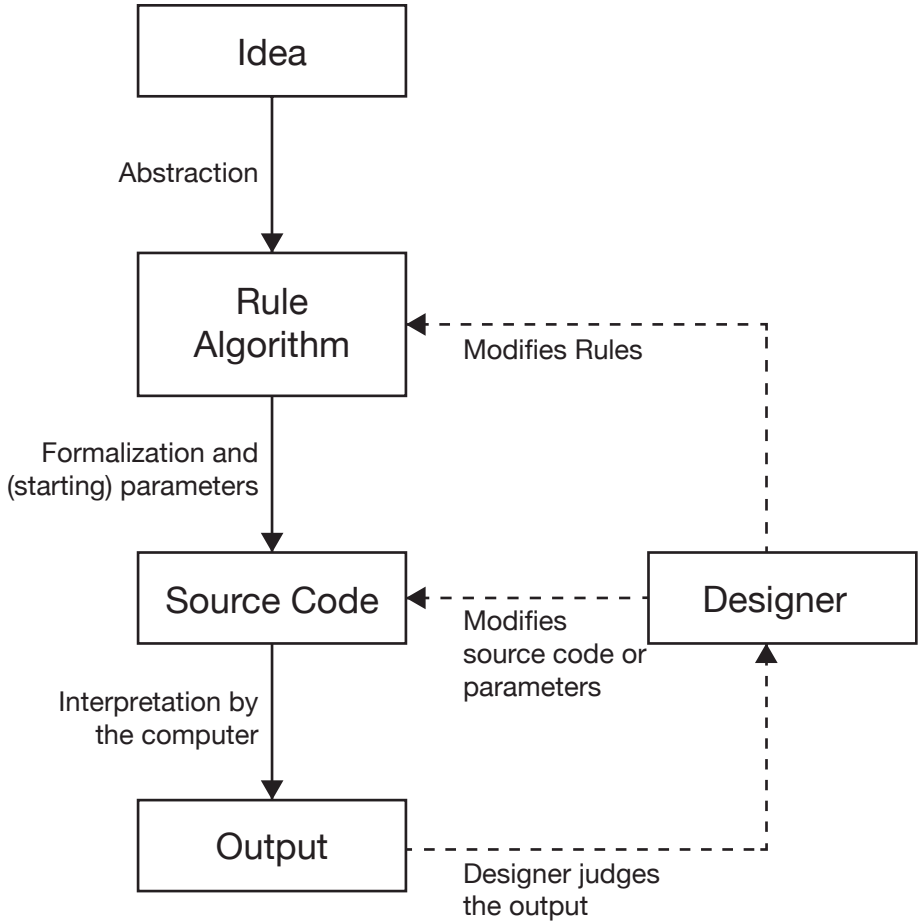
Systema is not about the output, it is about the algorithms that drive the program. These must act as an extension of the designer, following the same logic and patterns that governed the creation of the program. The ability to exclude myself from the final product is a benchmark for success: a fully functional and instructionally complete algorithm needs no designer to oversee its process.

The process diagram illustrated by Hartmut Bohnacker perfectly describes iteration across several stages of the generative design process. It all starts with an idea, which the designer abstracts and transliterates into source code. The code undergoes rigorous testing and tuning until the desired output matches expectations and specifications. Once the system has been completed and delivered, the process changes to almost entirely exclude the designer. Maintenance and debugging still fall to the designer, but the user interacts with the system independently.

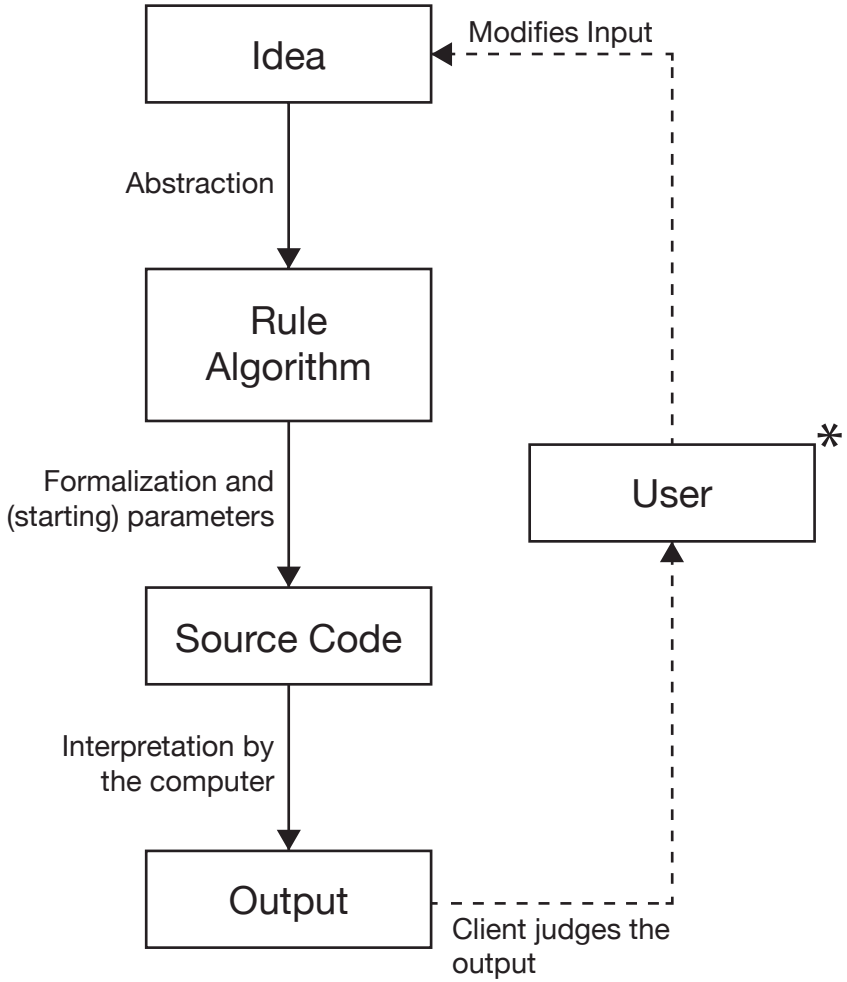
I have personally observed this fundamental shift in the generative process after working for the Virginia Bioinformatics Institute (now the Biocomplexity Institute). I developed a suite of programs designed to procedurally generate visual assets for the new brand. The creative director expressed her vision for the new identity for the institute while I created programs and formulae to fulfill that vision. Those programs are still used by the creative team to generate visual assets at the click of a button. Features such as adjustable parameters for size, target, and pre-approved color choices allow for curated flexibility and variety while maintaining brand integrity.

This is the future of branding: extensible, definable, complete systems that abstract identity into parameters and variables that can be easily manipulated by the user. Reas' initial goal to "minimize the technical aspects" of programming ("{Software} Structures") is directly analogous to the goal of Systema, which attempts to "minimize the technical aspects" of generative graphic design by abstracting variables and algorithms behind a user interface. Allowing users to create their own assets and materials gives them the ability to interpret their brand in new and unique ways using the same system.

Pre-delivery (generative process)

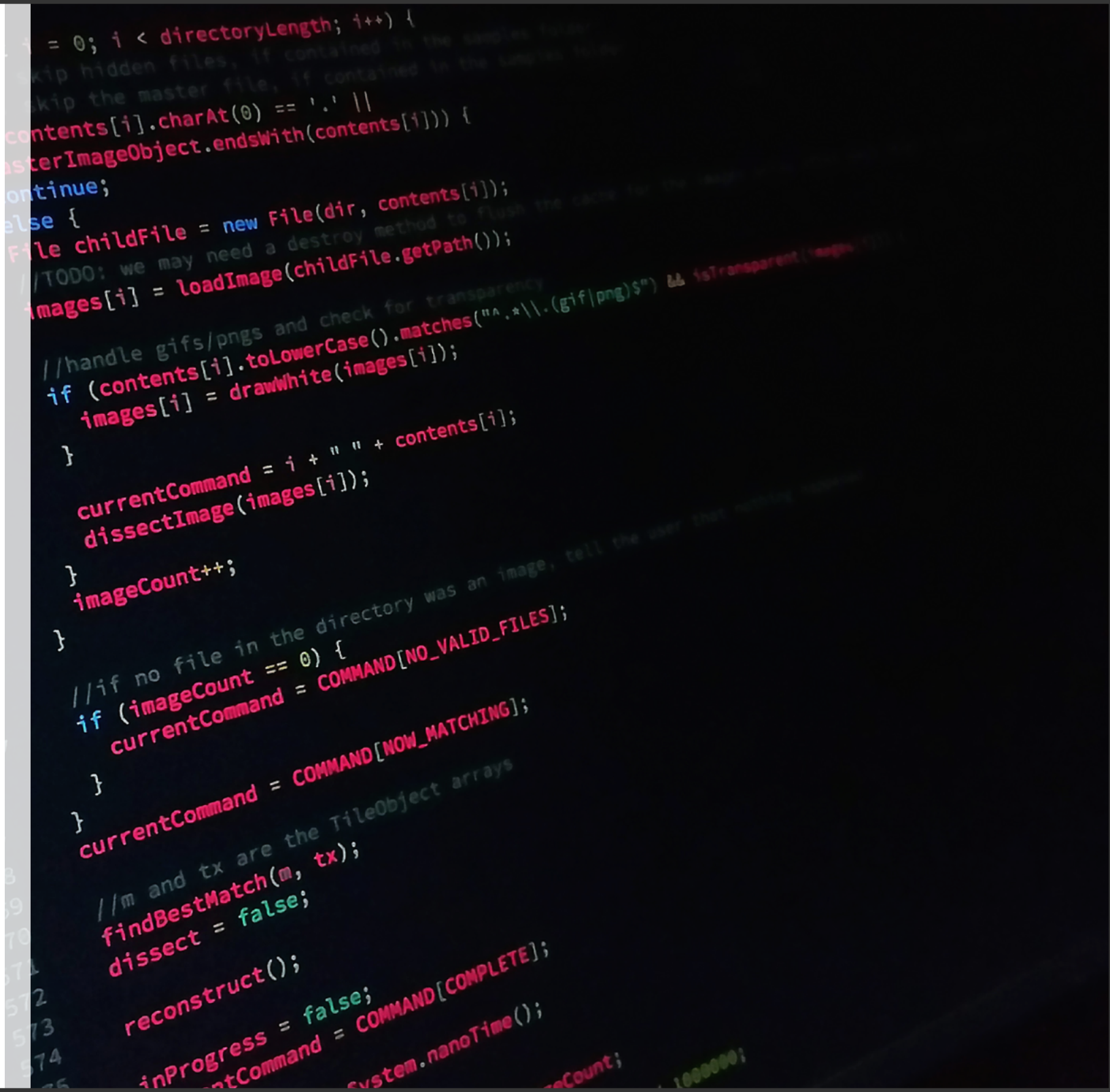


Post-delivery (user interaction)




Virginia Bioinformatics Institute
Research Symposium 2015
6:00 p.m. | April 9, 2015

Future


= 0; i < directoryLength; i++) {
ntents[i].charAt(0) == '.' ||
terImageObject.endsWith(contents[i])) {
ntinue;
lse {
ile childFile = new File(dir, contents[i]);
mages[i] = loadImage(childFile.getPath());
//handle gifs/pngs and check for transparency
if (contents[i].toLowerCase().matches("^.*\\.(gif|png)$") &&
images[i] = drawWhite(images[i]);
currentCommand = i + " " + contents[i];
dissectImage(images[i]);
imageCount++;
//if no file in the directory was an image, tell the user
if (imageCount == 0) {
currentCommand = COMMAND[NO_VALID_FILES];
currentCommand = COMMAND[NOW_MATCHING];
//m and tx are the TileObject arrays
findBestMatch(m, tx);
dissect = false;
reconstruct();
inProgress = false;
ntCommand = COMMAND[COMPLETE];
ystem.nanoTime();

Future

Systema is software. It is infinitely extensible to the limits of imagination and the constraints of the language in which it is written. Although I have conducted extensive testing over several years, the future of these projects remains open.

User testing is my first priority. I have been primarily the sole user of each project in Systema; only a handful of friends and peers used the software during the late stages of Mensa and the generative portrait studio. The user interface is rudimentary and chronologically organized. Every time I coded in a new feature, I added a button or slider at the bottom of the column of buttons and sliders. Systema is an example of code that focuses on function at the cost of form.

Beyond looks, these projects are a complete system. They are self-contained microcosms that follow instructions very well, but the instructions are years old, even at the time I am writing this document. Old instructions are like new technology: they fade or are rendered obsolete by the next big thing. Old technology follows a different trend, however. Chalk and slate have been around for quite some time, after all.

And Systema is only software.


Has just
changed
on state-

References

Albiac, Sergio. “I Am.” 2016. https://www.sergioalbiac.com/wall/i-am.html. Accessed Dec 2016.
“I Am.” is a generative portraiture series by the artist Sergio Albiac. It utilizes speech-to-text software, a microphone, and a webcam to automatically create a portrait of any given user. Albiac used this system to create over 10,000 portraits. I developed a project nearly identical to “I Am.” but was yet incomplete when Albiac’s was released. This citation is included to justify the change in direction in my research in late 2016.

Bohnacker, Hartmut, et al. Generative Design: Visualize, Program, and Create with Processing. Princeton Architectural Press, 2012.
This book is the generative design bible. More exhibition than exposition, this text walks the reader through various lessons and examples of well-programmed generative design sketches.

Reas, Casey. “Process Compendium (Introduction).” Vimeo, uploaded by Casey REAS, 27 April 2011. vimeo.com/22955812.
This video describes Casey Reas’ work “Process Compendium” and its many variations. Reas narrates the video and discusses the logic behind this series of designs. Reas has been a personal inspiration since I started learning Processing, and this video, in many ways, is one of the sources for the core concept of my thesis: building complexity from simplicity.

Reas, Casey. “{Software} Structures.” Whitney Museum of American Art, Aug 2004, artport.whitney.org/commissions/softwarestructures2016/text.html. Accessed 10 Oct 2017.
Reas discusses his 2004 work titled “{Software} Structures” in this article. Specifically, he relates the idea of software as art to Sol LeWitt’s wall drawings, a comparison critical to the development of Systema.

Smith, Hazel, and R. T. Dean, editors. Practice-Led Research, Research-Led Practice in the Creative Arts. Edinburgh, Edinburgh University Press, 2009.
Hazel and Dean’s justification of practice-led research as a valid contribution to creative arts contributed to the ideas in the introduction and conclusion of Systema.

Yowell, Jake. “Salve from an old Latin student.” Received by Evan Merkel. 26 Oct. 2017. E-mail.
The core concept of Systema comes from a lecture given by Yowell in 2007. In this email, I ask him to describe the history, motivations, and impact that this lecture has had on students over the years.

Zappaterra, Yolanda. “Generative design: Redefining the designer.” Communication Arts. www.commarts.com/columns/generative-design-redefining-the-designer. Accessed 15 Nov 2017.
Onformative, FIELD, and other agencies that specialize in generative design all contributed to this article. As leaders in generative design, their advice and process influenced some of the theory behind my projects.


Systema