Student ID: 25002544
Prae Sunsermsook

Unit 1
Methods of Iterating
Written Response

Draft 1

Replicating DIA Studio's generative poster application in p5.js shifted my practice from making visuals to translating intentions into commands. The poster's forms look simple, such as dots, curves, and adjustable parameters, but copying revealed how much the output depends on strict rules: syntax, execution order, and mathematical relationships that I am not used to. I spent a lot of time patching snippets from tutorials and documentation, then reorganising functions and renaming variables so I could reread my own code. Even then, the code often felt unreadable, and I had to look up its library as if it were a dictionary.

If coding means communicating between two sides: computer demanding precision and human needing clarity and literacy, what is a good code? How do languages for coding, such as abstract variable names, library-specific vocabularies, and optimisation practices, like minified code, redefine the experience of reading and authoring generative systems?

I will explore these questions by creating a generative tool that displays its own code as it runs. I'll iterate through different versions, such as readable versus minified, organised versus chaotic, and treat language and formatting as part of the visual output, not just a hidden system.

BIOGRAPHY

Dia Studio (2025) *Us, Me, You, Them, Here, There* [Digital art].
Available at: https://www.instagram.com/p/DOeV-szEnZm/?hl=en&img_index=1 (Accessed: 15 January 2026).

Student ID: 25002544
Prae Sunsermsook

Draft 2

INTRODUCTION

p5.js describes itself as a 'friendly' tool that allows creative practitioners learn coding in an 'intuitive' way similar to sketching in a notebook. However, as a graphic designer with limited coding experience, I found this far less intuitive than visual tools such as Adobe Creative Suite. While JavaScript is a programming language written by humans to command machines, it creates a conflict between our intuitive expression and the machine's demand for logic and precision. The rigidity of code syntax enforces a linear, uniform structure that excludes artistic expression, prioritising machine readability over human emotion.

This project investigates this tension by treating code as visible material through improvising and patching together existing code from library functions and various community resources. This process aligns with Adhocism, which emphasises the adaptation and recombination of existing resources (Jencks and Silver, 2013). I use this theory as a lens to examine the gap between p5.js's claims of accessibility and the barriers it conceals.

MECHANICAL EVOLUTION

Mechanical evolution in p5.js prioritises efficiency by simplifying complex operations into accessible functions. The quadraticVertex() function, for example, streamlines drawing curves for beginners. However, when I needed to calculate specific point positions the library did not provide, I had to implement calculations manually using the derivative of the Bézier formula: $B'(t) = 2(1-t)(P_1 - P_0) + 2t(P_2 - P_1)$. This creates an illusion of accessibility that disappears as practitioners move beyond standard functions, ultimately requiring a mathematical understanding that the library seeks to minimise. Optimisation extends to every code formatting. Monospaced typeface, shorthand, and linear structure reduce the expressive qualities of human writing to suit machine parsing. Minification strips away human-readable elements (e.g., comments, whitespace, meaningful variable names) to improve machine performance. This uniformity prioritises computational logic over expressive form, transforming the 'sketchbook' into an incomprehensible infrastructure.

Student ID: 25002544
Prae Sunsermsook

Draft 2

CRITICAL EVOLUTION

To challenge this mechanical evolution, I developed three forms of critical subversion through ad hoc methods:

Function subversion: I repurposed the translate() function as linguistic translation, running code through the Google Translate API across multiple languages. When combining all languages, it exposed how English-centric terminology breaks down into glitchy texture, a visual representation of the emotional friction and 'lost in translation' experience faced by non-native practitioners.

Form subversion: I expanded code into a solid black mass as an act of anti-minification. When layering this with blend modes, it created organic textures that resemble ancient sedimentary or inscription rock. This outcome reminded me that language has always adapted to the materials of its medium. Just as script once conformed to stone and paper, code is now shaped by the rigid logic of the screen. While code rigidity is inevitable, it reflects an intention to prioritise machine performance over human expression.

Action subversion: I created a drawing tool that used source code as digital ink, subverting the act of typing by turning it into drawing. This transformed precise syntax into an imprecise, graphite-like texture, with shaking movement. It represents the friction between the human hand and a machine structure that allows no room for error. Code transitions from computational instruction to expressive material, allowing human gestures and imperfections to re-enter a visual system of machine precision.

CONCLUSION

In conclusion, p5.js evolves as a form of mechanical optimisation rather than true inclusivity for beginners. Treating code as a malleable material reveals the linguistic barriers hidden under the claim of friendliness. These subversions reframe accessibility as a negotiated condition in which creative agency emerges through adaptation, misuse, and reinterpretation rather than technical mastery. In this context, coding transforms from a challenge of overcoming the machine into an ongoing negotiation between human expression and computational logic.

BIOGRAPHY

Jencks, C. and Silver T. (2013) *Adhocism: The Case for Improvisation*. Cambridge: MIT Press.

Student ID: 25002544
Prae Sunsermsook

Draft 3

INTRODUCTION

p5.js describes itself as a 'friendly' tool that allows creative practitioners learn coding in an 'intuitive' way similar to sketching in a notebook. However, as a graphic designer with limited coding experience, I found this far less intuitive than visual tools such as Adobe Creative Suite.

While JavaScript is a programming language written by humans to command machines, it creates a conflict between our intuitive expression and the machine’s demand for logic and precision. The rigidity of code syntax enforces a linear, uniform structure that excludes artistic expression, prioritising machine readability over human emotion.

This project investigates this tension by treating code as visible material through improvising and patching together existing code from library functions and various community resources. This process aligns with Adhocism, which emphasises the adaptation and recombination of existing resources (Jencks and Silver, 2013). I use this theory as a lens to examine the gap between p5.js's claims of accessibility and the barriers it conceals.

Student ID: 25002544
Prae Sunsermsook

Draft 3

MECHANICAL EVOLUTION

Mechanical evolution in p5.js prioritises efficiency by simplifying complex operations into accessible functions. The quadraticVertex() function, for example, streamlines drawing curves for beginners. However, when I needed to calculate specific point positions the library did not provide, I had to implement calculations manually using the derivative of the Bézier formula: $B'(t) = 2(1-t)(P_1 - P_0) + 2t(P_2 - P_1)$. This creates an illusion of accessibility that disappears as practitioners move beyond standard functions, ultimately requiring a mathematical understanding that the library seeks to minimise.
Optimisation extends to every code formatting. Monospaced typeface, shorthand, and linear structure reduce the expressive qualities of human writing to suit machine parsing. Minification strips away human-readable elements (e.g., comments, whitespace, meaningful variable names) to improve machine performance. This uniformity prioritises computational logic over expressive form, transforming the 'sketchbook' into an incomprehensible infrastructure.

Student ID: 25002544
Prae Sunsermsook

Draft 3

CRITICAL EVOLUTION:

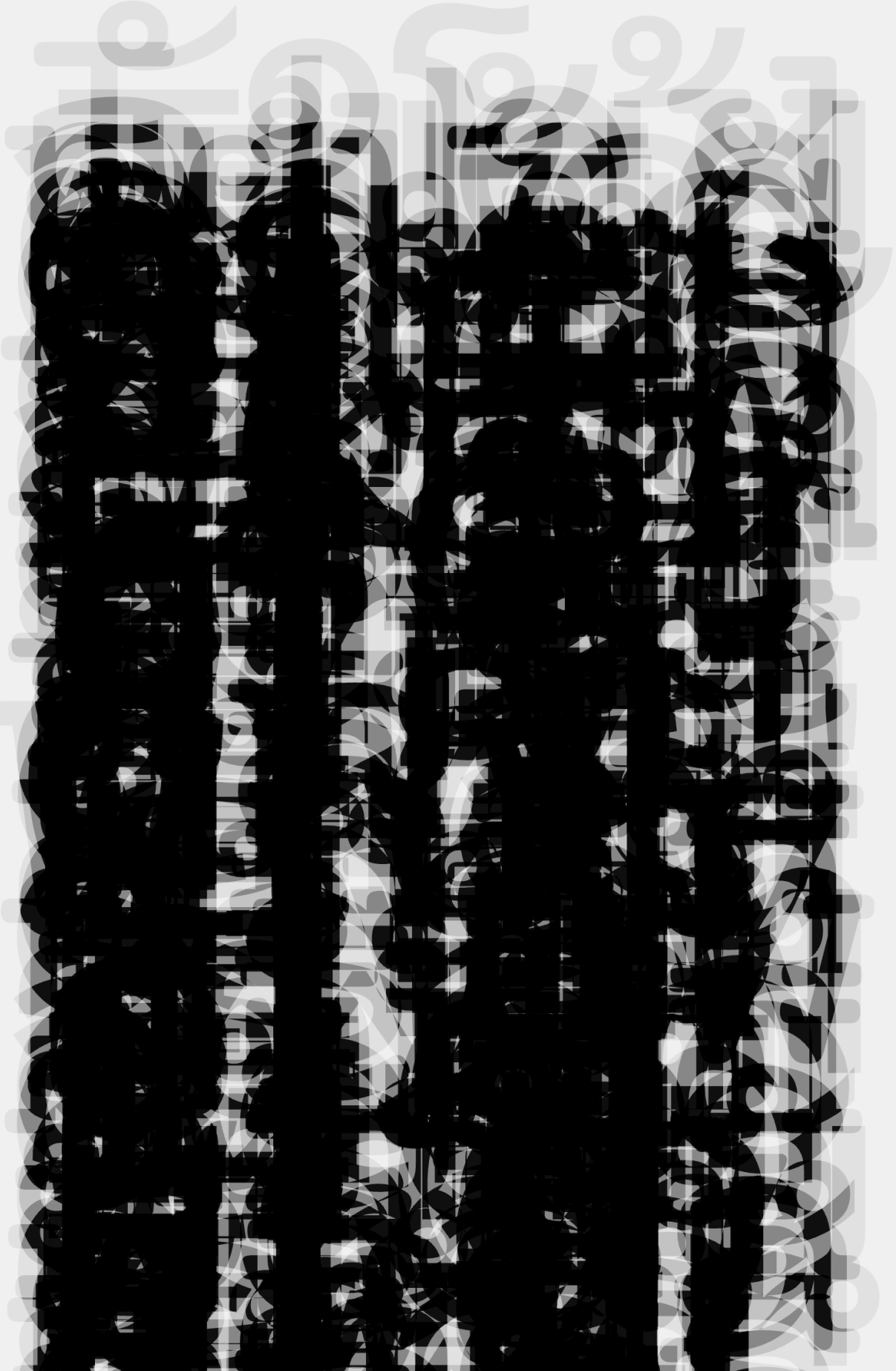
Function Subversion

Student ID: 25002544
Prae Sunsermsook

Draft 3

CRITICAL EVOLUTION:

Form Subversion



Student ID: 25002544
Prae Sunsermsook

Draft 3

CRITICAL EVOLUTION:

Action Subversion