torn at the seams: considering digital vernacular

Michael Murtaugh

> Processing is a free, open source programming language and environment used by students, artists, designers, architects, researchers and hobbyists for learning, prototyping, and production. Processing is developed by artists and designers as an alternative to proprietary software tools in the same domain. The project integrates a programming language, development environment, and teaching methodology into a unified structure for learning and exploration. […]
>
> Most of the examples presented in this book have a minimal visual style. This represents not a limitation of the Processing software, but rather a conscious decision by the authors to make the code for each example as brief and clear as possible. We hope the stark quality of the examples gives additional incentive to the reader to extend the programs to her or his own visual language.[1]

Teaching programming with free software to media design students for years, I've resisted Processing as it has always seemed to me to embody a particular kind of solipsism of digital interactivity and graphics that I want my students to avoid.

> Solipsism: a theory holding that the self can know nothing but its own modifications and that the self is the only existent thing; also: extreme egocentrism[2]

In the summer of 1996, as I was finishing my Masters degree, I met John Maeda who was just joining the MIT Media Lab to replace the recently deceased Muriel Cooper. Cooper was the first art director of the MIT Press, producing influential designs such as a 1969 catalog of the Bauhaus and the iconic MIT Press logo, a Bauhaus-inspired stylized graphical rendering of the letters "mitp". Cooper started the Visible Language Workshop, later one of the founding groups of the MIT Media Lab, to research the intersection of publishing, design, and computation.

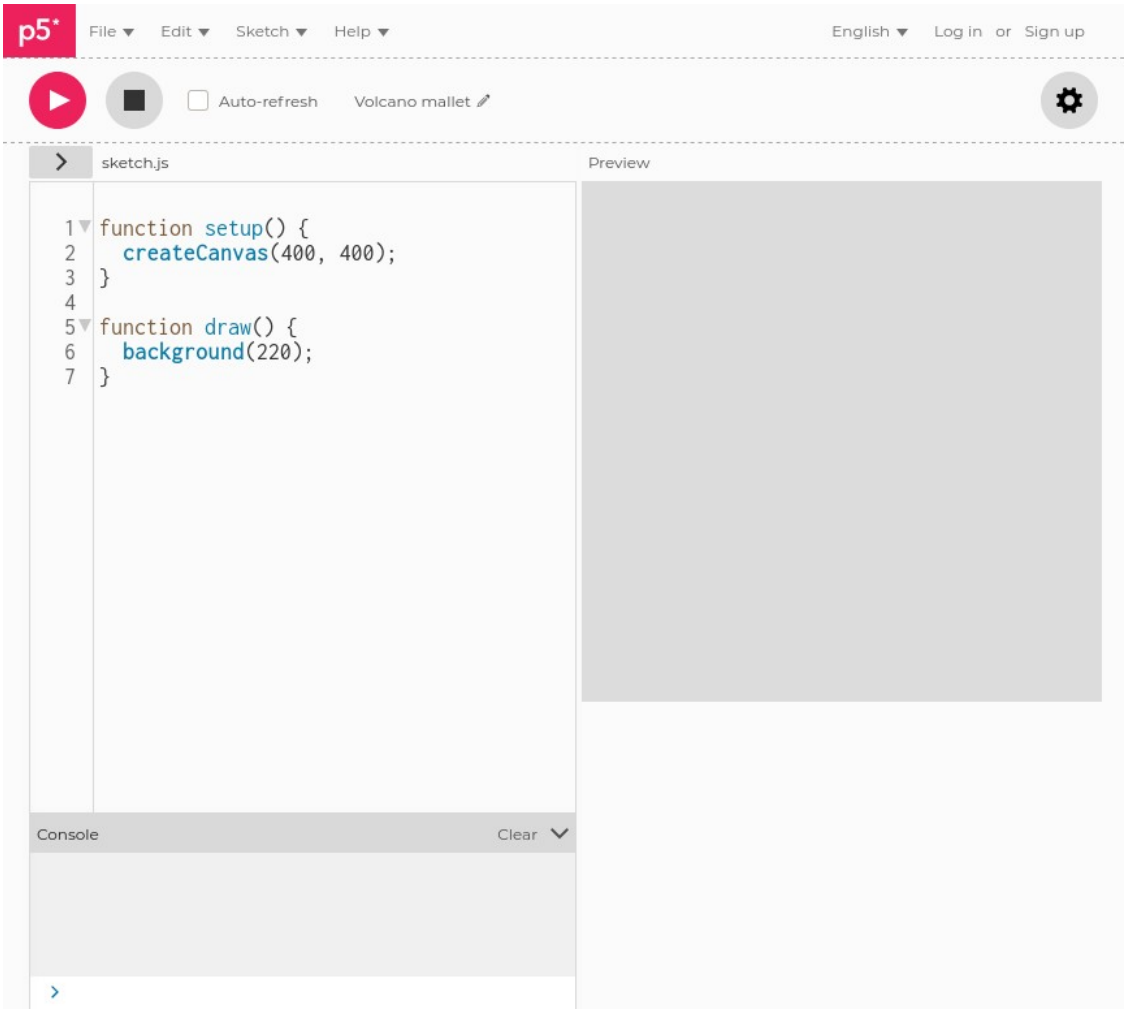
My friend Robin was a student in the VLW; through her I learned things like how Muriel would say "Swiss" instead of Helvetica. I took one class in typography at the VLW; among the many constraints we were given, we could only use one font, named for swiss typographer Adrian Frutiger.

---

[1] Casey Reas and Ben Fry, *Processing: A Programming Handbook for Visual Designers and Artists* (Cambridge: MIT Press, 2007), xxi-xxii.

[2] https://www.merriam-webster.com/dictionary/solipsism

Maeda created the Aesthetics and Computation group in part to continue Cooper's research. Ben Fry and Casey Reas were among Maeda's first students. Maeda published a software system called Design by Numbers. It had extreme constraints such as a fixed 100 by 100 pixel size and monochrome-only graphics. The accompanying print publication also had a square format. DBN would go on to inspire Reas + Fry to create Processing. When one starts the Processing application, the default starting code comprises two commands: createCanvas and background. The default canvas output is square, and a single numeric parameter to the background command produces a gray-scale value, both echoes of earlier DBN constraints. Both DBN and Processing were written in Java, a desktop application that exists outside of the web but which can be used to publish sketches as "applets" embedded in a web page and published online.



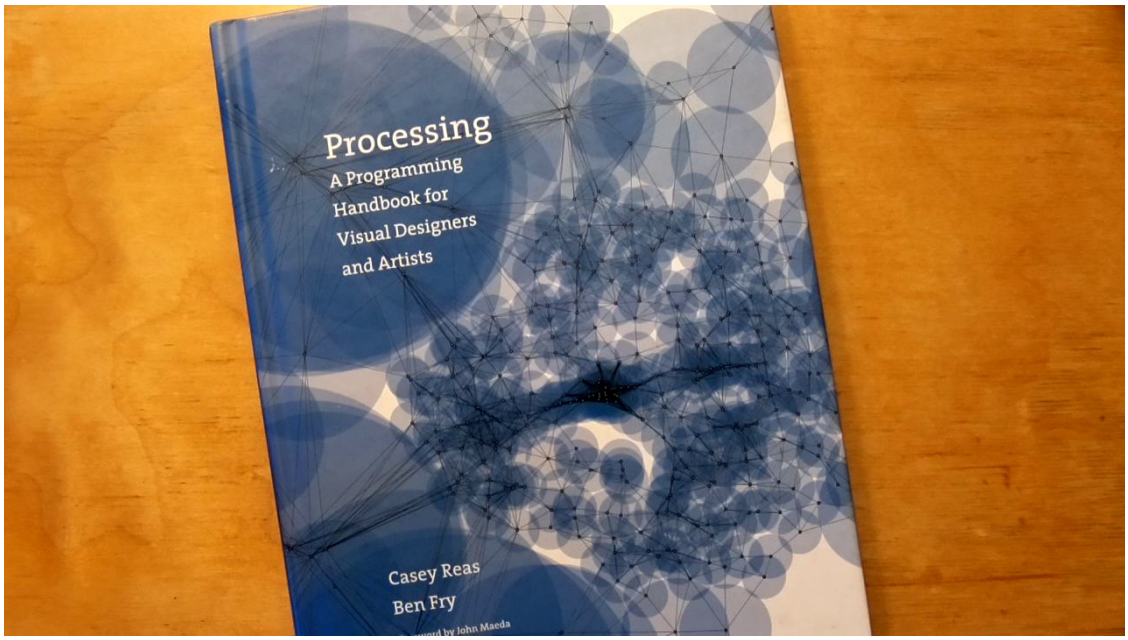
The p5js editor default starting code and output

In 2004, Reas co-developed an exhibition at the Whitney gallery called Software Structures. Inspired by Sol Lewitt's wall drawings, Software Structures presented a series of abstract rules for the production of an image, including some from Lewitt. The rules were then implemented using a variety of "materials": Processing, Flash MX, and C++.

> A benefit of working with software structures instead of programming languages is that it places the work outside the current technological framework, which is continually becoming obsolete. Because a software structure is independent from a specific technology, it is possible to continually create manifestations of any software structure with current technology to avoid retrograde associations.[3]

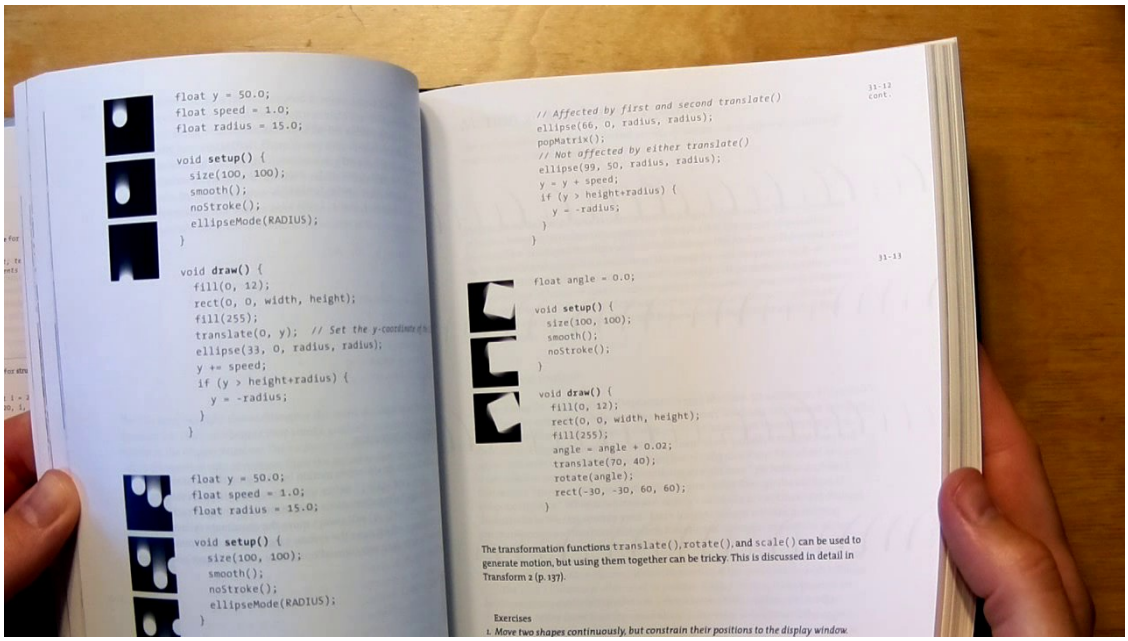
two books

In 2007 I attended a book launch of: Processing: A Programming Handbook for Visual Designers and Artists (MIT Press, Reas + Fry, with a foreword by Maeda).

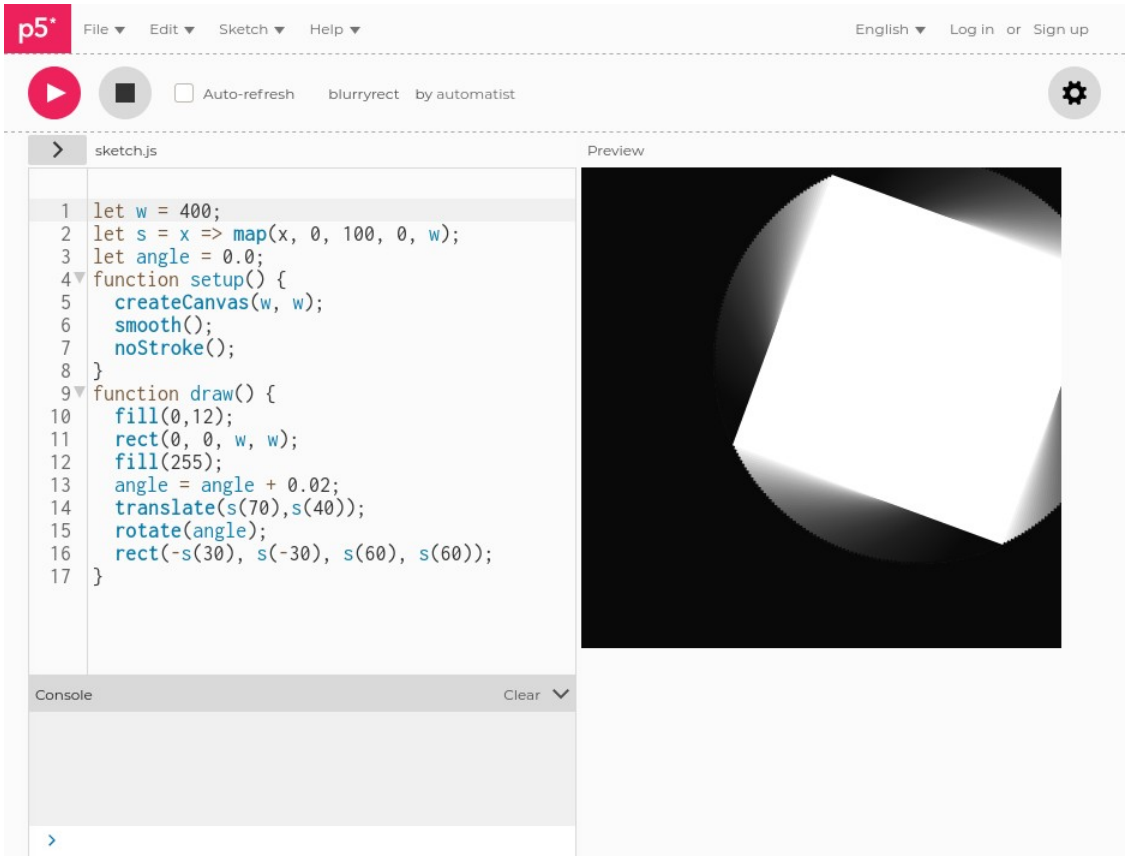


The book comprises over 700 pages and is organized by topics like: color, control, data, drawing, image, input, math, motion, structure, typography. The book also contains interviews with artists working with digital tools not limited to Processing. As indicated in the book's introduction, many examples are illustrated by "stark" small square images and concise code.

[3] Casey Reas, "Paragraphs on Software Art", Software Structures, retrieved October 28 2021, https://artport.whitney.org/commissions/softwarestructures/text.html#structure.



Processing commands typically address the canvas using x,y (Cartesian) coordinates. Shape commands, like circle, rectangle, and triangle allow for the drawing of graphical forms, while other functions control parameters like the color of fill or the width of the stroke (or outline) of each shape.



Input functions allow access to the mouse and keyboard to produce dynamic graphics that respond to the user. Processing sketches consists of (at least) two functions: *setup* which is invoked once and *draw* which is

invoked continuously; the default frequency being the refresh rate of the computer's display (typically 60 times per second). By using variables, and changing their values, graphics can be made dynamic. In addition, graphics are rendered using a technique known as "anti-aliasing" to appear "pixelated". This default behavior can be modified by using the noSmooth command.

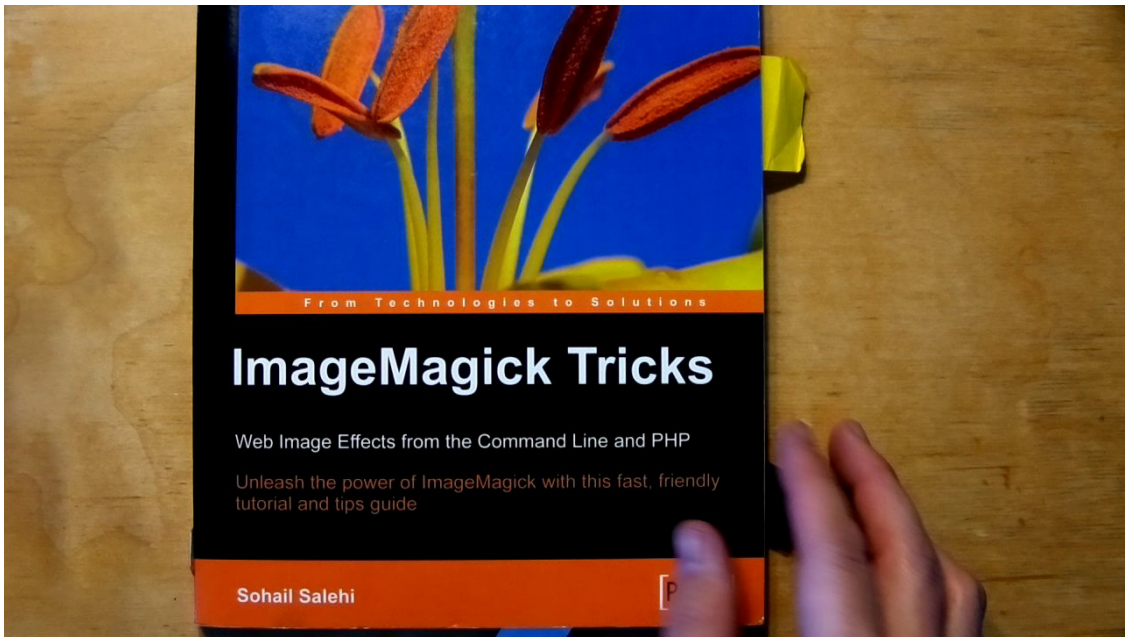
Earlier in the day I had bought another technical book "ImageMagick Tricks: Web Image Effects from the Command line and PHP" by Sohail Salehi[4]. While waiting for the presentation to begin, I crossed paths with Casey Reas at the back of the room. He was curious about the book I had with me and looked briefly at it. He had never heard of ImageMagick.

> ImageMagick started with a request from my DuPont supervisor, Dr. David Pensak, to display computer-generated images on a monitor only capable of showing 256 unique colors simultaneously. In 1987, monitors that could display 24-bit true color images were rare and quite expensive. There were a plethora of chemists and biologists at DuPont, but very were few computer scientists to confer with. Instead, I turned to Usenet for help, and posted a request for an algorithm to reduce 24-bit images to 256 colors. Paul Raveling of the USC Information Sciences Institute responded, with not only a solution, but one that was already in source code and available from USC's FTP site. Over the course of the next few years, I had frequent opportunities to get help with other vexing computer science problems I encountered in the course of doing my job at DuPont. Eventually I felt compelled to give thanks for the help I received from the knowledgeable folks on Usenet. I decided to freely release the image processing tools I developed to the world so that others could benefit from my efforts.[5]

---

[4] Sohail Salehi, *ImageMagick Tricks* (Birmingham: Packt Publishing, 2006)

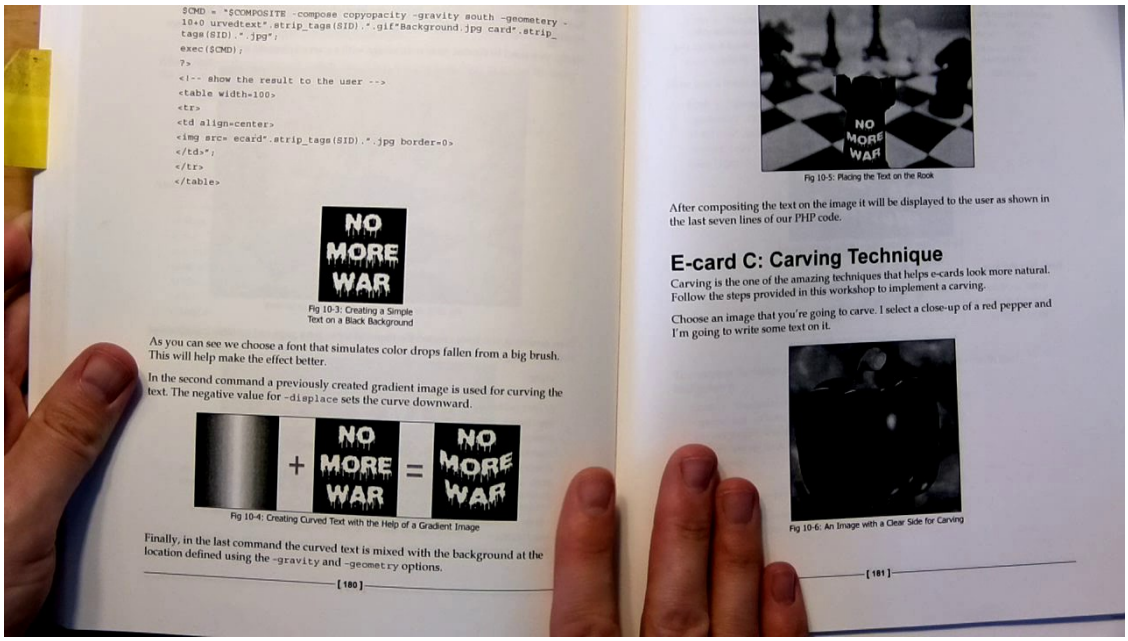
[5] John Cristy, "History", ImageMagick website. Retrieved from the Internet archive Oct 28, 2021, https://web.archive.org/web/20161029234747/http://www.imagemagick.org/script/history.php.



ImageMagick, first released in 1990, is a popular free software tool that's often referred to as a "swiss army knife" due to its ability to convert between hundreds of different image formats, and for the many built-in features to filter, manipulate and generate images. Thanks to the software being "not chemically or biologically based", Cristy was able to release the software under a Free license (echoing the way the UNIX operating system became free software due to its marginality to the interests of Bell Labs). The software is full of particularities. For instance, there are a number of built in images including a wizard (the mascot of the software) seated at a drawing table contemplating an image of the Mona Lisa.

ImageMagick is a command-line tool, designed to be used via textual commands. The typical usage of ImageMagick is to take one image as input, applying one or more transformations to it, and output a new image. In this way the tool can be used repeatedly in a so-called "pipelines", or otherwise composed together in structures called (BASH) scripts. In these scripts ImageMagick commands can be mixed with other commands from any software installed on the user's computer that also provides a command-line interface.

Salehi's book directly reflects the structure of ImageMagick, with chapter organized around various incorporated "tools": convert, mogrify, composite, montage, identify, display, conjure. The examples are practical, creating logos or adding captions or a border to an image. One example renders the word "Candy" with colorful stripes. Another series of examples duplicates and inverts the image and text of classical Persian poet "Hafez" to create a kind of playing card. Another example uses ImageMagick in conjunction with PHP and HTML to produce an online "e-card maker", a sequence of commands is demonstrated to render the text "No More War" (in a dripping paint font), deform it, and project it onto the side of a chess piece.



In another extended example, a British flag is constructed in steps. Rather than approaching the project as drawing geometric forms on a canvas, Salehi uses the diversity of ImageMagick's manipulations, performing a series of commands whose textual names invoke a sense of physical construction: blocks of color are skewed, sheared, cropped, flipped, flopped, and finally spliced (with "gravity" set to center). The approach creates a number of intermediate images, thus creating the digital equivalents of "cuttings" or leftover materials in the process. Below, the (intermediate) results from the same example slightly modified to take an image as input:

Source image:
https://en.wikipedia.org/wiki/Boris_Johnson#/media/File:Boris_Johnson_o

fficial_portrait_(cropped).jpg, Ben Shread / Cabinet Office, UK Open Government Licence v3.0 (OGL v.3)

```
convert -size 300x200 xc:'#002377' -fill white -draw 'skewX 128
image over 0,0 0,0 in/boris.jpg' background1.png
convert -size 300x40 xc:none -background transparent -fill '#ce201a'
-draw ' rectangle 0,0, 300,20' -shear 32 band.png
convert background1.png -draw ' image over -25,0 0,0 band.png'
background1.png
convert background1.png -draw ' image over -25,0 0,0 band.png'
background1.png
convert background1.png -crop 150x94+0+0 area1.png
convert area1.png -flip -flop area3.png
convert background1.png -draw ' image over -25,-21 0,0 band.png'
background2.png
convert background2.png -flop background2.png
convert background2.png -crop 150x94+150+0 area2.png
convert area2.png -flip -flop area4.png
convert -size 300x188 xc:none -draw ' image over 0,0 0,0 area1.png'
-draw ' image over 150,0 0,0 area2.png' -draw ' image over 0,94 0,0
area4.png' -draw ' image over 150,94 0,0 area3.png' mixed.png
convert mixed.png -background white -gravity center -splice 20x20 -
background '#ce201a' -splice 40x40 flag.png
```

constructivism

In the 1920s Russian avant-gardist El Lissitsky would move to Berlin and produce work that was highly influential to the then nascent Bauhaus. In 1923, Lissitsky illustrated a publication of poems by friend Vladimir Majakovskij. In it he created graphical forms by mixing typographic elements with geometric forms created by (mis-) using spacing elements (the “blind” elements typically used to create (negative/unprinted) space between lines of type, as positives producing geometric forms. This style, sometimes called constructivism, was part of an effort to make a radical break from traditional styles of typographic layout and illustration using the means then available for print. The book is notable for its “interactivity” via iconic tabbed pages.

For the Voice, El Lissitzky designer; Image from the archive of Guttorm Guttormsgaard. Used with permission.
https://arkiv.guttormsgaardsarkiv.no/node/19/item/39

In the 1970s Seymour Papert co-developed a pedagogy for teaching children mathematics and programming based on the LOGO programming language. Part of the system was a (virtual) robotic turtle that could be programmed to draw figures. The system, known as Turtle graphics, had commands like: forward, turn left, turn right, pen up, pen down, that directly addressed the "turtle" to draw shapes while moving.

> The process reminds one of tinkering: learning consists of building up a set of materials and tools that one can handle and manipulate. Perhaps most central of all, it is a process of working with what you've got. .... This is a science of the concrete, where the relationships between natural objects in all their combinations and recombinations provide a conceptual vocabulary for building scientific theories. Here I am suggesting that in the most fundamental sense, we, as learners, are all *bricoleurs*.[6]

[6] Seymour Papert, *Mindstorms* (Cambridge: MIT Press, 1980), 173.

Made with http://logointerpreter.com/turtle-editor.php

Papert's approach is part of a pedagogic project (also) called constructivism. In a key example, Papert describes how students can be taught about circles by imagining (or better yet themselves enacting) the turtle repeatedly performing the instruction sequence "go forward a little, turn a little". He contrasts this with the formal equation of a circle ($x^2 + y^2 = r^2$) typically taught in an elementary school geometry class.

In Belgium, where I currently live "bricolage" is the French language equivalent to "DIY" and is often used in a derogatory sense to indicate that something is made in an amateurish way. Papert embraces the term, a central point being how informal/intuitive methods not only appeal to "common sense" but also engage more profoundly with the materiality of its subject than would a formal approach. In the case of the circle, the "turtle" method is not only a way for the student to imagine the problem physically, it also relates to methods of differential calculus, something the algebraic formulation misses completely.

> For most there is a total dissociation between these live activities and the dead school math. We have stressed the fact that using the Turtle as metaphorical carrier for the idea of angle connects it firmly to the body geometry. We have called this *body syntonicity*. Here we see a *cultural syntonicity*: The Turtle connects the idea of angle to navigation, activity firmly and positively rooted in the extraschool culture of many children.[7]

tearing at the seams

Initially I came to the writing of this essay as part of a proposed dialog with the artist Winnie Soon. Unfortunately because of time constraints it wasn't able to happen. Together with Geoff Cox, Soon has recently published "Aesthetic Programming: A Handbook of Software Studies"[8]. The book itself, designed by Open Source Publishing, makes promiscuous

[7] Seymour Paper, *Mindstorms*, 68.

[8] Winnie Soon and Geoff Cox, *Aesthetic Programming: A Handbook of Software Studies* (Open Humanities Press, 2020)

use of different tools and techniques, such as the diagramming tool graphviz, HTML, Pelican (a CMS), and paged.js, and showcases an expressive use of typography (including a reinterpretion of a font originally drawn by Swiss typographer Adrian Frutiger).

Where Reas and Fry's Handbook is demure, Soon and Cox's is generous, voluptuous even, richly mixing programming exercises with diagrams, flowcharts, and illustrations. The discussions are explicit about the sociality of code and intersections with software art and software studies. The text is densely annotated with theoretical references and links to related critical artistic projects. Directly addressing the "dark sides" of technology such as the extractive and aggregative corporate practices of big data, this handbook explicitly positions coding as an act of activism. Most of the code examples are written using p5.js. As they describe in the first section:

> This book will use p5.js, a Javascript library which was created by artist Lauren McCarthy in 2014 for the purpose of what we call "aethetic programming." … Importantly, the core idea of p5.js is not just to deploy Processing as a web-based platform, but to address diversity and inclusivity explicitly, and take these issues seriously in software development and communication"[9]

In 2016, the Whitney published a "restored" version of Software Structures.[10] As technologies like Java and Flash had then for reasons both technical and commercial fallen out of popular use on the web, the new version featured many of the processing sketches adapted by Reas to use p5.js.

Despite the project's earlier stated interest in exploring diverse "materialities", it's telling that rather than considering the older processing implementations as a different material and presenting screenshots of them as was done for the Flash and C++ examples, the "restoration" maintains the illusion of a certain "permanence" to the processing sketches, making them thus appear closer to those imagined "software structures" than to "retrograde" technologies like Java or an out-dated browser version. In addition this "adaptation" hides the quite significant work that has gone into (1) the development and subsequent implementation in different browsers of the newly standardized canvas element[11], and (2) McCarthy's work creating the p5.js library to bridge from the legacy processing code to this new standard.[12]

---

[9] Winnie Soon and Geoff Cox, 31.

[10] https://whitney.org/exhibitions/software-structures

[11] https://html.spec.whatwg.org/multipage/canvas.html#the-canvas-element

[12] Despite the seeming similarity of names, Java and Javascript are two completely independent and quite different programming languages. Adapting software from one to the other is thus not trivial.

Alfred North Whitehead, writing on the sciences, established an influential idea of a “fallacy of misplaced concreteness”. The idea is that making abstractions, such as what happens when a particular phenomenon is named, is a simplification that works by suppressing “what appear to be irrelevant details”.[13] In Media Ecologies, Matthew Fuller extends this thinking to consider technical standards as “a material instantiation”[14] of Whitehead’s misplaced concreteness, and considers how technical devices through a process of *objectification* “expect in advance the results that they obtain”.[15] Fuller cites the example of Laurence Lessig, who makes an argument for regulation of the Internet based on the assertion of standard objects (such as networking protocols) being layered and reconfigurable.

> [Lessig’s argument] allows us to recognize another characteristic of the standard object: that, while it may be simultaneously embedded within multiple compositions wherein it may be involved in many, separate, disjunctive, contiguous, or contradictory processes, it does provide a threshold, either side of which is differentiated enough for significant political, technical, aesthetic, and social conjunctions or conflicts to occur.[16]

Susan Leigh Star takes Whitehead’s “misplaced concretism” and proposes a feminist methodology specific to information technology.[17] Her essay develops the idea of “standards” as one type of “Boundary object”, which she describes as:

> […] those scientific objects which both inhabit several communities of practice and satisfy the information requirements of each of them. Boundary objects are thus objects which are both plastic enough to adapt to local need and common identity across sites.[18]

Star cites Donna Haraway, who wonders in *A Manifesto for Cyborgs*:

> How do I then act the bricoleur that we’ve all learned to be in various ways, without being a colonizer…. How do you keep

---

[13] Alfred North Whitehead, *Science and the modern world* (New York: Free Press, 1967), retrieved from the Internet archive Oct 28, 2021 https://archive.org/details/sciencemodernwor00alfr/page/52/mode/2up

[14] Matthew Fuller, *Media Ecologies* (Cambridge: MIT Press, 2005), 127.

[15] Matthew Fuller, 104.

[16] Matthew Fuller, 129.

[17] Susan Leigh Star, “Misplaced Concretism and Concrete Situations: Feminism, Method, and Information Technology” (1994), *Boundary Objects and Beyond* (Cambridge: MIT Press, 2015)

[18] Susan Leigh Star, 157.

> foregrounded the ironic and iffy things you're doing and still do them seriously [...]

Star draws on a tradition of diverse feminist thinking through the "articulation of multiplicity, contradiction, and partiality, while standing in a politically situated, moral collective" to synthesize and propose what she calls the important attributes of a feminist method:

1. experiential and collective basis;
2. processual nature;
3. honoring contradiction and partialness;
4. situated historicity with great attention to detail and specificity; and
5. the simultaneous application of all of these points.[19]

For me tools like ImageMagick embody *collectivity* from it's origins as a way to "give back" to a community sharing code over usenet, through to its continued development by multiple authors and relation to the larger free software community as an invaluable toolbox for extremely diverse pratices. I find the *experiential* in the highly flexible commandline interface, itself also an example of honoring *contradiction* and *partialness*, with often more than one way to express the same transformation. The *processual* is implicit in its construction as a tool of transformation, encouraging an exploratory iterative approach to composing transformations to arrive at a desired outcome, often leading to missteps and errors that can be happy accidents and lead one to reconsider one's goals. Finally, in its extreme support of hundreds of different formats, ImageMagick use often leads to the discovery and exploration of diverse image formats, each with related practices, and contexts.

In contrast, behind a seemingly "neutral" aesthetic, Processing I think embodies a very particular set of values and assumptions. The "visual minimalism" claimed in the Processing handbook, belies the project's expansive claims on representing a pedagogic approach to a broad intersection of programming and the visual arts. The project's "neutral" aesthetics while dimly echoing a once-radical Bauhaus aesthetic, ignores the larger pedagogic program of the historical Bauhaus' experimentation with the materials of their (contemporary) technical production. The framework valorizes smoothness and fluidity, that leads one to prioritize interactivity as that which happens on the surface of a sketch, rather than say in the network, or among collaborators. As a pedagogic project, Processing has historically seemed uninterested in its own underlying materiality, preferring its students to explore the "world at large" by adding additional layers of technology in the form of sensors, rather than considering all the ways the technologies they use are *already engaged* and impacting the world.

> A concatenation of operations of misplaced concreteness thus allow the gaps, overlaps, and voids in the interrelated capacities

[19] Susan Leigh Star, 148-149.

> of such systems to construct a more “accurate” account of its own operations.[20]

Vernacular rejects the “false neutrality” of the seamless universal design solution, embracing instead the tips and tricks of specific tools, in specific contexts. A vernacular composition tears open its seams proudly displaying the glitches and gaps as a badge of honor of a world view of truth as multi-threaded, incomplete, and sometimes uncomfortable, and is engaged with the world and not afraid to be seen as a monster.

[20] Matthew Fuller, 104.