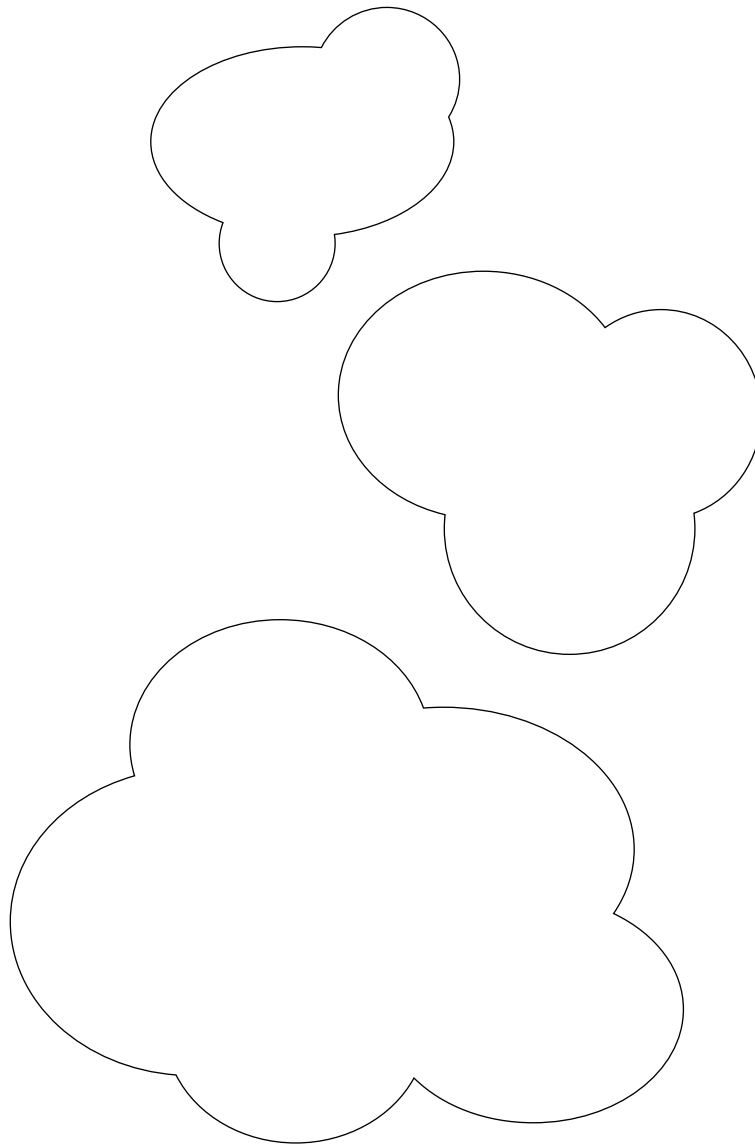


Birk Weiberg, «Against Programming: On the Development of Cultures of Coding in Art and Design», in: *Nummer*, no. 11, *Update Available: Transforming Education in Design, Film and Fine Arts*, eds. Orlando Budelacci, Jacqueline Holzer and Birk Weiberg, Luzern 2023, pp. 52–55.
doi:10.5281/zenodo.7418207

Entire issue: doi:10.5281/zenodo.7417851

CC BY-NC-ND 4.0



Against Programming

On the Development of Cultures of Coding in Art and Design

Birk Weiberg

...the predominant cultural technique of digitality for art and design seems to belong to another discipline, and thus requires strategies of appropriation.

The newly developed interdisciplinary curriculum of the Lucerne School of Art and Design is organised around seven competences, which have been identified as relevant for the future across all specialised Bachelor's programmes. Digitality is one of them and, as with any transdisciplinary competence, the question is how to adapt it to the specific context in which it is expected to become productive. The following thoughts sketch one way to do so but do not claim universality.

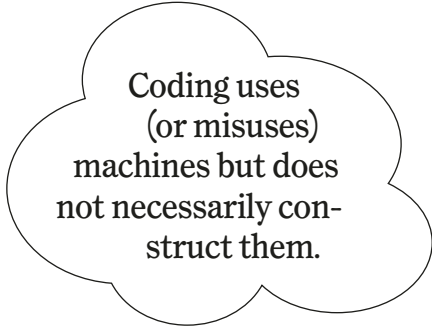
In academia, digitality, or «being digital», as Nicholas Negroponte audaciously called our condition in contemporary technoculture,¹ has first and foremost been the domain of computer science. This is also the case for the primary practice of digital authorship: programming. Of course, other forms of digital authorship (by means of non-textual interfaces) exist, but they would not do so without programming. From this perspective, the predominant cultural technique of digitality for art and design seems to belong to another discipline, and thus requires strategies of appropriation. One form of symbiotic relationship between the disciplines is that developers program smart tools which enable artists and designers to create equally smart things. But in recent years, computer science has lost its monopoly on teach-

...computer science has lost its monopoly on teaching people how to write software...

...an approach that today goes by the name creative coding.

ing people how to write software, so that more and more initiatives not only address additional groups but also develop independent approaches to writing code.² As Annette Vee observed in her book *Coding Literacy*, «programming is too useful to too many professions»³ to be left to a single discipline alone and its understanding of what programming actually is. This is all the more important as since the late 1950s computer science has successfully framed programming as a form of engineering, a practice that aims to solve a specific problem or task efficiently and reliably. This instrumental conception of algorithms makes perfect sense when thousands of programmers write millions of lines of code that make planes fly or cars drive autonomously. But it has little to do with what individual artists and designers might want to achieve when they waive much more intuitive user interfaces in favour of structured text.

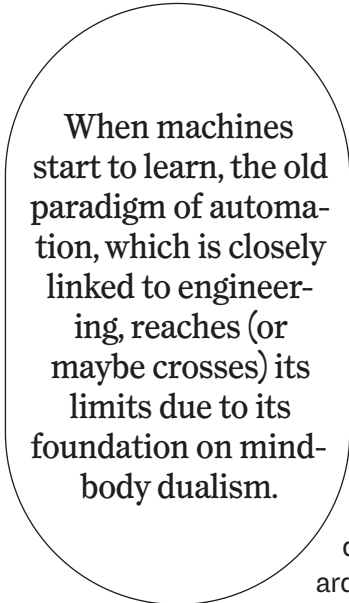
Parallel to the history of computers in science and engineering there have, of course, been parallel trajectories in art and design that often go back to the same period in the 1950s but lack the successful institutionalisation of computer science. Computer art, to give these parallel developments a single comprehensive name, has been much more fragile, emerging locally with changing focal points



Coding uses
(or misuses)
machines but does
not necessarily con-
struct them.

or cultures, which often sailed in the slipstream of computer science, e.g. by using its infrastructures for rendering graphics during night shifts.⁴ A broader academic institutionalisation of computer art started some twenty years ago not as art but as design, an approach that today goes by the name creative coding. This development began with John Maeda's software (and book) *Design by Numbers* and his eponymous book *Creative Code*, where he argues that designers should embrace algorithms as a new tool.⁵ And the development has probably come to a conclusion with the recent *Code as Creative Medium*, edited by Golan Levi and Tega Brain, in which a dozen successful educators in the field of creative coding reflect on their teaching practices.⁶

Though the frequent use of the noun 'code' in this context suggests a specific approach or autonomy, the verbs 'coding' and 'programming' are usually used synonymously. Likewise, I am not aware of any discussion on the questions of whether it makes sense to distinguish practices in computer science from those in art and design more clearly by giving them different names.⁷ But this is what I want to suggest to better understand how practices of digital authorship in art and design differ from those in computer science and how they can be developed further specifically in educational contexts. From that point of view, the objective of



When machines
start to learn, the old
paradigm of automa-
tion, which is closely
linked to engineer-
ing, reaches (or
maybe crosses) its
limits due to its
foundation on mind-
body dualism.

programming is the production of a coherent, functional piece of software, a program, a little (or not so little) machine that is hopefully neatly constructed to do a specific thing. Coding, on the other hand, can be seen as a much more profane denomination of the practice of writing code, a piece of machine-readable text that usually does *something* but does not need to fulfil the same standards as a program. Coding uses (or misuses) machines but does not necessarily construct them. And lest this sounds as if coding is simply an underdeveloped form of programming, the humbleness that comes with my reading of the term also has the advantage of reminding us that the act of formalising language is at the core of human-machine relations and a chance to reflect upon them.⁸

The crux of successful practices like creative coding is that the development of tools and frameworks can lead to restrained applications and aesthetics (e.g., the typical complex hairlines of early processing). When increased freedom of use is put forward as an argument for working with code rather than with GUI tools, which are based on the separation of complicated, functional code from creative usage, this distinction is potentially



↑ Fig. 1 Joana Chicau and Renick Bell, *Círculo e Meio / Circle & Half*, live coding performance, 2018, Spektrum Berlin, photographed by Henrique Palazzo

Perhaps surprisingly, the research subject of code studies is usually not pre-existing code, which is often inaccessible and simply too extensive, but pieces of one's own code that can interact with that of others.

blurred as creative coding tools become ever easier to handle. The better and more professional such tools become, the more one should be open to alternative approaches that cultivate transdisciplinarity as a site of critique. One reference comes from Philip E. Agre, who was trained and worked in AI in the 1970s and 1980s before turning to humanities to better understand the blind spots of his disciplines: «A critical technical practice will, at least for the foreseeable future, require a split identity – one foot planted in the craft work of design and the other foot planted in the reflexive work of critique.»⁹ While Agre uses the term design here to mean the purposeful approach he found in AI programming, we might take it as a placeholder for any distinctly applied practice that may have to limit its critical potential to remain functional. The limits of intradisciplinary critique are something Agre encountered, and AI is a good example of the need for a space to think about the future of co-creation between humans and machines. When machines start to learn, the old paradigm of automation, which is closely linked to engineering, reaches (or maybe crosses) its limits due to its foundation on mind-body dualism.

A field in the humanities that developed in parallel with creative coding in design is software studies, also called critical code studies.¹⁰ What started as

a critical look, first at applications and then at interfaces of New Media, has led at least some scholars to code itself. Perhaps surprisingly, the research subject of code studies is usually not pre-existing code, which is often inaccessible and simply too extensive, but pieces of one's own code that can interact with that of others. Coding here occupies an interesting ambiguous space between humanities' genuine medium, text, and something that is directly actionable and potentially expressive and creative. The blurring of the distinction between traditional text and computer code is further supported by references to J. L. Austin's speech act theory, which is an ascription for the first but description of the latter. Thus, the insight that code is an actionable language that connects humans and machines can be seen as ground zero for any critical approaches to code.¹¹

These inquiries by humanities scholars have paved the way for new hybrid forms of teaching coding without losing a critical distance to its applications. An excellent example here is *Aesthetic Programming* by Winnie Soon and Geoff Cox, which combines an introduction to the popular coding framework p5.js with critical theories.¹² Similar to more conventional introductions to

The ontological shifts that come with this practice, which does not categorically distinguish between material, tool, notation and art piece, are characteristic for more than functional usages of code.

- 1 Nicholas Negroponte, *Being Digital*, New York 1995.
- 2 One exemplary grass-roots initiative, which develops workshops for various disciplines to empower them when it comes to dealing with code and data, is The Carpentries, <https://carpentries.org>. Other initiatives address specific, marginalised groups that are under-represented in computer science without considering specific disciplines at all.
- 3 Annette Vee, *Coding Literacy. How Computer Programming Is Changing Writing*, Cambridge, MA 2017, p. 12.
- 4 Cf. Grant D. Taylor, *When the Machine Made Art. The Troubled History of Computer Art*, New York 2014.
- 5 John Maeda, *Creative Code*, New York 2004.
- 6 Golan Levin and Tega Brain, *Code as Creative Medium. A Handbook for Computational Art and Design*, Cambridge, MA 2021.
- 7 The composite «creative coding» aims to do this, of course, but misses the opportunity to position coding as a counter-practice to programming for the sake of a simple alliteration. Likewise, the attribution of creativity to the usage of code in art and design is taken as easy prey and without much contemplation. For an in-depth analysis of the concept of creativity, see the contribution by Orlando Budelacci in this issue, pp. 56–59, doi:10.5281/zenodo.7418222.
- 8 One might also think here of code in relation to semiotics – a connection that is beyond the scope of this article.
- 9 Philip E. Agre, «Toward a Critical Technical Practice. Lessons Learned in Trying to Reform AI», in: *Social Science, Technical Systems, and Cooperative Work. Beyond the Great Divide*, eds. Geoffrey C. Bowker et al., New York 1997, pp. 131–157, here p. 155.
- 10 *Software Studies. A Lexicon*, ed. Matthew Fuller, Cambridge, MA 2008; Mark C. Marino, *Critical Code Studies*, Cambridge, MA 2020.
- 11 Cf. Inke Arns, «Read_me, run_me, execute_me. Code as Executable Text: Software Art and Its Focus on Program Code as Performative Text», in: *Media Art Net* (2004), http://www.medienkunstnetz.de/themes/generative-tools/read_me/scroll/ (retrieved 1 Sept 2022).
- 12 Winnie Soon and Geoff Cox, *Aesthetic Programming. A Handbook of Software Studies*, London 2020, <https://www.aesthetic-programming.net> (retrieved 1 Sept 2022).
- 13 *Ibid.*, 167.
- 14 Winnie Soon, «Vocable Code», in: *MAI* (10 Nov 2018), <https://maifeminism.com/vocable-code/> (retrieved 1 Sept 2022).
- 15 <https://www.geometries.xyz>; Joana Chicau and Renick Bell, «Choreographies of the Circle & Other Geometries», in: *Critical Coding Cookbook. Intersectional Feminist Approaches to Teaching and Learning* (2022), <https://criticalcode.recipes/contributions/choreographies-of-the-circle-other-geometries> (retrieved 1 Sept 2022).

programming, the individual chapters dive into specific features of JavaScript and the p5.js framework, but do so in connection with theoretical concepts and art pieces. So, the chapter «Vocable Code», named after an installation/performance by the authors, does explain how to load data from a JSON file, how conditional structures work and how to animate text in a browser window. But it also demonstrates how «code mirrors the instability inherent in human language in terms of how it expresses itself, and is interpreted.»¹³ With its practice-based approach, the book here makes intelligible what Soon elsewhere described as «constrained writing»,¹⁴ i.e., our sensation of writing code according to the rules of the machines but which echoes societal functions of language.

A critical coding practice is also central to the work of Joana Chicau and Renick Bell in their artistic research project «Choreographies of the Circle & Other Geometries», where they explore alternative conceptions of the web browser space by means of live coding. The project makes the framework developed by the artists available as an instrument for others and provides instructions on how to use it in the form of a recipe.¹⁵ The ontological shifts that come with this practice, which does not categorically distinguish between material, tool, notation and art piece, are characteristic for more than functional usages of code.

These examples are far from being normative, but they stand for a diversification of coding and programming practices that must be seen as vital for the formation of the digital technoculture we live in. With its new curriculum, the Lucerne School of Art and Design is attempting to develop trans-disciplinary modules and transfer a long-standing tradition of combining theory and praxis to digital environments. To position *coding* in art and design

schools against *programming* as it is taught in computer science does not put one above the other, but rather argues for an independent and self-assured claim on digital technoculture by art education, one that explores the situatedness and contingencies of technologies, and sees them as a means not only to employability but also to critical inquiry and public participation.

To position *coding* in art and design schools against *programming* as it is taught in computer science does not put one above the other...