DHQ: Digital Humanities Quarterly
2023
Volume 17 Number 2

Introduction: Situating Critical Code Studies in the Digital Humanities

Mark C. Marino <markcmarino_at_gmail_dot_com>, University of Southern California https://orcid.org/0000-0003-2034-3433

Jeremy Douglass <jeremydouglass_at_gmail_dot_com>, University of California, Santa Barbara https://orcid.org/0000-0001-7798-8801

Abstract

In this foreword from the editors we present a brief introduction to the field of Critical Code Studies, a reflection on its genesis and evolution, and a summary of the many and varied author contributions to Part 1 of this remarkable special collection.

Almost two decades ago, at the start of Critical Code Studies (CCS), we asked two provocative questions: “What does computer source code *mean*?” and “What do we discover when we read code with the interpretive tools of the humanities?” 1

At the 2006 MLA Convention in Philadelphia [PMLA 2006] where we launched the manifesto, some humanities scholars in the audience asked whether reading code was a conversation better suited for a technology conference. Conversely, some computer scientists reacted to our endeavor by asking why English majors were reading code at all. Incidentally, that MLA panel was organized by Rita Raley and one of the speakers was John Cayley, both of whom are included in this collection, so you might say this first volume is a bit of a reunion — appropriately so, since from the beginning CCS was a collaborative project, inviting and requiring a diverse group of thinkers and makers. 2

On roads paved by cultural studies, semiotic analysis, science and technology studies, and media archaeology, we set out on the journey to understand code. Along the way we launched critical explorations — sometimes alone, but more often in groups. Since 2010 we have convened seven biennial gatherings of the Critical Code Studies Working Group (“the major online think tank for critical code studies, a hub of dialogue and collaborative inquiry that generates major thrust in the reading of code.”) Those multi-week gatherings gave rise to multi-author books such as *10 PRINT CHR$(205.5+RND(1)); : GOTO 10* [Montfort et al. 2013], monographs such as *Critical Code Studies* [Marino 2020], and numerous conference presentations and journal articles, including early publications in *Digital Humanities Quarterly* 2013 [Montfort et al. 2013] [Marino 2013] edited by Lisa Swanstrom and Jessica Pressman. 3

Following these initial publications, we are delighted to present this sequence of two Digital Humanities Quarterly special issues on Critical Code Studies as the latest scholarly work in the field. Their contributions extend and develop CCS through close readings of code and theoretical interventions, offering new methods of reading and interpretation while introducing new programming languages, expanding the scope of code studies even as they refine its methods and practices. 4

Critical code studies is the application of the hermeneutics of the humanities to the interpretation of the extra-functional significance of computer source code. “Extra” here does not mean “outside of” or “apart from” but instead it refers to a significance that is “growing out of” an understanding of the functioning of the code. While the initial manifesto spoke of treating code as a “text”, in later clarification (https://hcommons.org/deposits/item/hc:19537/), Mark has explained that “text” refers to a cultural object, rather than a collection of words and symbols. More significantly, if early definitions positioned the code object as the *ends*, over time, the code has proven instead to be an entry point, a *means* to open up 5 conversations about a wide variety of topics in techno-culture. This may be part of what drives the intersectionality of critical code studies with related subfields in cultural studies and technology. CCS is as much a field constituted by methods (code reading) as it is by particular objects of study (code), and as such it can provide new approaches into many areas of investigation; indeed, the revealed object of study is often not the code *per se* but instead “the border,” “the lunar lander program,” et cetera.

Still, reading code, even *without* interpreting its cultural significance, can be no easy task. Ask a professional programmer who inherits legacy code to maintain or, worse yet, to improve, and they will tell you about the dread of sorting out just-in-time code, minimally documented, written with hasty patches, full of compromises and workarounds. Even those who write their code in artistic projects can be shy about sharing their code out of embarrassment and self-consciousness. This shame is a product of the “encoded chauvinism” of programming culture, one that can be fostered on the internet as much as it is in classrooms [Marino 2020, 148]. 6

Trying to interpret code, in the humanities sense of interpretation, compounds the challenge of reading code, which may be why scholars so often eschew the attempt. In spite of the groundwork we have tried to lay in the form of methods and models, workshops and working groups, most essays and books about software objects either tend to present code writing in functional and utilitarian terms from computer science / software engineering perspectives or else tend merely to gesture toward the existence of code from humanities / social sciences / cultural studies perspectives — but rarely to analyze it. Surely, the task can be arduous. Even we editors have admitted to feeling a sense of doubt when faced with a new code object to interpret, wondering whether an examination will lead anywhere at all. In our classrooms, both of us have discovered the challenges of teaching students how to engage in the practice. The daunting challenge of interpreting code is part of what makes this collection such a milestone. 7

Origins & Extensions

Critical code studies grew out of a moment when many interconnected groups of scholars were bringing new forms of attention to digital and computational objects. The namings of multiple new groups, organizations, journals, conferences, and subfields in the period around ~1999-2009 were often both sudden emergences and simultaneously culminations: “electronic literature,” “game studies,” “software studies,” “platform studies,” “digital humanities” and more were all ascendent at this time. 8

For years, scholars such as Jay David Bolter, George Landow, Kathleen Fitzpatrick, Paul Saint-Amour, Janet Murray, and Brenda Laurel had examined the new media forms of literary hypertext and digital theatre. The Electronic Literature Organization was founded in 1999, organizing conferences and ELC digital literary anthologies around bringing sustained critical attention to digital literary objects. In Game Studies, Espen Aarseth’s *Cybertext: Perspectives on Ergodic Literature* (1997), Gonzalo Frasca’s “ludology” (1999), the new journal of *Game Studies* (2001), and the Digital Games Research Association (2003) brought a focus on the analysis of rules, procedures, and processes in (often digital) games. 9

After decades of “humanities computing” research, a new banner term “digital humanities” was popularized by the influential anthology *A Companion to Digital Humanities* (2004) edited Susan Schreibman, Ray Siemens, and John Unsworth, followed by the formation of The Alliance of Digital Humanities Organizations (ADHO) (2005) and thereafter its first issue of this journal, *Digital Humanities Quarterly* 1.1 (2007) under editors Julia Flanders, Wendell Piez, and Melissa Terras. In the United States, funding through the NEH Digital Humanities Initiative (2006) / Office of Digital Humanities (2008) further helped DH become the organizing term for what would eventually become the DH “big tent” — incorporating two factions that Kathleen Fitzpatrick would describe as “scholars who use digital technologies in studying traditional humanities objects and those who use the methods of the contemporary humanities in studying digital objects.” [Fitzpatrick 2011] 10

Within this broader scene, an additional crucial context for the emergence of critical code studies were the respective calls of both Lev Manovich (*The Language of New Media*, 2001) and N. Katherine Hayles (*Writing Machines*, 2002) for scholars to employ media-specific analysis to attend to unique features and material conditions, to develop new approaches more suited to these new and emerging forms. AS if in answer to those calls came a stampede of “studies,” 11 including critical code studies, software studies, and platform studies. Software studies brought interdisciplinary attention to software systems and their social and cultural effects with The Software Studies Workshop (2006), Software Studies Initiative (2007), and the Softwhere Studies Workshop (2008), along with *Software Studies: a Lexicon* edited by Matthew Fuller [Fuller 2008]. In 2006, Ian Bogost and Nick Montfort also announced the Platform Studies book series they would edit for the MIT Press [Bogost and Montfort 2006] alongside their forthcoming *Racing the Beam: The Atari Video Computer System*, and defined platform studies as investigating “the relationships between the hardware and software design of computing systems and the creative works produced on those systems.” Alongside these came the continued development of media forensics as practiced by Matthew Kirschenbaum (e.g., *Mechanisms: New Media and the Forensic Imagination*, [Kirschenbaum 2012]), and of media archaeology, whether by Kittler and his students or by Lori Emerson.

The development in critical reading practices has been attended by an expansion in the way programming itself is presented in works such as Nick Montfort’s introduction to *Exploratory Programming for the Arts and Humanities* [Montfort 2016] and in *Code as Creative Medium* [Levin and Brain 2021] by Tega Brain and Golan Levin. Taking up questions of cultural meaning, Geoff Cox and Winnie Soon released their *Aesthetic Programming* [Soon and Cox 2021] on Gitlab and invited readers to fork it, an invitation Mark took up with Sarah Ciston when they added a chapter of their own. We would also be remiss to omit Daniel Shiffman’s *The Nature of Code* [Shiffman 2012], which has become a staple for novice programmers. 12

The number of books examining the culture of programs has also increased, beginning with David Berry’s *The Philosophy of Software* [Berry 2011]. Furthermore, Annette Vee’s *Coding Literacy: How Computer Programming is Changing Writing* offers a cogent argument about the nature of programming knowledge [Vee 2017]. *Speaking Code: Coding as Aesthetic and Political Expression* by Alex McLean and Geoff Cox offers a kind of duet with Cox reading McLean’s code and vice versa [Cox and McLean 2013]. Recent books have taken up subcultures of programming, such as *Live Coding: a user's manual* [Blackwell 2022]. Algorithms have received their own attention, as in Jeffrey M. Binder’s *Language and the Rise of the Algorithm* [Binder 2022]. More recently, James Brown’s *Ethical Programs: Hospitality and the Rhetorics of Software* [Brown 2015] and Kevin Brock’s *Rhetorical Code Studies* [Brock 2019] offered even more methods of interpreting code. Interpreting the code of a digital object has contributed to as a larger *Reading Project* [Pressman et al. 2015], our collaboration with Jessica Pressman, which includes an analysis of the source code and source files of William Poundstone’s *Project for Tachistoscope {Bottomless Pit}* [Poundstone 2005]. 13

As more critical attention takes up software, scholars have turned their attention to racial bias and software, following the work of pioneers such as Alondra Nelson. More recently Safiya Noble in *Algorithms of Oppression* [Noble 2018] and Joy Buolamwini with her Algorithmic Justice League have worked to bring the topic of racial discrimination in software to the forefront. Ruha Benjamin has, likewise, traced out bias in code with her extension of “race critical code studies [Benjamin 2019].” Historians are documenting the role of race in programming spaces in books such as Clyde W. Ford's *Think Black: A Memoir* [Ford 2019] and with respect to gender as in Mar Hicks’ *Programmed Inequality* [Hicks 2017]. micha cárdenas’ *Poetic Operations: Trans of Color Art in Digital Media* [cárdenas 2022] has examined algorithms with respect to trans of color. The social side of code and computers continues to be the subject of scholars as in the recent collection *Your Computer is on Fire* [Mullaney et al. 2021] and in the groundbreaking work of Wendy Chun. 14

At the same time, code poets have been publishing their code. Some examples include Nick Montfort’s *#!* (pronounced she-bang) and the *The Truelist*, JR Carpenter’s *Generation(s)*, Lillian-Yvonne Bertram’s *Travesty Generator*, or Milton Laufer’s *A Noise Such as a Man Might Make*. Code poets have played with code from the creole of *mezangelle* by Mez Breeze to the code poems of Margaret Rhee, collected in *Love, Robot*. And for something completely different, Angus Croll’s *If Hemingway Wrote Javascript* offers humorous though insightful renditions of the writing styles of famous natural language creative writers (from Jane Austen to Virginia Woolf) adapted into programming languages [Croll 2014]. 15

This may be an odd claim to make, but not everyone who writes about code is a (digital) humanities scholar. A collection called *Beautiful Code: : Leading Programmers Explain How They Think* asks programmers to discuss their favorite lines of code [Oram and Wilson 2007]. In *Once Upon an Algorithm* [Erwig 2017], Martin Erwig offers lessons in programming 16 in the language of storytelling. Books about the cultural meaning of code surely go back to Don Knuth's *Literate Programming* [Knuth 1992] and even *The Structure and Interpretation of Computer Programs* [Knuth 1984]. Though these authors might not write in the language of critical theory or hermeneutics, the vast insight from their works, drawn out of lifetimes of work in the field of programming and computer science, continue to direct and instruct critical code studies.

In addition to these special issues of DHQ, scholars can look to a number of collections of publications on *electronic book review*, which has been a major outlet for critical code studies since its inception. There, readers can find write-ups and overviews from some of the CCS Working Groups. Among other postings are the overview of the original working groups, the weekly discussions from 2010 (1, 2, 3, 4, 5). In 2020, *ebr* published an overview, and intros to weekly discussions re-introducing CCS, Indigenous Programming, and Feminist AI. Of course, *electronic book review* was also the publication of the original manifesto. 17

These special issues of DHQ are the first fruits of our work to foster the development of critical code studies through conference presentations and biennial working groups over these past two decades. Over the course of the seven working groups so far, participants have conducted fruitful investigations into several bodies of code, from Joseph Weizenbaum's ELIZA to William Crowther's ADVENTURE. More recent working groups have looked at the code for the Apollo Moon Lander. We have taken up issues of race and gender, creative coding and the ethics of code. We have explored platforms for annotating code, from repositories to annotation tools on Google Docs and ANVC Scalar. Scalar itself has served in readings of the Transborder Immigrant Tool and FISHNETSTOCKINGS. And we have speculated about alternatives to dominant models of code, as in our discussions of feminist AI and a feminist programming language. Every working group takes up new themes, as the international group of scholars, artists, and programmers, and every combination therein post code critique threads to see what could be said about a menagerie of objects made out of code. 18

We continue to develop new venues for critical code studies practices as well. In the past three years, we have launched an Anti-Racist Critical Code Studies Reading Group, inspired by the work of Noble, Benjamin, and Buolamwini, as well as the Knit&Perl group co-organized with Anne Sullivan and Anastasia Salter, a sewing circle of scholars which looks at the intersection of coding and stitchcraft or the fibre arts [Salter and Sullivan 2022]. We have also launched a subgroup of the Humanities and Critical Code Studies (HaCCS) Lab that advocates for the public release of code (such as predictive policing code) that affects, governs, and shapes the lives of citizens. This new group is called the Sunshine Source Force, drawing its name from the movement for legal transparency, known as the Sunshine Laws. Surely, new initiatives are on the horizon. 19

Code is never one thing and is as dynamic as any semiotic form, constantly in flux. As we write this, machine-learning code generators, such as Github's Copilot, are emerging as a major part of programming assistance even as LLMs also assist essay writing. We suspect, and our recent experiments have confirmed that suspicion, that they will also offer assistance in the interpretation of code. Like the seas of natural language, the ecology of computer source code is constantly shifting and so there is always a need for more reading practices and, of course, a wider and more diverse set of scholarly and creative minds embarking on this endeavor. 20

In this issue

Before you is the first special collection of critical code studies readings. This landmark publication, offered in the first of two issues, presents a variety of interpretations and theoretical reflections that apply and extend the methods of critical code studies and also offer a resource for scholars looking for models of what these readings can accomplish. In addition to demonstrating established methods and best practices, scholars in this issue offer new and nuanced approaches to a wide range of code objects as well as developing new approaches, expanding the realm of what can be analyzed through critical code studies — accompanied by in-depth readings performed by top scholars in the field. This first issue presents three groupings of articles: 1) exemplary close readings of code, 2) new directions in critical code studies (such as code legibility and Critical AI), and 3) new work in programming languages and linguistics (including esoteric programming languages and indigenous programming languages). 21

First, this issue contains exemplary close readings of code. In “Reverse Engineering the Gendered Design of Amazon’s Alexa: Methods in Testing Closed-Source Code in Grey and Black Box Systems,” Lai-Tze Fan examines the gendered design of Amazon Alexa’s voice-driven capabilities, or, “skills,” (despite closed source impediments) in order to better understand how Alexa, as an AI assistant, mirrors traditionally feminized labour and sociocultural expectations. In “BASIC FTBALL for Everyone and Computer Programming for All,” Annette Vee puts the 1965 BASIC program FTBALL in the historical, cultural, gendered context of “computer programming for all” while gesturing to the tension between a conception of “all” and FTBALL’s context in an elite, all-male college in the mid-1960s. In “Computational art Explorations of Linguistic Possibility Spaces: comparative translingual close readings of Daniel C. Howe’s Automatype and Radical of the Vertical Heart 忄,” John Cayley elaborates a comparative, transculturally implicated, code-critical close reading of two related works, by Daniel C. Howe, which explore linguistic possibility spaces in English and Chinese. This reading engages distinct and code-critically significant programming strategies, and underappreciated comparative linguistic concepts with implications for the theory of writing systems, of text, and of language as such. In “‘Any Means Necessary to Refuse Erasure by Algorithm:’ Lillian-Yvonne Bertram’s Travesty Generator,” Zach Whalen creates an expansive reading of Bertram’s “challenging, haunting, and important achievement of computational literature” while digging more broadly and deeply into how specific poems work to better appreciate the collection's contribution to the field of digital poetry. In “Poetry as Code as Interactive Fiction: Engaging Multiple Text-Based Literacies in *Scarlet Portrait Parlor*,” Jason Boyd examines how various text-based literacies (procedural, poetic, ludic) can, when used together, elucidate the meanings of an Inform7-programmed interactive fiction in the form of a sonnet. This examination suggests how critical code studies may engage in more nuanced discussions of natural language programming. 22

Second, this issue contains new directions in critical code studies, pursuing areas such as machine learning software and the limits of code, non-code, and “nonsense code”. In “How to Do Things with Deep Learning Code,” Minh Hua and Rita Raley consider the feasibility and critical potential of CCS as a method when the object of study is deep learning code. Calling for a “critical urgent” need for basic understanding of the composition and functioning of large language models, they extract a representational map of OpenAI’s GPT-2 which they then verify through case studies of two popular GPT-2 applications: the text adventure game, *AI Dungeon*, and the language art project, *This Word Does Not Exist*. In “Tracing ‘Toxicity’ Through Code: Towards a Method of Explainability and Interpretability in Software,” David M. Berry examines how we can use concepts of explainability and interpretability drawn from computer science in critical code studies. By examining a set of code artifacts, the paper looks at how following conceptual traces in concrete source code layers can contribute to understanding and explaining them. In “Nonsense Code: A Nonmaterial Performance”, Barry Rountree and William Condee analyze three case studies in which a literal reading of each program’s code is effectively nonsense; however, the programs generate meaning in performance. Using the framework of nonmaterial performance (NMP) and its four tenets (code “abstracts”, “performs”, “acts within a network”, and “is vibrant”), they consider the 1950s UNIVAC 1 “Happy Birthday,” the Firestarter processor stress test, and the Platypus family of side-channel attacks to decenter text from its privileged position and to recenter code as a performance. 23

Finally, we offer reflections on code, language, and linguistics, particularly esoteric and indigenous programming languages. In “ᐊᒐᐦᑭᐯᐦᐃᑲᓇ ᒫᒥᑐᓀᔨᐦᐃᒋᑲᓂᐦᑳᓂᕽ | acahkipehikana mâmitoneyihicikanihkânihk | Programming with Cree# and Ancestral Code: Nehiyawewin Spirit Markings in an Artificial Brain,” Jon Corbett discusses his project “Ancestral Code,” which consists of an integrated development environment (IDE) and the Nehiyaw (Plains Cree) based programming languages called Cree# (pronounced: Cree-Sharp) and ᐋᒋᒧ (âcimow). These languages developed in response to western perspectives on human-computer relationships, which Corbett challenges and reframes in Nehiyaw/Indigenous contexts. In “The Less Humble Programmer,” Daniel Temkin explores the aesthetics of how esoteric programming languages (esolangs) break from the norms of language design by explicitly refusing practicality and clarity. Through examples that make code disordered (e.g., Malboge) or even impossible to write (e.g., Unnecessary), esolangs may challenge or reaffirm wider ideas in programming culture and in how computer science is taught: specifically the sometimes-contradictory aesthetics of humbleness and computational idealism. 24

Acknowledgments

We are grateful to our tireless authors, reviewers, and editors, as well as to *Digital Humanities Quarterly* for their

support of critical code studies through these special issues. We are excited to see what they inspire!

Works Cited

Benjamin 2019 Benjamin, Ruha. *Race After Technology: Abolitionist Tools for the New Jim Code*. 1st edition, Polity, 2019.

Berry 2011 Berry, David M. (David Michael). *The Philosophy of Software: Code and Mediation in The Digital Age*. Palgrave Macmillan, 2011.

Binder 2022 Binder, Jeffrey M. *Language and the Rise of the Algorithm*. First edition, University of Chicago Press, 2022.

Blackwell 2022 Blackwell, Alan F., et al. *Live Coding: A User’s Manual*. The MIT Press, 2022.

Bogost and Montfort 2006 Bogost, Ian and Nick Montfort. Platform Studies website. 2006. http://platformstudies.com/.

Brock 2019 Brock, Kevin. *Rhetorical Code Studies: Discovering Arguments in and around Code*. University of Michigan Press, 2019.

Brown 2015 Brown, James J., Jr. *Ethical Programs: Hospitality and the Rhetorics of Software*. U OF M Digt Cult Books, 2015.

Cox and McLean 2013 Cox, Geoff, and Alex McLean. *Speaking Code: Coding as Aesthetic and Political Expression*. The MIT Press, 2013.

Croll 2014 Croll, Angus. *If Hemingway Wrote JavaScript*. No Starch Press, 2014.

Erwig 2017 Erwig, Martin. *Once Upon an Algorithm: How Stories Explain Computing*. The MIT Press, 2017.

Fitzpatrick 2011 Fitzpatrick, Kathleen. “The Humanities, Done Digitally.” *The Atlantic*, May 8, 2011. https://archive.ph/FNyko.

Ford 2019 Ford, Clyde W. *Think Black: A Memoir*. Amistad, 2019.

Fuller 2008 Fuller, M. ed., 2008. *Software Studies: a lexicon*. The MIT Press.

Hicks 2017 Hicks, Marie. *Programmed Inequality: How Britain Discarded Women Technologists and Lost Its Edge in Computing*. 1st edition, The MIT Press, 2017.

Kirschenbaum 2012 Kirschenbaum, M.G., 2012. *Mechanisms: New Media and the Forensic Imagination*. The MIT Press.

Knuth 1984 Knuth, D.E., 1984. “Literate programming.” *The Computer Journal*, 27(2), pp.97-111.

Knuth 1992 Knuth, D.E. (1992) “Literate Programming.” Number 27 in CSLI Lecture Notes. *Center for the Study of Language and Information*, pp.349-358.

Levin and Brain 2021 Levin, Golan, and Tega Brain. *Code as Creative Medium: A Handbook for Computational Art and Design*. Annotated edition, The MIT Press, 2021.

Marino 2006 Marino, Mark C. “Critical Code Studies”. *Electronic Book Review*, vol. electropoetics, Winter 2006, http://www.electronicbookreview.com/thread/electropoetics/codology.

Marino 2013 Marino, Mark C. *Code as Ritualized Poetry: The Tactics of the Transborder Immigrant Tool*. *Digital Humanities Quarterly*, no. 1, 2013. http://www.digitalhumanities.org/dhq/vol/7/1/000157/000157.html.

Marino 2016 Marino, Mark C. “Why We Must Read the Code: The Science Wars, Episode IV”. In *Debates in the Digital Humanities*, edited by Matthew K. Gold and Lauren F. Klein, vol. 2, U of Minnesota Press, 2016, http://dhdebates.gc.cuny.edu/debates/text/64.

Marino 2018 Marino, Mark C. “Reading Culture through Code”. In *Routledge Companion to Media Studies and Digital Humanities*, edited by Jentery Sayers, Routledge, 2018, pp. 472–82. https://hcommons.org/deposits/item/hc:19537/.

Marino 2020 Marino, Mark C. *Critical Code Studies*. The MIT Press, 2020.

Montfort 2016 Montfort, Nick. *Exploratory Programming for the Arts and Humanities*. 1 edition, The MIT Press, 2016.

Montfort and Strickland 2013 Montfort, Nick, and Stephanie Strickland. “Cut to Fit the Tool-Spun Course”. *Digital Humanities Quarterly*, vol. 7, no. 1, 2013, http://www.digitalhumanities.org/dhq/vol/7/1/000149/000149.html.

Montfort et al. 2013 Montfort, Nick, et al. *10 PRINT CHR$(205.5+RND(1)); : GOTO 10*. The MIT Press, 2013.

Mullaney et al. 2021 Mullaney, Thomas S., et al., editors. *Your Computer Is on Fire*. MIT Press, 2021, https://doi.org/10.7551/mitpress/10993.001.0001.

Noble 2018 Noble, Safiya Umoja. *Algorithms of Oppression: How Search Engines Reinforce Racism*. Illustrated edition, NYU Press, 2018.

Oram and Wilson 2007 Oram, A. and Wilson, G., 2007. *Beautiful Code: Leading Programmers Explain How They Think*. O’Reilly Media, Inc.

PMLA 2006 “Program of the 2006 Convention”. *PMLA*, vol. 121, no. 6, 2006, pp. 1801–2000.

Poundstone 2005 Poundstone, W. (2005) *Project for Tachistoscope {Bottomless Pit}*. Electronic Literature Collection.

Pressman et al. 2015 Pressman, Jessica, et al. *Reading Project: A Collaborative Analysis of William Poundstone’s Project for Tachistoscope {Bottomless Pit}*. 1 edition, University Of Iowa Press, 2015.

Salter and Sullivan 2022 Salter, Anastasia, and Anne Sullivan. “Week 2: Of Textiles and Technology - Discussion Starter”. *2022 CCS Working Group*, 29 Jan. 2022, https://wg.criticalcodestudies.com/index.php?p=/discussion/113/week-2-of-textiles-and-technology-discussion-starter.

Shiffman 2012 Shiffman, Daniel. *The Nature of Code: Simulating Natural Systems with Processing*. 1st edition, The Nature of Code, 2012.

Soon and Cox 2021 Soon, Winnie, and Geoff Cox. *Aesthetic Programming: A Handbook of Software Studies*. Open Humanities Press, 2021.

Vee 2017 Vee, Annette. *Coding Literacy: How Computer Programming Is Changing Writing*. The MIT Press, 2017.

cárdenas 2022 cárdenas, micha. *Poetic Operations: Trans of Color Art in Digital Media*. Duke University Press Books, 2022.