49 CRITICAL UNMAKING

Toward a Queer Computation

Jacob Gaboury

If there is an ethos that drives our contemporary digital culture, it is the belief that technology improves over time and that improvements in technology bring improvements to our lives. It is this desire for the new that supports our faith in the potential good of technology in bringing about a better future (Kurzweil 1990). Simply put: technology drives progress. This belief is by no means new—indeed, this progressivist faith in technology is a defining feature of the postindustrial west—but it has been accelerated by computational technologies whose exponential growth has become the self-fulfilling prophecy of our digital age (Moore 1965). With the release of each new software update, each new generation of smartphone or game console, and each new operating system, we update and upgrade. We want what is faster and newer, or perhaps what we have no longer functions as it should or once did. We are compelled to engage in this drive toward the new—this push toward a technological future that is always on the horizon, always deferred.

Left out of these narratives of progress are the many ways in which technology fails us and the ways we fail with and through technology. Not all digital technologies work as they should. Many fail to make money or disrupt, transform, and revolutionize the world as they so often claim. Often, technologies fail to be adopted by users, or simply fail to work altogether. Yet our belief in technological progress asks us to ignore the bugs, glitches, and obsolescence built into technical systems (Slade 2006; Krapp 2011; Parikka 2011). Moreover, it asks us to limit our use of these objects to their most productive forms, and to deny any use outside these sanctioned boundaries, disavowing them as somehow failed. Under these terms, failure becomes much more than an unfortunate or unforeseen error; it becomes a refusal of this impulse toward compulsory production, a disruptive withholding of one's identity, activity, and process time. Indeed, we might argue that the production of failure within technological systems in this way constitutes a radical queer practice.

Radical, perhaps, but why queer? To be sure, queerness is not the only means through which we might understand technological failure, and indeed many forms of failure or misuse might explicitly work against queer forms of life. One need only look to the disruptive and often violent use of technology to harass and threaten women, queer and trans people, and people of color online to see the limits of non-normative technological use as a radical practice (Massanari 2015). Likewise, failure is often reframed and recuperated as a useful step along the road to innovation and success (Farson & Keyes 2003; Harford 2011; Firestein 2015). Nonetheless, there is great value in thinking about queerness and computation together, as digital media technologies continue to have a transformative effect on the ways we construct identity and build community. As such, it seems crucial that we find new ways to make queer theory speak to technology on its own terms. To this end, this chapter looks to "compute queerness" by both making it subject to the logic of computation and asking it to act computationally; that is, to become executable (Galloway 2004: 165). In doing so, it proposes a practice of critical unmaking, foregrounding queer techniques of refusal, misuse, and disruption that must nonetheless work with and through contemporary digital technologies.

Fail

Queerness and queer theory offer us an ideal lens through which to understand and critique the progressivist teleology that drives the production of contemporary technology. Just as failure offers us a way to think outside the compulsion toward progress through technological production (Nunes 2011; Menkman 2011), queer theory has for decades worked to critique the demands of biological reproduction under compulsory heterosexuality and the drive toward a future that is inaccessible to queer forms of life.

This connection between queerness and failure is embodied in a wide range of theoretical practices and intellectual histories reaching back to the start of the twentieth century. As literary theorist, Heather Love, notes in her work on loss and the politics of queer history, "same-sex desire is marked by a long history of association with failure, impossibility and loss," such that "homosexuality and homosexuals serve as scapegoats for the failures and impossibilities of desire itself" (2009: 21). Queerness here is marked by failure in that it not only exists outside a given norm, but cannot be made legible or useful to a given society. For early queer theorist, Guy Hocquenghem, it is capitalism that marks the homosexual as failed, where failure is understood as an incapacity for proper reproductive love. Without a clear connection between sociality, relationality, family, sex, desire, and consumption as afforded to heterosexual reproduction, homosexuality cannot be made productive to capital and is re-territorialized as a failed state of being (Hocquenghem 1993 [1972]: 93–112). For literary scholars such as Lee Edelman, the queer subject has always been epistemologically bound to negativity, occupying the space of the social order's death drive, an irrecuperable excess whose ethical value lies in "accepting its figural status as resistance to the viability of the social while insisting on the inextricability of such resistance from every social structure" (2004: 3). Queerness here is marked by its own illegibility to the social order, a social order that Edelman identifies as reproductive futurism. Indeed, this illegibility is its very value. For Edelman, we must resist the legibility of a secure political identity lest we lose the radical difference that queerness offers. Refusing this move away from the political, Jack Halberstam (2011) has taken up the radical potential of queer failure to suggest that living within failure and refusing the terms of success pushed onto us by capitalism, patriarchy, heteronormativity, neoliberalism, and other compulsory norms might allow for new ways of being in the world that need not disavow the possibility of a political imaginary, that may indeed form the basis for a politics of refusal. Failure here becomes a radical practice, one that is not without a future but instead reimagines how that future might come into being.

Simply put, queer theory has, for decades, engaged in the difficult and at times contradictory task of marking a political identity bound by a refusal to be made useful or productive. Deploying queer theory in this way asks us to think through failure as not only a disruptive technical practice but also a radical lived experience that might allow for an embodied critique of technological futurism. To compute queerly, then, is to acknowledge, embrace, and enact a practice of radical technological failure. It is to engage in critical unmaking: to make central those externalities—exploits, bugs, breakdown, abuse, and misuse—of our digital culture that, while pervasive, we nonetheless disavow. To compute in this way is to work against the neoliberal drive toward the capture and exploitation of the self by technology; to work against the demands of pervasive visibility by means of always-on devices, against the quantification of affect, leisure, and solitude for the purpose of value extraction; and to disrupt the fantasy that technology—through the aggregation and quantification of data—allows us access to some unmediated truth (boyd & Crawford 2012; Crary 2013). In acknowledging, accepting, and even producing failure, queer computation seeks to make clear the values and assumptions that drive our culture of technological development and to offer alternate modes for living with and through technology. But what might a queer computing look like?

Glitch

There are several objects to which we might turn to identify a queer computational form, but perhaps the most immediately recognizable would be small but playful forms of disruption such as the error or glitch. A glitch is a temporary malfunction within a technical, usually computational, system. Glitches are unexpected, but importantly they do not shut down a system entirely. They temporarily transform a technical object by producing an unintended result or error, and in the process lay bare its material function. A glitch is an outlier, an aberration. As such, it should be no surprise that it has been for decades an extremely productive site for artistic and critical investigation into the aesthetics and politics of digital media. Early net art artists such as the European art collective, Jodi, have, since the 1990s, used markup languages and videogame modification as platforms for disruptive play (Jodi 2001; Adang 2013). This interest in shattering the illusion of a medium by laying bare the processes and materials that produce its technical form is a longstanding tradition in modernist art and the avant-garde. Nonetheless, the glitch offers a unique engagement not simply with the specificity of its medium, but in its negotiation with failure as a temporally delimited and unexpected transformation in the function of computational devices.

In recent years, the aesthetics of glitch have grown in both visibility and popularity, with a number of artists adopting techniques for producing glitch in a variety of visual works to both critical and aesthetic ends. The artist, Rosa Menkman, deals explicitly with the disruptive potential of glitch as both an unexpected and engineered form. In her 2010 work, "A Vernacular of File Formats," Menkman offers a didactic visual text that points at ways to exploit and deconstruct the organizations of file formats into new designs. Repeatedly glitching an image of herself in a variety of file formats (e.g., JPEG, BMP, TIFF, and RAW), Menkman attempts to demonstrate the particularity of each, making visible its method of compression, its bitmap structure, and its analog to digital conversion. In one example, Menkman databends a GIF file, producing a horizontal blur that resembles a VCR tracking error. The resulting image appears warped and smeared across the frame in jagged interlaced repetitions. Alongside the image, Menkman notes:

> The GIF format uses a four pass dimensional interlacing strategy. This basically means that the image, consisting of different rows of pixels, decodes some rows of pixels before other rows. The example image shows the displacement of the different rows

during weaving (the putting together of the two layers), resulting in "combining artifacts" with "jagged edges."

(Menkman 2011: 21)

Bending each format multiple times, Menkman produces a surprisingly diverse range of visual glitches. In breaking the technology in this way, Menkman shows that the verisimilitude of each image is highly dependent on the formal specificity of the image file. What we are left with after the glitch is not the image itself but the file format revealed (Sterne 2012).

Here, the glitch offers both a compelling and frustrating example of queer computation. To be sure, the work of Menkman and others demonstrates the disruptive and didactic potential of technological failure, but it is also readily recuperable as a visual aesthetic divorced from critique. The proliferation of glitch as an aesthetic practice in this way diminishes its radical potential, a transformation that Menkman openly acknowledges. Differentiating glitch from failure itself, Menkman notes, "while failure is a phenomenon to overcome, the glitch is a phenomenon that will be incorporated into new processes and conditions of technological design or cultural meaning" (2011: 27). Glitch is, like noise, a relative and subjective phenomenon. All channels are noisy channels. All systems contain glitch. These phenomena have no essential quality; they are simply that which we disavow as irregular, unwanted, or queer (Chang 2013; Halberstam 2013).

This contradiction, whereby the radical potential of glitch is transformed into a productive aesthetic practice, echoes the concerns of Edelman and others with regards to queerness. In marking failure as an explicit and repeatable practice, it is abstracted from its critical potential and recuperated by technological futurism. Here, we find ourselves in a double bind. While we may desire to identify or prescribe a queer computational form that we might use to disrupt or transform our digital landscape, we must acknowledge that any such prescription opens queerness to a radical visibility such that it may be repurposed to serve a normative ideology. How can we break free of this bind? What queer forms cannot be recuperated in this way?

Taken to an extreme, we might logically argue that the only truly queer computational form is one that fully embodies the radical potential of failure. If any investment in productivity or futurity is normatively recuperable, then the only queer computer is broken, nonfunctional, or destroyed. Surely, complete annihilation is the most radical, most extreme form of failure possible. Yet, while it may be compelling to smash our computers in an act of queer rebellion, the radical potential of such a gesture ends there. A broken machine cannot compute, queerly or otherwise. It is a brick, a doorstop; it has no radical potential for computation as it has no computational function. Likewise, it may be compelling to simply opt out of digital technologies altogether, supposing that digital media are irreconcilable with a radical queer politics. While Luddism is certainly a form of critique, it is deeply limited in its efficacy here. We cannot simply ignore the pervasive influence of digital technologies on all forms of contemporary life, queer and otherwise, and we must find ways to negotiate queer technological practices while working against the recuperative drive toward productivity and futurity.

Norm

It would appear that queer computation cannot simply offer an antinormative critique of digital media. Instead, it must offer a reframing of the goals, drives, and interests of these media as technologies in which queerness is necessarily situated. This presents a challenge, given how closely queer theory and indeed queer life hews to a politics of antinormativity. Indeed, antinormativity is in many ways the ground on which queer theory was formed. As Robyn Wiegman and Elizabeth Wilson have argued, “nearly every queer theoretical itinerary of analysis that now matters is informed by the prevailing supposition that a critique of normativity marks the spot where queer and theory meet” (2015: 1). And yet computation is built on and requires norms in order to function at all.

The computer is a precisely engineered machine, and computer science is a discipline made possible through the arrangement of incredibly complex systems that must function with consistency and accuracy if they are to function at all. This belief in the exceptional perfection of computation is a founding principle of the field. Writing in 1948, John von Neumann—a key conceptual inventor of the stored program digital computer—notes that:

> A computing machine is one of the exceptional artifacts. They not only have to perform a billion or more steps in a short time, but in a considerable part of the procedure (and this is a part that is rigorously specified in advance) they are permitted not a single error. In fact, in order to be sure that the whole machine is operative, and that no potentially degenerative malfunctions have set in, the present practice usually requires that no error should occur anywhere in the entire procedure.
>
> (von Neumann 1961: 292)

Computers require near perfect conditions to operate. A single error might produce an incorrect calculation or, worse still, cause a machine to fail altogether. Yet this ideal of a perfectly functioning technology is precisely that: an ideal. Far more common than the successful execution of a given calculation—particularly in von Neumann’s own time—were bugs, errors, and failure. This was due in no small part to the high degree of human labor involved in the operation of a mechanical computer at that time. Programming, data input, and the interpretation of output were all dependent on mathematical calculation and interpretation by individuals known as either “human computers” or “computers,” many of whom were women (Light 1999; Grier 2007; Ensmenger 2010). Likewise, the machines themselves were highly unstable, reliant on unreliable vacuum tubes and experimental hardware long since phased out of our modern machines. Yet our belief in the scientific precision of computation persists, due in part to the increasing malleability of computational systems, which have automated many of these processes in an attempt to account for human error, unexplained glitches, and other forms of inconsistency at those sites where the analog world interfaces with the logic of the digital. Put another way, in spite of this ideal of scientific precision, modern technical systems are, to a degree, incredibly tolerant of failure and misuse.

What, then, might we understand as the “normal” conditions under which a modern computer functions? If this ideal of a perfectly running machine is little more than a fantasy, perhaps a more accurate description might be that computation requires certain base conditions to be met, a set of rules or procedures known as protocols. These protocols constitute the control structure of a given computational form. Unless they are satisfied, a machine will not boot, run, or compute. Yet, problematically, these protocols are often designed and described as being without an explicit politics, that is, as postideological. On the topic of distributed network protocols, Alexander Galloway notes:

> [I]deology is a problem to be forgotten or subsumed: networks are specifically conceived and designed as those things that both are nonideological in their conception

> (we just want to "get things done"), but also post-ideological in their architecture (in that they acknowledge and co-opt the very terms of previous ideological debates, things like heterogeneity, difference, agency, and subject formation).
>
> (Galloway 2014)

Protocols are, by design, expansive and inclusive, but this disavowal of their own political agency also makes them insidious, even dangerous. While technical protocols are intended to facilitate communication and interaction rather than restrict use, they nonetheless shape what is permissible to a given technology. As such, while a technology may allow for queer uses, it will nonetheless demand strict adherence to certain base assumptions. In this way, protocols set the conditions for discourse itself.

This normalization of the terms of engagement can have wide-reaching effects, particularly on minority populations whose needs, desires, and bodies are often excluded from the norms that structure protocological assumptions. Scholars such as Dean Spade (2011), Simone Browne (2015), and Shoshana Magnet (2011) have shown the many ways that the norms we program into our technologies have a disproportionate effect on non-normative bodies and identities, enacting a double violence in which subjects are made hyper-visible in their difference and also violently excluded from those sites at which normalizing technologies are used: airport security, police surveillance, and state and national identification documentation, for instance.

In the face of these protocological norms that set the very terms of engagement in our digital society, antinormativity offers us very little. How, then, can we queer, if not as a resistance to the norms that govern a particular system?

Code

I would suggest that our queer imperative must be to identify the ideological assumptions that produce protocologial norms and then subvert them—to make visible through a queer critical practice the values that structure our technology. If it is not possible to work outside the conditions for engagement produced by a given technology, then we must work with technical practices to critique and disrupt the values and assumptions that structure that technology. We must acknowledge the value in *making* as a productive practice that creates the conditions for political intervention, as well as the need for a critical *unmaking* as both a means to resist recuperation and a mode of being that exists outside of protocological norms—that is, as a form of life that cannot be made productive to technology. In other words, we must begin by asking ourselves a question: What does our protocol value?

To begin to think through this challenge, I put forward here an exemplary practice: a playful, half-serious, yet deeply political form of computational subversion known as the esoteric programming language or *esolang*. Esolangs are a group of weird, largely unused technical languages built for the purpose of testing the limits of programming language design. They seem well suited to the kind of queer computation described here as they play at the limits of computational logic but generally strive to be computationally complete. Esolangs can theoretically be used to program anything that a more standard programming language could be used for. However, they are largely avoided due to their bizarre or intentionally frustrating form. Esolangs take on many forms, from playful parodies of existing languages to maddeningly complex languages that test the very limits of what we understand as code (Mateas & Montfort 2005).

The first known esolang is INTERCAL, and it was developed in 1972 by Don Woods and James Lyon (Raymond 2015). INTERCAL began largely as a send-up of programing languages from the 1960s, such as COBOL, FORTRAN, and APL. Borrowing and transforming the vocabulary and structure of these languages, INTERCAL "reads" as a series of clever puns and reversals designed for a technical audience. It confounds where it should clarify, plays language games, and makes bad jokes; it frustrates utility, but it also opens new avenues for playful expression not available to more rigid or purposeful linguistic forms. The name INTERCAL, for example, stands for "Compiler Language with No Pronounceable Acronym." The compiler itself, called simply "ick," is designed to skip over any text it does not understand and cannot compile, rather than return a compiling error as with all standard compilers. This makes debugging a particular challenge, but it also allows programmers to comment and converse in-program through the addition of noncompilable text.

This commentary form of esolang is also the most common for artists and scholars looking to critique the normative structure of technical language design through the lens of gender and sexuality. For example, the artist, Zach Blas, produced the transCoder programming language as part of his Queer Technologies (2007–12) suite. Drawing on queer linguistic traditions of coded and obfuscated language, such as the Polari cant slang used by queer men in Britain during the nineteenth and twentieth centuries (Baker 2002), transCoder seeks to produce new computational forms not normally accessible to technical languages. The language relies heavily on statements meant to play on the function of existing languages, such as C, replacing commands like INCLUDE and FOR with tongue-in-cheek formations such as finger(), which stimulates data, and qTime(), which permits the executions of a program to run outside of conventional computational narratives. In this sense, transCoder plays on a major feature of esolangs, a kind of structured play that allows for "double coding," where meaning may be expressed as computationally complete and compilable code, but also, on another level, as legible human language that either supplements or subverts that code.

Still other esolangs work in precisely opposite ways, obscuring code such that it loses all human-readable reference. One of the most famous esolangs to exhibit this feature is the Brainfuck language, which allows for an extremely limited set of only seven commands, each of which is represented by a single nonalphanumeric character. All commands in Brainfuck are produced through the assembly of these seven characters, but this lack of alphanumeric text alienates the language from anyone attempting to read it as text. This extreme minimalism and refusal of human-readable language are what make Brainfuck notorious, but the language is in fact Turing complete, which is to say it can be used for most any functional programming task. While we need not be too literal in our interpretation, Brainfuck's name seems apropos, given how it plays on our expectation of what a programming language is and can do, fucking with our desire for a language that prioritizes clarity and functionality. Read another way, the language fucks with our desire for simplicity, offering up an offensively simple set of characters that render the language all but useless. Here, we see the ways in which queer technologies might acknowledge and redeploy the assumptions that structure a technical object or practice, and in so doing draw attention to the ideological function of its protocols, that is, those ideologies that its producers and users disavow.

Brainfuck has also inspired a number of derivative languages, but most notable here is the bodyfuck language developed by artist and programmer, Nik Hanselmann, in which each of the seven Brainfuck commands is assigned a body movement or gesture. To "write" in bodyfuck, the programmer/performer must use their body in discrete and specific ways:

> Jumping increases the memory register, ducking decreases the memory register, moving left or right either swaps the memory register or sets a point in the "magnetic strip" by which the head will return upon the zeroing of the preceding register (a loop).
>
> (Hanselmann 2009–10)

In adding an additional layer of difficulty onto an already tedious process, and invoking the body as a physical and material apparatus that makes possible the coding of technical systems, bodyfuck renders explicit the ways in which embodiment is both inscribed and erased into digital technologies. As Hanselmann notes:

> bodyfuck as well as the performances that it engendered places conventional computability in tension with movements of the body. While bodyfuck maintains its ability as to automatically compute, the process of transcribing the code into the computer becomes an arduous task. The embedding of physical difficulty into the creation of a computer program reifies the gaps between machine performance and physical performance. The computer didn't get sore. I did.
>
> (Hanselmann 2009–10)

bodyfuck does not break the function or concept of the programming language, but instead opens it up to new forms of failure and play by asking its programmers to consider their body and its performance as part of a technical practice. While it is unlikely bodyfuck will ever be used outside its initial artistic context, it points to the great potential of queer interventions that take up and subvert the very technologies they employ.

End

To compute queerness, we must begin by acknowledging what queerness offers to a critique of computation. In doing so, we are left with few clear answers and are instead asked to imagine new ways to work against the normalizing influence of our technical culture while maintaining the general functionality of the systems we inhabit. While we need not give up on the radical potential of queerness as a means of imagining a future that is not yet here (Muñoz 2009), we must nonetheless acknowledge how futurity has been colonized by the cultural logic of contemporary technology, and as such cannot serve as the primary vector for queer computational critique. Thus, rather than mobilize queerness as a useful technological apparatus, we might deploy it as part of a critical practice of unmaking.

Further Reading

Blas, Z. and W. Schirmacher (eds.) (2011) *The Transreal: Political Aesthetics of Crossing Realities*, New York, NY: Atropos Press.

Gaboury, J. (2013) "On Uncomputable Numbers: The Origins of a Queer Computing," *Media-N: Journal of the New Media Caucus*, retrieved from median.newmediacaucus.org/caa-conference-edition-2013/on-uncomputable-numbers-the-origins-of-a-queer-computing.

Keeling, K. (2014) "Queer OS," *Cinema Journal* 53(2), 152–57.

Magnet, S. (2011) *When Biometrics Fail: Gender, Race, and the Technology of Identity*, Durham, NC: Duke University Press.

Mateas, M. and N. Montfort (2005) "A Box, Darkly: Obfuscation, Weird Languages, and Code Aesthetics," in *Proceedings of the 6th Digital Arts and Culture Conference*, IT University of Copenhagen, pp. 144–53.

References

Adang, L. (2013) "Untitled Project: A Cross-Disciplinary Investigation of Jodi's Untitled Game," New York: Rhizome, retrieved from media.rhizome.org/artbase/documents/Untitled-Project:-A-Cross-Disciplinary-Investigation-of-JODI%E2%80%99s-Untitled-Game.pdf.

Baker, P. (2002) *Polari—The Lost Language of Gay Men*, London and New York, NY: Routledge.

Blas, Z. (n.d.) TransCoder, retrieved from www.zachblas.info/projects/queer-technologies.

boyd, d. and K. Crawford (2012) "Critical Questions for Big Data: Provocations for a Cultural, Technological, and Scholarly Phenomenon," *Information, Communication & Society* 15(5), 662–79.

Browne, S. (2015) *Dark Matters: On the Surveillance of Blackness*, Durham, NC: Duke University Press.

Chang, E. (2014) "Queer Glitches, or, the Recuperation of Vanellope von Schweetz," *Society for Literature, Science, and the Arts Annual Conference*, Notre Dame, IN: University of Notre Dame.

Crary, J. (2014) *24/7: Late Capitalism and the Ends of Sleep*, London: Verso Books.

Edelman L. (2004) *No Future: Queer Theory and the Death Drive*, Durham, NC: Duke University Press.

Ensmenger, N. L. (2010) *The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise*, Cambridge, MA: MIT Press.

Farson, R. and R. Keyes (2003) *The Innovation Paradox: The Success of Failure, the Failure of Success*, New York, NY: Simon and Schuster.

Firestein, S. (2015) *Failure: Why Science Is So Successful*, New York, NY: Oxford University Press.

Galloway, A. R. (2004) *Protocol: How Control Exists after Decentralization*, Cambridge, MA: MIT Press.

Galloway, A. R. (2014) "Network Pessimism," 11 November, retrieved from cultureandcommunication.org/galloway/network-pessimism.

Grier, D. A. (2005) *When Computers Were Human*, Princeton, NJ: Princeton University Press.

Halberstam, J. (2011) *The Queer Art of Failure*, Durham, NC: Duke University Press.

Halberstam, J. (2013) "Queer Gaming: Gaming, Hacking, and Going Gaga," *Queer Games Conference*, UC Berkeley, retrieved from www.twitch.tv/qgcon/c/3156423.

Hanselmann, N. (2009–10) "There Is No Hardware," M.F.A. Thesis, UC Santa Cruz, retrieved from nik.works/bodyfuck.

Harford, T. (2011) *Adapt: Why Success Always Starts with Failure*, New York, NY: Farrar, Straus and Giroux.

Hocquenghem, G. (1993 [1972]) "Capitalism, the Family and the Anus" in *Homosexual Desire*, Durham, NC: Duke University Press, pp. 93–112.

Jodi (2001) www.jodi.org/, retrieved from jodi.org.

Krapp, P. (2011) *Noise Channels: Glitch and Error in Digital Culture*, Minneapolis, MN: University of Minnesota Press.

Kurzweil, R. (1990) *The Age of Intelligent Machines*, Cambridge, MA: MIT Press.

Light, J. S. (1999) "When Computers Were Women," *Technology and Culture* 40(3), 455–83.

Love, H. (2009) *Feeling Backward*, Cambridge, MA: Harvard University Press.

Magnet, S. (2011) *When Biometrics Fail: Gender, Race, and the Technology of Identity*, Durham, NC: Duke University Press.

Massanari, A. (2015) "#Gamergate and The Fappening: How Reddit's Algorithm, Governance, and Culture Support Toxic Technocultures," *New Media & Society*, 9 October, retrieved from nms.sagepub.com/content/early/2015/10/07/1461444815608807.

Mateas, M. and N. Montfort (2005) "A Box, Darkly: Obfuscation, Weird Languages, and Code Aesthetics" in *Proceedings of the 6th Digital Arts and Culture Conference*, IT University of Copenhagen, pp. 144–53.

Menkman, R. (2010) "A Vernacular of File Formats," *Sunshine in My Throat*, retrieved from rosa-menkman.blogspot.ca/2010/08/vernacular-of-the-file-formats-2-workshop.html.

Menkman, R. (2011) *The Glitch Moment(um)*, Amsterdam: Institute of Network Cultures.

Moore, G. (1965) "Cramming More Components onto Integrated Circuits," *Electronics* 38(8), 114–17.

Muñoz, J. E. (2009) *Cruising Utopia: The Then and There of Queer Futurity*, New York and London: NYU Press.

Nunes, M. (ed.) (2011) *Error: Glitch, Noise, and Jam in New Media Cultures*, New York, NY: Continuum Books.

Parikka, J. (2011) "Mapping Noise: Techniques and Tactics of Irregularities, Interception, and Disturbance," in E. Huhtamo and J. Parikka (eds.) *Media Archaeology: Approaches, Applications, and Implications*, Berkley, CA: University of California Press, pp. 256–77.

Raymond. E. S. (2015) The INTERCAL Resources Page, retrieved from catb.org/esr/intercal.

Slade, G. (2006) *Made to Break: Technology and Obsolescence in America*, Cambridge, MA: Harvard University Press.

Spade, D. (2011) *Normal Life: Administrative Violence, Critical Trans Politics, and the Limits of Law*, Brooklyn, NY: South End Press.

Sterne, J. (2012) *MP3: The Meaning of a Format*, Durham, NC: Duke University Press.

von Neumann, J. (1961) "The General and Logical Theory of Automata," in A. H. Taub (ed.) *Collected Works: Vol. 5, Design of Computers, Theory of Automata, and Numerical Analysis*, Oxford: Pergamon Press, pp. 288–326.

Wiegman, R. and E. A. Wilson (2015) "Introduction: Antinormativity's Queer Conventions," *differences* 26(1), 1–25.