# ‘Device Not Found’: Failure and Frustration in Critical Digital Methodologies

### Emma Fraser and Clancy Wilmott

## Introduction

From their inception, all digital media technologies are failing.

Our everyday devices, phones, and computers are digital media machines, and from the moment new computation technologies are developed they enter a cycle of inevitable obsolescence. As such, the defining moment of a piece of digital media technology is in its relation to perfection and failure. Each release is more advanced and complete and more perfect, than the technologies that came before.[[1]](#footnote-2) Simultaneously, the success of digital media is already eclipsed by whatever new computational innovation is on the horizon. Every new app, platform, or piece of hardware or software is implicitly a failure in relation to the next more perfect update, version, or model.

In *Updating to Remain the Same*, Wendy Hui Kyong Chun argues that in order to maintain the status quo of media consumption, constant updates to software and hardware are needed in an enduring cycle of crisis and digital maintenance.[[2]](#footnote-3) This chapter presents a reconfiguration of Chun’s original position: rather than updating to remain the same in a constant state of renewal and progression, we argue that all digital media technologies are never finished or complete, but always in a state of comparative and increasing failure and decay. As Chun writes in her conclusion, what matters is ‘what and how things linger’, rather than ‘the new and fading—the bleeding edge of obsolescence’.[[3]](#footnote-4)

From this position, we ask: what would it mean to encounter digital media as already failing — and what would it mean to use this as a starting point for establishing a different kind of critical digital methodology which critiques notions of contemporaneity and completeness? Such questions shed new light on our relation to all — past, present and future — digital media technologies. We have never been modern.[[4]](#footnote-5) Software and hardware (like fashion) have never been up to date.[[5]](#footnote-6)

If we accept that digital media fail to be emergent, and only succeed in being outmoded and perpetually unfinished (lingering on as the ephemera of the contemporary technological moment that is always in-progress), we can develop a critical and pedagogical frame informed by Chun and others’ critiques of both digital media technology, and dominant framings of computation and technological development. Central to this approach is a move away from the fetishization of the new, opening up playful and critical encounters with contemporary and historic digital media technologies as spaces of perpetual failure and reframing them from perfect machines with imperfect users to imperfect machines: glitch-filled, design-limited, dust-ridden, bound by bodged code, and held together by loose connections.

## Device Not Found

In digital media technologies — unique compositions of software and hardware, computation, and silicon — failure is apparent across the material and the symbolic. Consider two computers sitting side-by-side in a teaching studio: a Macintosh 512k (c.1985), and a Macintosh SE (c. 1987). Their exteriors are covered in a residue of dust and grime left behind by many hands and many commands ground into their ‘Snow White’[[6]](#footnote-7) design over the years, the patina of old technology.

Curious students discover with disappointment that the Mac 512k is broken. A feature of the Mac 128k/512k is a small parameter RAM (PRAM) battery slot in the back, which was used to hold basic memory functions while the computer was off — a technical fix for the problem of a portable computer that would otherwise wipe essential data (like time and date, or user data) whenever it was unplugged and transported. A portable personal computer that was already failing in its most basic function from the very start. At some point in its previous 38 years of life, a small 4.5v Eveready No. 523 battery has been inserted into this slot and then forgotten — left to corrode, and eventually burst open. As a result of design failure and human neglect — but also entropy and obsolescence — corrosive material now coats the battery slot and has begun to corrode the computer’s circuitry itself.

The newer version — or a more recently *failed* version — of this early model, the Mac SE, was released without this external PRAM battery feature, and as such it still turns on and functions as designed. For students encountering this machine, the imperfections and failures of the much-fetishized original Mac combine with a strong familiarity with the fundamentals of its design. The antecedents of their own screen, keyboard, and mouse and the Apple ecosystem provide a strange encounter with failure as both a problem and possibility of modern technology. Once they identify the power switch on the back, the Macintosh SE turns on with a chime once familiar to many, but largely unrecognizable to this generation of students. Students begin to explore a suite of System (Mac O.S.) 4 software and applications, exclaiming as they make links between old and new interfaces (playing with menus and system preferences), and as features they take for granted (like the Spotlight search) are frustratingly absent. Searching through their own indexes of computation technology, they experiment between the awkwardness of the machine and the magic of computing.

Between material and usability failures, students learn that digital media technologies are bound to let us down, to break, to fail. With the appropriate attitude, they can be met intentionally in their imperfect state — expressly, critically. Here we understand ‘digital instantiations of imperfection as elements of friction capable of challenging the digital’s problematically frictionless veneer’.[[7]](#footnote-8) This frictionless veneer is in part a cultural perception that overlooks ‘social and machinic relations’ in order to fetishize the ‘magic’ workings of source code and programming, and invisibilizes the contact points between user, ‘machine’, hardware, and software.[[8]](#footnote-9) It also includes the mystifying, ideological framing of computation as the drive to perfect code and representation; computers as machines that simply execute the desires of the masterful programmer.

Consolidating knowledge, information, process, and meaning, code becomes logos. Code-as-logos inserts the abstraction of ‘software’ in place of executed code and conflates meaning into action in the pursuit of some essence of command and control through calculation.[[9]](#footnote-10) Automated code (in the form of, for example, compiled instructions or higher-order programming languages that short-cut the dreary repetition of raw computation via libraries and source code) becomes an exercise in truth-telling. To command code is to render knowledge and information into a unified language or form, to control a language without ambiguity that simply ‘does what it says’; instruction and result in tandem.[[10]](#footnote-11) In programming languages, ‘[o]ne’s word creates something living’[[11]](#footnote-12) — yet this creation is, always, imperfect, whether in the gap between meaning and action, user and machine, software and hardware, or writing and execution, or in the friction between instrumental reasoning and perpetual, illogical, obsessive reinvention.

Against such perfection and efficiency and the alchemical ‘sourcery’ of source code,[[12]](#footnote-13) we develop a pedagogical framing of failure as an alternative mode for the study and design of all computational technologies. The tension between playful, disruptive, and messy experiences of technology, and the idealized fetish of mastery and completion is the source of our claim that all digital media technologies are always failing: not just in the way that media in general fail to neatly signify (one of the interpretations of logos), but in the way that computational technologies at the heart of the digital media ecosystem fail to be error-free, fail to remain up-to-date, fail to function as expected or intended. In their failure, digital media technologies demand that we (academics, students, designers) figure out through trial and error; that we push past the veneer of perfection to find the productive frictions between function and obsolescence, which rest in failure.

## States of Failure

The challenge for us as researchers and educators is to find critical digital methodologies that reveal the frictions and patinas of digital media technology from the point of failure, rather than the axiomatic framework of perfect execution, novelty, and newness — at both the level of pedagogy and of research. Key to this is a resistance against the fantasy of command and total control that comes with what Chun describes as the ‘executive power’[[13]](#footnote-14) in programming — in our case, by making room for students to explore so-called ‘old’ digital technologies without guides or instructions, without prerequisite knowledge, and without the default assumption of immediate success or legibility. Old technologies provide an insight into the past and future of contemporary tech, but also a uniquely low-risk space in which failure can be embraced. In failure, the ‘executive power’ is humbled, and we see ourselves and our computation critically. Further, this failure is more transparent — we can pick apart the intersections between machine and code and see the system more clearly in its unfamiliar familiarity.

Thus, we invite students to sit with frustration and uncertainty, to flounder or make mistakes that might result in dead machines, missing files, incorrect outputs, or error messages. ‘Device not found’ here means execution incomplete; connection not quite made; attempt failed. But it also marks the meeting point of user and machine and failure and method. What does it mean for a device — a physical object that you can see and touch — to be invisible to another device? What problem does it pose, and what is the real failure here? Is it one of knowledge; of hardware; of code or compatibility; of representation; of obsolescence? Or is it the wider failure of the ever-present expectation that a machine (anthropomorphized by language and logos) *knows* something we don’t? That the machine’s code and software *does* something magical that we ourselves cannot do? Is it we who are failing to work with the machine, or is the machine always failing to meet us where we are?

By asking students to engage with technologies as they have (always) failed, we also aim to draw students away from what Chun calls the ‘sourcery’ of source code: the reification of the executable command that invisibly stitches language to action, reducing, or rendering invisible, the steps of compilation and interpretation in-between.[[14]](#footnote-15) Practically, this approach requires a veritable scrapyard of ‘old media tech’ — failed tech: Apple Macintoshes from the 1980s; Blackberries and Palm Pilots; a Nokia 3310; Atari and ColecoVision gaming consoles; floppy disks; and outdated operating systems like Fortran. Importantly, it is not only devices familiar in the history of computing — like the Macintosh Classic — that should be included, but those which have failed to have any significant legacy at all, like the lesser-known Intellivision gaming console by Mattel, or the iPod’s (more) failed peers like Microsoft Zune or SanDisk Sansa.

From the moment it was developed, the Macintosh 128k was already failing in relation to its successors: the 512k, which provided a more appropriate quantity of RAM to run the desired programs of the time, but also the Apple SE, the Macintosh Plus, and the (now classic) Macintosh Classic, all of which contained updates to their hardware and software that addressed prior failures and rendered existing models obsolete. To use a computer from 1984 today is to encounter failures known at the time, but also the failures that emerge more clearly as we have become accustomed to working with these machine’s successors. These are failures of design (the choice of 128k RAM to keep costs down meant the first Macintosh couldn’t run many common programs), time-induced hardware failures of disk drives and batteries, but also stark software failures with respect to accessibility and usability.[[15]](#footnote-16)

Simple tasks — finding the power button, for instance — become an exploration of failure. Not just failure of the broken machine, but of connection, and of recognition. ‘Why that symbol?’, students ask, referring to the ‘0’ and ‘|’ at the top and bottom of the power switch. These seem familiar, because they are contemporary signifiers for powering on and off (see Figure 2.1). The answer lies in the basic function of the machine: it depicts an electric circuit, closed or open — the most fundamental aspect of analogue computers and binary code: on and off — obfuscated by the increasing distance between command and execution, action, meaning, and function with each iteration of computing.

A picture containing graphical user interface

Description automatically generated

Figure 3.1: The evolution of the ‘power’ button on Mac, 1984-2020.[[16]](#footnote-17)

This isn’t simply a failure of young people with old tech. This common experience of failure is cross—generational and speaks to the permanent failure of all digital media technology in one way or another: we (the authors) did not know how to turn on and off our new Macbook Pro machines. The on/off signification has been ‘smoothed’ over in newer models, demonstrating — through failure — that the distance between command and action is increasingly obscured by ‘the digital’s problematically frictionless veneer’.[[17]](#footnote-18) In older technologies the same failure resides in a boxy mouse that is difficult to maneuver, a cursor that drags across the screen more slowly than we are used to, a program that requires shortcuts to run (and the associated knowledge), or a switch or button where there might now be a menu or a voice command. What we can learn from these moments of failure and disconnect is the fetishization of code and programming, which invisibilizes hardware, but also the fundamental experience of using technology as one of never quite being there, never finally arriving, and never actually taking off: a promise that is never really fulfilled.

## Critical Digital Methodologies

In digital media ‘the ephemeral endures’ — even as it fades. Just as our old Apple Macintosh hardware is degenerating, our social-technological relations are determined within a ‘present that is always degenerating’, but simultaneously resuscitated from ‘undead’ information accessed via digital media machines.[[18]](#footnote-19) When the students finally get the Apple SE working, they view files unchanged since the 1990s: saves of undergraduate essays and PhD data, ephemeral media salvaged through lines of code and saved in file sizes (4kb, 8kb) that they find difficult to comprehend. ‘This is hard’ they say, as they play games like *Lode Runner* (see Figure 2.2), struggling with the slow feedback from the keyboard and the blunt space of the blocky black and white landscape. ‘You have to think ahead. Press the spacebar early!’ Too late, another death and another failure. The game restarts and another student takes the keyboard. Another set of hands wear down the plastic, the connections, and the circuitry, trying to ‘figure out how’ in the space between execution and action.

Diagram

Description automatically generated with medium confidence

Figure 3.2: Screenshot of *Lode Runner* (128k port for Macintosh, c.1985).[[19]](#footnote-20)

We developed this pedagogical approach with failure as guiding principle from the start. Coding and programming are forged in failure. Machines and code are fetishized as world-making media even as they degenerate. As a result, our students don’t encounter technology just expecting it to successfully run or function. Through doing, they learn to understand digital media technologies as systems, devices, and interfaces that *never* ‘just work’, but in fact constantly fail. Without critical intervention, source code remains a mysterious backend, obfuscating the machine in a way that emphasizes the power of the programmer as commander, without questioning this knowledge, and without thinking about the gaps between meaning and execution.

Critical digital practices can operate in a mode perpendicular to more traditional forms of computer science; where the latter tries to solve problems through technology, the former attempts to intervene in technology *as both problem and possibility*. If we only focus on teaching students perfect mastery of program, code, and hardware, we deny them space to work things out through play, to learn from errors and disconnections, and to understand computation not as the increasing collapse of meaning and action, or an idealized practice that will eventually disappear the machine, but in fact a profound meeting of intention and chance, user and hardware, mistake, cover-up, invention, and perpetual failure. Rather than approaching failure as a nuisance element of computational practice — an element that stymies the search for perfect execution, and which is the exception rather than the norm — this chapter has argued that critical practices *must* embrace the constant frustration and uncertainty of computing, particularly in the digital media sphere. This awareness can be built up through working with old technology and broken components and sensors and working against the culture of novelty, mastery, and invisibility built through computational disciplines. It can also be recognized in the many false starts, misfires, and random connections of contemporary technological design. To resist the fetish of computing[[20]](#footnote-21), digital and computational pedagogy and research must engage in a more failure-oriented method of teaching and interpreting digital media technologies, a method that challenges the smooth veneer of perfection, where critical potential can be found against the illusion of frictionless computation, the mastery of programming, and the hubris of the new.

1. Wendy Hui Kyong Chun, *Updating to Remain the Same: Habitual Media*, Cambridge MA: MIT Press, 2016. [↑](#footnote-ref-2)
2. Chun, *Updating to Remain the Same.* [↑](#footnote-ref-3)
3. Chun, *Updating to Remain the Same*, 171. [↑](#footnote-ref-4)
4. Bruno Latour, *We Have Never Been Modern,* trans. Catherine Porter, Cambridge, MA: Harvard University Press, 1991. [↑](#footnote-ref-5)
5. For Walter Benjamin, for example, history is antiquity dressed up as novelty, though everything — including fashion and media — immediately becomes out of date. Walter Benjamin, *The Arcades Project*, trans. Howard Eiland and Kevin McLaughlin, Cambridge, MA and London, England: Harvard University Press, 1999. [↑](#footnote-ref-6)
6. ‘Snow White’ describes the minimalist design principle of the early Macintosh computers, with very little ornamentation, casings lightly colored with a fog color or similar (like putty or platinum), and using horizontal and vertical lines to hide ventilation and other functions. [↑](#footnote-ref-7)
7. Jakko Kemper, ‘Silicon Ashes to Silicon Ashes, Digital Dust to Digital Dust: Chronolibido and Technological Fragility in *Glitchhiker*’, in Caleb Kelly, Jakko Kemper, and Ellen Rutten (eds), *Imperfections: Studies in Mistakes, Flaws, and Failures*, New York, London and Dublin: Bloomsbury Academic, 2022, pp. 165-188, 167. [↑](#footnote-ref-8)
8. Chun, Wendy Hui Kyong, *Programmed Visions: Software and Memory*, Cambridge MA: MIT Press, 2011, 51. [↑](#footnote-ref-9)
9. Chun, *Programmed Visions,* 20-23. [↑](#footnote-ref-10)
10. Chun, *Programmed Visions,* 22. [↑](#footnote-ref-11)
11. Chun, *Programmed Visions,* 47. [↑](#footnote-ref-12)
12. See chapter ‘On Sourcery and Source Codes’ in Chun, *Programmed Visions.* [↑](#footnote-ref-13)
13. Chun, *Programmed Visions,* 27-28. [↑](#footnote-ref-14)
14. Chun, *Programmed Visions*, chapter ‘On Sourcery and Source Codes’. [↑](#footnote-ref-15)
15. Of course, the Apple II was also a successful product (manufactured from 1977-1993), and there were many contemporary competitors and precursors like the IBM Personal Computer or the Commodore 64. [↑](#footnote-ref-16)
16. Original figure created by Clancy Wilmott. [↑](#footnote-ref-17)
17. Kemper, ‘Silicon Ashes,’167. [↑](#footnote-ref-18)
18. Chun, *Programmed Visions*, 172-173. [↑](#footnote-ref-19)
19. The Internet Archive, https://archive.org/details/mac\_Lode\_Runner. [↑](#footnote-ref-20)
20. Chun, *Programmed Visions*, 18.

    ## Bibliography

    Benjamin, Walter. *The Arcades Project*, trans. Howard Eiland and Kevin McLaughlin, Cambridge, MA and London, England: Harvard University Press, 1999.

    Chun, Wendy Hui Kyong. *Updating to Remain the Same: Habitual Media*, Cambridge MA: MIT Press, 2016.

    \_\_\_\_. *Programmed Visions: Software and Memory*, Cambridge, MA: MIT Press, 2011.

    Kemper, Jakko. ‘Silicon Ashes to Silicon Ashes, Digital Dust to Digital Dust: Chronolibido and Technological Fragility in *Glitchhiker*’, in Caleb Kelly, Jakko Kemper, and Ellen Rutten (eds), *Imperfections: Studies in Mistakes, Flaws, and Failures*, New York, London and Dublin: Bloomsbury Academic, 2022, pp. 165-188.

    Latour, Bruno. *We Have Never Been Modern,* trans. Catherine Porter, Cambridge, MA: Harvard University Press, 1991.

    The Internet Archive, https://archive.org/details/mac\_Lode\_Runner. [↑](#footnote-ref-21)