


Minimal Computing from the Labor Perspective

Tiffany Chan <tjychan_at_uvica>, University of Victoria Libraries

Jentery Sayers <jentery_at_uvica>, University of Victoria  <https://orcid.org/0000-0002-7181-1703>

Abstract

The process of making digital objects available and discoverable demands a great deal of labor, from digitization, to creating metadata, to preservation, to importing it into a digital asset management system, and finally to presenting it. We begin this essay with a case study of one such system, called “Vault,” in the University of Victoria Libraries, and the work required to migrate from a Software as a Service (SAAS) model (called ContentDM) to a free and open-source software (FOSS) model (a customized instance of Samvera).

Vault illustrates what we call “minimal computing from the labor perspective,” which seeks to reduce the opacity of software through “low-tech” practices such as pseudocode, thereby reducing the alienation of practitioners from their projects. Drawing from feminist ecological work on capitalism, affective labor, and care, we advocate for the “degrowth” of digital projects by resisting tendencies to reinvest surplus labor and value into increased productivity. Instead, degrowth as minimal computing prompts practitioners to articulate a project’s needs and desires; what work is required and from whom; and how or whether to sustain this labor for the future.

For some practitioners, research begins in library collections or archives at the moment of contact with an object of interest. In the case of digital objects, this encounter is usually mediated by a digital asset management system. Although screens afford the allure of immediacy, the process of making digital objects available and discoverable demands a great deal of labor, from digitizing a physical object to creating metadata, to maintaining the object, to importing it into the system, and finally to presenting it. 1

We begin this essay with a case study of one such digital asset management system, called “Vault,” in the University of Victoria Libraries, and we focus on the work required to not only keep it running but also advance the migration process from Software as a Service (SAAS)^[1] to free and open-source software (FOSS).^[2] Vault does not seem like a typical minimal computing project. It does not involve a static site generator, small single-board computer, or lightweight markup language; nor does it espouse a minimalist aesthetic. Yet it does inform and enact what we call “minimal computing from the labor perspective,” which seeks to “degrow” digital projects. 2

Here, our use of “degrow” corresponds with ongoing feminist ecological work on capitalism, labor, and care [Barca 2019] [Akbulut 2017]. For these scholars, particularly Stefania Barca and Bengi Akbulut, degrowth “ultimately means eliminating the productive reinvestment of surplus value” [Barca 2019, 208]. For us, degrowth in the context of computing — or degrowth as minimal computing — prompts people to articulate what they need or want from a project, what work is required when, and who will do that work, all without appealing to increased efficiency or productivity. Minimal computing from a labor perspective is thus first and foremost about social relations between people, projects, and labor before minimalism is reified into a form, technique, tool, or feature. Practically, it defines an explicit endpoint to resist scope creep and potentially endless maintenance. Politically, it acknowledges and seeks to address power structures in claims to minimalism, including the hierarchies of labor they entail. In the following study of Vault, for example, we discuss how software maintenance resists binaries of masculinized programming culture (“real” technical labor) and feminized carework (devalued organizational or affective labor). Finally, there is a design angle to the labor 3

perspective, whereby minimal computing reduces the opacity of digital projects to make labor apparent. Through methods such as annotation and pseudocode,^[3] it demonstrates out how computation transforms objects and information, and it demystifies projects to avoid proverbial black boxes. As is the case with Vault, minimal computing from the labor perspective eschews SAAS in favor of FOSS while recognizing and supporting the everyday labor demanded by the latter.

Taken together, these practical, political, and design approaches to digital projects speak to the primary aim of minimal computing from the labor perspective: to degrow computation's alienation of practitioners from their own projects and from social organization and collective expertise. Rather than asserting this perspective and then applying it to examples, we begin with Chan's argument, based on experience, which describes the conditions and practices that gave rise to Vault and her contributions to it. From there, we extrapolate lessons from Vault to further define and historicize minimal computing from the labor perspective and outline its real and potential effects on the habits, cultures, and values of computing.

4

Case Study: Vault at the University of Victoria

I am one of two developers at University of Victoria Libraries ("Libraries") who knows and uses Ruby on Rails.^[4] Ruby is a general-purpose programming language akin to Python, while Rails is a server-side framework for building web applications with Ruby. I am responsible for developing and maintaining Vault,^[5] which is the Libraries' asset management system. Written in Ruby on Rails, Vault is a custom instance of Samvera's open-source repository software Hyku. It is further part of a multi-year, collaborative effort unfolding across occupations and groups within the Libraries, including digitization (Page DeWolfe, Leanne Gibbs, and Kathy Mercer), metadata and cataloging (Karen Dykes, Dean Seeman, Maggie Tan, and Shelley Coulombe), software development (Braydon Justice, Ethan Getty, and me), library systems (John Durno), and digital scholarship and strategy (Lisa Goddard and J. Matthew Huculak). Importantly, Vault is not the first platform the Libraries have used for asset management. Like most platforms, it arrived in the middle of our work at the University of Victoria (UVic), after we used the CONTENTdm software.

5

Changing platforms from CONTENTdm to Vault will gradually increase public access to the Libraries' collections by reducing UVic's use of proprietary software.^[6] This aspect of Vault aligns with some goals of the FOSS movement, where people "have the freedom to run, copy, distribute, study, change, and improve the software" [GNU 2021]. There is also a keen appetite among libraries and other memory institutions for open-source platforms. The list of Samvera adopters, for example, contains 485 organizations, including many university and college libraries as well as high-profile organizations such as the British Library and the Electronic Literature Organization [Colvard 2021]. That number will likely increase; several institutions recently reported migrating from CONTENTdm to open-source repositories, and many institutions worldwide are facing a major open-source repository upgrade from Fedora 3 to Fedora 4-6 [Dohe 2019] [Hardesty and Homenda 2019].

6

At the Libraries, switching from CONTENTdm to Vault will also help to reduce SAAS costs; however, implementing open-source software, and implementing it well, is obviously not free.^[7] When choosing between SAAS and FOSS, each institution should take into account their own needs, resources, and local contexts. One cost that institutions might have to bear for complex, server-side FOSS platforms is the infrastructure that supports software. They may also have to pay internal or external developers to develop, maintain, and/or migrate data to their instance of that software, and software localization may pose challenges in the process. When developers have to update FOSS, no automatic patch, built-in support, or documentation may exist for custom features designed for a particular institution. Developers such as myself thus consider each feature on a case-by-case basis to determine the best way to port the functionality from one version to the next. Sometimes this process is relatively simple (e.g., copy and pasting code that other developers have already tested), but it can also be very laborious.

7

Although maintenance is rarely discussed in research and academic publications, it is part of the routine work and decision-making processes in memory institutions such as the Libraries.^[8] It also points to palpable contradictions at play within project design and development, where increasing access may not reduce the labor of maintenance.^[9] Such

8

tensions assert themselves when projects and institutions have limited staff and faculty, are underfunded and under-resourced, and/or face significant material constraints. Kate Dohe observes that open-source software initiatives are mostly dominated by wealthy, historically white institutions and are still relatively opaque to non-specialists [Dohe 2019]. She adds that this disparity is compounded by tensions between the masculinized micro-culture of software development and the feminized library workforce. “The end result,” Dohe declares, is “elite institutions making products for other elite institutions, and every year the technical and economic barriers to entry grow higher” [Dohe 2019]. If libraries strive to support and customize software, and they recognize, as Samvera’s development team does, that contradictions between support and customization are the rule of, rather than the exception to, product and service design, then we must ask who can afford to develop and maintain FOSS [Frost 2015].^[10]

While Hyku may be more transparent than CONTENTdm, the technical expertise required to read or write its code nevertheless creates barriers to participation that existing governance structures often exacerbate [Dohe 2019]. For instance, one piece of software is composed of many different components working in concert. In the case of Vault, those components include Fedora 4 (repository software), Blacklight (discovery service), Solr (indexing software), and the International Image Interoperability Framework (IIIF), among others.

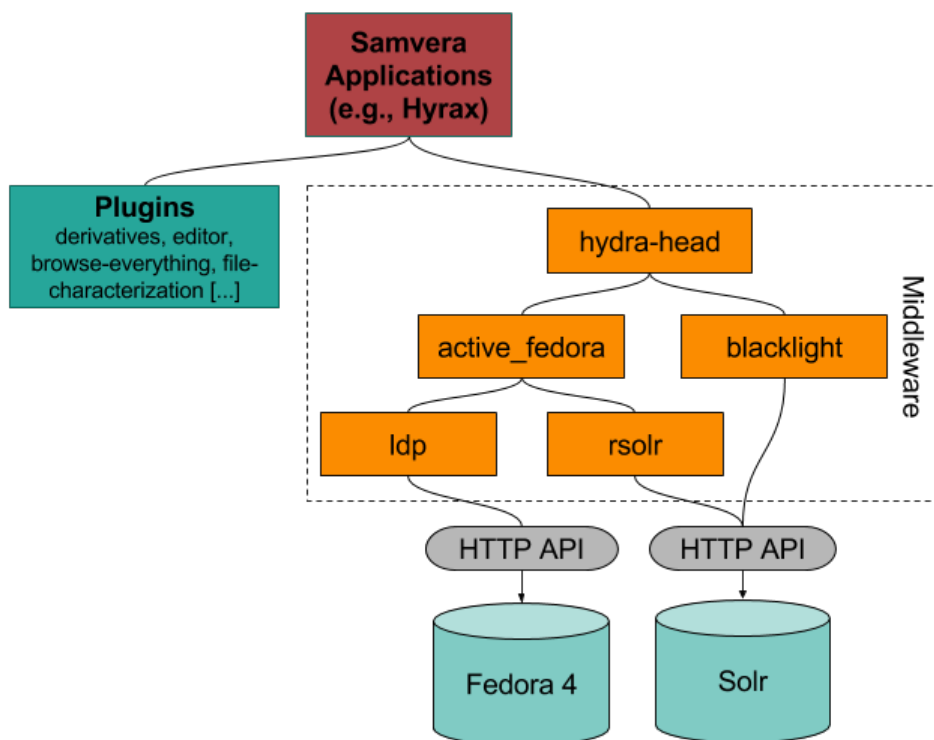


Figure 1. A diagram of components in Samvera’s digital repository software. For more, see <https://samvera.org/samvera-open-source-repository-framework/technology-stack/>.

My typical day can involve working with three to six types of code, depending on the task and how “code” is defined. Coding a single web page for Vault requires knowledge of HTML (the marked-up content of the page), CSS (the style or display), JavaScript (animation and interface elements), and Ruby on Rails (the page interacting with the server or database). Each of these languages has its own syntax and quirks while also relying or building on other languages. Programming a simple hide/show animation in JavaScript, for example, requires knowing which HTML element to animate — an element I will likely mark with a CSS selector or class before passing it into JavaScript. A task that may at the outset appear simple requires familiarity with three different languages that, in turn, require a non-trivial amount of time and labor to learn. The difficulties of this work are compounded when people are not compensated or supported for developing technical expertise, or where a lack of resources or staff means one person must juggle many separate tasks in addition to debugging software.

Errors and error messages are, like maintenance, largely ignored in broader discussions of technological innovation and software, except in minute conversations between developers or as lingering comment threads in forums such as Stack Overflow.^[11] They are unplanned interruptions and undesirable hiccups in the “seamless” experience of technology, and they are mostly forgotten as soon as a problem is fixed and an error disappears. But they are also ubiquitous, not to mention fundamental, to the composition of code and the maintenance of software such as Vault. Errors highlight the messiness of migration, localization, and upkeep. They emphasize how code resists exact or straightforward replication over time or across platforms. Although some code can be reused, people still need to modify it to fit a different context (e.g., renaming variables), match changes elsewhere in the software (e.g., updating to a code module needed for an existing application), or accommodate an unexpected edge case (e.g., data that does not easily fit within an existing database schema and thus requires an update to that database’s structure). Other common maintenance and debugging issues I might anticipate during a project like Vault include typos or syntax errors, libraries deprecated for performance issues or security vulnerabilities, adding or deleting columns in a database, cascading errors introduced by a bug fix, reference errors, or — worst of all — code that fails silently, without any indication of where the problem may be. Such errors are in fact so common they become banal, at least for developers.^[12]

11

Software development is associated mostly with writing new code; however, in practice I work with non-functioning code — that is, code that does not operate how my colleagues or I want or expect it to — as often as I do with functioning code. And I am not alone. Nathan Ensmenger notes that software maintenance is “the single most time consuming and expensive phase of [software] development,” representing 50-70% of total expenditures from the early 1960s to 2014 [Ensmenger 2014, 2]. Code is in constant need of repair. As Ensmenger declares, “All software has bugs; the question is simply whether [or] not they are known, and the degree to which they affect the general consensus on whether or not the software is ‘working’” [Ensmenger 2014, 4]. Ensmenger’s statement echoes research in maintenance studies that stresses the necessary but invisible and devalued acts of care and repair performed by people who keep the machine running [Rosner and Ames 2014] [Jackson 2014] [Chachra 2015] [Mattern 2018].

12

The development and maintenance of Vault, like research across maintenance studies, demonstrates quite clearly why code is not *logos*, or the one true “source [or] . . . representation of action”: something that simply works, untouched by the messy world of people and things [Chun 2013, 19]. As Wendy Hui Kyong Chun argues in *Programmed Visions*, fetishizing source code frames it as a seamless set of instructions that propel execution without regard for the many systems, agents, and social relationships in between [Chun 2013]. Chun writes, “[Fetishization] assumes no difference between source code and execution, between instruction and result” [Chun 2013, 21]. If code requires constant maintenance, then it cannot stand as a monolithic “enclosed object” — not only since it is always breaking down, but also because it is deeply enmeshed in social and material relations that enable it to run and continue running [Chun 2013, 54]. As Ensmenger puts it, “software . . . is like a contract, a constitution, or a covenant” under constant revision and negotiation [Ensmenger 2014, 11]. It is “history, organization, and social relationships made tangible” [Ensmenger 2014, 11], especially under the SAAS paradigm, where everything from networking and storage to applications and data may be hosted and managed remotely by a company or organization. This arrangement means SAAS customers usually play little to no role in programming and maintenance decisions related to access, discovery, search, metadata, customization, and the storage limits of their project. They are *here*, and software is *there*. The irony, then, is that the consequences of code as *contract* may not differ significantly from the consequences of code as *logos*. While they operate under distinct assumptions — “code does what it says” versus “code is a service” — both isolate code from social dimensions of its workflow. Projects such as Vault and Hyku may be understood as a response, if not a corrective, to this isolation. With them comes increased, localized attention to the contingencies of code: software moves from *there* to *here*.

13

As Vault demonstrates, code and software are contingent on the needs of the Libraries’ catalogers. Responding to those needs, Justice and I implemented a CONTENTdm migrator tool, which lets catalogers map metadata terms from one system to another before exporting CONTENTdm data to a comma-separated values (CSV) file with a row for each item and its metadata. Importantly, each row contains a file path to the digital object that the metadata describes so Vault can link them together. After refining the CSV, catalogers upload it to Vault. Vault parses the CSV, ingests both the metadata and objects, and performs various other tasks, such as indexing metadata or creating thumbnails. Previously, an empty

14

file path in the CSV would cause the entire batch upload job to fail without any notification to the uploader. So, based on feedback, Justice and I also implemented a file path checker tool that determines whether a file actually exists at the path indicated before Vault tries to import the object or metadata. The path checker then prints a list of empty paths for catalogers to correct. Accurately typing long file paths is often difficult and frustrating for people, especially if they are accustomed to graphical interfaces or different operating systems (for instance, Linux paths use forward slashes while Windows paths use backslashes). During the migration and localization processes, the Libraries are able to address some of these difficulties and frustrations through customization by way of automation, which may not be available to SAAS customers.

Vault's features are also contingent on what the Libraries do not need or choose to avoid. Justice removed Hyku's built-in notifications feature to decrease the number of internal system requests, which impede server performance. To restrict access, he also removed the option to register for an account without an explicit invitation from an administrator. I included an option for the Libraries to allow people who are not affiliated with UVic to view an item without being able to download it. Hyku's default settings coupled "viewable" with "downloadable," but the Libraries needed to restrict downloads for items that are in copyright or have protocols governing their use. These sorts of customizations fall under the broader umbrella of maintenance as they not only speak to the ongoing, local needs of the Libraries but also affect, or cascade across, Vault as a whole over time. Although they are quite technical in the particular cases of programming and debugging, they need to be communicated more generally to everyone in the Libraries who relies on Vault. Otherwise, code and software are once again isolated from who and what keep them running.

15

Pseudocode, or an informal description of how a computer program operates, is useful for such translation practices, and for instilling a sense of trust when labor is divided and individuals contribute only to specific parts of a platform based on their roles within the Libraries. Consider an example based on writing a Ruby on Rails web application ("app"). When digitizing objects such as periodicals or books for archiving purposes, the Libraries' digitization team scans each page as a .tiff image (Tagged Image File Format, or "TIFF") at about 600 dots per inch (DPI) and a file size of hundreds of megabytes each. However, people who access the Libraries' collections may prefer to view these images as .pdf documents (Portable Document Format, or "PDF") in which every image is a page. Justice, Greg Lanning, and I begin by talking with the Libraries' digitization unit about which Ruby functions and features may help to address this issue of format needs and preferences. Then the three of us compile a list of specifications, written in rather plain language. The list may look something like this:

16

- People will select a folder of TIFFs (through a dialog box) on a shared library storage drive for conversion.
- The app will ignore all non-TIFF files in this folder.
- The app will check the DPI of each TIFF, print a warning if the TIFF is under 600 DPI, and resample the TIFF to 600 if it is above 600 DPI.
- The resulting PDF will be linearized or "web-ready."
- Up to four people should be able to use the app at the same time.

This imagined app considers the fact that converting the TIFFs to a linearized PDF is a time-consuming task that cannot be done easily through a graphical user interface (GUI). Even proprietary software such as Adobe Acrobat, which allows for some automatic conversion, will crash when given a significant number of high-quality, information-dense image files. And so, rather than jumping immediately into the Adobe Creative suite, I begin writing pseudocode for the Libraries.

17

Pseudocode enables me to draft code as a series of actions the code will perform. To increase accessibility and bypass assumptions of shared technical language, I write in prose instead of code. "Linearize the PDF" would, for example, be expressed in Ruby as ``qpdf #{input_pdf_name} - linearize #{output_pdf_name}``. Here is sample pseudocode for the specifications list above:

18

1. Take a folder containing TIFFs as the input.
2. Create a list of TIFF files to convert.
3. Check the DPI of each TIFF and resample, do nothing, or print a warning if necessary.

4. Convert each TIFF to a PDF.
5. Combine all the PDFs (in the correct order) into one PDF file.
6. Linearize the PDF and save it somewhere.
7. Notify the user when the process is done and allow them to download the PDF.

Pseudocode is important for describing what a project should or must do without becoming tangled in the particular syntax of a programming language. As intellectual work, it helps practitioners to clarify what they need and how to judge whether, how, or to what degree a resulting program succeeds. It is also advantageous from a maintenance perspective because the potential for error in digitization projects is high, and it is far easier to debug a small program (in terms of number of lines) than a large one. By avoiding resource-heavy interfaces and scope creep, and by adding complexity incrementally and only when necessary, the Libraries can thoroughly test one part of a program and confirm it is working before moving to another feature or function of Vault.

19

Along the way, I may use a “low-tech,” non-GUI interface such as a text editor, command line, or interactive shell or console called a Read-Eval-Print-Loop (REPL) when composing code. I may also draw or sketch a workflow on paper before touching a keyboard. Such interfaces narrow attention and tend to reduce complexity when testing and debugging. They focus the work on describing details concretely. What functions are needed? What do those functions need — which data types, variables, or parameters? What are some errors that could occur? What alternative functions or workarounds exist if initial attempts fail? What information is needed from people, and what can be provided, calculated, or expressed by the computer? If the Libraries are creating profiles in Vault, for example, do we need a person’s first or last name or email address? Where and how will this data be stored, used, displayed, and retrieved?

20

After writing preliminary pseudocode, I build these steps using logical statements — such as if/then clauses or loops — to guide the computer in its decision-making process. If/then statements are like forks in the road: they describe two (or more) conditions that a computer calculates and pair each condition with a specific action the computer will take if the corresponding condition is met. Loops are repeated or repeatable if/then statements. They say, “As long as condition X is met, repeat action Y.” For instance, I can loop through a list of files in a folder (“for every file . . .”). Here is the revised pseudocode for the TIFF-to-PDF conversion process, including if/then statements and loops:

21

1. Take a folder containing TIFFs as the input. The function needs a parameter or argument: the folder’s name. This name must be unique and unambiguous.
2. Create a list of TIFF files to convert based on the folder’s contents. For every file:
 1. If the file is a TIFF, then add it to a list of TIFFs.
 2. If the file is not a TIFF, then ignore it.
3. For every TIFF in the list, check the DPI of that file.
 1. If the DPI is over 600, then resample it to 600 DPI.
 2. If the DPI is equal to 600, then continue to the next file.
 3. If the DPI is under 600, then warn the user but keep the file in the list for conversion.
4. Convert each TIFF file to a PDF file.
5. Combine all the PDFs (in alphabetical order based on the TIFF’s original filename) into one big PDF. Each PDF becomes a page of the resulting PDF.
6. Linearize the PDF and save it in a folder for download.
7. Email the user with a link to the PDF when the process is done.

After I create a pseudocode roadmap for my app, I begin to translate it into Ruby on Rails. Ruby contains a number of built-in methods: algorithms that transform some input into an output. Where possible, I look for these built-in methods as well as external code libraries (called “gems” in Ruby) to achieve common programming tasks such as creating, moving, or removing files. This step avoids reinventing the wheel. Why write code entirely from scratch if someone else has already created an effective approach to the problem? In some cases, writing code entirely from scratch is not reasonable. When converting a TIFF into a PDF, for example, I would have to know the byte-by-byte values of image headers (data placed at the beginning of a file that tells a viewer to parse or interpret the file as a TIFF) and replace

22

them with PDF headers, all without corrupting the content of the file itself. In this scenario, programmers usually rely on image manipulation libraries, such as ImageMagick, for conversion. Offloading the task in this way would allow me to concentrate on the scope of the app I am writing. It also saves me significant time and labor in the short term. Yet shortcuts like ImageMagick depend on external code not written or hosted by the Libraries. The use of such dependencies as part of Vault may therefore be vulnerable to obsolescence or deprecation, thus yet another tension at play in project maintenance. A desire to reduce the complexity of code conflicts with another desire to reduce dependencies and maximize a project's persistence. Such tensions are, for good reason, more likely to be accepted than resolved in projects like Vault.

Defining the Labor Perspective

What does my work with Vault tell readers of this journal about minimal computing? This may at first sound like a rather odd or misguided question. After all, Vault is substantial in both its size and reach, and it draws upon a significant array of resources in the Libraries.^[13] Again, Vault is neither the product of a static site generator nor stored on a small, single-board computer.^[14] It is also not a project about the art of programming, elegant code, or teaching the fundamentals of computing.^[15] It is, however, an informative case study for minimal computing from what Sayers and I call the labor perspective, where — drawing from work across media and technology studies, especially McKenzie Wark's *Molecular Red* — labor is an activity operating both within and against computation,^[16] where the combination of “within” and “against” is key to our approach [Wark 2015]. Contra Marx, we need not find humans, and specifically the universal human subject, always at the center of labor.^[17] To quote Wark, “Labor is the mingling of many things, most of them not human” [Wark 2015, 217], hence our attention to both activity and infrastructure throughout the following inquiry.

23

The logic of the labor perspective unfolds like so:

24

- Computation is historical, not abstract. Minimal computing starts in the middle of institutional work — *in medias res*.^[18] It engages computation as mediation, beyond the packaging of minimalism as a canned style, transhistorical feature, or list of fundamentals intended for download and exchange. From the labor perspective, a minimal computing project is its files and the relationships between them; it is also the ongoing activity of maintenance, customization, and care moving into the future with the past in mind.^[19]
- Computation subsumes labor. Or, labor is found within computation [Wark 2015, 19].^[20] Minimal computing happens within infrastructures greater, or larger, than itself. From the labor perspective, a minimal computing project attends to the power and persistence of these infrastructures and determines which ones are worth maintaining.
- Yet labor also experiments with computation [Wark 2015, 19]. Or, it is an activity of resistance and friction, especially with respect to software's default settings. Minimal computing projects find new uses for computing outside SAAS paradigms. From the labor perspective, a minimal computing project positions technical work as creative and critical work beyond familiarity with management systems.
- Minimal computing from the labor perspective may thus be defined as *the reduction of computation's alienating effects*: the alienation of projects but also of practitioners from social organization and collective expertise. From the labor perspective, a minimal computing project degrows computation's tendency to reinvest in productivity by investing instead in shared structures and common activities, also called “convivial computing” [Barca 2019] [Sterne 2007].^[21]

All of these aspects are more than metaphors, and they can be unpacked with attention to the particulars of Vault as a computing project and labor issue.

25

We would be remiss, however, if we did not first recognize how minimalism is often an expression of power, if not an exaltation of it. Google's reduction of search to a single input field on a mostly blank page is a canonical example of such power.^[22] The cost of Apple products, known for their minimalist design, is yet another. Beyond computing, we may — at the risk of digressing for a moment — look to minimalist sculpture in the U.S. starting in the 1960s. Donald

26

Judd once described such sculpture as “plain power” [Judd 2002], and Frank Stella said, “What you see is what you see” [Stella 1995, 158]. Here, the exposure of means is the medium, what Anna Chave deems not only a will to power but also the face of patriarchy and capital [Chave 1990, 51–8].^[23] In the New York’s minimalist sculpture scene during the 1960s, that will to power was also predominantly white and male, and a lot of the work was factory-made, or it evinced the factory-made.^[24] Chave notes that Tony Smith once boasted, “I didn’t make a drawing; I just picked up the phone and ordered it” [Chave 1990, 52]. Minimalist sculpture of that scene rarely left a trace or signature of its production, and it did not appear to address a particular subject or community of subjects [Foster 1996].^[25] It functioned more like an impersonal, austere, monolithic object: as “anti-artifice” rather than art in the making [Strickland 1993, 13]. It did not readily disclose its process of composition, either, reminding audiences that “laying bare” the means does not necessitate attribution, let alone transparency, of labor.

This labor issue persists in the exhibition and care of minimalist sculpture found in public parks and plazas. Consider, for instance, Chave’s analysis of Richard Serra’s *Tilted Arc*, constructed in 1981 and commissioned by the U.S. General Services Administration’s Art in Architecture Program:

27

In its site on Federal Plaza in lower Manhattan, Serra’s mammoth, perilously titled steel arc formed a divisive barrier too tall (12 feet) to see over, and a protracted trip (120 feet) to walk around. In the severity of its material, the austerity of its form, and in its gargantuan size, it served almost as a grotesque amplification of Minimalism’s power rhetoric. Something about the public reaction to that rhetoric can be deduced from the graffiti and the urine that liberally covered the work almost from the first, as well as from the petitions demanding its removal (a demand met last year). [Chave 1990, 59]

28

Here, minimalism was so assertive, so aggressive in its will to power, that people responded by tagging it and relieving themselves on it.^[26] Employees of the Department of Health and Human Services then had to attend to it — all to help maintain Serra’s public work. After almost a decade of this back-and-forth, *Tilted Arc* was dissected into three pieces, removed, and delivered to a nearby scrap-metal yard [PBS 1999].

29

While minimalist sculpture in New York may not appear immediately relevant to Vault, it does provide an example of how minimalism is value-laden, and it may serve as a model for how *not* to approach minimal computing from the labor perspective. Rather than assuming that a minimalist project is the impersonal expression of means or the exaltation of a lone artist’s patriarchal power and capital, the labor perspective attends to the social and material mingling of activities with infrastructure over time. We approach this mingling through technical, organizational, and affective labor in particular.

30

Technical Labor: On the Recalcitrance of Computing

The technical labor of Vault engages the “recalcitrance” of computing [Wark 2015, 18], and it starts in the middle. In the narrowest senses of mediation and history, it migrates and localizes materials from the “there” of CONTENTdm to the “here” of Vault in the Libraries. This by no means makes Vault unique. Thousands of these stories exist across the world, and — as noted previously in this article — more and more institutions are moving from SAAS to FOSS for the purposes of management. But *in medias res* persists as a theme even when we speak more generally about Vault; web and software projects rarely start from scratch and, of course, memory institutions by necessity engage with historical materials in their collections. In the case of UVic, these include collections such as Victorian Serial Novels; Herbert Geddes; Medieval and Early Modern Manuscripts; Food Not Bombs; and the Rikki Swin Institute: Gender Education, Research, Library, and Archives.^[27] Digitizing, stewarding, and caring for these materials is labor-intensive, and localizing them with customizations of Hyku written in Ruby underscores how migration is not a mere copy-and-paste job from CONTENTdm.

31

When viewed in the aggregate, these issues of mediation demonstrate why maintenance, rather than innovation or disruption, is a fundamental term for Vault and the labor around it. The lived reality of Vault and other FOSS projects is one of constant negotiation with software, of moving from one moment of stabilization to the next, between past and

32

future. As Vault reduces UVic's alienation from its own collections, and the Libraries from their own labor and computing, all as an alternative to proprietary software in the academy, it also makes everyone involved more aware of the means available for asset management. This awareness of means underscores the fact that technical labor — like all creative and critical work — carries both negative and affirmative connotations, from managing endless bugs and implementing updates to experimenting with code, building knowledge around historical materials, and imagining new uses for computing.^[28]

An awareness of means has been an appeal of minimalism for some time now. In his examination of minimalist tendencies in music and plastic arts, Edward Strickland asserts: “In much Minimal music . . . overt and immediately audible repetition of simple, even simplistic material, is the predominant structural principle. In dance, film, sculpture, and literature, similarly, Minimalism exposes the components of its medium in skeletal form” [Strickland 1993, 13]. The same may be argued for the technical labor of minimal computing; for instance, the components, skeleton, and even the principles of infrastructure are foregrounded in the design of open-source microcontrollers and static site generators.^[29] People accessing or contributing to such projects are presumably not estranged from computation's component parts and, in some cases, the connections between those parts and their dependencies. The materials are presented in plain sight, as if they *are* the interface. Vault does something similar through the simplicity of its online presentation. Descriptions, metadata, and items are all shared together on the same page, and a collection's size (in bytes), number of items, provenance, and permalink are readily available. Additionally, since Vault is a custom instance of Hyku — and Hyku is FOSS designed for modification and reuse — the Libraries may create new features or adapt existing ones to suit workflows specific to UVic.^[30] In CONTENTdm, there is no easy way for people to download either specific image items or images of an item's constituent parts, such as particular pages in a codex. The components and skeletal form of CONTENTdm are not readily apparent to local developers, and modifying its code is very difficult and likely illegal. CONTENTdm is thus an example of the proverbial black box.^[31] Developers and libraries must rely on the vendor (the Online Computer Library Center) to determine costs of features, which of those features to offer, and when.

With Vault, the Libraries see the component parts and their relations, and the UVic team may adjust the platform's settings to allow or disallow file downloads and increase granularity in access protocols, making only a subset of collections downloadable or permitting downloads of specific works in a given collection. This sort of technical labor, including customization, applies to the general maintenance and care for Vault: the living activity of ensuring the social and technical reproduction of the Libraries' holdings through code and software. It is also central to reproducing scholarship and, by extension, the academy [Shirazi 2017]. As we discuss later in this article, smaller projects, such as a journal built with Jekyll or an exhibit made with Wax gems, may integrate Vault, with items from UVic's holdings becoming featured materials for curation and interpretation and the platform itself serving as infrastructure for the description and persistence of assets.^[32] This aspect of Vault and projects like it is critical to approaching minimal computing from the technical labor point of view, as it prompts practitioners to account for the entire stack of scholarly communication, and to render visible its structures and components, even if no one is an expert in all the areas and technologies involved.^[33] While some layers of this stack, such as storage, servers, digitized assets, and metadata, may be centralized according to standards at institutions like the Libraries, other layers, such as applications and even data, may be decentralized.^[34] But focusing solely on particular layers of a stack — on only a journal or exhibit application and not the storage and server, for instance — risks abstracting an otherwise social process and treating its features as immediate or ahistorical.

Low-tech practices such as pseudocode help to render an otherwise recalcitrant stack intelligible to a group — to keep code mingling, and to degrow its alienating effects. The labor question is not whether everyone in the Libraries knows how to code (where programming is a means to power) or whether code does what it says (where code as *logos* is the means becoming the medium).^[35] It is, more generally, whether people follow how *this* becomes *that*: how, for example, a TIFF becomes a PDF with a simple program that begins as a specifications list and is then translated into Ruby, which is then written into infrastructure. Such shared working knowledge brushes against “what you see is what you see,” “plain power” assumptions of common sense, also popular in minimalist design principles like “Keep It Simple, Stupid” (KISS), where simplicity is presented overtly in ableist terms, as if it is obvious or readily apparent to whomever.^[36] In

reality, the unfolding of technical labor is not obvious, uniform, immediate, or even coherent. As pseudocode for Vault reveals, simplicity is value, and it must be iterated, learned, and communicated through composites of knowledge and experience that gradually decrease collective estrangement. Otherwise, the recalcitrance of computing and the layers of a stack are naturalized: assumed to be an abstract quality of a project or an inherent property of computers. “I’m not good with computers,” we might, for good reason, hear someone say, and minimalist expressions of plain power and assertions of technical labor’s simple sense may be reasons why.

Organizational Labor: On Bringing People Together around Tasks

The organizational labor of Vault addresses people’s everyday negotiations with computing but also “thread[s] people together around their tasks” [Wark 2015, 18–19]. Vault is a collaboration across digitization, metadata and cataloguing, software development, systems, and digital scholarship and strategy in the Libraries. Each of these groups has their own areas of concentration, across which they work and communicate. Organization is vital for these reasons. Vault is developed, maintained, and used by unionized employees at UVic; those unions agree upon and follow collective agreements, which are pivotal to stability, advocacy, and governance given the increasing number of precarious positions in and beyond the academy, the expanding breadth of job qualifications in the gig economy, the rise of for-profit colleges, and the maintenance, carework, and digital labor involved in FOSS [Aneesh 2006] [Scholz 2013] [Fuchs 2014] [McMillan Cottom 2017] [CUPE 2018].^[37] Such agreements also foster a collective awareness of a mediating apparatus, if you will, without assuming consensus among all involved.

Expounding Donna Haraway’s cyborg point of view, Wark says an “apparatus renders to the human a world that isn’t for the human” [Wark 2015, 151].^[38] Wark is not examining computing here, but it fits. While practices such as pseudocode matter for the comprehension of computation — for understanding in general terms how this becomes that — a mediating apparatus affects apprehension, or the aspects of a computing stack that are not intended for us. Components of code may “speak” more often to each other, or to machines, than to (most) people. Recall, for instance, this line of Ruby: ``qpdf #{input_pdf_name} - linearize #{output_pdf_name}``. It is not exactly human-readable, even if aspects of it are intelligible. Or, approaching the business of mediation more broadly, people are unable to witness what a project like Vault is doing “behind the scenes” or “on the back end.” So much process is indecipherable or invisible, and this can all be quite alienating in the absence of organizational labor. After all, people rely upon software and instill their trust in it, usually without being proximate to those who keep it running.^[39] Ensuring that an apparatus is understood as social and technical labor supported by collective agreements decreases the potential for estrangement without aspiring for immediacy. It also increases the visibility of technical labor, affirms it, and underscores why it is essential, in part by framing maintenance as research publishable in a journal such as this one. Furthermore, collective awareness of an apparatus — a shared apprehension of how it renders to us what is not for us — can shape the language and practice of organizing around Vault — “project” rather than “product,” for example, or minimal computing as labor first, then style, specification, file, or theme, if need be. With such changes, a project’s design process may begin with maintenance considerations rather than ignoring them (acting as if they do not exist), deferring them (acting as if they matter less than other considerations), or attending to them as they emerge (acting as if they are isolated incidents rather than patterns or cascades).

Affective Labor: On Being Directed by Computing and Computing Projects

The affective labor of Vault engages the feeling of “being directed” by computing and computing projects [Ahmed 2017, 43]. As we mentioned earlier, the challenges of technical labor are compounded when people are not compensated or supported for developing their expertise, or where a lack of resources or staff means one person must wear many hats and fix bugs in code as they arise. While the organizational labor of a collective agreement may help to alleviate issues related to compensation and professional development, the occupational aspect of “many hats” involves various cultural factors that are entwined with the everyday experiences of programming and carework. A labor approach to Vault underscores the need for structures and guidelines that protect employees from outside demands for increased efficiency or motivate institutions to reinvest a project’s surplus in productivity. UVic’s *Library Services for*

36

37

38

Grant-Funded Research Projects unfolds along these lines, communicating to researchers what the Libraries do and do not provide, when, and the monetary value of in-kind contributions.^[40] Where affect matters here is in the articulation of maintenance with service and happiness: fixing bugs that impede people's access to historical materials, for instance, or designing simple interfaces to decrease friction during research, not to mention enduring recurring acts of mansplaining and routinely being asked to communicate one's position in a feminized library workforce.^[41] These are topics, no doubt associated with expectations, that likely go unmentioned in a collective agreement. But, as Sara Ahmed makes clear in her work on building feminist worlds [Ahmed 2017], such expectations are norms, and in computing they are norms entangled with how software and its values govern people, especially white women, nonbinary folks, and people of color, in how to act and feel. Ahmed calls this "power as directionality" [Ahmed 2017, 43], and included within it is the heteropatriarchal command (if tacit) to appear happy and busy during maintenance and care, despite the fact that frustration, delays, negotiation, bugs, skepticism, confusion, and complaints are not just inevitable but necessary aspects of any inventive computing project, especially if and when people are wearing many hats.^[42]

Minimal computing from the labor perspective is thus premised on supporting (and not just "accommodating") people who are marginalized or silenced by systemic directionality. It aims to name problems, call attention to norms, slow down investments in productivity, discuss difficult topics, experiment, and refuse to feign happiness when maintenance and carework are devalued. This means the degrowth of computing's alienating effects must at the same time recognize alienation as a critical position, even if the result is a contradiction.^[43] Ahmed states: "If alienation is sensation, it is not . . . just or only the sensation of negation" [Ahmed 2017, 41]. It is "studious"; it is "wonder" [Ahmed 2017, 41]. Ahmed offers a guide to such alienation in her "killjoy survival kit" [Ahmed 2017, 235–249], and other cultural precedents for engaging affect in computing include *Bodies of Information*, edited by Jacqueline Wernimont and Elizabeth Losh; zines by Julia Evans, Mimi Onuoha, and Mother Cyborg (Diana Nucera); and recommendations made by Katrina Anderson et al. [Wernimont and Losh 2018b] [Anderson et al. 2016].^[44] Each of these publications succeeds at moving beyond heroification ("Hail the maintainers!") and romantic celebrations of maintenance without devaluing or abstracting it from social and material relations.^[45] Parallel steps may include recognizing how minimal computing manifests from both a necessity and desire to degrow. This is the path to which we now turn.

39

Degrowing Digital Projects: Necessity and Desire

Degrowth in minimal computing instigates a shift from using technologies to reinvest in productivity ("crunch," increased output, longer hours, more data, jobs rather than careers, and technology as a service, for instance) to designing shared structures that support technical, organizational, and affective labor as critical and creative activities. This is frequently achieved by reducing scope creep, and thus we might echo Alex Gil's question, "What do we need?" [Gil 2015]. We might also look to *archipelagos — a journal of Caribbean digital praxis* for examples. With co-editor, Kaiama L. Glover, Gil describes that journal in the following way:

40

[W]e have thought through our platform with the same critical eye we cast on the archive, and our resulting infrastructure embodies our principles. We are fully open access and charge no author fees. Our authors retain their copyright. We pursue best indexing, accessibility, and archival practices. We emphasize the primacy of "sustainable authorship in plain text."^[46] Our website and PDFs are generated from the same markdown files using Jekyll and ConTeXt, respectively. The resulting website is light-weight and mobile-friendly, acknowledging the importance of mobile phones, bandwidth differentials, and data costs in the Caribbean. [Glover and Gil 2020]

The skeleton of the project, not to mention its editorial and commit history, are accessible to readers along with the journal's technological stack. Readers are also "allowed to read, download, copy, distribute, print, search, or link to the full texts of the articles in [the] journal without asking prior permission from the publishers" [Glover and Gil 2020]. The publication process relies on Markdown and format redundancy (PDF and HTML) rather than extensive markup or word processing, and it is enhanced by Dublin Core for general metadata needs.^[47] The interface, like the project itself, privileges content and responsive design. There are no ads or contrast errors, there is only one menu, the HTML is mostly valid, and articles are available in a single column (without distracting sidebars). The journal also bypasses a

41

content management system and database and, in doing so, reduces the frequency of potential errors and hacks. All of this work is done by necessity to facilitate access in the Caribbean and reduce the likelihood of alienating readers and researchers there. At one point, Glover and Gil ask, “How might we encourage collaboration with, increase accessibility for, and otherwise work to narrow the gap between Caribbeanist researchers, especially those in the North Atlantic academy, and the communities we are committed to serving?” [Glover and Gil 2020].

From the labor perspective, Glover and Gil also articulate minimalism with sustainability. They publish the journal’s workflow, render clear its standards and structure, bypass subscription fees (including SAAS fees), credit all team members (including the designer, architect, and editorial board), and ensure the journal is available in English, French, and Spanish. Even though *archipelagos* relies on GitHub for storage and distribution, the entire journal (213.3 MB as of this writing) could easily be downloaded to a USB stick and deposited with a library, press, or nonprofit organization, without any need for a running version of Ruby or Jekyll.^[48] What’s more, Glover, Gil, and the *archipelagos* team provide a low-maintenance, light-footprint model for other projects that could integrate and interpret collections housed by open-source platforms such as Vault, thereby bridging small, nimble, and focused projects with large, persistent, and extensive ones often found in libraries. Their model may especially appeal to groups and institutions that are underfunded, under-resourced, and/or understaffed. Certainly, the project’s files and open-source repository and the files contained within it are crucial here; however, the technical, organizational, and affective labor involved in *archipelagos*’ production is equally important, framed around a dedicated team with a clear sense of the sort of scholarship and knowledge work they want to see in the world. The project may be what Jonathan Sterne had in mind when, in 2007, he called for “a whole convivial system of digital components, a convivial digital infrastructure” [Sterne 2007, 28].

42

The labor of minimal computing may also emerge from a desire to degrow “forever” projects. Here, we might ask what relationships we want with technologies and each other as we are propelled into the future.^[49] Keeping in mind that academic FOSS initiatives are largely dominated by wealthy, historically white universities, what kind of work can other institutions maintain? Or do they wish to maintain? Which infrastructures are worth keeping? Which projects are, or should be, prioritized, updated, and routinely debugged?^[50] For how long, and by whom? Which tasks do we want to automate or offload to scripts, and which do we want to conduct manually?^[51] If, as Susan Brown et al. indicate, “Users now firmly expect that scholarly digital publications will be kept up-to-the-minute and respond to user suggestions” [Brown et al. 2009], then minimal computing from the labor perspective addresses user (or consumer) expectations directly by accepting and encouraging others to accept the fact that the internet and digital materials, like all other materials, age, rot, and degrade [Kirschenbaum 2008]; that code does not always do what it says [Chun 2013]; that capturing content and archiving the web is a “messy affair” [Davis 2014]; and that even public projects must stop growing at some point. Initiatives such as Vault are in positions to organize around this issue and educate people on the lived realities of preservation, planned obsolescence, and digital labor, where projects are published yet presumably never done [Brown et al. 2009] [Kirschenbaum 2009].^[52]

43

Part of this degrowth may imply attention to, if not an embrace of, ephemerality in scholarly communication [Donaldson 2020] [Sample 2010]. Or, in some cases, the labor response may very well be “done” or “enough,” a refutation of the assumption that practitioners are service-providers who accumulate piles of technical debt that must be paid indefinitely back to users expecting fresh content on demand.^[53] And yet others may focus on computing and labor’s potential role in environmental justice. As Stefania Barca argues, “Alienation of the producers from the products of their work is what leads to the reinvestment of surplus into increased production,” and “weak unions and virtually non-existent enforcement of labor regulations play a major role in determining the environmental impact of production” [Barca 2019, 209–10]. Each of these possibilities demonstrates why, in the last instance, necessity may be inextricable from desire.

44

At UVic, the Endings Project (2020) at the Humanities Computing and Media Centre (HCMC) has begun addressing degrowth via technical labor practices.^[54] Although they do not use the term “degrow,” they argue that “projects — even digital ones — need to end” [HCMC 2020]. We agree from the vantage of both desire and necessity. The HCMC outlines “endings principles for digital longevity,” consisting of five primary components: data, products, processing,

45

documentation, and release management. Various aspects of these principles correspond with dimensions of minimal computing, such as “build a static website with no databases,” “[d]ata is stored only in formats which conform to open standards and which are amenable to processing,” and “every entity in the site has a unique page with a simple URL” [HCMC 2020].^[55] One potential benefit of the Endings Project is it could nudge researchers, especially faculty members, to start their projects with an ending in mind and — recalling Tony Smith’s boast in 1966 — curb the likelihood of people “phoning in” projects for others to build and maintain.^[56] Starting a project with an ending, and a roadmap for how to get there, should increase researcher awareness of projects like Vault as well as the labor required to publish content and keep the machine running. It could also encourage more researcher involvement in maintenance and carework and, by extension, degrow researcher alienation from their own projects. Importantly, all of these possibilities extend beyond the habituated use of management systems — a tendency encouraged by many SAAS paradigms.

And ultimately, that is what we hope minimal computing from the labor perspective may achieve through degrowth: a change in habits, culture, and values, not just technologies. We admit it is no small task.

Notes

[1] “Software as a Service” is a business model where a vendor owns and deploys software from a centralized server (such as a cloud server) and customers subscribe to that service for a recurring fee [Dictionary of Computer Science 2016b] [Dictionary of Information Science and Technology 2013]. Examples of SAAS include Adobe Creative Suite, Netflix, and Amazon Web Services.

[2] Free and Open Source Software, or Free/Libre Open-Source Software (FOSS), encompasses software that anyone (with expertise) is “free” to obtain, modify, and/or redistribute. Often, software based on FOSS must also be distributed with the same free license as the original [Dictionary of Information Science and Technology 2013]. Open-source software stems from the belief that “widespread inspection, modification, and correction of the source code” by interested parties fosters high quality software [Dictionary of Information Science and Technology 2013]. For a historical perspective on the FOSS movement, see [Jullien 2009].

[3] Pseudocode comprises natural language (such as English) statements mixed with the structure of code elements (such as if/then conditional statements or loops). Although pseudocode may mimic instructions, it is meant for human interpretation and not machine execution [Dictionary of Computer Science 2016a].

[4] “I” here and throughout this article is Tiffany Chan.

[5] For more about Vault, see vault.library.uvic.ca or the code repository at <http://github.com/UVicLibrary/Vault>.

[6] Jentery Sayers defines this aspect of minimal computing as “maximum access”: “reduc[ing] the use of proprietary technologies and paywalls to increase access to content, data, and/or source files” [Sayers 2016].

[7] For more on migrating from CONTENTdm, see Amanda Mita et al., who “provide a feasible workflow for breaking with CONTENTdm and strategies for modifying alternative systems to accommodate archival digital collections” [Mita et al. 2018]. See also Heather Gilbert and Tyler Mobley’s description of “breaking up” with CONTENTdm [Gilbert and Mobley 2013].

[8] Discussions of maintenance in *Digital Humanities Quarterly* include [Engel et al. 2018] [Gold 2009] [Kretzschmar, Jr. 2009] [Smithies 2011] [Wernimont 2013]. Drawing from [Lientz and Swanson 1980], Nicolas Gold writes about costs: “Although much emphasis is placed on the delivery of new systems, the maintenance of existing software consumes at least 50% of the lifetime cost of a software system” [Gold 2009]. William Kretzschmar underscores the relationship between maintenance and funding: “The largest humanities computing projects are likely to require continuing care and maintenance, if not more radical representation and reinterpretation in light of the advance of scholarship, and yet they seem unlikely ever to be funded comprehensively for these tasks. The best way forward is to create some sort of stable institutional setting for large projects that will provide continuity and baseline resources for the work” [Kretzschmar, Jr. 2009]. Meanwhile, James Smithies writes about methodologies: “We are only just beginning to find processes that will genuinely assist our software engineering tasks. There are now a handful of broadly accepted methodologies that claim [to] — and generally do — make the task of software engineering and maintenance easier (RUP, AGILE, PRINCE 2, ITIL), but few organisations have either the will or the means to deploy these approaches in a ‘purist’ sense” [Smithies 2011]. Jacqueline Wernimont engages maintenance, labor, and funding via intersectional feminism and work by women: “Both [Women Writers Online] and Orlando depend on scholarly collaboration to create and maintain their materials. At this point in most digital scholarly projects, collaboration is happening between a small set of trained graduate students, faculty, and IT and library staff. This is often due to a complex nexus of concerns, including interest, scholarly expectations, expertise, and where funding and labor cycles are consistently available”

[Wernimont 2013].

[9] Sayers defines this aspect of minimal computing as “minimal maintenance,” or “reduc[ing] dependencies and the use of features to decrease the labor of updating, moderating, and stewarding a project over time” [Sayers 2016].

[10] For its part, Samvera has taken steps to make its community more welcoming, including creating a code of conduct, anti-harassment policy, and an incident response team [Samvera 2021]. Additionally, the Samvera Branch Renaming Working Group has renamed the “master” branch of their GitHub repositories (i.e., the default or core version of their code) to “main.” The former term, often used in coding jargon with “slave,” both originates from and perpetuates racist language. See <https://groups.google.com/g/samvera-community/c/pbKs4nj5gBU> for more.

[11] For more on errors and maintenance, see Adam Barr’s *The Problem with Software: Why Smart Engineers Write Bad Code* (2018). Barr includes a history of software engineering (1960s through the 1980s) and stresses a split between post-secondary education, where students write small pieces of software, and industry, where people design and maintain large pieces of software. Building on work by [Dijkstra 1972] and [Knuth 1974], he draws a distinction between the performance and design of software, noting how software written to perform well may actually require more maintenance [Barr 2018]. An emphasis on optimizing a project now ignores long-term issues, such as when groups will need to fix and update code.

[12] Consider a simple example of a software error written in JavaScript, as typed into an interactive console in the Chrome browser (the first two lines are from the programmer and the last is a reply from the computer):

```
var test = 4 tst + 6 Uncaught ReferenceError: tst is not
Error: "tst" was not defined defined at <anonymous>:1:1
```

Pseudocode:

```
Create an entity called "test" that has the value 4. What
is tst plus 6 equal to? (I don't know what "tst" is)
```

In the example above, someone might reasonably guess that “tst” is a misspelling of “test” and therefore that `tst + 6` should equal 10. But the computer in this scenario has labelled this a reference error because the computer assumes there is an entity “tst” to which the programmer has forgotten to assign a value. By the computer’s understanding, the programmer is referring to something that does not exist. This is a very simplified version of a miscommunication, but it demonstrates how the assumptions people and machines make are not necessarily identical. Is “tst” a misspelling of a variable “test,” or is it a variable on its own? In this scenario, if not generally speaking, computers do not tolerate ambiguity well; they rely on and require both assistance and interpretation.

[13] For one example, see the growing popularity of Jekyll, a static site generator considered to be an alternative to resource-heavy, database-driven management systems such as WordPress.

[14] For an example of research conducted on small-single board computers, see James Smithies’s essay, “Full Stack DH: Building a Virtual Research Environment on a Raspberry Pi.” Near the end of his argument, Smithies makes a point about maintaining his <https://jamessmithies.org> minimal computing project: “[A]t the end of my career I hope to have a single project that consolidates and presents a lifetime of digital scholarship. This aim is perhaps the larger challenge posed by jamessmithies.org. For people with the requisite skills, does a minimal computer + open source web framework + cloud service integration amount to the ultimate VRE? The need for ongoing maintenance suggests not. Conceived as a decades-long personal project, jamessmithies.org is likely to stay live for some time as a by-product of intellectual and technical engagement, but it is far from a model for general use” [Smithies 2018].

[15] See, for instance, Donald Knuth’s *The Art of Computer Programming*, first published in 1968.

[16] “We” from here through the end of the article are Chan and Sayers, and we engage Wark via a technique she calls “substitution” by rewording her approach to labor in *Molecular Red*: “Labor finds itself in and against nature” [Wark 2015, 19–20]. In our argument, “nature” becomes “computation” or “computing.” We are also striving to avoid what Wark identifies as a limitation of this philosophical approach: “The result tends to be the thought of activity without matter or of matter without activity” [Wark 2015, 19].

[17] See *Capital*, Volume One, for instance. Labor is “a process between man and nature, a process by which man, through his own actions, mediates, regulates and controls the metabolism between himself and nature” [Marx 1976, 283]. The “between” here — the boundary between people (“man”) and nature — differs from our use, via Wark, of “within and against.” For a critique of the universal human subject in digital

praxis, see Roopika Risam, who writes: “With the move to generate software and algorithms that replicate ‘human’ processes, particular forms of ‘human’ are authorized. As postcolonial scholars have argued, the Enlightenment gave rise to the idea of a homogeneous definition of ‘human,’ which centers the European subject and, in turn, marginalizes all whose cultures, lifestyles, and values deviate from the universal. Postcolonial theory, crucially, has made the case for the importance of the particular, grounded in the idea that, indeed, cultures — specifically the cultures of colonized or formerly colonized communities — are left out by universalist discourse” [Risam 2018].

[18] Sayers elaborates on starting *in medias res* in his introduction to *Making Things and Drawing Boundaries* [Sayers 2017].

[19] On carework in particular, see Bengi Akbulut, who defines it both narrowly and generally: “The most straightforward (yet admittedly narrow) definition of carework is labor performed to fulfill the needs of those who cannot do so themselves, such as food provision, cleaning, health, etc. Broader understandings of carework stress that such work is often performed in tandem with and complementary to other types of (unpaid) reproductive labor and cannot be considered separate from the broader sphere of social reproduction. That is to say, carework is better seen as the *more comprehensive field of paid and unpaid labor that ensures social reproduction in general*” [Akbulut 2017, emphasis original]. See also [Moraga and Anzaldúa 1983] and [Ferguson and Folbre 2000]. Technologies are of course one area among many where such social reproduction occurs and is regulated by patriarchal biopolitics and racialized protocols. As Michelle Murphy notes: “A century of feminist calls to seize the means of reproduction, to take control of one’s own body, to love oneself, to embrace reproductive rights, to end racism, to denounce reproductive technologies, to enjoy sex, to situate bodies intersectionally and so on are all quintessentially biopolitical” [Murphy 2012, 10].

[20] Here we again draw directly from Wark, specifically her engagements with Alexander Bogdanov and Karl Marx in *Molecular Red*: “The labor point of view has to reject ontologies of abstract exchange with nature. . . . Labor is always firstly in nature, subsumed within a totality greater than itself. Labor is secondly against nature. It comes into being through an effort to bend resisting nature to its purposes. . . . Labor experiments with nature, finding new uses for it” [Wark 2015, 19].

[21] For more on degrowth from the labor perspective, see Stefania Barca who writes: “[T]he project of building a degrowth society can only start from fostering dealienation by reopening the possibility for workers control and economic democracy, from the workplace to society at large” [Barca 2019, 209]. For more on convivial computing, see Jonathan Sterne’s work. Engaging Ivan Illich’s work on technology in an essay about waste and computer trash [Illich 1973], Sterne writes: “Illich uses the term ‘conviviality’ to connote the following characteristics of technologies: ease of use, flexibility in implementation, harmony with the environment, and ease of integration into truly democratic forms of social life. Obviously, Illich’s vision is a utopian one, but his measure of a technology’s conviviality seems relevant to the question of computer trash. We need a ‘convivial’ computer” [Sterne 2007, 28].

[22] Perhaps the most common way people talk about minimalism, especially in Europe and North America, is through the language of reduction. Encyclopedia definitions of minimalism are rife with reduction [Oxford 2007] [Encyclopedia Britannica 1998], which Jon Barth finds in writing by everyone from Robert Browning (“less is more”), Emily Dickinson, Edgar Allan Poe, Henry James, Ernest Hemingway, and Samuel Beckett to Frederick Barthelme, Ann Beattie, Bobbie Ann Mason, and Raymond Carver, not to mention the rationalism of a certain mathematician and philosopher: “[A]fter the excesses of scholasticism comes Descartes’s radical reductionism — let us doubt and discard everything not self-evident and see whether anything indubitable remains upon which to rebuild” [Barth 1986, 1]. Modernist painter, John D. Graham uses reduction to define minimalism in the visual arts: “Minimalism is the reducing of painting to the minimum ingredients for the sake of discovering the ultimate, logical destination of painting in the process of abstracting” [Graham 1971, 33]. Marc Botha acknowledges Graham as well as critics Frances Colpitt and Cynthia W. Hallett, in his transhistorical and transdisciplinary theory of minimalism, where he frames reduction as a minimalist practice and concept: “The minimalist story . . . speaks of a simplicity arrived at through the disciplined process of reduction” [Botha 2017, 78] [Graham 1971] [Hallett 1996]. From a more situated perspective, James Meyer contextualizes reduction, linking it to the new austerity movement of the mid-1960s: “The popular concept of the minimal as an aesthetic of refusal, of reduction, became so omnivorous that it quickly overtook the practices it was first applied to” [Meyer 2001, 80]. Hal Foster acknowledges this historical prominence of reduction in definitions of minimalist art, only to the object, calling reduction a “great misreading” [Foster 1996, 40]. Meanwhile, Hartmut Obendorf anchors his human-computer interaction (HCI) methodology for “designing simplicity” and “designing the minimal” in a pragmatist reduction resonating with the economy of Occam’s razor: “The practitioner should be able to consciously choose his/her tools and make informed design decisions To this end, this book defines an ideal for design, focuses on reduction as a technique, and draws on the notion of minimalism to differentiate understanding of simplicity” [Obendorf 2009, 4]. Invested more in ideas and lifestyle minimalism than computing or interaction design, the first chapter of Kyle Chayka’s *The Longing for Less: Living with Minimalism* is titled, “Reduction,” where the term serves as a starting point for understanding minimalism and what else it may mean or do [Chayka 2020].

[23] Chave draws from Michel Foucault’s work to make this argument, but she accompanies that influence with a critique of Foucault’s inattention to gendered and patriarchal power in the process: “A persuasive case can be made, after all, that the patriarchal overevaluation of power and control — at the expense of mutuality, toleration, or nurturance — can be held to account for almost all that is politically

reprehensible and morally lamentable in the world. The case can be made as well that what is most badly needed are, at least for a start, visions of something different, something else” [Chave 1990, 56] [Balbus 1986] [Foucault 1980].

[24] Cameron Glover extends this observation in her work on contemporary lifestyle minimalism. She writes: “While minimalists preach universality — you don’t have to be rich to own fewer things — the authors behind the most influential minimalist-themed titles and #minimalism posts tell a different story. Influencers that frequently appear on lists of influential minimalist Instagram accounts are predominantly white and East Asian (today’s Japanese minimalism arises from Zen and Buddhist traditions). The best known minimalist authors today — Marie Kondo, Joshua Fields Millburn and Ryan Nicodemus, Francine Jay, Joshua Becker, Fumio Sasaki — fit within the same demographics” [Glover 2017]. She then shifts to highlight minimalist work being done by Black minimalists: “Despite the new movement’s lack of diversity on the surface, there are a small number of black individuals who proudly call themselves minimalists in 2017” [Glover 2017]. Among them is Roe Cummings, who defines minimalism this way: “Minimalism, if defined to me, is the practice of looking around at what you have and living your abundant life in the now” [Glover 2017].

[25] Hal Foster argues in “The Crux of Minimalism” that “minimalism considers perception in phenomenological terms, as somehow before or outside history, language, sexuality, and power. In other words, it does not regard the subject as a sexed body positioned in a symbolic order any more than it regards the gallery or the museum as an ideological apparatus” [Foster 1996, 43].

[26] The urination may also be explained by the small degree of public privacy afforded by *Titled Arc* as a sort of shield installed in Federal Plaza. Meanwhile, employees of the Department of Health and Human Services noted that the graffiti tended to include racial slurs and religious epithets [Hornblower 1985].

[27] These and other UVic collections may be viewed online at <https://vault.library.uvic.ca>.

[28] The history of women in computing, or women as computers or operators, is foundational to understanding the nuances of technical labor as, or aligned with, invisible, uncompensated, and/or unattributed activities of reproduction [Plant 1997] [Hayles 2005] [Chun 2013] [Edwards and Harris 2017] [Hicks 2017]. We have also written about one moment in this history (the optophonics work of Mary Jameson) with Mara Mills [Chan et al. 2018].

[29] For a history of one such open-source microcontroller (Arduino) and the relation of its design to power, control, and subjectivity, see Shaun Macpherson’s “‘A Computer for the Rest of You’: Human-Computer Interaction in the Eversion” [Macpherson 2014].

[30] The code for Hyku is available on GitHub at <https://github.com/samvera/hyku>. It is described thusly: “Hyku: A multi-tenant Hyrax application built on the latest and greatest Samvera community components. Brought to you by the Hydra-in-a-Box project partners and IMLS; maintained by the Hyku Interest Group.”

[31] For more on the notion of a black box, see *Science in Action*, where Bruno Latour uses the term to differentiate between “ready made science” and “science in the making”: “The word black box is used by cyberneticians whenever a piece of machinery or a set of commands is too complex. In its place they draw a little box about which they need to know nothing but its input and output” [Latour 1988, 2–3]. The black box is abstract, not historical. Later in *Science in Action*, Latour also describes it as that which “closely resembles an organised whole” [Latour 1988, 131].

[32] For more on Wax, see <https://minicomp.github.io/wax/>. There, the credits for Wax read as follows: “Wax is a minimal computing (minicomp) project led by Marii Nyröp. The project is currently maintained by Marii Nyröp and Alex Gil at Columbia University Libraries. It uses open source libraries and frameworks including Jekyll, IIIF, OpenSeaDragon, Rake, and ElasticLunr. Wax builds upon work by Peter Binkley, David Newbury, and others.”

[33] The stack for Vault includes Hyku, Fedora 4, Blacklight, Solr, and IIIF, among other technologies.

[34] This logic was central to early work on the Scalar platform for multimodal scholarly communication (see <https://scalar.me/anvc/>). Sayers published an article on Scalar with Craig Dietrich, noting that Scalar’s “authors are not creating small, isolated archives on the Scalar server. Although they can upload their own videos, audio, and images to that space, they are instead encouraged to house them with a partner archive. Or, in the case where assets are already online, they are encouraged to embed those assets in their Scalar projects. That way, systems point to existing URIs rather than duplicating resources and producing redundancies. Here, the advantage is that media playback is overseen by groups that not only have institutional support but also specialise in metadata, asset categorisation, provenance, and interoperability” [Sayers and Dietrich 2013].

[35] See [Posner 2012] on the problem with exhorting people, especially white women and people of color, to code.

[36] For more on common sense, Stuart Hall and Alan O'Shea describe common sense as "a form of 'everyday thinking' which offers us frameworks of meaning with which to make sense of the world. It is a form of popular, easily-available knowledge which contains no complicated ideas, requires no sophisticated argument and does not depend on deep thought or wide reading. It works intuitively, without forethought or reflection. It is pragmatic and empirical, giving the illusion of arising directly from experience, reflecting only the realities of daily life and answering the needs of 'the common people' for practical guidance and advice" [Hall and O'Shea 2013, 8–9]. They add that "Common sense, tends to be socially conservative, leaning toward tradition" [Hall and O'Shea 2013, 9]. Consider, for instance, the treatment of computing practices as something you simply "do" or "keep doing." Without attention to the labor perspective (among other social and cultural approaches), the minimalist dimensions of minimal computing lend themselves to demands for the easily-available, intuitive, pragmatic, empirical, and socially conservative. On common sense, see also the work of Antonio Gramsci as well Linda Åhäll, who considers common sense to be a vehicle for apolitical activity [Gramsci 1971] [Åhäll 2018]. For more on common sense as *sensus communis* (shared sensation) and aesthetic rationality, see [Chuh 2019]. For critical responses to the socially conservative dimensions of computing and digital work, see #TransformDH at <https://transformdh.org>.

[37] The Canadian Union of Public Employees (CUPE) wrote in 2018: "More Canadians are working in precarious conditions, employed in contract, temporary, and/or part-time jobs with low wages. The increase has been concentrated in accommodation and food services, education, information, culture and recreation services — and particularly among young workers aged 15 to 24 and older workers aged 65+. The [Canadian Centre for Policy Alternatives] recently reported that among the university and college workforce, as many as half are employed in precarious conditions. A disproportionate share of those are the 68,000 CUPE workers employed in this sector. Permanent jobs make up a declining share of the overall jobs in the post-secondary workforce. They're being replaced by people working in temporary, involuntary part-time and multiple jobs" [CUPE 2018]

[38] Another description of the apparatus may be useful here. In *Meeting the Universe Halfway*, Karen Barad writes: "(1) apparatuses are specific material-discursive practices (they are not merely laboratory setups that embody human concepts and take measurements); (2) apparatuses produce differences that matter — they are boundary-making practices that are formative of matter and meaning, productive of, and part of, the phenomena produced; (3) apparatuses are material configurations/dynamic reconfigurings of the world; (4) apparatuses are themselves phenomena (constituted and dynamically reconstituted as part of the ongoing intra-activity of the world); (5) apparatuses have no intrinsic boundaries but are open-ended practices; and (6) apparatuses are not located in the world but are material configurations or reconfigurings of the world that re(con)figure spatiality and temporality as well as (the traditional notion of) dynamics (i.e., they do not exist as static structures, nor do they merely unfold or evolve in space and time)" [Barad 2007, 146]. Barad approaches the apparatus from a science studies perspective and builds on work by Haraway, Niels Bohr, Judith Butler, and Foucault. This perspective distinguishes the apparatus as defined by structuralists such as Louis Althusser.

[39] See Chun's *Programmed Visions* for more on this issue of not seeing or grasping computation (Chun 2011). For more on how it applies to maintenance, such as the carework of moderation, see Sarah Roberts's *Behind the Screen: Content Moderation in the Shadows of Social Media* [Roberts 2019]. For more from a game studies perspective, specifically the invisible labor at play in occupations such as goldfarming, see Lisa Nakamura's "Don't Hate the Player, Hate the Game: The Racialization of Labor in World of Warcraft" [Nakamura 2009]. The invisibility of such labor practices is directly tied to worker alienation; those who are not seen are not only deprived of recognition and attribution but also refused fair wages and subjected to exploitation.

[40] See <https://www.uvic.ca/library/research-teaching/research-grant-support/grants.php> for the Libraries' "Grant menu." For more on in-kind contributions in a Canadian context, see the Social Science and Humanities Research Council's guidelines: https://www.sshrc-crsh.gc.ca/funding-financement/policies-politiques/cash_inkind-especes_en_nature-eng.aspx.

[41] For more on the gender and power dimensions of service in projects, see Susan Brown's "Delivery Service: Gender and the Political Unconscious of Digital Humanities" [Brown 2018].

[42] In "On Being Directed," Ahmed explains how technologies, or things, play a role in naturalizing power as directionality:

As you become aware of how the social world is organized, norms appear as palpable things. I think of those times, say, when you walk into a toy shop and it is striking. You might pick up the vacuum cleaner, a toy vacuum cleaner, and feel like you are holding the future for girls in a tangible thing. You can pick up a toy gun, and also feel this: the future for boys held as a tangible thing. Norms become striking: holdable as palpable things. Once we are stricken, there is still much work left to do. The hardest work can be recognizing how one's own life is shaped by norms in ways that we did not realize, in ways that cannot simply be transcended. A norm is also a way of living, a way of connecting with others over or around something. We cannot 'not' live in relation to norms [Ahmed 2017, 43].

[43] Ahmed explains "how feminism can be experienced as life alienation, how we can become estranged from the lives we are living in the very process of recognizing how our lives have been shaped or have taken shape" [Ahmed 2017, 43]. Later in the book, she also develops the

notion of being an “affect alien” [Ahmed 2017, 57].

[44] See, for instance, Evans’s *Help! I’ve Got a Manager!* as well as *A People’s Guide to AI* by Onuoha and Mother Cyborg [Evans 2018] [Onuoha and Mother Cyborg 2018]. Elsewhere, in their introduction to *Bodies of Information*, Wernimont and Losh unpack an acronym (MEALS) that could easily serve as a foundation for minimal computing from the labor perspective: “In our work we use the acronym MEALS as shorthand for a feminist emphasis on how the ‘material, embodied, affective, labor-intensive, and situated character of engagements with computation can operate experientially for users in shared spaces’” [Wernimont and Losh 2018a]. Also see their essay, “Wear and Care: Feminisms at a Long Maker Table” [Wernimont and Losh 2018a]. Elsewhere, Anderson et al. “recommend that affective labour be formally recognized, through: Acknowledgement on project, course, and department websites. Acknowledgement in publications and other forms of dissemination. Evaluation for tenure and promotion. Evaluation for grants and other forms of funding” [Anderson et al. 2016].

[45] For an example of such heroification, see [Russell and Vinsel 2016]. Even if well-intended, heroification of maintainers and careworkers risks furthering their alienation, masking systemic problems, and ignoring labor conditions. Bengi Akbulut writes: “What is largely missing from the celebration of care as the cornerstone of the post-growth transition is how carework is to be organized in a socio-ecologically just future. This is crucial, since re-centering a society around care does not imply gender justice. Quite the contrary, carework has historically been one of the most exploitative, flexible and invisible forms of labor performed by women” [Akbulut 2017].

[46] In this quoted text, Glover and Gil encode a link to Dennis Tenen and Grant Wythoff’s “Sustainable Authorship in Plain Text using Pandoc and Markdown” [Tenen and Wythoff 2014].

[47] *archipelagos* also has a pedagogical dimension in this regard. Gil noted in email communications with us that contributing graduate students learn about the technical labor involved in scholarly editing and publishing. They are compensated for that labor, with attribution, along the way.

[48] Although popular among researchers and academics, GitHub’s politics, and the politics of using it, have been subject to scrutiny given the platform company’s contract with U.S. Immigration and Customs Enforcement (ICE). Kevin Truong with Vice News reports: “The Microsoft-owned software development platform company continues to contract with ICE, despite CEO Nat Friedman sharing in a blog post last year that he strongly disagrees with the Trump administration’s immigration policies” [Truong 2020].

[49] Here we are drawing from Walter Benjamin’s treatment of Paul Klee’s *Angelus Novus* in the ninth thesis “On the Concept of History” [Benjamin 1940].

[50] See John D. Martin and Carolyn Runyon’s work for an analysis of how digitization projects are prioritized through funding. They write: “Of the projects analyzed in this study, 26 focused on digitizing the work and intellectual legacies of individual people. Of these, only one woman was singled out for individual treatment: the ‘Ida M. Tarbell Papers Digitization Project,’ awarded \$30,000. Similarly, only one African-American was at the center of a project: ‘Digitizing W.E.B. Du Bois,’ awarded \$314,787. This means that projects on individual women and black Americans were awarded only 8% of the total \$4,225,061 awarded to projects on individuals. All of the rest focused on white men of historical importance. Several, such as Walt Whitman and Thomas Jefferson, had multiple projects representing them” [Martin and Runyon 2016, 25–26]. They add: “While white men were likely to be treated as individuals in a given project, other race/ethnic categories and women were treated as groups almost exclusively. Instead of a project focusing on specific historical figures, the narrative and documentary history of these groups is considered at the aggregate level. This disparity is important to note because it speaks to a larger social phenomenon whereby great (white) men stand out for their achievements, but other groups have been largely left to be remembered for their collective struggle” [Martin and Runyon 2016, 26]. See also the work of Martha Nell Smith, Moya Z. Bailey, Tara McPherson, Kim Gallon, and Christina Boyles [Smith 2007] [Bailey 2011] [McPherson 2012] [Gallon 2016] [Boyles 2018].

[51] Automation is always about labor and alienation: what work people do and do not want to do, which work they wish to protect, and how the quality and pleasure of work changes with a decrease in manual contribution.

[52] For more on planned obsolescence, see Giles Slade’s *Made to Break: Technology and Obsolescence in America* and Kathleen Fitzpatrick’s *Planned Obsolescence: Publishing, Technology, and the Future of the Academy* [Slade 2006] [Fitzpatrick 2011]. The history of planned obsolescence in America follows a capitalist growth paradigm.

[53] As part of the *Lift Every Voice and Sing* project, Donaldson links ephemerality, memory, and minimal computing with Black lives and Black digital humanities: “In this moment, one in which we are forced to confront our own fragility, our own ephemerality as (potentially) disappearing bodies amid a global pandemic, minimal computing impels us toward a reassessment of our individual and communal practices and makes apparent the ways we might reject circulating notions of a ‘new normal’ as we think about the potential to reclaim and reshape community. What

might that look like? What is sustainable? What is not? What do we need?" [Donaldson 2020]. Elsewhere, Sample asks: "[M]ust everything be permanent? Must we insist that every cultural object be subjected to the archive?" [Sample 2010]. Sayers also speaks to ephemerality in the context of minimal computing, which may be understood as "maximum ephemerality": "Reduce an impulse to inscribe, measure, or visualize with technologies in order to increase the likelihood of experimentation and collective participation" [Sayers 2016].

[54] See <https://projectendings.github.io>. Their website includes their endings principles as well as resources related to preserving digital projects. The Endings Project team consists of Claire Carlin, Ewa Czaykowska-Higgins, Janelle Jenstad, Elizabeth Grove-White, Corey Davis, John Durno, Lisa Goddard, J. Matthew Huculak, Martin Holmes, Stewart Arneill, Greg Newton, Emily Comeau, Sarah Kell, Daniel Martin, Tye Landels-Gruenewald, Jennifer Polack, and Joseph Takeda.

[55] The HCMC's "endings principles" would make for an interesting comparison with *archipelagos*, especially on the topics of which programming languages to use (XSLT versus Ruby, for instance) and whether to commit to extensive markup (such as TEI XML) for the purposes of longevity.

[56] See also Joseph Takeda's "Ending Your Digital Humanities Project from the Start" [Takeda 2018].

Works Cited

#TransformDH *#TransformDH*. Accessed October 14, 2021. <https://transformdh.org/>.

Åhäll 2018 Åhäll, Linda. "Affect as Methodology: Feminism and the Politics of Emotion." *International Political Sociology* vol. 12.1 (2018): 36–52. <https://doi.org/10.1093/ips/olx024>.

Ahmed 2017 Ahmed, Sara. *Living a Feminist Life*. Durham, NC: Duke University Press, 2017.

Akbulut 2017 Akbulut, Bengi. "Carework as Commons: Towards a Feminist Degrowth Agenda." *Degrowth.info*, February 2, 2017. <https://degrowth.info/en/blog/carework-as-commons-towards-a-feminist-degrowth-agenda-2>.

Anderson et al. 2016 Anderson, Katrina. et al. "Student Labour and Training in Digital Humanities." *Digital Humanities Quarterly* vol. 10.1 (2016). <http://www.digitalhumanities.org/dhq/vol/10/1/000233/000233.html>.

Aneesh 2006 Aneesh, A. *Virtual Migration: The Programming of Globalization*. Durham, NC: Duke University Press, 2006.

Bailey 2011 Bailey, Moya Z. "All the Digital Humanists Are White, All the Nerds Are Men, but Some of Us Are Brave." *Journal of Digital Humanities* vol. 1.1 (2011). <http://journalofdigitalhumanities.org/1-1/all-the-digital-humanists-are-white-all-the-nerds-are-men-but-some-of-us-are-brave-by-moya-z-bailey/>.

Balbus 1986 Balbus, Isaac D. "Disciplining Women: Michel Foucault and the Power of Feminist Discourse," *Praxis International* vol. 5.4 (1986): 466–483.

Barad 2007 Barad, Karen. *Meeting the Universe Halfway: Quantum Physics and the Entanglement of Matter and Meaning*. Durham, NC: Duke University Press, 2007.

Barca 2019 Barca, Stefania. "The Labor(s) of Degrowth." *Capitalism Nature Socialism* vol. 30.2 (2019): 207–216. <https://doi.org/10.1080/10455752.2017.1373300>.

Barr 2018 Barr, Adam. *The Problem with Software: Why Smart Engineers Write Bad Code*. Cambridge, MA: MIT Press, 2018.

Barth 1986 Barth, John. "A Few Words about Minimalism." *New York Times*, December 28, 1986. <https://archive.nytimes.com/www.nytimes.com/books/98/06/21/specials/barth-minimalism.html>.

Benjamin 1940 Benjamin, Walter. "On the Concept of History", trans. H. Zohn. In *Selected Writings*, vol. 4, 1938–1940, edited by Eiland, Howard and Michael W. Jennings, 389–400. Cambridge, MA: Belknap Press of Harvard University Press 2003.

Botha 2017 Botha, Marc. *A Theory of Minimalism*. NY: Bloomsbury, 2017.

Boyles 2018 Boyles, Christina. "Counting the Costs: Funding Feminism in the Digital Humanities." In *Bodies of Information: Intersectional Feminism and Digital Humanities*, edited by Jacqueline Wernimont and Elizabeth M. Losh, 93–107. Minneapolis: University of Minnesota Press, 2018. <https://dhdebates.gc.cuny.edu/read/untitled-4e08b137-aec5-49a4-83c0-38258425f145/section/6a48cd20-cfa5-4984-ba32-f531b231865f#ch07>.

Brown 2018 Brown, Susan. "Delivery Service: Gender and the Political Unconscious of Digital Humanities." In *Bodies of Information: Intersectional Feminism and Digital Humanities*, edited by Jacqueline Wernimont and Elizabeth M. Losh,

261–286. Minneapolis: University of Minnesota Press, 2018. <https://dhdebates.gc.cuny.edu/read/untitled-4e08b137-aec5-49a4-83c0-38258425f145/section/7c07a9eb-fbc7-4b69-8777-82676a3c64ab#ch15>.

- Brown et al. 2009** Brown, Susan. "Published Yet Never Done: The Tension Between Projection and Completion in Digital Humanities Research." *Digital Humanities Quarterly* vol. 3.2 (2009). <http://www.digitalhumanities.org/dhq/vol/3/2/000040/000040.html>.
- CUPE 2018** Canadian Union of Public Employees (CUPE). "Precarious Work on the Rise." CUPE, March 15, 2018. <https://cupe.ca/precarius-work-rise>.
- Chachra 2015** Chachra, Debbie. "Why I Am Not a Maker." *The Atlantic*, January 23, 2015. <https://www.theatlantic.com/technology/archive/2015/01/why-i-am-not-a-maker/384767/>.
- Chan et al. 2018** Chan, Tiffany, Mara Mills, and Jentery Sayers. "Optophonetic Reading, Prototyping Optophones." *Amodern* 8 (2018). <https://amodern.net/article/optophonetic-reading/>.
- Chave 1990** Chave, Anna. "Minimalism and the Rhetoric of Power." *Arts Magazine* vol. 64.5 (1990): 44-63.
- Chayka 2020** Chayka, Kyle. *The Longing for Less: Living with Minimalism*. NY: Bloomsbury, 2020.
- Chuh 2019** Chuh, Kandice. *The Difference Aesthetics Makes: On the Humanities "After Man."* Durham, NC: Duke University Press, 2019.
- Chun 2013** Chun, Wendy Hui Kyong. *Programmed Visions: Software and Memory*. Cambridge, MA: MIT Press, 2013. <https://doi.org/10.1177/1461444811429927d>.
- Colpitt 1990** Coplitt, Frances. *Minimal Art: The Critical Perspective*. Ann Arbor, MI: University of Michigan Press, 1990.
- Colvard 2021** Colvard, Chris, Brian Hole, and Kevin Kochanski. "Hyku Implementations." Accessed October 29, 2021. <https://wiki.lyrasis.org/display/hyku/Hyku+Implementations>.
- Davis 2014** Davis, Corey. "Archiving the Web: A Case Study from the University of Victoria." *Code4Lib Journal* 26 (2014). <https://journal.code4lib.org/articles/10015>.
- Dictionary of Computer Science 2016a** "Open-source." In *A Dictionary of Computer Science* (7th ed), edited by Andrew Butterfield, Gerard E. Ngondi, and Anne Kerr. Oxford, UK: Oxford University Press, Oxford, 2016.
- Dictionary of Computer Science 2016b** "Saas (Software as a Service)." In *A Dictionary of Computer Science* (7th ed.), edited by Andrew Butterfield, Gerard E. Ngondi, and Anne Kerr. Oxford UK: Oxford University Press, 2016.
- Dictionary of Information Science and Technology 2013** "Software as a Service (SaaS)." In *Dictionary of Information Science and Technology* (2nd Edition), edited by Mehdi Khosrow-Pour, 827. Hershey, PA: IGI Global, 2013. <https://doi.org/10.4018/978-1-4666-2624-9>.
- Dijkstra 1972** Dijkstra, Edsger W. "The Humble Programmer." 1972 ACM Turing Award Lecture, Communications of the Association for Computing Machinery, October 1972. <https://doi.org/10.1145/355604.361591>.
- Dohe 2019** Dohe, Kate. "Care, Code, and Digital Libraries: Embracing Critical Practice in Digital Library Communities." *In the Library with the Lead Pipe*, 2019. <http://www.inthelibrarywiththeleadpipe.org/2019/digital-libraries-critical-practice-in-communities/>.
- Donaldson 2020** Donaldson, Sonya. "Singing the Nation: On Memory, Ephemerality, and the Minimal Computing Model." *Lift Every Voice and Sing*, 2020. <https://elotroalex.github.io/lift/posts/singing-the-nation/>.
- Edwards and Harris 2017** Edwards, Sue B. and Duchess Harris. *Hidden Human Computers: The Black Women of NASA*. North Mankato, MN: Abdo Publishing, 2017.
- Encyclopedia Britannica 1998** Britannica, The Editors of Encyclopedia. "Minimalism." Encyclopedia Britannica, 20 July 1998, <https://www.britannica.com/art/Minimalism>. Accessed 9 September 2022.
- Engel et al. 2018** Engel, Deena, Lauren Hinkson, Joanna Phillips, and Marion Thain. "Reconstructing *Brandon* (1998-1999): A Cross-Disciplinary Digital Humanities Study of Shu Lea Cheang's Early Web Artwork." *Digital Humanities Quarterly* vol. 12.2 (2018). <http://www.digitalhumanities.org/dhq/vol/12/2/000379/000379.html>.
- Ensmenger 2014** Ensmenger, Nathan. "When Good Software Goes Bad: the Surprising Durability of an Ephemeral Technology." The Maintainers Conference, Hoboken, New Jersey, April 9, 2014. <https://homes.luddy.indiana.edu/nensmeng/files/ensmenger-mice.pdf>.

- Evans 2018** Evans, Julia. *Help! I've Got a Manager*. Wizard Zines. <https://wizardzines.com/zines/manager>.
- Ferguson and Folbre 2000** Ferguson, Anne and Nancy Folbre. "Women, Care and the Public Good: A Dialogue." In *Not for Sale: In Defense of Public Goods*, edited by Anton Anatole, Milton Fisk, and Nancy Holmstrom, 95–108. Boulder: Westview Press, 2000.
- Fitzpatrick 2011** Fitzpatrick, Kathleen. *Planned Obsolescence: Publishing, Technology, and the Future of the Academy*. NY: New York University Press, 2011.
- Foster 1996** Foster. Hal. *Return of the Real: The Avant-Garde at the End of the Century*. Cambridge, MA: MIT Press, 1996.
- Foucault 1980** Foucault, Michel. *Power/Knowledge: Selected Interviews and Other Writings, 1972-1977*, Colin Gordon (ed). NY: Pantheon Books, 1980.
- Frost 2015** Frost, Hannah. "Community Input, Plus Thoughts on Customization and Configuration." *Hyku, Samvera Community*, December 16, 2015. hyku.samvera.org/2015/12/16/community_input.html.
- Fuchs 2014** Fuchs, Christian. *Digital Labour and Karl Marx*. NY: Routledge, New York, 2014.
- Fuller 2005** Fuller, Matthew. *Media Ecologies: Materialist Energies in Art and Technoculture*. Cambridge, MA: MIT Press.
- GNU 2021** GNU. "What Is GNU?" *GNU Operating System*, August 7, 2020. <https://www.gnu.org/home.en.html>.
- Gallon 2016** Gallon, Kim. "Making a Case for the Black Digital Humanities." In *Debates in the Digital Humanities 2017*, edited by Matthew K. Gold and Lauren F. Klein, 42–49. Minneapolis: University of Minnesota Press, 2016. <https://dhdebates.gc.cuny.edu/read/untitled/section/fa10e2e1-0c3d-4519-a958-d823aac989eb>.
- Gil 2015** Gil, Alex. "The User, the Learner, and the Machines We Make." *GO::DH Minimal Computing Working Group*, May 21, 2015. <https://go-dh.github.io/mincomp/thoughts/2015/05/21/user-vs-learner/>.
- Gilbert and Mobley 2013** Gilbert, Heather and Tyler Mobley. "Breaking Up with CONTENTdm: Why and How One Institution Took the Leap to Open Source." *Code4Lib Journal* 20 (2013). <https://journal.code4lib.org/articles/8327>.
- Glover 2017** Glover, Cameron. "Is Minimalism for Black People?" *Pacific Standard*, November 23, 2017. <https://psmag.com/social-justice/is-minimalism-for-black-pepo>.
- Glover and Gil 2020** Glover, Kaiama L. and Alex Gil. "About Us." *archipelagos — a journal of Caribbean digital praxis*. Accessed September 29, 2021. <http://archipelagosjournal.org/about.html>.
- Gold 2009** Gold, Nicolas. "Service-Oriented Software in the Humanities: A Software Engineering Perspective." *Digital Humanities Quarterly* vol. 3.4 (2009). <http://digitalhumanities.org/dhq/vol/3/4/000072/000072.html>.
- Graham 1971** Graham, John. *System and Dialectics of Art*. Baltimore: Johns Hopkins University Press, 1971. Originally published in 1937.
- Gramsci 1971** Gramsci, Antonio. *Selections from the Prison Notebooks of Antonio Gramsci*, translated by Quentin Hoare and Geoffrey Nowell-Smith. NY: International Publishers, 1971.
- HCMC 2020** Humanities Computing and Media Centre (HCMC). *The Endings Project*, August 8, 2020. <https://endings.uvic.ca>.
- Hall and O'Shea 2013** Hall, Stuart and Alan O'Shea. "Common-sense Neoliberalism: The Battle over Common Sense Is a Central Part of Our Political Life." *Soundings* 55 (2013): 8-24.
- Hallett 1996** Hallett, Cynthia J. "Minimalism and the Short Story." *Studies in Short Fiction* vol. 33.4 (1996): 487-495.
- Hardesty and Homenda 2019** Hardesty, Juliet and Nicholas Homenda. "The Ecosystem of Repository Migration." *Publications* vol. 7.1 (2019). <https://doi.org/10.3390/publications7010016>.
- Hayles 2005** Hayles, N. Katherine. *My Mother Was a Computer: Digital Subjects and Literary Texts*. Chicago: University of Chicago Press, 2005.
- Hicks 2017** Hicks, Mar. *Programmed Inequality: How Britain Discarded Women Technologists and Lost Its Edge in Computing*. Cambridge, MA: MIT Press, 2017.
- Hornblower 1985** Hornblower, Margot. "New Yorkers, Artists Tilt Over 'Arc.'" *Washington Post*, March 7, 1985. <https://www.washingtonpost.com/archive/lifestyle/1985/03/07/new-yorkers-artists-tilt-over-arc/a9132351-a9cb-472b->

- Illich 1973** Illich, Ivan. *Tools for Conviviality*. NY: Harper & Row, 1973.
- Jackson 2014** Jackson, Steven J. "Rethinking Repair." In *Media Technologies: Essays on Communication, Materiality and Society*, edited by Tarleton Gillespie, Pablo J. Boczkowski, and Kristen A. Foot, 221–240. Cambridge, MA: MIT Press, 2014.
- Judd 2002** Judd, Donald. "Specific Objects." In *Donald Judd, Early Work 1955-1968*, Kellein, Thomas (ed). NY: Distributed Art Pub Inc, 2002.
- Jullien 2009** Jullien, Nicolas. "A Historical Analysis of the Emergence of Free Cooperative Software Production." In *Encyclopedia of Multimedia Technology and Networking*, edited by Margherita Pagani, 605–612. Hershey, PA: IGI Global, 2009. <http://doi:10.4018/978-1-60566-060-8.ch001>.
- Kirschenbaum 2008** Kirschenbaum, Matthew *Mechanisms: New Media and the Forensic Imagination*. Cambridge, MA: MIT Press, 2008.
- Kirschenbaum 2009** Kirschenbaum, Matthew. "Done: Finishing Projects in the Digital Humanities" *Digital Humanities Quarterly* vol. 3.2 (2009). <http://www.digitalhumanities.org/dhq/vol/3/2/000037/000037.html>.
- Knuth 1968** Knuth, Donald. *The Art of Computer Programming, Volume 1: Fundamental Algorithms*. Boston: Addison-Wesley, 1968.
- Knuth 1974** Knuth, Donald. "Structured Programming with Go To Statements." *ACM Computing Surveys* vol. 6.4 (1974). <https://doi.org/10.1145/356635.356640>.
- Kretzschmar, Jr. 2009** Kretzschmar, Jr., William A. "Large-Scale Humanities Computing Projects: Snakes Eating Tails, or Every End is a New Beginning?" *Digital Humanities Quarterly* vol. 3.2 (2009). <http://www.digitalhumanities.org/dhq/vol/3/2/000038/000038.html>.
- Kriss 2019** Kriss, J. "Minimal Computing." *Jesse Kriss*, January 31, 2019. <https://tilde.tinyserver.club/~jkriss/writing/minimal-computing>.
- Latour 1988** Latour, Bruno. *Science in Action: How to Follow Scientists and Engineers through Society*. Cambridge, MA: Harvard University Press, 1987.
- Lientz and Swanson 1980** Lientz, Bennet P. and E. Burton Swanson (eds). *Software Maintenance Management*. Boston: Addison-Wesley Publishing, 1980.
- Macpherson 2014** Macpherson, Shaun. "A Computer for the Rest of You': Human-Computer Interaction in the Eversion." M.A. thesis, 2014. University of Victoria.
- Martin and Runyon 2016** Martin, John D. and Carolyn Runyon. "Digital Humanities, Digital Hegemony: Exploring Funding Practices and Unequal Access in the Digital Humanities." *ACM SIGCAS Computers and Society* vol. 46.1 (2016). <https://doi.org/10.1145/2908216.2908219>.
- Marx 1976** Marx, Karl. *Capital: A Critique of Political Economy*, translated by Ben Fowkes. NY: Vintage, 1976.
- Mattern 2018** Mattern, Shannon. "Maintenance and Care." *Places Journal* (2018). <https://doi.org/10.22269/181120>.
- McMillan Cottom 2017** McMillan Cottom, Tressie. *Lower Ed: The Troubling Rise of For-Profit Colleges in the New Economy*. NY: The New Press, 2018.
- McPherson 2012** McPherson, Tara. "Why Are the Digital Humanities So White? or Thinking the Histories of Race and Computation." In *Debates in the Digital Humanities*, edited by Matthew K. Gold, 139-160. Minneapolis: University of Minnesota Press, 2012. <https://dhdebates.gc.cuny.edu/read/untitled-88c11800-9446-469b-a3be-3fdb36bfd1e/section/20df8acd-9ab9-4f35-8a5d-e91aa5f4a0ea>.
- Meyer 2001** Meyer, James. *Minimalism: Art and Polemics in the Sixties*. New Haven: Yale University Press, 2004.
- Mita et al. 2018** Mita, Amanda, Zachary Pelli, Kimberly Reamer, and Sharon Ince. "CONTENTdm to Digital Commons: Considerations and Workflows." *Journal of Archival Organization* 15 (2018): 58-70. <https://doi.org/10.1080/15332748.2019.1609308>.
- Moraga and Anzaldúa 1983** Moraga, Cherríe and Gloria Anzaldúa. *This Bridge Called My Back: Writings by Radical Women of Color*. Latham, NY: Kitchen Table, Women of Color Press, 1983.

- Murphy 2012** Murphy, Michelle. *Seizing the Means of Reproduction: Entanglements of Feminism, Health, and Technoscience*. Durham, NC: Duke University Press, 2012.
- Nakamura 2009** Nakamura, Lisa. "Don't Hate the Player, Hate the Game: The Racialization of Labor in World of Warcraft." *Critical Studies in Media Communication* vol. 26.2 (2009): 128-44.
- Obendorf 2009** Obendorf, Hartmut. *Minimalism: Designing Simplicity*. London: Springer-Verlag, 2009.
- Onuoha and Mother Cyborg 2018** Onuoha, Mimi and Mother Cyborg. *A People's Guide to AI*. Detroit, Allied Media Projects, 2018.
- Oroza 2006** Oroza, Ernesto. *For an Architecture of Necessity and Disobedience*. Self-published (2006). <http://architectureofnecessity.com>.
- Oxford 2007** "Minimalism." In *The Concise Oxford English Dictionary of Music*, edited by Michael Kennedy and Joyce B. Kennedy. Oxford: Oxford University Press, 2007. <https://doi.org/10.1093/acref/9780199203833.001.0001>
- PBS 1999** "Richard Serra's *Tilted Arc*." *Culture Shock*. WGBH Interactive and the Public Broadcasting Service (PBS), 1999. <https://www.pbs.org/wgbh/cultureshock/flashpoints/visualarts/tiltedarc.html>.
- Plant 1997** Plant, Sadie. *Zeroes and Ones: Digital Women and the New Technoculture*. NY: Doubleday, 1997.
- Posner 2012** Posner, Miriam. "Some Things to Think about before You Exhort Everyone to Code." *Miriam Posner's Blog*, February 29, 2012. <https://miriamposner.com/blog/some-things-to-think-about-before-you-exhort-everyone-to-code/>.
- Risam 2018** Risam, Roopika. "What Passes for Human?: Undermining the Universal Subject in Digital Humanities Praxis." In *Bodies of Information: Intersectional Feminism and Digital Humanities* edited by Jacqueline Wernimont and Elizabeth M. Losh, 39-56. Minneapolis: University of Minnesota Press, 2018. <https://dhdebates.gc.cuny.edu/read/untitled-4e08b137-aec5-49a4-83c0-38258425f145/section/34d51cdb-2a89-4e4b-9762-bf6461cf0bb7#ch03>.
- Roberts 2019** Roberts, Sarah T. *Behind the Screen: Content Moderation in the Shadows of Social Media*. New Haven: Yale University Press, 2019.
- Rosner and Ames 2014** Rosner, Daniela K. and Morgan G. Ames. "Designing for Repair? Infrastructures and Materialities of Breakdown." *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, Baltimore, Maryland, February 2014. <https://doi.org/10.1145/2531602.2531692>.
- Russell and Vinsel 2016** Russell, Andrew and Lee Vinsel. "Hail the Maintainers." *The Maintainers*, April 7, 2016. <https://aeon.co/essays/innovation-is-overvalued-maintenance-often-matters-more>.
- Sample 2010** Sample, Mark. "The Archive or the Trace: Cultural Permanence and the Fugitive Text." *Sample Reality*, January 18, 2010. <https://www.samplereality.com/2010/01/18/the-archive-or-the-trace-cultural-permanence-and-the-fugitive-text/>.
- Samvera 2021** "Samvera Code of Conduct and Anti-Harassment Policy," 2021. <https://samvera.org/home/samvera-code-of-conduct-and-anti-harassment-policy/>
- Sayers 2016** Sayers, Jentery. "Minimal Definitions (tl;dr version)." *GO::DH Minimal Computing Working Group*, October 3, 2016. <https://go-dh.github.io/mincomp/thoughts/2016/10/03/tldr/>.
- Sayers 2017** Sayers, Jentery. "Introduction: 'I Don't Know All the Circuitry.'" In *Making Things and Drawing Boundaries: Experiments in the Digital Humanities*, edited by Jentery Sayers, 1–18. Minneapolis: University of Minnesota Press, 2017. <https://dhdebates.gc.cuny.edu/read/untitled-aa1769f2-6c55-485a-81af-ea82cce86966/section/7d8fca82-c6ca-480f-bf17-1df4a2cdb577>.
- Sayers and Dietrich 2013** Sayers, Jentery and Craig Dietrich. "After the Document Model for Scholarly Communication: Some Considerations for Authoring with Rich Media." *Digital Studies/Le Champ Numérique* vol. 3.2 (2013). <http://doi.org/10.16995/dscn.237>.
- Scholz 2013** Scholz, Trebor (ed). *Digital Labor: The Internet as Playground and Factory*. NY: Routledge, 2013.
- Shirazi 2017** Shirazi, Roxanne. "Reproducing the Academy: Librarians and the Question of Service in the Digital Humanities." In *Making Things and Drawing Boundaries: Experiments in the Digital Humanities*, edited by Jentery Sayers, 86–94. Minneapolis: University of Minnesota Press, 2017.
- Slade 2006** Slade, Giles. *Made to Break: Technology and Obsolescence in America*. Cambridge, MA: Harvard University Press, 2006.

- Smith 2007** Smith, Martha Nell. "The Human Touch Software of the Highest Order: Revisiting Editing as Interpretation." *Textual Cultures* vol. 2.1 (2007): 1–15.
- Smithies 2011** Smithies, James. "A View from IT." *Digital Humanities Quarterly* vol. 5.3 (2011). <http://digitalhumanities.org/dhq/vol/5/3/000107/000107.html>.
- Smithies 2018** Smithies, James. "Full Stack DH: Building a Virtual Research Environment on a Raspberry Pi." In *Making Things and Drawing Boundaries: Experiments in the Digital Humanities*, edited by Jentery Sayers, 102-114. Minneapolis: University of Minnesota Press, 2018. <https://dhdebates.gc.cuny.edu/read/untitled-aa1769f2-6c55-485a-81af-ea82cce86966/section/859c8bb8-df9d-46e9-8372-334bbbb71926>.
- Stella 1995** "Bruce Glaser: Questions to Stella and Judd." In *Minimal Art: A Critical Anthology*, edited by Gregory Battcock, 148-164. Berkeley: University of California Press, 1995.
- Sterne 2007** Sterne, Jonathan. "Out with the Trash: On the Future of New Media." In *Residual Media*, edited by Charles R. Acland, 16-31. Minneapolis: University of Minnesota Press, 2007.
- Strickland 1993** Strickland, Edward. *Minimalism: Origins*. Bloomington, IN: Indiana University Press, 1993.
- Takeda 2018** Takeda, Joseph. "Ending Your Digital Humanities Project from the Start." *projectEndings*, October 2, 2018. https://github.com/projectEndings/Endings/tree/master/presentations/UBC_DH Mixer.
- Tenen and Wythoff 2014** Tenen, Dennis. and Grant Wythoff. "Sustainable Authorship in Plain Text using Pandoc and Markdown." *The Programming Historian* 3 (2014). <https://doi.org/10.46430/phen0041>.
- Truong 2020** Truong, Kevin. "The Open Source Community Is Calling on Github to 'Drop ICE.'" *Vice News: Motherboard*, July 20, 2020. https://www.vice.com/en_ca/article/m7jpgy/open-source-community-changing-github-avatars-drop-ice.
- Wark 2015** Wark, Mackenzie. *Molecular Red: Theory for the Anthropocene*. London: Verso, 2015.
- Wernimont 2013** Wernimont, Jacqueline. "Whence Feminism? Assessing Feminist Interventions in Digital Literary Archives." *Digital Humanities Quarterly* vol. 7.1 (2003). <http://www.digitalhumanities.org/dhq/vol/7/1/000156/000156.html>.
- Wernimont and Losh 2018a** Wernimont, Jacqueline and Elizabeth M. Losh. "Wear and Care Feminisms at a Long Maker Table." In *The Routledge Companion to Media Studies and Digital Humanities*, edited by Jentery Sayers, 97-107. NY: Routledge, 2018. <https://scholarworks.wm.edu/asbookchapters/5/>.
- Wernimont and Losh 2018b** Wernimont, Jacqueline and Elizabeth M. Losh (eds). *Bodies of Information: Intersectional Feminism and Digital Humanities*. Minneapolis: University of Minnesota Press, 2018.



This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.