Retrocomputing as Preservation and Remix

Yuri Takhteyev
University of Toronto
yuri.takhteyev@utoronto.ca

Quinn DuPont
University of Toronto
quinn.dupont@utoronto.ca

Abstract

This paper looks at the world of retrocomputing, a constellation of largely non-professional practices involving old computing technology. Retrocomputing includes many activities that can be seen as constituting "preservation." At the same time, it is often transformative, producing assemblages that "remix" fragments from the past with newer elements or joining together historic components that were never combined before. While such "remix" may seem to undermine preservation, it allows for fragments of computing history to be reintegrated into a living, ongoing practice, contributing to preservation in a broader sense. The seemingly unorganized nature of retrocomputing assemblages also provides space for alternative "situated knowledges" and histories of computing, which can sometimes be quite sophisticated. Recognizing such alternative epistemologies paves the way for alternative approaches to preservation.

Keywords: retrocomputing, software preservation, remix

Recovering #popsource

In late March of 2012 Jordan Mechner received a shipment from his father, a box full of old floppies. Among them was a 3.5 inch disk labelled: "Prince of Persia / Source Code (Apple) / ©1989 Jordan Mechner (Original)." Mechner's announcement of this find on his blog the next day took the world of nerds by storm.[1] *Prince of Persia*, a game that Mechner single-handedly developed in the late 1980s, revolutionized computer games when it came out due to its surprisingly realistic representation of human movement. After being ported to DOS and Apple's Mac OS in the early 1990s the game sold 2 million copies (Pham, 2001).

Mechner's original 1989 version, however, was written for Apple II, a platform already somewhat outdated at the time. The original version of the game thus featured much more modest graphics and sound than the later DOS and Mac releases. This early version is still remembered — and played — by aficionados, however, being easily available on the Internet in the form of disk image files derived from a "crack" of the game produced around 1990, credited to "The Crasher" and associates, and bearing a curious dedication to "Nebraska Cement Factory."[2]

Prince of Persia images is to load them on one of the many Apple II emulators available online. For the more dedicated fans, however, there is the option of using the original hardware. For some, this original hardware is, of course, Apple II. For others, however, it may be *other* 1980s computers, including some that could not run the game at the time. For example, in 2011 a programmer known as "mrsid"

[1] Our discussion of Mechner's discovery and subsequent recovery of Prince of Persia source code is based in part on Mechner's blog (Mechner, 2012a) and Twitter stream (Mechner, 2012b), as well as the Wired article written by a person who was present during the recovery process (Mastrapa, 2012).
[2] Our dating is based on the "crack screen," included in the game which could be a forgery.

Acknowledgements: This research was supported by Insight Development Grant from the Social Sciences and Humanities Research Council of Canada.
Takhteyev, Y., DuPont, I.Q. (2013). Retrocomputing as Preservation and Remix. *iConference 2013 Proceedings* (pp. 422-432). doi:10.9776/13230

successfully completed porting the Apple II version of *Prince of Persia* to Commodore 64, a task that took him two and a half years (mrsid, n.d.). (Mechner's comments on the announcement noted that the game was not originally released on Commodore 64 because the platform was considered obsolete in 1989.) Projects such as those undertaken by mrsid would be much easier if the source code of the game were available. The code, however, had long been presumed lost. Mechner's discovery of the floppy thus generated quite a lot of excitement.

Prince of Persia images is to load them on one of the many Apple II emulators available online. For the more dedicated fans, however, there is the option of using the original hardware. For some, this original hardware is, of course, Apple II. For others, however, it may be *other* 1980s computers, including some that could not run the game at the time. For example, in 2011 a programmer known as "mrsid" successfully completed porting the Apple II version of *Prince of Persia* to Commodore 64, a task that took him two and a half years (mrsid, n.d.). (Mechner's comments on the announcement noted that the game was not originally released on Commodore 64 because the platform was considered obsolete in 1989.) Projects such as those undertaken by mrsid would be much easier if the source code of the game were available. The code, however, had long been presumed lost. Mechner's discovery of the floppy thus generated quite a lot of excitement.

The find, however, also presented a challenge. "I will now begin working with a digital-archeology-minded friend to attempt to figure out how to transfer 3.5″ Apple ProDOS disks onto a MacBook Air and into some kind of 21st-century-readable format," Mechner wrote on his blog. Mechner's call for assistance brought two men to his door a few weeks later. One was Jason Scott, best known as the maintainer of textfiles.com, a website originally dedicated to preserving thousands of ASCII files shared on bulletin-board systems (BBS) in the 1980s and early 1990s, but then expanded to collect shareware CD images, audio files, and other digital artifacts from the era. Until recently Scott pursued those interests while employed as a system administrator. In 2011, however, he joined the Internet Archive to focus his full-time efforts on preservation of old software. The other man was Tony Diaz, a collector of Apple II hardware and maintainer of the website apple2.org, dedicated to photographic images of Apple II. Each man arrived with somewhat different tools. Scott brought DiscFerret, a small open-hardware device designed to read raw pattern of magnetism from a floppy, leaving the analysis and digitization of the pattern to a software tool, thus offering flexible support for a wide range of approaches for storing data, as well as an ability to circumvent many antique copy-protection schemes. Diaz arrived with a van full of Apple II hardware — original, though rigged with substantial hardware and software modifications, including support for Ethernet, not available on the original Apple II.

With their help, Mechner's original files were transferred to his MacBook Air, in a live-tweeted session tagged "#popsource" that attracted so much attention that Mechner's website collapsed from the traffic. The source code was then quickly made available on GitHub, a site widely used for sharing open source code. Within hours, GitHub user "st3fan" made a modification commenting out the copy-protection code (st3fan, 2012). This move was purely symbolic, since the posted code was incomplete at the time and could not actually be compiled and run. A few days later, however, a programmer working on an Apple II emulator credited the posted source code as a source of information leading to an improvement in the emulator.[3]

The story presented above provides a glimpse into the world of retrocomputing, a set of diverse practices involving contemporary engagement with old computer systems. Such practices are primarily private and non-professional, though this is not always the case — there is also a substantial economy providing products and services. And to the extent that retrocomputing participants are "hobbyists," in the sense of not being paid for their work, they are hardly unskilled amateurs. Rather, their practice often demonstrates deep sophistication. In other words, many of them are "hobbyists" only in the same sense as many of the contributors to open source software, which today underlies much of the world's computing infrastructure.

Many of the activities that make up retrocomputing can be seen as constituting collection and preservation, and many retrocomputing enthusiasts in fact recognize preservation of computer history as one of their key goals. Such activities involve efforts to collect and restore old hardware, develop

[3] The remaining Prince of Persia files were posted on Github in May, making it possible, in theory, to compile the code. At the moment it is unclear if anyone has been successful in doing so.

emulators, and build substantial digital collections of old software. For example, it does not take long to find on the Internet disk images for *Prince of Persia* for Apple II, as well as a *variety* of emulators that can run them. Retrocomputing also involves development of sophisticated tools for digitization, such as the DiscFerret tool mentioned above.

At the same time, closer attention to those projects reveals that they cannot be easily understood as just a matter of preservation in the narrow sense of keeping objects from the past fixed in their "original" form. Instead, retrocomputing is often transformative and involves construction of assemblages that "remix" fragments of old systems with newer elements, such as old software running on freshly debugged emulators or original hardware enhanced with contemporary networking. It can also involve a mixture of historic components that were never combined in the past, as in the case of mrsid's porting of *Prince of Persia* to Commodore 64.

Such "remixes" are sometimes born out of necessity: retrocomputing usually emphasizes active engagement, actually *running* old systems when possible. This sometimes requires replacing pieces that have become unavailable with newer alternatives. Quite often, however, retrocomputing is transformative because simple preservation is not the participant's only (or even main) goal. Instead, retrocomputing enthusiasts engage with old computing for a variety of reasons, including a simple desire to have fun with old technology. When doing so, they often seek to adapt systems to their contemporary needs — for example, finding ways to connect old computers to newer ones via modern networking.

While such "remixing" may seem contrary to preservation, it fulfills the goals of preservation in a broader sense. In addition to reassembling fragments of computer history into running systems, retrocomputing enthusiasts also do the work of reintegrating fragments into a living contemporary practice. This provides an important set of resources on which more traditionally focused preservation projects can draw. The most obvious of these are the collections of preserved artifacts and tools. Much more important, perhaps, is the retrocomputing ecosystem itself: a persisting system of cultural and economic relationships that keeps alive a circulation of material and digital artifacts as well as deep technical knowledge.[4] It is this system of relationships that ensured that Mechner's call for assistance was quickly answered by people who arrived equipped not only with the appropriate knowledge but also with the necessary tools.

However, appreciating retrocomputing just for the resources it can provide may miss its deeper value. The practice of retrocomputing also provides space for ongoing circulation of meaning and divergent "situated knowledges," and for alternative histories of computing. Consequently, it may not only provide us with new insights into the *how* of digital preservation, but also into the *what* and *why*. We may therefore need to recognize the value of retrocomputing projects on their own terms and look for ways to provide them with support, while respecting their own objectives.

In the rest of the paper we start by looking at some of the ways in which retrocomputing aligns with the more traditional approaches to preserving the past. We then look at how it deviates from such approaches and engages in "remix." We argue that such remix may fulfill the goals of preservation in a broader sense. We then turn to the importance of looking at retrocomputing on its own terms, recognizing the divergent situated knowledges embodied in this practice.

Retrocomputing as Preservation

Preserving the history of computing can take different forms (Aspray, 1984; Bruemmer, 1987). One could focus on preserving the original hardware and original media, perhaps restoring them to the state where they can actually run (Burnet & Supnik, 1996). One can build digital collections of software, to be run either in emulators or on original hardware (Shustek, 2006). One can also collect documents relevant to the history of computing and its historical cultural milieu (Cortada, 2002; Mahoney, 2008). If such documents were born physical, they can be collected in the original physical form or can be digitized. Some such documents, such as text files shared on bulletin board systems in the 1980s, were "born digital" and can be collected as such, perhaps with some adjustments for contemporary encoding methods (Cloolan, 2007). Taking photos of old computers and video recording them in use provides yet

[4] See also Lowenthal (1989), who argues that preservation should be less concerned with whole, material artifacts, and instead should look for fragments and processes, and Eng-Wilmot (2008) on collage.

another route.[5] All of these approaches are represented in retrocomputing. Without prejudging the relative value of each of these approaches, we focus this section on retrocomputing participants' efforts directed towards digital preservation of software, in order to document the substantial sophistication of the participants' preservation efforts.

Emulation of old platforms in software has been widely recognized as one of the promising strategies for digital preservation, and possibly the only viable long term strategy for preservation of software (see Rothenberg, 1995, 1999; Ross 2000[6]). Development of emulators, however, is difficult work requiring substantial expertise both in contemporary software platforms as well as the old systems that are being emulated. For example, developing an Apple II emulator to run inside a web browser requires knowledge of modern JavaScript as well as deep understanding of the workings of Apple II. This work is made particularly challenging by the large number of old platforms. Development of an emulator is also, in an important sense, never quite complete. First, and most obviously, emulators themselves run on platforms that eventually become obsolete, sometimes quite quickly. For example, the ActiveGS emulator for Apple II works with the 2011 version of the Firefox browser, but not with the 2012 version. Additionally, emulating new software sometimes requires adjustments to the emulator due to the imperfect match between the emulator and the original platform. Due to these difficulties, organizing development of emulators as paid professional work can be extremely costly and institutional software preservation initiatives would find it difficult or impossible to commission emulators for all the software that they may want to preserve (Von Suchodoletz et al, 2009).

A distributed, open source style "peer production" approach provides a possible solution. Retrocomputing enthusiasts often have both the requisite knowledge and the interest in doing this difficult work.[7] Their efforts have so far resulted in a large number of emulators, covering a variety of platforms. In the case of relatively popular platforms such as Apple II, one can find emulators that run on Windows, Mac OS X, Linux, as well as the above mentioned browser-based ActiveGS. Less obviously, there are Apple II emulators for platforms that are themselves quite "retro," such as Amiga OS (Tzvetkov, n.d.) — an example of "remix" to which we turn in the next section. One can also find many emulators for lesser known platforms, such as Apple's Lisa or the 1974 (pre-Altair) SCELBI-8H.

Some of the emulators are produced by open source projects in the full sense of the term: the code is developed using open source methodology and shared under an open source license. Many are "not quite open source." For example, a popular MAME emulator is distributed under a license that allows redistribution, but prohibits commercial use, which runs contrary to open source licensing.[8] One of the reasons for this "not quite open source" approach appears to be historical. While retrocomputing in many ways resembles open source and increasingly emulates it, the practice originates from a different subculture, tied more closely to the world of 1980s personal computing, where "shareware" was the dominant form of sharing software.[9] (In contrast, free / open source software's roots are best traced to academic computing based on Unix.) This slow transition to open source (impeded in part by legacy licensing) highlights the importance of recognizing retrocomputing itself as a historically situated practice.

While use of emulation has often been recognized as one of the main strategies in digital preservation, this approach has multiple problems. It is important to recognize, however, that retrogaming enthusiasts are often quite aware of such problems and seek sophisticated solutions. For example, last year *Ars Technica* published an essay by "byuu" the anonymous author of BSNES emulator (byuu, 2011). byuu's essay points to many challenges facing accurate emulation, including, for example, the computational complexity of ensuring the proper timing of emulated instructions. byuu argues that the correct solution would involve re-synchronizing the clocks of the real and emulated processors after each

[5] Lowood (2011) extends this to screen capture and Machinima.

[6] It should be noted that emulators were in use in the retrocomputing community before Rothenberg suggested their use 1995, for example Nutria, a ZX Spectrum emulator developed in 1991 or ZXAM, a Spectrum emulator for Amiga developed in 1993.

[7] Development of emulators by retrocomputing enthusiasts is briefly noted by Rothenberg (1999). Other authors often cite retrocomputing emulators to argue for feasibility of emulation (*e.g.,* Ross 2000), but do not usually discuss the provenance of these tools.

[8] According to Open Source Consortium's "Open Source Definition," an "open source" license cannot restrict the use of software "in a specific field of endeavor," a clause specifically meant to exclude licenses that prohibit commercial use. Free Software Foundation's definition of free software also requires that the recipient be allowed to run the software "for any purpose," and further that he or she should be free to charge others for copies.

[9] Retrocomputing is also closely linked with the "demo scene" subculture of the 1980s.

instruction, which would require more computational power than today's hardware can provide.[10] byuu discusses the tradeoffs between emulation efficiency and accuracy, urging the users to invest in more powerful machines and run more accurate (if slower) emulators, arguing that wide use of power-intensive emulators would help the community to raise expectations for accuracy and make progress towards identifying and fixing discrepancies. byuu's analysis thus showcases not just sophisticated *technical* thinking, but also attention to the social elements of the collective projects in which he (or she) is involved.

Emulation requires "images" of original software — files containing the bits extracted from the original media. This would often need to include images of the original system software (for example, Apple II system ROMs), as well as the application software that is to be emulated.[11] Such images are in fact widely available on hobbyist sites today, often with elaborate metadata. Distribution of such images is, as a general rule, illegal, as it would require permission from the copyright holder, which is usually difficult or impossible to obtain.[12]

This situation creates a serious challenge for preservation projects undertaken by traditional institutions, which feel that they must keep old software under lock and key to avoid charges of copyright infringement. However, it presents a lesser problem for hobbyists, who are somewhat shielded by their relative anonymity and lack of major assets.[13] Additionally, retrocomputing participants employ a number of strategies to limit their exposure yet further. One of them is "splitting" the legal risk, by relying on other parties to host the most "radioactive" assets. For example, some Apple II emulators do not include system ROMs but instead provide users with instructions where to obtain them. Such risk-sharing also provides a potential point of collaboration with institutional efforts: institutions may be able to "offload" some legal risk by relying on the retrocomputing community to perform some of these legally risky tasks.[14]

Disk images collected and circulated by the hobbyists come from sundry sources. Some are actually quite old: the most commonly available image for the Apple II *Prince of Persia* appears to have originated around 1990. Some are more recent, but still produced by running image ripping software on old machines — or at least *older* machines. For example, one can read Apple II disks using early PC hardware with special software. This method can be quite challenging due to copy protection, as well as the gradual disappearance of older hardware and knowledge of how to operate it. Perhaps the most sophisticated solution for this problem is exemplified by DiscFerret and KryoFlux. Both are hardware products that sit between a floppy disk drive and a contemporary computer, allowing the latter to scan the raw pattern of magnetization from a disk's surface, leaving the "parsing" of this pattern to a later step, implemented in software. This allows, among other things, to handle copy-protection methods that rely on specific features of the original hardware to "trick" it (*e.g.,* writing data between the tracks or relying on timing intricacies). Both projects are run by private groups. The more established KryoFlux is proprietary, while DiscFerret is organized as an open source and "open hardware" project.

Running software in emulators is, of course, only one approach. Retrocomputing quite often also involves using the original hardware. While this can mean working with the original media (*e.g.,* the original floppies), this method is problematic since such media can be easily damaged in the process, either through wear and tear or because data may be modified.[15] The original media is also becoming

[10]For an extreme example of this approach byuu cites DICE, an emulator that aims to emulate discrete logic systems (without a CPU) by simulating individual transistor propagation delays.

[11] In case of applications, such software serves dual purpose. First, it provides the raison d'être for the emulator. Second, a diverse collection of application software helps improve the accuracy of emulation.

[12] This difficulty often has less to do with the copyright holders' desire to keep old software away from users, and more with the difficulty that aspiring image distributors encounter in even getting a response. In many cases, it is impossible to identify who the copyright holders are, since the companies that produced the software have by now gone through series of mergers and sales or liquidations. In those cases where the current owners can be identified, their legal departments are often reluctant to grant explicit permission, seeing limited upside in doing so, or simply being too busy attending to their current needs.

[13] As RIAA's lawsuits against people participating in MP3 sharing have shown, neither anonymity nor limited assets provide protection against motivated copyright holders. In case of old software, however, the copyright holders are rarely motivated to pursue their rights (and, in fact, may not even be aware that they are the owners of the software).

[14] Another potential strategy, though one we have not seen yet realized in retrocomputing, would involve "crowdsourcing" legal research in the same way Wikipedia does with images: countless volunteers take time to identify the copyright status of the images and when necessary engage with the copyright holders to secure proper licensing.

[15] Perhaps the most extreme case of this is presented by "Agrippa": a limited edition diskette released in 1992 containing a poem by William Gibson and designed so that the content of the poem would be irreversibly encrypted after being briefly displayed to the user (Kirschenbaum 2007).

increasingly rare. As a result, old hardware is frequently used in combination with downloaded images. One way to do so involves writing downloaded images to a new floppy disk. This method, however, requires functioning floppy drives, which are among the least reliable peripherals for old computing platforms. A common alternative is to connect old computers to peripherals capable of reading modern media such as SD cards. Such solutions usually involve a combination of hardware and software, which is today available for a number of platforms, including Apple II ("Pseudo Disk][Controller Card," n.d.). The resulting assemblages of hardware and software from different time periods illustrate the issue of "remix" to which we turn in the next section.

Retrocomputing as Remix

While aspects of retrocomputing can be easily recognized as constituting preservation, retrocomputing is often transformative, producing assemblages of physical and digital fragments originating from different time periods and "remixed" in novel ways.

Such remix can be done out of simple necessity to substitute missing pieces in order to "restore" a old system to working order. This restoration can be carried out quite conservatively, as is usually the case when it is undertaken by the more traditional memory institutions. For example, when the Computer History Museum in California undertook the project of restoring a PDP 1, effort was made to ensure that any modifications were reversible. Some retrocomputing projects exercise a similar amount of care for preserving the authenticity of restored systems. Quite often, however, they involve creation of assemblages that clearly violate the principles of minimalism in preservation, as illustrated by many of the examples introduced earlier. One can observe cases of old hardware linked to contemporary peripherals, such as SD card readers or Ethernet networking cards. Such hardware may need new software to interact with such devices. Sometimes the original operating system is replaced with something newer altogether. For example, the early 1980s Commodore 64 might be updated to run the 1989 release of JiffyDOS system software, which provides a substantial boost in performance, while some Amiga computers from the 1980s may be setup to run a 2006 version of MorphOS. (Alternatively, an updated version of Amiga OS is also available and can be made to run on 2006 hardware.) Such systems may then be used to run images of software downloaded from the Internet — original, modified (*e.g.,* to remove copy protection), or brand new.

We conceptualize these transformative aspects of retrocomputing as a form of "remix" — a term popularized by Lessig (2008). Like the closely related concept of "collage," the term "remix" refers to a creative and often playful reassembly of fragments of earlier works into something new.[16] While the reasons for chimeric assemblages described above is sometimes pragmatic, at other times it is simply playful, carried out for fun. At a gathering of Commodore enthusiasts attended by one of the authors, a participant demonstrated an old Commodore 64C system that he had skillfully painted bright blue. He had also hacked it to play stereo sound and wrote his own Commodore 64 software that took advantages of the sound, since none of the original software was designed for stereo sound. When asked about his reasons for repainting the machine, the man explained that the computer's original white plastic had turned an "ugly" yellow over time. Repainted blue, it looked "awesome." In other cases, however, the pursuit of fun and beauty often cannot be easily separated from the "pragmatic" motivation for remixing fragments of old computing. Much like Linus Torvalds describing his development of Linux as "just for fun" (Torvalds, 2001), this notion of fun usually implies getting satisfaction in finding solutions to technical problems, thus fusing "pragmatic" and "playful" motivations.

Playful remix inherent in much of retrocomputing may at first blush seem to be in contradiction to efforts preserving the history of computing. This contradiction, however, dissipates with further analysis. Even in the seemingly extreme case of re-painting an old machine to a new color — a step that cannot be undone — the act preserves in that it restores the machine to the "awesomeness" that it once possessed. A machine that was once a source of joy becomes capable of bringing joy once again.

[16] Appropriately for our look at retrocomputing, the term "remix" has also come to connote open-ended and distributed reassembly. That is, in a "remix" fragments of the old are often understood to be assembled not to be frozen in a new configuration (as might be the case in a "collage"), but rather to be taken apart and further remixed by others.

More generally, the remix inherent in retrocomputing allows continuous reintegration of elements of past computing systems into an ongoing, living practice. We understand practice as the system of activities comprised of people, ideas, and material objects, tied by shared meanings and joint projects (Takhteyev, 2012; see also Lave and Wenger, 1991).[17] Computing artifacts are born into such systems and have power and meaning because of their linkages to other elements. Over time, however, some elements of these systems disappear or enter into new relationships and abandon the old ones. Commodore International, a company that once made Commodore computers has long since disappeared. Companies that once wrote software for the Commodore have moved on to other projects. Most of the users have also "upgraded" to other computers. Hidden in a basement old computers may remain functional in principle, but become dead in the sense of no longer being incorporated into an ongoing system of activities. With the dissolution of relationships comes the fading of tacit knowledge that once made its use possible (*see* Galloway, 2011; *cf*. Collins, 1974; MacKenzie and Spinardi, 1995). Such processes of social decomposition may often be much more damaging to old computing systems than the physical breakdown of the hardware or storage media, and it cannot be stopped by isolating the fragments and shielding them from sunlight or improper temperature and humidity.

The decay of the socio-technical practice, in which the antiquated elements of computing history were once embedded, is partly stopped or even undone in retrocomputing, as ancient fragments are reintegrated into ongoing activities, becoming part of a *contemporary* living practice. Such integration allows for maintenance (and sometimes recovery) of tacit knowledge. It also makes possible continuous circulation of tools and resources. While retrocomputing is often understood to be the domain of "hobbyists," it is in fact undergirded by a complex ecology of commercial, hobby, and grey market products and services. Many of the projects mentioned above are undertaken by either for-profit companies or by people and organizations that combine non-profit and commercial activities. It is this complex, interrelated ecosystem that allowed Mechner's files to be transferred with such seeming ease from the 1980s floppies to GitHub.

One of the surprising (and perhaps disconcerting) aspects of retrocomputing assemblages is the fact that it often involves an admixture of elements from a variety of historical periods: not just from the time of the "original" and contemporary, but everything in between. The transfer of Mechner's files from floppy disks to GitHub involved not only old and contemporary software and hardware, for example, but also Ethernet hardware and drivers originating from the late 1990s. Someone looking for the Apple II release of *Prince of Persia* online would likely encounter a cracked version with a dedication to a cement factory in Nebraska and a screen crediting the crackers. This crack — also in some ways a "remix" — appears to be quite old, likely from the early 1990s. Such time-warping, however, reflects the fact that retrocomputing is itself a historically unfolding practice, closely tied to other forms of computing remix. Thus remix may need to be recognized not as corruption of the original but as representing the vibrant nature of computing. When thinking about what it would mean to preserve *Prince of Persia*, we may need to consider that the version of the game produced by "The Crasher" and his associates may have been played over the years by more people than Mechner's original, and thus may be as worthy of preservation. Yet more importantly, the cracked version not only preserves the gameplay of *Prince of Persia* as experienced by many players at the time, it also reflects the legal and cultural battles over intellectual property rights in the 1980s and 1990s, bearing witness to players' resistance to a property regime that was still fairly new, and painting a richer picture of computing in the 1980s and early 1990s than does the original game.

Retrocomputing and Situated Histories

As we have demonstrated above, the efforts undertaken by retrocomputing participants can potentially be of great value for the more traditional efforts to preserve computing history. At the same time, many other aspects of the practice present challenges to conventional preservation, in particular when it comes to what may seem like wanton disregard for authenticity. While authenticity is a thorny issue theoretically, in practice "authentic" archival objects are usually desired: we normally want archived object and records to be properly contextualized and to resemble the original as much as possible.

[17] This approach to practice also has important parallels to the agency-oriented approach advocated by Dallas (2007).

Retrocomputing, however, presents us with chimeras such as *Prince of Persia* running on Commodore 64, remixing fragments of 1980s computing into something that may *seem* like an authentic artifact from the period to a naive observer, yet in fact presents an entirely novel combination. We may also further worry about the seeming indifference to authenticity exhibited by sites that showcase the "cracked" version of *Prince of Persia* without bothering to even make note of the fact that this version contains modifications added by the crackers. And of course, there is the issue of retrocomputing participants' focus on the seemingly trivial domains of computing — in particular games — at the expense of the more "serious" ones.[18]

Our reactions to such chimeras may lead us to educate the participants about proper curatorial practices. Alternatively, we could treat them as a mostly harmless play. We would like to argue, however, that a better reaction may be to use retrocomputing to challenge our own notions of authenticity and to look at retrocomputing armed with Haraway's notion of "situated knowledges" (1988). Haraway's approach aims to find a middle ground between a traditional positivist epistemology that seeks objectivity through a singular disembodied "view from nowhere" and, on the other hand, relativism that rejects the notion of objectivity altogether. Instead, in Haraway's view, knowledge is only available in the form of a multitude of partial and situated perspectives.[19]

Applying this approach to the history of computing would lead us to avoid measuring retrocomputing against a singular conception of "the history" of computing, but instead to look at the multitude of histories. From this perspective, a Marxist's history of computing can be contrasted to an engineer's history of computing, which can be contrasted to that of a gamer (see also Mahoney, 2005). The question then becomes not whether retrocomputing actually preserves the history of computing, but rather whose histories and knowledges are reflected in it, how, and why.

One criticism of retrocomputing that we have mentioned earlier concerns its heavy focus on computer games. While retrocomputing participants do engage with non-gaming applications, games are in fact preserved much more than, for example, business applications and platform. Haraway's perspective alerts us, however, to the fact that this criticism itself reflects a backgrounding of a situated perspective that pre-assigns value to different domains of computing, deeming some of them as more worthy of serious attention than others. But what if we suspend such assumptions and instead look at the history (and future) of computing through the eyes of people who preserve and play old games? What emerges then is an alternative history of computing, one concerned less with pure engineering innovation and the politico-economic history of the twentieth century, but instead reflecting in much more vivid colors the computer's changing place in relation to human psyche (see also Turkle, 1984). A proper reading of such history of computing would take seriously the notion of the computer as potentially an "awesome" machine, and may give some credence to the idea that a freshly repainted Commodore playing stereo sound may be more reflective of the 1980s experience of personal computing than a disconnected machine encased in discolored plastic.

Approaching retrocomputing from the perspective of situated knowledges led us to ask what *other* situated histories may be reflected in it. One example we have encountered is the vibrant community of people focused on the history of computer chess. Taking a form of a loose network of websites, the community combines the dedication and zeal that we saw in many retrocomputing subcultures. In fact, our outsider perspective struggled to reduce this community to any simple examples. Nonetheless, "The Spacious Mind" website appears to be characteristic. The site boasts a very large collection (hundreds or perhaps thousands) of chess computing devices and software and corresponding old hardware ("The Spacious Mind," n.d.). Instead of cataloging the results of human play against chess computers, the chess computer community pits computer against computer in a battle of artificial wits. The Spacious Mind maintains its own records of these games (with interactive playback) but also participates in a global network of computer versus computer chess tournaments, crowning victors in various categories. To keep the matches interesting, the chess computers are typically arranged in to national allegiances for

[18] While old games are by far the most active domain of retrocomputing, though, many of the other domains are in fact represented, if not as actively.

[19] Haraway's rejection of a single "objective" perspective of reality is distinct from relativism in that it embraces situated perspectives rather than aiming to rise above them. In Haraway's view relativism and traditional positivism are similar in that "both deny the stakes in location, embodiment, and partial perspective [...] both are 'god tricks' promising vision from everywhere and nowhere equally and fully" (584).

appropriate years (usually the year of production), such as the "1980 World HC&CC Championship Revisited Swiss" tournament.

In addition to tournament play, the chess computer community has developed a set of tests to compare chess devices and software. These tests are used to determine if an identical chipset was used by two distinct manufactures, or if programming was copied from one maker to another. These tests require careful comparisons of timing to test the response to each move, in addition to the actual change of position. Because some chess computers have built-in randomness, other tests force the computer to make certain moves and then, over a two hour game record the moves in an extensive spreadsheet to spot where derivations occur.

To organize this large collection the curator of the Spacious Mind has established a metadata scheme so rich and complex it is hard to imagine that it could arise in any other context. Each record contains 31 metadata fields, two sets of matches against the curator's collection (often repeated multiple times), two sets of matches against the chess computer community (cross-referenced against other chess computer websites), and a web-based display re-enacting the moves played. Other retrocomuting subcultures also develop their own informal but relatively stable set of metadata fields appropriate to the specific domain — what Bowker and Star would call "local classification schemes" (1999). Such local schemes frequently go well beyond common archive or library metadata.

Recognizing the value of the situated knowledges of the past, as embodied in retrocomputing, does not mean we need to idealize those perspectives. Such situated histories *are* partial and incomplete. Recognizing the value of partial perspectives, however, points us to a very different set of critiques and solutions: instead of noting the ways in which those histories diverge from "the history" of computing, we may want to ask whose knowledges of the past fail to be registered. (We may note, for example, the shortage of women's voices.) Additionally, many situated histories may be represented yet easily overlooked among the many others. We can then ask, what can be done to *broaden* the diversity of perspectives rather than trying to narrow them down.

The approach advocated here has important parallels in some of the new movements in museology, which increasingly recognizes museums as inherently situated agents, reflecting biases and assumptions of specific cultures (Stam, 1993),

rather than possessing a privileged perspective of the kind that Haraway would call "a view […] from nowhere." This perspective is most easily observed in the way institutions increasingly approach artifacts belonging to other cultures, especially former colonies of the West, where the history of eurocentrism is hard to ignore.[20] At a broader level, however, such approaches increasingly recognize the need for more flexible and relational understandings of the notions of objectivity and authenticity. Authenticity is no longer understood to be an inherent or essential part of the object, but rather is "constructed" from patrons' individual and collective sense of authenticity (Jones, 2010).

Retrocomputing and the Preservation Community

Retrocomputing presents challenges and opportunities to academic and institutional preservation communities. Perhaps the simplest response (other than just ignoring retrocomputing altogether), is to appreciate the technical knowledge possessed by retrocomputing participants and the work they have done, while closing our eyes to those aspects of retrocomputing that may seem to violate the principles of preservation. (After all, in most cases nothing is lost in those remix projects.) This stance would then open the way for traditional institutions to make use of the products of retrocomputing labor and perhaps even steer the participants' efforts towards projects deemed more important. This approach may in fact be beneficial to institutional preservation efforts and may well be welcomed by the retrocomputing community. It appears that a growing number of institutions are in fact starting to venture in this direction.

[20] One response to such prior eurocentrism has involved paying more attention to other cultures' "indigenous" curatorial practices around their own objects (*e.g.,* Kreps, 2009). One could note a parallel with our case: retrocomputing participants are often people who have had deep experience with historic computing artifacts back at the time when they were new, either as users or even as their creators. Their ways of handling those artifacts today may therefore deserve attention and respect in the same way as advocated by Kreps. This parallel, of course, is far from precise, since the power relationships in the two cases are quite different.

Approaching retrocomputing from the position of what can be gained from it, however, risks missing the deeper value of the practice. We have attempted to demonstrate that remix practices can be understood as a practical consequence of situated knowledges for the various subgroups of participants. Consequently, we suggest that it may be useful for the preservation community to approach retrocomputing from the perspective of Haraway's epistemology, to embrace new community-driven practices such as remix, and to work with the communities to understand what drives their projects.

Such engagement does not need to be limited to observation. The institutional preservation community (and the iSchool community more broadly) ought to be able to assist retrocomputing projects. Such assistance, however, should start by asking not what retrocomputing practitioners are doing wrong from the perspective of institutional practice or academic scholarship, but rather, looking at practitioners' *own* objectives and asking which of those objectives they are failing to achieve and what institutions may be able to offer. For example, instead of asking how private collections of old software could be "ingested" into institutional repositories, it may be more appropriate to ask in what ways retrocomputing practice could be enhanced if we offered the participants access to more powerful infrastructure.[21]

The preservation community (and some of the other subcomunities within the iSchool movement) can help generalize and extend bottom-up practices within the retrocomputing community, function as a liaison between the sometimes distant retrocomputing communities, help bridge their gaps, and adopt common (exchangeable) practices. The challenge with situated knowledge is that it is, precisely, *situated*, and thus difficult to benefit from broadly. We caution against approaches that attempt to *exploit* these knowledges, or reduce them to "mere resource" as Haraway warned. We ought to ask, however, what can be done to help such knowledges become more mobile and thus of greater benefit to all.

References

Aspray, W. (1984). Literature and institutions in the history of computing. *Isis, 75*(1), 162–170.

Bowker, G. C., & Star S. L. (1999). *Sorting things out: Classification and its consequences*. Cambridge, Mass.: The MIT Press.

Bruemmer, B. H. (1987). *Resources for the history of computing.* Minneapolis, MN: Charles Babbage Institute, University of Minnesota.

Burnet, M. M., & Supnik, R. M. (1996). Preserving computing's past: Restoration and simulation. *Digital Technical Journal, 8*(3), 23–38.

byuu. (2011). Accuracy takes power: One man's 3GHz quest to build a perfect SNES emulator. In *Ars Technica.* Retrieved from http://arstechnica.com/gaming/2011/08/accuracy-takes-power-one-mans-3ghz-quest-to-build-a-perfect-snes-emulator/

Cloonan, M. V. (2007). The paradox of preservation. *Library Trends, 56*(1), 133–147.

Collins, H. M. (1974.) The TEA set: Tacit knowledge and scientific networks. *Science Studies 4*(2): 165–185.

Cortada, J. W. (2002). Researching the history of software from the 1960s. *Annals of the History of Computing,* IEEE, *24*(1), 72–79. doi:10.1109/85.988584.

Dallas, C. (2007). An agency-oriented approach to digital curation theory and practice. In J. Trant and D. Bearman (eds.), *International Cultural Heritage Informatics Meeting (ICHIM07): Proceedings,* Toronto: Archives & Museum Informatics.

Eng-Wilmot, G. (2008, April 24). *The decay of memory and matter: Material transformation in the new artistic archive* (Masters). Georgetown University, Washington, DC, USA.

Galloway, P. (2011). Retrocomputing, archival research, and digital heritage preservation: A computer museum and iSchool collaboration. *Library Trends*, *59*(4), 623–636. doi:10.1353/lib.2011.0014.

Haraway, D. (1988). Situated knowledges: The science question in feminism and the privilege of partial perspective. *Feminist Studies, 14*(3), 575–599. doi:10.2307/3178066.

Jones, S. (2010). Negotiating authentic objects and authentic selves: Beyond the deconstruction of authenticity. *Journal of Material Culture*, *15*(2), 181–203. doi:10.1177/1359183510364074.

Kirschenbaum, M. (2007). *Mechanisms: New media and the forensic imagination.* MIT Press.

[21] This approach is now to some extent taken by the Internet Archive, which some other ways straddles the worlds of institutional preservation and retrocomputing.

Kreps, C. (2009). Indigenous curation, museums, and intangible cultural heritage. In L. Smith & N. Akagawa, *Intangible heritage*. 193–208. London: Routledge.
Lave, J., & Wenger E. (1991). *Situated learning: Legitimate peripheral participation.* Cambridge: Cambridge University Press.
Lessig, L. (2008). *Remix*. London: Bloomsbury Publishing PLC. Retrieved from http://www.bloomsburyacademic.com/view/Remix_9781849662505/book-ba-9781849662505.xml
Lowenthal, D. (1989). Material preservation and its alternatives. *Perspecta*, *25*, 67–77. doi:10.2307/1567139
Lowood, H. (2011). Perfect capture: Three takes on replay, machinima and the history of virtual worlds. *Journal of Visual Culture, 10*(1), 113–124.
MacKenzie, D., & G. Spinardi. (1995). Tacit knowledge and the uninvention of nuclear weapons. *American Journal of Sociology, 101*(1): 44–99.
Mahoney, M. S. (2005). The histories of computing(s). *Interdisciplinary Science Reviews*, *30*(2), 119–135. doi:10.1179/030801805X25927.
Mahoney, M. S. (2008). What makes the history of software hard. *Annals of the History of Computing,* IEEE, *30*(3), 8–18. doi:10.1109/MAHC.2008.55.
Mastrapa, G. (2012) The geeks who saved *Prince of Persia's* source code from digital death. In *Wired.com*. Retrieved from http://www.wired.com/gamelife/2012/04/prince-of-persia-source-code/
Mechner, J. (2012 a). *Jordan Mechner / Making video games*. [Multiple entries from March and April 2012.] Retrieved from http://jordanmechner.com/blog/2012/04/source/
Mechner, J. (2012 b). *Jordan Mechner (jmechner) on Twitter.* [Multiple posts from April 2012.] Retrieved from https://twitter.com/jmechner
mrsid. (n.d.). *Prince of Persia C64 – Development blog.* Retrieved from http://popc64.blogspot.ca/
Pham, Al. (2001, April 25). Prince of Persia creator Jordan Mechner is still in the game. *Los Angeles Times*. Retrieved from http://articles.latimes.com/2011/apr/25/business/la-fi-himi-mechner-20110425
Pseudo disk][controller card. In *AppleLogic*. Retrieved from http://www.applelogic.org/PseudoDisk2.html
Ross, S. (2000). Changing trains at Wigan: Digital preservation and the future of scholarship. Retrieved from http://eprints.erpanet.org/archive/00000045/01/seamusross_wigan_paper.pdf
Rothenberg, J. (1995). Ensuring the longevity of digital documents. *Scientific American, 272*(1): 24-29.
Rothenberg, J. (1999). Avoiding technological quicksand: Finding a viable technical foundation for digital preservation. *A report to the Council on Library and Information Resources*.
Shustek, L. (2006). What should we collect to preserve the history of software? *Annals of the History of Computing,* IEEE, *28*(4), 112–111. doi:49E7500D-B671-4972-8D0E-A1017B2B878B.
st3fan. (2012). Removed copy protection checks. In *github*. Retrieved from https://github.com/st3fan/Prince-of-Persia-Apple-II/commit/d232cb1e
Stam, D. C. (1993). The informed muse: The implications of "the new museology" for museum practice. *Museum Management and Curatorship, 12*(3), 267–283. doi:10.1080/09647779309515365.
Stebbins, R. A. (2007). *Serious leisure: A perspective for our time.* New Jersey: Transaction Publishers.
Takhteyev, Y. (2012). *Coding places: Software practice in a South American city.* Cambridge, MA: The MIT Press.
The spacious mind. Retrieved from http://www.spacious-mind.com/index.html
Torvalds, L. (2001). *Just for fun: The story of an accidental revolutionary*. New York: Harper Business.
Turkle, S. (1984). The second self: Computer and the human spirit. London: Granada.
Tzvetkov, V. (n.d.). KEGS (Amiga). In *Hirudov.com*. Retrieved from http://hirudov.com/amiga/KEGS.php
Von Suchodoletz, D., Rechert, K., Schröder, J., & Van Der Hoeven, J. (2009). Seven steps for reliable emulation strategies solved problems and open issues. *7th International Conference On Preservation Of Digital Objects*. Vienna, Austria. Retrieved from http://www.ifs.tuwien.ac.at/dp/ipres2010/papers/vonsuchodoletz-53.pdf