Presented at the Electronic Media Group Session, AIC 40th Annual Meeting, May 8–11, 2012, Albuquerque, NM.

CONSERVATION IN COLLECTIONS OF DIGITAL WORKS OF ART

BEN FINO-RADIN

ABSTRACT

Rather than addressing entropy and deterioration of objects, paintings, prints, or textiles, many collections must now contend with ensuring the conservation of works that are entirely born-digital. The constantly shifting compatibility of technologies, and looming wave of obsolescence dictates that this emerging practice is often preemptive, in some cases even leaning towards the archival perspective. Since 1999, Rhizome at the New Museum of Contemporary Art, New York, has maintained an online archive called the ArtBase, a collection containing works of art that employ materials such as software, code, websites, moving images, games, and browsers, towards aesthetic and critical ends. This paper offers three case studies on the conservation of works from Rhizome's collection. These three works will be used specifically to illustrate conservation methods for:

- Works that exist as performative actions within commercial web services or social networking platforms
- Performative web-based works that employ real-time data
- The obsolescence of crucial client-side JavaScript in web based works

Offered in conclusion is a discussion of metadata, from a collection management standpoint, to ensure the long-term sustainability of digital conservation measures.

INTRODUCTION

Rhizome is a non-profit institution (established in 1996) devoted to the creation, presentation, preservation, and critique of emerging artistic practices that engage technology. In 1999 the Rhizome ArtBase was founded as an online archive that sought to collect, preserve, and contextualize the creative practices that had emerged in the preceding decade, along with the advent of the world wide web. The ArtBase collects works that employ materials diverse as software, code, websites, moving images, games, and web browsers towards aesthetic and critical ends. As the years have passed, shifting threats to the longevity of these works have emerged, along with a discourse and philosophy of the conservation of born-digital works of art. In 2011 Rhizome published a paper titled "Digital Preservation Practices and the Rhizome ArtBase" (Fino-Radin), which explored in detail the fundamental threats to its collection, and extrapolated a theoretical framework for approaching the management of these risks. This paper, in contrast, focuses explicitly on methods and techniques of digital conservation and preservation, through the documentation of three case studies. All methods described and documented herein make use of open source tools and common technologies and standards, yielding workflows that are reproducible in similar scenarios. Discussed first is *Legendary Account* (2008–2010) by American artist Joel Holmberg (b. 1982). This artwork will present an approach for preserving works whose primary materials exist within a 3rd party commercial web platform not controlled by the artist or collecting institution. The provided methodology is grounded in web archiving practices, and makes use of a Python tool for the batch production of archival screenshots of web resources. Second, we will consider *i'm here and there .com* (2011), by Dutch artist Jonas Lund (b. 1984), which poses the problem of how to collect and preserve a dynamic web based artwork that included a dimension of time and performance. The approach discussed offers a model of conservation that produces an archival solution without compromising the sensitive, real-time, and performative nature of the work. Finally we will consider *Pulse* (1999) by American artist Mark Napier (b. 1961). This work fell victim to obsolescence of crucial client-side JavaScript. This section will provide a qualitative analysis of various emulation strategies that restored function to the work, as well as consideration of the scalability of maintaining code-based works in collections and archives.

LEGENDARY ACCOUNT

As is often the case in the broad scope of new media and technology based works of art, Holmberg's *Legendary Account* initially posed a non-technical challenge to conventions of curation, collection, and ontology of artworks. In a sense, the work could be said to have no primary form or medium, existing originally as a disparate set of actions. Over the course of two years, the artist engaged users on Yahoo! Answers, asking questions (fig. 1) that range from unanswerable provocations (e.g., "How possible is it to convince people that you are an artist?" June 6, 2008, 4:24 PM), to the absurdly mundane (e.g., "where did you get that chair you are sitting in?" June 19, 2008, 2:28 AM). The work was an attempt to mildly subvert the platform's intended mode of interaction, exposing its inherent limitations, and to engage its users in a manner oddly personal.

The artist exhibited a manifestation of the work in the 2010 exhibition *Free* at the New Museum of Contemporary Art. For the exhibition, the artist painted a gallery wall to match the background color of the Yahoo! Answers website, and mounted digital prints of the work to the wall. This manifestation was a temporary installation, its photographic documentation being the only surviving artifact. The artist had at some point in the work's history documented each question he had asked through his Yahoo! Answers account, as screenshots. These were saved in the JPG format with severe lossy compression, causing the images to display compression artifacts. Surveying these manifestations, instantiations, and documents of the work, it emerges that they as a whole constitute the work, rather than one discrete artifact. The goal then

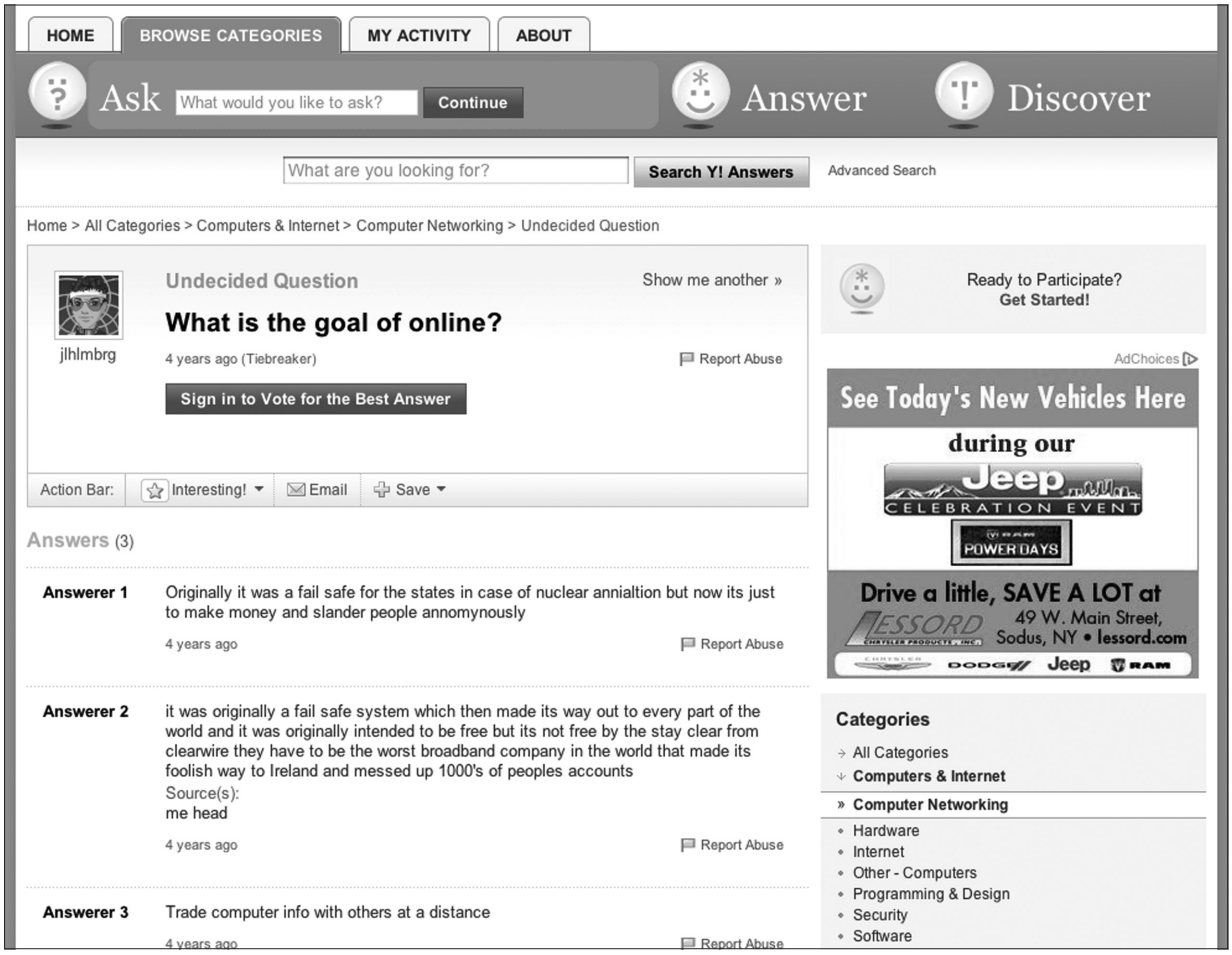

Fig. 1. Holmberg asks: "What is the goal of online?"

was not so much the conservation and preservation of the material as the artwork itself, but rather, accurate and future-proof documentation of evidence of the work. The material that was the target of this documentation was the original collection of web pages that contained the individual questions belonging to Holmberg's Yahoo! Answers account and his profile page, which contained links to each of these individual questions. It was desired to archive these pages such that the links between all of them were preserved, so that an offline version, or one hosted in a repository would be browsable as it was originally within the Yahoo! Answers platform.

The requirements for selecting the appropriate tool included primarily that: (1) the tool offered the ability to perform a precise, targeted crawl of a list of specific URLs, and (2) that the tool provided the ability to output the Web ARChive file format (WARC) (National Digital Stewardship Alliance 2011), as well as the retrieved materials in an uncompressed directory structure. In typical large-scale web archiving settings, where the intent is to perform broad crawls of entire websites, recursion is heavily employed. The first page hit by the crawler is parsed to identify hyperlinks so that these linked pages may be also crawled, archived, and parsed for additional pages, at which point the process continues indefinitely depending on the level of recursion specified. Recursion was not an option in this case, as this would cause the crawler to extend far beyond scope, eventually downloading Yahoo! Answers in its entirety. Thus, it was cru-

cial that the tool used allow for a list of targets to be specified. While WARC is widely considered to be the standard archival format for web resources, it requires additional software (such as the Internet Archive's Wayback software) (Internet Archive 2011) in order to render the archived materials. For this reason it was desired that in addition to WARC, to also output the crawled content in a standard directory structure, preserving the relative directory structure. This format of output yields a collection of files that may be hosted and served from any web server, without special software. The GNU Wget (GNU 2012) software was selected as the tool of choice, as it offers all of the aforementioned aspects of utility and has a very diverse community of users, which yields a support system. Wget is quite adaptable to small scale, highly specific and customized crawls, and as of recently, offers the option to output crawls in WARC format as well as a directory structure.

The methodology and procedure used with Wget was to first compile the list of URLs to be included in the crawl. This list was saved as a plaintext file named "url_list," with each URL on a new line, without a delimiting character. Wget is a non-interactive command line application, meaning it is issued as a command, with all desired options following the "wget" command as flags. The following options were used at the time of crawl:

wget -e robots=off --random-wait -p -H -nd --restrict-file-names=windows -k -i url_list --warc-file=LEGENDARY_ACCOUNT --warc-header="Operator: Ben Fino-Radin"

The first option, "-e robots=off" instructs Wget to ignore robots.txt, a file that defines the areas that a website's crawlers may and may not access. While robots.txt was originally well intentioned by preventing crawlers from overloading servers with requests (Charlie 2007), it can prevent significant amounts of material from being archived. It is advisable to ignore robots.txt with care, as it is poor etiquette to conduct broad crawls that ignore the intentions of the web site designer to control the audience for their content. In this case our crawl was quite limited, and crucial components would have been omitted were robots.txt to be respected. HTTP based crawlers are only able to access that which is publicly accessible, so ignoring robots.txt would seem to pose no privacy threat. The second option, "--random-wait" is another option that must be used with caution and care. Some servers are configured to detect patterns in multiple requests from the same user, for the purpose of blocking a crawler making requests on a predictable interval. "--random-wait" causes Wget to randomize the intervals at which it makes requests of the server, thus eluding detection (GNU 2013c). The third flag, "-p" tells Wget to retrieve all page "requisites." Without this flag, none of the images, video, sound, linked CSS (cascading style sheets), and JavaScript files—essentially any embedded content—would be retrieved. The fourth option, "-H" specifies that the crawl may span hosts. If the pages listed in "url_list" were at the http://answers.yahoo.com host, but all embedded images were hosted at http://l.yimg.com/, without the "-H" flag these images would not be retrieved, even with the "-p" flag having been invoked. When conducting a recursive crawl (which we are not) this flag is highly problematic, as it allows Wget to follow hyperlinks to pages on other hosts, causing the crawl to potentially spiral out of control, recursing across the entire web. This is one of the major flaws of Wget. It currently does not offer the ability to conduct a full recursive crawl of a website, including embedded content hot-linked from other hosts, without traversing across pages on another site. The fifth flag, "-nd" specifies that we do not want Wget to produce a directory structure that preserves the original URL structure of pages and resources (GNU 2013b). Conventionally, this would be inadvisable, as it is best practice to maintain this provenance. Here we take this measure, as certain assets will require post-crawl processing, and a complicated system of relative paths would make this process painstaking. The sixth flag, "--restrict-file-names=windows" was used, as many image assets included characters in their filenames that would be problematic further down the road.

For example, question marks present in the filenames of images served by Yahoo! Answer's content delivery system is the result of variables passed to their system. In hosting and serving the archived files, these characters would be mistakenly interpreted by Apache as being variables, thus causing the images to not load embedded in their respective pages. The seventh flag, "-k" tells Wget that after completing the crawl, all links should be converted. All links between downloaded pages, as well as the URLs of downloaded page requisites will be modified to point to the relative path of the local resource. The eighth flag, "-i" indicates that rather than providing one URL to start the crawl from, Wget should read the contents of the proceeding text file (in this case "url_list"), and crawl any URLs included in this file (GNU 2013a). The final two flags relate to the WARC format. "--warc-file=LEGENDARY_ACCOUNT" sets the file name of the WARC file, and "--warc-header="Operator: Ben Fino-Radin"" allows for the optional inclusion of inherent metadata. Here we are simply specifying the person that was the operator of the crawl.

Completion of a crawl with these parameters results in a complete, archival capture of the content served by the URLs in "crawl_input". According to conventional web archiving practices, this would be sufficient; however, Rhizome's standards of quality control place a strong emphasis on preserving the original look and feel of all resources. Seven pages included in the crawl contained a graph or visualization of Holmberg's account activity. These visualizations were Adobe Flash-based, and did not function properly in a crawled version of the pages. This is due to the fact that the embedded SWF objects were attempting to make calls to data still hosted by Yahoo. Such activity is prevented by Adobe Flash's default security settings, and, in this case, a simple HTML table is rendered in its place. In order to preserve the look and feel of these visualizations, a lossless PNG screenshot of the functional visualization was produced that was cropped precisely to the size of the Flash resource. The source code of the pages containing the Flash resource was modified to instead contain this PNG as an embedded image. The original Flash-based visualization was non-interactive, so in terms of look and feel, this PNG replacement was a perfect stand-in, and provides a stable solution for the future. This concluded the crawl and quality control process. The directory containing the crawl was uploaded to the server that hosts the ArtBase's repository. All pages, both functionally and visually, are identical mirrors of their original state on Yahoo! Answers.

The next stage of documentation was to produce archival quality screenshots of the pages. This step was undertaken not only for the purpose of having image-based documentation for the ArtBase catalog, but also for the purpose of preserving the look and feel of how the pages are rendered by web browsers contemporary to the time of the work's creation. For this process, an open source Python tool called webkit2png was used (Hammond 2013). This is a command line tool that produces PNG images of websites as rendered by the webkit browser engine (used both by Apple Safari and Google Chrome web browsers). Initially an attempt was made to produce PDF file captures rather than PNG, as this would maintain the machine readability of the page's content. While there is a forked version (Yig 2013) of webkit2png that also offers PDF output, this approach was found to be less trustworthy in terms of digital longevity. The PDFs created by the webkit2png PDF fork do not adhere to the PDF/A (Adobe's PDF/Archive) specification, and conversion to PDF/A was found unstable due to the heterogeneity of fonts. It was decided that since these captures were ancillary to the web archived pages, which were, of course, machine readable and indexable, that the image captures need not maintain this aspect. Moreover, PNG is a widely used format with little to no preservation risk. The webkit2png tool was employed as follows:

```
cat url_list | webkit2png -W 2000 -F --delay=6 -
```

Cat is a standard Unix program that reads the data in any file and prints its contents to the command line (standard out, or stdout). Here it is used to read the same list of URLs that were fed to Wget. The "|" character that follows is known as a pipe. This allows the use of programs and pieces of software together as building blocks. In this case, it enables passing the text output from cat to webkit2png. The first flag following the webkit2png command, "-W" and its following integer, defines the width of the image to be captured. It should be noted that varying this size is synonymous with resizing a browser window. It does not increase the size of elements in the web resource, but simply increases the visible area of the browser window. The "-F" flag states that only the full size PNG should be produced (without this a thumbnail image is produced by default). A delay of six seconds is then set, ensuring that all content has completely loaded before the image is captured. The final character in the command, "-" tells webkit2png, that instead of entering one URL, it is to read from standard input. This along with the pipe that precedes the webkit2png command, is what allows one to "cat" the URL list, and pass it to the webkit2png program.

With this, and the Wget process as outlined, one is left with three forms of documentation: a WARC file of the entire crawl (which requires the Wayback software in order to be rendered); the crawled content in a directory; and a directory containing the PNG captures of the pages. Given the performative nature of *Legendary Account*, these documents are not the work per se, but are the most complete and accurate forms of documentation that can be retrieved.

i'm here and there .com

Jonas Lund's (b. 1984) piece *i'm here and there . com* (2011) is a web-based work that deals with privacy, transparency, and performance. Upon visiting http://imhereandthere.com, the viewer is presented with large black text on a white background—the text being the URL of another website. In most cases, if the viewer stays longer than thirty seconds or so, the URL in black text will change. This is because *i'm here and there . com* is a window into the artist's web browsing. The URL displayed on the website is whatever URL happens to be open in the artist's web browser at that precise moment (within the second).

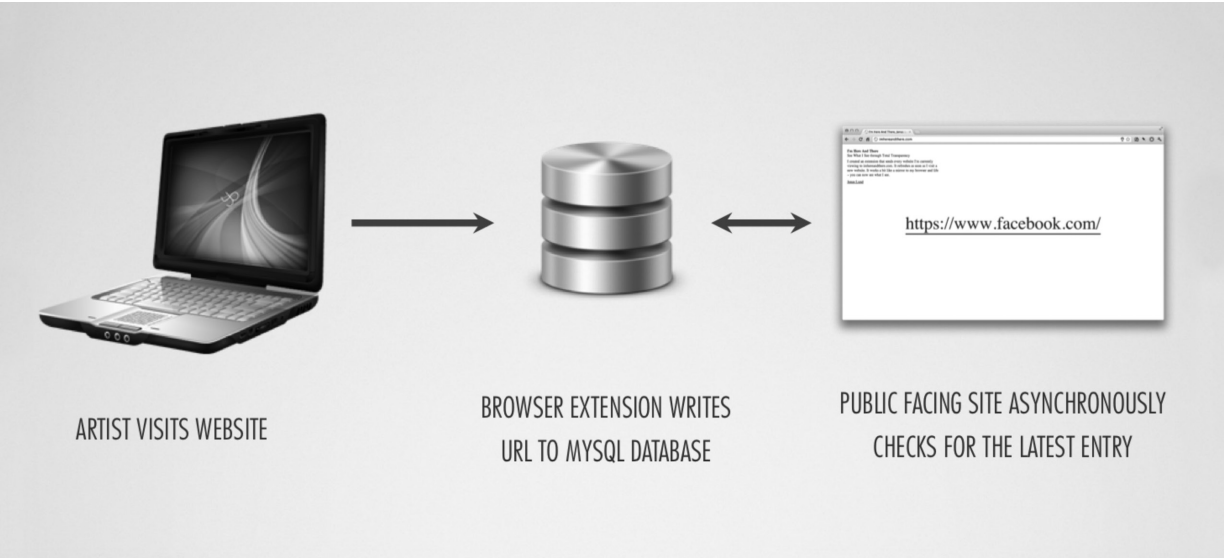

Fig. 2. Technical flow of Lund's imhereandthere.com.

The work functions through the cooperation of a browser extension, a MySQL database, and a web page with asynchronous javascript. The process can be illustrated by following the data from the point of creation (fig. 2). When Lund visits a new website, or switches tabs to a website that is already open, the custom extension present in his web browser communicates with a MySQL database hosted on his server, writing a new row. The database consists of only one table, with three columns: an auto-incrementing integer that serves as the primary key; the timestamp of when the data was recorded; and the URL that was visited. In essence, it is a minimally formatted browser history database. The public facing website consists of an index.php page containing the markup and structure of the page, the PHP includes a configuration file with database credentials, an embedded local copy of jquery 1.5.1, and an embedded file called "script.js". These files work together such that when the viewer loads http://imhereandthere.com, the script calls to the database, retrieving the most recent row in the database—which is the last URL to be visited by Lund. The response from MySQL is passed to a PHP variable, which is used to echo the URL inside a specific DIV container on the index.php page.

Our mission in conserving this piece was to acquire a functional version of the display page, and to have a locally hosted mirror of Lund's database in order to host a fully functional or "living" mirror of the work in our archive, without compromising the real-time performative dimension of the work. In order to accomplish this, the artist provided two essential components, the first of which was an SQL dump of his database. SQL dumps of relational databases contain not only the contents of a database, but SQL commands for creating the tables and relations needed to structure the data and provide meaning to it. Rhizome created a new database on the archive ArtBase MySQL host, and imported the SQL file, thus creating an exact duplicate of Lund's database at the time the data was dumped. Lund then modified his

Fig. 3. Reworked and archival flow of imhereandthere.com.

browser extension so that in addition to writing data to his own database, it would also push the same data to Rhizome's database. Second, Lund provided a directory containing index.php, style.css, jquery-1.5.1.min.js, script.js, xhr.php, and config.php—all of the files required for rendering the page, and for communication with the database. Config.php contained the MySQL hostname and login credentials, which Rhizome modified to look to the database hosted in the ArtBase archive. In the end, this process yielded a completely dynamic and living version of the work, hosted entirely by Rhizome (fig. 3). As long as the artist continues his project, the work will continue to live, both in the wild—and in the archive. In the event that the artist's browser extension were to stop functioning, any data not recorded would be identical across both the Rhizome database, and the artist's database.

Looking forward, there is likely only one aspect of the work remotely at risk of obsolescence—and that is the jquery-reliant JavaScript that asynchronously communicates with the database. While the jquery library used by the work is hosted locally, the script that commutates with the database executes on the client side—meaning that it is reliant on support by the JavaScript interpreter of the viewer's web browser. This method of asynchronous data retrieval by way of jquery is completely commonplace, but it is entirely feasible (if not probable) that it will eventually obsolesce. Given the utter ubiquity of jquery this may seem improbable, however, it is indisputable that in twenty years it will be an antiquated technology. In such an event, this method of communication with the database could be replaced with any other asynchronous client side code. Due to the technical simplic-

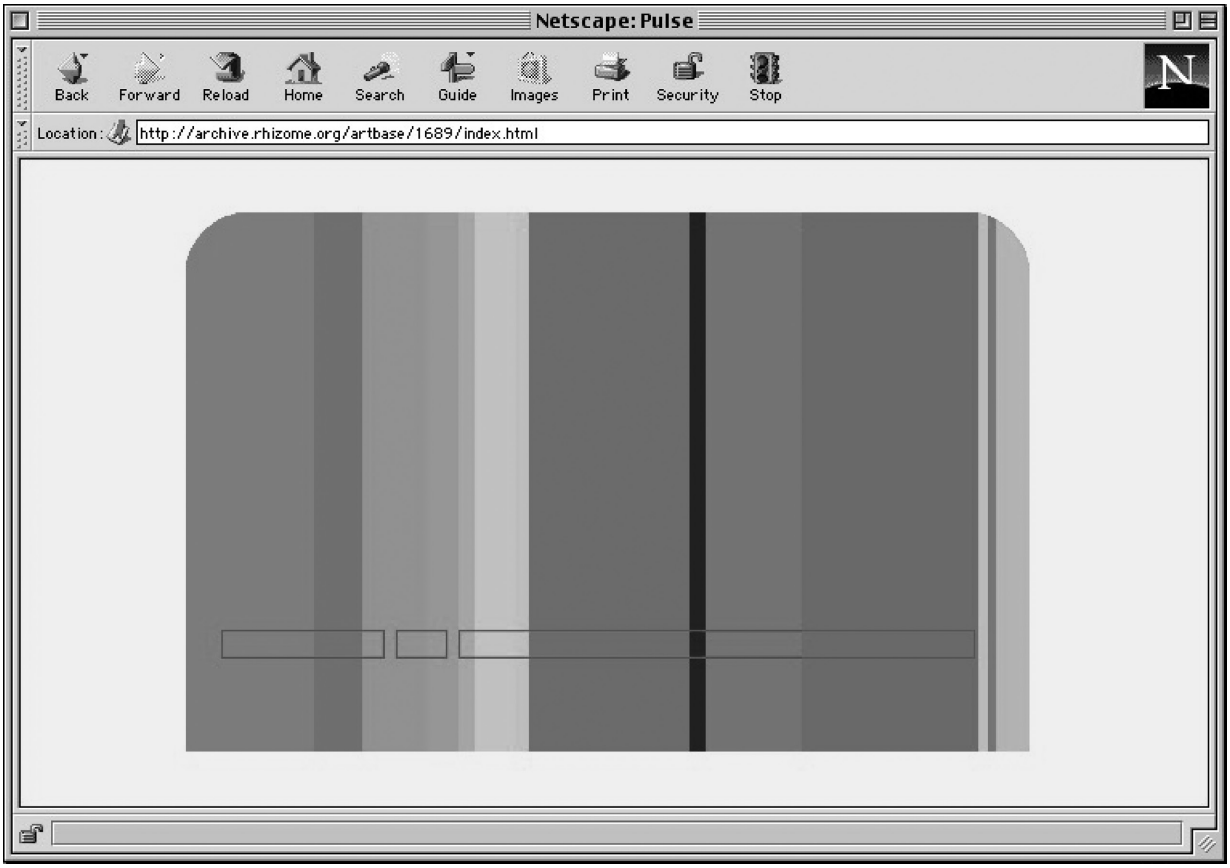

Fig. 4. Screenshot of Pulse (1999) by Mark Napier.

ity of the work, all behind the scenes technologies are replaceable—especially the database technology being used. In this example, the technologies themselves lend no artifactual authenticity or aura to the work. As seen in the next artwork, this is seldom the case.

PULSE

The third and final work to be considered is *Pulse* (1999) by Mark Napier (b. 1961). The work is a web-based piece that consists of a simple, interactive abstract pattern (fig. 4). A rectangle divided into vertical stripes reacts to the movement of the viewer's mouse, oscillating between colors, and reacting to clicks of the mouse. The work, which speaks to the time in which it was created, is a simple browser experiment–it explored the capabilities of JavaScript in 1999. Unfortunately, the work has entirely ceased to function due to the obsolescence of its JavaScript. The web page loads, however no activity whatsoever occurs, and it does not respond to user interaction.

With JavaScript, elements of a page are manipulated by their name in the Document Object Model (DOM) (W3C 2005). In the early web, during the "browser wars," users were often informed by a website (often in the form of an animated gif—such as "Netscape Now!"), that the current site should be viewed in Netscape Navigator, rather than Microsoft Internet Explorer (or vice versa). Sometimes this was a mere preference of the user that created the site, but in some cases, it was due to actual functional aspects. Netscape Communications originally created JavaScript in 1996, specifically for the Netscape Navigator browser. Initially, any JavaScript written for Netscape simply would not function in Internet Explorer. An official standard for JavaScript was not published until 1997. For this reason, the source code of Napier's work includes a few lines that conduct what is called "browser sniffing." In other words, he wanted the work to function in both popular browsers of 1999, and so his code was written to provide alternate paths depending on the browser being used by the viewer. For example:

```
function setbgColor(fader, r, g, b) {
        hr = hex(r); hg = hex(g); hb = hex(b);
        if (NS) {
                fader.styleObj.bgColor                =
"#"+hr+hg+hb;
        } else if (IE) {
                fader.styleObj.backgroundColor        =
"#"+hr+hg+hb;
        }
}
```

These lines define a function that sets the background color of specific DOM elements. Napier's code tells the browser that if the user is running Netscape, to set the "bgColor" of a DOM element, and if they are running Internet Explorer, to set the "backgroundColor" of the element. The key though is that this is defined through an "if/elseif" statement. In other words, if Netscape Navigator is being used, do this; if Internet Explorer is being used, do that. However, there is no exception or alternative offered. This means that if the viewer were using neither browser, the code would simply not execute, and so the work itself ceases to function. Of course it is doubtful that anyone would be using a Netscape browser these days. Furthermore, even if an exception were to be provided, or if the user were to be using Internet Explorer, DOM manipulations from 1997 would simply not work. The standard has evolved, and the elements referred to in the code have been renamed.

In approaching the conservation of this work, Rhizome conducted an initial investigation of the artist's Javascript. In the research process, the artist was interviewed in order to aid in the documentation of his original code and to attempt to isolate the issue. Interestingly enough, before the issue was isolated, the artist repaired the code himself. While in hindsight, the cause of the issue is quite clear upon reading the code, it presented quite the challenge upon initial investigations. Reading another person's JavaScript can be an arduous task, and the artist, knowing his code quite well, was able to im-

mediately isolate the problem. The updated code now functions in modern browsers, preserving its original behavior entirely. Beyond function however, one must wonder if something is lost by interacting with the work on modern systems. While it is certainly desirable for computer based works in a public archive to migrate across platforms when possible, for ease of access to the viewer, the software that surrounds the work—the web browser and operating system, provide insight into the aesthetic, temporal, and cultural context of a work. *Pulse* is an example of a work that relies solely on its formal, functional, and interactive aspects—all of which seem insignificant when viewed on a modern platform. However, when viewing the work within a web browser of the period (such as Netscape Navigator version 4.03), as well as an appropriate operating system (such as Mac OS 8), new life is breathed into the web-based work. The work is then placed within a specific historic place and time—the software and computing ecosystem that surrounded the work at its time of creation. It is easier to experience how it must have felt to experience the work for the first time in 1997, when browser based interactivity carried a degree of novelty.

The absolute ideal environment for such cases of a preferred authentic rendering, is to employ period-specific (vintage) hardware. Fortunately, Macintosh and PC systems from the late 1990s are still widely available. However, in the long term these machines will inevitably cease to exist. Thus, the sustainable path forward for any sort of viably authentic environment, is emulation. It hardly warrants explanation, but emulation is the process of simulating the processor architecture of specific vintage hardware, within a host environment of a modern or contemporary computer. The simulation of this vintage hardware allows one to run obsolete operating systems and software that are compatible with the given emulated environment. This practice is used considerably within the diverse digital preservation community, and quite extensively within communities of gamers and computing enthusiasts. Fortunately there are many free, open-source emulation platforms, which have wide adoption across many communities. This bodes well for the sustainability of emulation as a conservation solution, as adoption is the strongest safe-guard against obsolescence.

Producing an emulation with high fidelity to performance level on original hardware can prove challenging. Despite running on a very powerful host machine, certain operating systems and emulation environments can be sluggish. In testing a variety of emulation environments (see table 1), the most accurate and responsive was found to be an emulation of Mac OS 8.1, running in Basilisk II (Bauer 2013a), simulating a Macintosh Quadra with a Motorola 68040 processor, 64 MB of RAM, and Netscape Navigator version 4.03 as the web browser. This configuration exhibited the highest degree of fidelity to the perfor-

Emulator	Emulated System	Browser	Performance
Basilisk II	Mac Quadra, 68040 CPU, OS 8.1, 64 MB RAM	Netscape 4.03	4/5
SheepShaver	Mac OS 9, 512 MB RAM	Netscape 4.03	3/5
VirtualBox	Windows 95, 512 MB RAM, 1 CPU core, 100% execution cap	Netscape 4.03	2/5
VirtualBox	Windows 98, 1082 MB RAM, 1 CPU core, 100% execution cap	Netscape 4.03	2/5

Table 1.

mance of the original code on native or vintage hardware. For each of these tests, the host system was a mid-2011 Apple iMac, with a 2.7 GHz Intel Core i5 processor, 16 GB of RAM, and running OS X 10.8. Other environments tested included SheepShaver (Bauer 2013b), running Mac OS 9, and VirtualBox (Oracle 2013) running both Microsoft Windows 95 and 98. Despite the challenges in executing emulation that exhibits high fidelity to a native environment, this does offer a solution far more scalable to large collections. The type of maintenance and hands-on migration of source code (in this case executed by the artist) is costly, and denies the work a historically accurate aesthetic context.

TYING IT ALL TOGETHER: METADATA

The task of tracking and documenting preservation actions with born-digital works, or variable new media works, is inherently challenging. While there are plenty of collections management, digital asset management, and digital preservation systems, none of these quite meet the requirements for collections of born-digital works of art. The ArtBase's current management system is a Django- based web application, which has served the collection well for several years, but lacks the ability to document and track preservation actions and subsequent versions of works. While it offers standards-based structural metadata, the system does not enforce metadata content standards. Rather than improving upon this database, or reinventing the wheel by building a complete collections management platform from scratch, Rhizome has opted to implement a pre-existing, open-source, highly configurable collections management system called Collective Access. This system was designed such that it is completely customizable according to the individual needs of a given institution. This is accomplished through an XML file called a "profile." Rhizome has developed (currently in draft stage) a profile that is finely tuned for the needs of the ArtBase, and is adaptable to any large collection of new media and born-digital works of art.

There are two main components of this configuration that aid in monitoring works for obsolescence: documenting technical requirements and documenting conservation actions. The profile defines two top-level object types, artworks and technologies. The artwork sub-types are: image, moving image, performance, software, sound, and web. The technology sub-types are: code, file format, hardware, operating system, and software. The key component however is the ability to define relationships between works and technologies, and from one type of technology to the other. These relationships are established through an ontology consisting of the following types of relationships within the Collective Access platform: creating application, required for display, preferred for display, required for edit, and known compatibility. This improves greatly upon the ArtBase's former system, which allowed technologies to be correlated with works, but made no semantic distinction as to the nature of the relationship. This new ontology affords finer granularity in the specification of relationships between technologies and artworks, as well as the ability to specify relationships between technologies.

For example, in the case of *Pulse* (1999), one could establish the "Preferred for Display" relationships between the work and Netscape Navigator 4.03. This in and of itself is useful metadata, but it is made all the more stronger by the Netscape 4.03 entry having an established relationship between itself (object type software), and a record for Mac OS 9, Windows 95/98 (Known Compatibility being the relationship type), and any other compatible operating systems (object type Operating System). The records for these operating systems would in theory have established "Known Compatibility" with either specific models of hardware or emulation environments. This degree of relation in the system would over the course of time create an incredibly useful database for documenting operational requirements, for exhibiting works, and also for monitoring of obsolescence across the collection. If a particular work ceases to function,

and the issue is identified as an obsolete format, language, or software, all works in the collection that share this technology are easily identifiable. Furthermore, this new system should prove to be an invaluable curatorial tool, for example, having the ability to search for all works that should be displayed on Netscape Navigator 3.

CONCLUSION

Despite presenting highly unique scenarios, and specific technical methodologies of conservation, the hope is that these three cases may provide a framework for future endeavors. Looking across all three cases, the following important concepts emerge: First, with time, software provides a cultural and aesthetic context. While challenging in terms of fidelity, emulation offers an elegant and scalable solution, arguably more sustainable than the hands-on maintenance and migration of source code. Second, digital conservators must posses the ability to build new tools, or create mash-ups of existing ones. Every work presents unique challenges, for which there is no magic out-of-the-box solution. Finally, preemptive conservation is crucial with born-digital works. The best preservation policy is one that by design ensures the longevity of the work on a fundamental level, before the need for action is truly dire. Direct collaboration between conservator and artist is a must, for documentation of preservation rights, technical specifications, and original intent.

REFERENCES

Bauer, C. 2013a. The official Basilisk II home page. http://basilisk.cebix.net (accessed 09/01/12).

Bauer, C. 2013b. The official SheepShaver home page. http://sheepshaver.cebix.net (accessed 09/01/12).

Charlie. 2007. "Historical footnote: where robots.txt came from." Posting on Slashdot on December 2, 2007. http://yro.slashdot.org/comments.pl?sid=377285&cid=21554125 (accessed 09/01/12).

Fino-Radin, B. 2011. Digital preservation practices and the Rhizome ArtBase. http://media.rhizome.org/artbase/documents/Digital-Preservation-Practices-and-the-Rhizome-ArtBase.pdf (accessed 09/01/12).

GNU. 2012. Introduction to GNU Wget. Free Software Foundation, Inc. www.gnu.org/software/wget (accessed 09/01/12).

GNU. 2013a. "2.4 logging and input file options." GNU Wget 1.13.4 Manual. Free Software Foundation, Inc. www.gnu.org/software/wget/manual/wget.html#Logging-and-Input-File-Options (accessed 09/01/12).

GNU. 2013b. "2.6 directory options." GNU Wget 1.13.4 Manual. Free Software Foundation, Inc. www.gnu.org/software/wget/manual/wget.html#Directory-Options (accessed 09/01/12).

GNU. 2013c. "2.5 download options." GNU Wget 1.13.4 Manual. Free Software Foundation, Inc. www.gnu.org/software/wget/manual/wget.html#Download-Options (accessed 09/01/12).

Hammond, P. 2013. webkit2png. www.paulhammond.org/webkit2png (accessed 03/14/13).

Internet Archive. 2011. Wayback. http://archive-access.sourceforge.net/projects/wayback (accessed 09/01/12).

National Digital Stewardship Alliance. 2011. "WARC, Web ARCive file format." Sustainability of digital formats: Planning for Library of Congress collections, compilers A. Arms and C. Fleischhauer. www.digitalpreservation.gov/formats/fdd/fdd000236.shtml (accessed 09/01/12).

Oracle. 2013. VirtualBox. www.virtualbox.org (accessed 09/01/12).

W3C. 2005. "What is the document object model." Document Object Model (DOM). www.w3.org/DOM/#what

Yig. 2013. "webkit2png." Posting on Github, n.d. https://github.com/yig/webkit2png (accessed 09/01/12).

Ben Fino-Radin
Rhizome at the New Museum
235 Bowery
New York, NY 10002
ben.finoradin@rhizome.org
bfinoradin@gmail.com