Acid-Free Bits

Recommendations for Long-Lasting Electronic Literature

v1.0 June 14, 2004

By Nick Montfort and Noah Wardrip-Fruin

Contents

Preface: Acid-Free Bits and the ELO PAD Project

With the release of *Acid-Free Bits* (version 1.0), the Electronic Literature Organization (ELO) brings to the public concerns that have been debated here for at least two years. The document is a plea for writers to work proactively in archiving their own creations, and to bear these issues in mind even in the act of composition. The destinations of print literature — in stores, as a node on Amazon, on the library shelf and ultimately in a consensual canon — are by now so prevalent that few print writers give much thought to preservation. No such destinations exist, however, for born-digital writing.

The best that current digital repositories have accomplished is a home, not for literature, but for scholarly journals. Text files linked with graphics are preserved, not programs; individual works are indexed, not networks; and the most advanced use of technology is the hyperlink. We have a great deal of information but very little structure or context. Contrary to expectations, the electronic archive has not been conducive to the creation and circulation of literary writing. Writers of electronic literature have needed, to a degree unimaginable in print, essentially to build the environment for their own work. Those who neglect basic preservation issues very quickly discover the mess that technological obsolescence makes, not only of individual works, but of the connections among texts, images, and code-work that are crucial for any sustained, community-wide literary practice.

In response, the ELO committee for the Preservation, Archiving, and Dissemination of electronic literature (PAD) commissioned Nick Montfort and Noah Wardrip-Fruin to draft an appeal directly to authors, in the hope that the creative component does not separate out from the curatorial. *Acid-Free Bits* initiates a campaign, online and in pamphlet form, that will develop in three stages. Stage 2 will add to the mix further explanations of metadata and community-owned standards, preferred presentation languages, and complementary preservation efforts worldwide (for example, Archiving the Avant-Garde). This stage of the ELO preservation campaign will be addressed not just to authors but also publishers, librarians, software developers, and others. Stage 3 will create, through the efforts of archivists and programmers, a working framework of standards and practices as well as implementation tools for authors and publishers. This third stage of the effort will depend on widespread support from institutions and granting agencies in addition to continued, coordinated input by individual authors and members of ELO.

Readers will notice that the Web edition of *Acid-Free Bits* is itself a model of standards and best practices circa summer 2004, created in a non-proprietary, open, XML-based language that facilitates the kind of cross-referencing, mixing, and institutional networking that the Organization as a whole, working collectively, hopes to produce in the coming months and years. As the campaign continues, it will include a site where authors can come, communicate, and contribute to the preservation effort, and know that they are not creating in an isolated world. Before long, the issues identified and addressed by Montfort and Wardrip-Fruin will broaden out to a key area for ELO investigation — namely, X-Lit. X-Lit will be an XML framework for representing and migrating works of e-literature (including both content and dynamic behaviors). Supplemented by implementation tools, X-Lit will be designed so that authors, publishers, archivists, academics, and progammers can more easily collaborate in the preservation and dissemination of e-literature. The concept for X-Lit, and the development of open-

source interpreters and emulators designed specifically for e-literature, will be the subject of the next ELO release, a white paper by Alan Liu.

For the ELO Board and PAD Core Group,

—Joseph Tabbi

1 Keeping E-Lit Alive

Electronic literature doesn't come on bound, offset-printed pages. Keeping it on a shelf doesn't mean that it will be easy, or even possible, to read it in the future. Even putting it into a vault with controlled temperature, light, and humidity won't ensure its availability.

The new possibilities of electronic literature come from its being as much software as document, as much machine as text. For electronic literature to be readable, its mechanisms must continue to operate or must be replaced, since changes in the context of computing will complicate access to important works of literature on the computer. The context of computing includes operating systems, applications, the network environment, and interface hardware — and this context is constantly evolving. A piece of electronic literature written for a Macintosh in the 1980s may be unreadable on the Macs in a college computer lab today. But e-lit can become unreadable much more quickly, as an upgrade to the next version of the authoring or reading software introduces unexpected problems.

Some approaches to creating e-lit are more likely than others to result in work that is preservable. *Acid-Free Bits* provides information to help authors find ways to create long-lasting e-lit, ways that fit their practice and goals. To explain why this information is being offered, we'll first answer two central questions:

Why preserve electronic literature?

If special efforts aren't made now, students, professors, authors, and readers won't be able to access many important works of electronic literature in the future. For electronic literature to contribute to our culture, it's important to have works that readers can return to later, literature people can recommend to others with some assurance that the work will still be available in readable form. Preserving e-lit, and creating e-lit that will remain available, is essential to the very concept of electronic literature, the basic idea that the computer can be a place for new literary works that make use of its capabilities.

Who should preserve electronic literature?

Preservation is always the work of a community. Ultimately, preserving electronic literature will be the work of a system of writers, publishers, electronic literature scholars, librarians, archivists, software developers, and computer scientists. But today, and even once such a system is in place, the practices of authors and publishers will determine whether preserving particular works is relatively easy or nearly impossible. The goal of this document is to help electronic writers make informed decisions about their practices. What authors do today will not only

shape the future of their own work, it will help direct the overall course of electronic literature.

2 Acid-Free Bits

Artists who sketch on acidic newsprint generally know that, in a few years, the paper they use will be brittle and discolored, perhaps rendering their work illegible. Probably only a few who used the HTML extensions introduced with Netscape 4 realized that their work might meet a similar fate.

At least HTML is stored in plain text and is human-readable, so people can see what an HTML file says, even if the special browser it was written for isn't easily located. The situation can be worse for writers who used authoring systems that produced binary files. These files may be unintelligible except by proprietary programs no longer available — or except by programs that would not run on today's operating systems, even if a copy could be located.

Authors of electronic literature may choose to work with materials that are less than durable, just as sculptors choose to work with cardboard and even ice. Performances, too, are by nature temporary. But even with such temporary art, there is often more that remains than with electronic writing that has gone 404 or will no longer run on any systems that are on hand. Sculptors using cardboard may still do their preliminary sketches in acid-free notebooks. Ice sculptures are photographed. A production of an off-Broadway play leaves a script, a playbill, photographs, reviews, and probably a videotape in the Lincoln Center's library. It's possible to appreciate, and learn from, the documentation of a show that closed a decade ago. Web writing from the same era is sometimes only known about from brief entries that remain in unpruned hotlists.

There are a number of different approaches to the preservation of born-digital art, such as electronic literature. This guide doesn't advocate that all authors take all such approaches. Nor is the advice here supposed to train authors to be their own digital media archivists. Rather, this is an outline of some important principles that, when used to guide literary practice on the computer, can make it easier for e-lit to remain accessible, functional, and enjoyable for future readers.

3 How Electronic Literature is Preserved

3.1 Old Hardware Is Preserved to Run Old Systems

This is a costly and difficult option, but one that has been managed in a few places with old video game cabinets and other historical electronic art. It is one of the ways individuals and institutions may preserve electronic literature in its original material form — but even if this method is successful, only those who visit the preserved hardware will be able to access the work.

3.2 Old Programs Are Emulated or Interpreted on New Hardware

Free emulators and interpreters have made it possible to run many older programs on modern computers. Emulation and interpretation allow the original files of an e-lit work to be used, providing mechanisms that bridge between the original file formats and current platforms. As long as some strong interest in work from certain older platforms remains, it is likely that emulation or interpretation of them will be an option — new emulators and interpreters will continue to be developed for new platforms. For platforms or programs that are obscure or programs that are particularly demanding (requiring network access or special hardware, for instance) this option may not always be realistic.

3.3 Old Programs and Media Are Migrated to New Systems

By converting programs and data files from their original format to modern formats, e-lit works can be made to run again. These sorts of migrations are specific to particular programs and data formats — when they can be done, the results are of varying fidelity. In addition, the conversion may have to be done several times (from the original format to a 1998 format, from a 1998 format to today's platform, and again in the future) and each migration risks introducing new problems. In these cases, the migration process becomes part of how the e-lit received by current readers was authored, and documentation of the transformations should be available to them.

3.4 Systems Are Documented Along with Instructions for Recreating Them

Although this option does not preserve people's ability to interact with an e-lit work as they did originally, it often provides a practical way for people to get an idea of what an e-lit work was like. If documentation is thorough, it may even allow a work to be re-created in the future.

4 Principles for Creating Long-Lasting Work

4.1 Prefer Open Systems to Closed Systems

An open system is one whose essential workings are fully, publicly documented; an open standard is published and available to anyone. Those who use open systems and adhere to open standards when creating electronic literature have a much better chance that the format of their literary works will be supported, or decipherable, in the future. The small group of people in charge of a closed system or standard may lose interest and stop developing software, or the small group may change the system or standard without warning, so that older works of electronic literature no longer work on new platforms. (This is particularly a risk when electronic literature is not the main purpose of a system, and may be obliterated incidentally.) Open systems and formats can be most easily migrated and emulated, since their specifications are publicly known. Closed systems are far more difficult to migrate and emulate.

A closed system may provide important capabilities that are otherwise not available, and some closed systems may be very well suited for the type of literary creation in which authors are interested, so there may be good reasons for authors

to use a particular closed system. However, authors should be aware that such a choice could affect the longevity of their works. As a result, authors may wish to document such projects more thoroughly.

4.2 Prefer Community-Directed Systems to Corporate-Driven Systems

A corporate-driven or proprietary system is one in which the underlying technologies are controlled by a corporation, even if they may be fully documented and therefore open. PDF, Flash, and Java are examples, although Sun (and to some extent Adobe and Macromedia) have taken measures to involve the community in the development of these technologies. However well-documented an e-literature technology may be, there are specific risks to the longevity of corporate technologies that aren't present in free and academic systems. The case of multimedia authoring software mTropolis is illustrative. The teetering company mFactory Inc., which produced this software, was seemingly saved by a buyout from the desktop publishing company Quark in 1997 — only to be shut down the following year just before the next version of mTropolis was to be released.

> **Flash** is a system used to create interactive media and programs, usually to be delivered on the Web. It began as a tool for developing interactive scaling animations. It is a corporate-controlled system, with developer Macromedia Inc. directing the system and its formats. Project files (.fla) are in a proprietary binary format. Files for distribution (.swf) are in a publicly documented binary format. These .swf files can be generated and parsed by Macromedia's products (including browser plugins) and by other software, such as Apple's Quicktime products. ¶ Flash has been widely adopted, and it does offer capabilities that are essential to some electronic literature authors' practice. However, we recommend against using Flash to create elements that can be easily created in open, non-proprietary systems. These include still images and panes of text with scrollbars. For Flash projects (or projects in any other format) that require a browser window with certain characteristics, it is recommended that a launch page that can create such a window be kept directly with the work, not off on a table of contents in a different directory. Formats that are more preservable than Flash may emerge as useful in the future; one candidate is SVG, the Scaling Vector Format, an application of XML. **"Macromedia Flash Considered Harmful"** provides arguments against the use of Flash in accessible Web sites; **"Is 'Open' and 'Shut' Really Open-and-Shut?"** is a response on these issues from Macromedia.

Corporate-driven systems may allow migration, if the corporation decides to provide a facility for migration. (In the realm of proprietary, corporate systems, authors would do well to prefer systems that can optionally export into open formats.) In general, however, community-driven systems have a better track record in terms of enabling migration and emulation.

4.3 Consolidate Code, Supply Comments

Ideally, the same bit of code should exist in only one place in an e-lit project. For example, since the advent of Actionscript 2, the same Flash code can be stored in several locations within the authoring environment or in a single externally-editable class function declaration; the latter option makes code easier to locate and easier to migrate. Centralizing code and eliminating duplication is also good software engineering and programming practice and offers benefits during development. Unfortunately, as of this writing, programs such as Dreamweaver encourage exactly the opposite practice — by default placing the same redundant code in each HTML file of a Web project that uses the same functions on multiple pages.

Code should also include comments, not only for the sake of archivists but so even the original authors can make sense of the code in the future. Comments should explain how each declared variable is used, for instance, and should make it clear not just what is being done (which can be learned from inspection of the code) but why things are being done (which is seldom as obvious).

4.4 Validate Code

Computer code, whether written in a programming language, a markup language, or a document presentation language such as CSS, can be either "valid" or "invalid," independent of whether it does the right thing. Validity means the code is of the right form — it actually is HTML or Java or whatever language it is supposed to be. In the ordinary language of mathematical equations, for example, "2+2=5" is a valid equation (false, but of the right form), while "=4" isn't a valid equation, since it doesn't equate two expressions.

The formal definition of HTML (or XHTML) is hidden from most people who read and write Web pages, who can see much more quickly that a page looks right in one particular browser. This doesn't mean the page will render correctly in every browser. Web browsers try to be nice; they display some pages correctly even if they are not written in valid HTML. As a result, a browser does not serve as a validator; it does not ensure that a page is valid.

> **Inform** consists of a compiler and libraries for interactive fiction. It has been maintained by the creator, Graham Nelson, with many community contributions. Inform files (.inf) written in ASCII can be compiled to either of two open binary formats: Z-Machine (.z5, .z8) or Glulx (.ulx). The Inform language is an open standard. Both binary formats are open and cross-platform. Source code is available for the interpreters that run both Z-Machine and Glulx formats. See ***The Inform Designer's Manual*** for more information about the system. ¶ Inform is a key part of an open, widely-used interactive fiction system. While multimedia Inform/Glulx does not work perfectly across platforms, Inform/Z-Machine interactive fiction has worked well across a large number of platforms for more than a decade and has very good prospects for remaining accessible.

Validating a page or site, using a service like the W3C Validator or the validator built into BBEdit, ensures that all browsers that comply with World Wide Web Consortium standards, now and in the future, will deal with the page correctly. Not all browsers are completely standards-compliant, but this is a much better

guarantee than is available from just checking a page in a browser — or even a handful of browsers.

This principle applies to other sorts of languages, also, but validity is more obvious with compiled programming languages. An invalid Java program won't compile, so the author/programmer is alerted to this problem right away.

4.5 Prefer Plain-Text Formats to Binary Formats

A plain-text format, such as HTML (or such as XML formats like XHTML or SVG), can be edited, read, and inspected on many platforms. This accessibility remains even if the program that created it, or the program that was meant to interpret it, is no longer available (or exists in a radically different and incompatible version). A plain-text file's contents remain accessible even if documentation of the file format is unavailable. This accessibility isn't true of binary formats, like early Microsoft Word formats, which cannot be edited or read without knowledge of the file's structure (which may not be public) and a specialized program. The compiled version of a work may be in a binary format because of the nature of the work, but it's reasonable to expect the source code to be in plain text.

4.6 Prefer Cross-Platform Options to Single-System Options

For preservation reasons as well as modern-day accessibility reasons, it's best if work can run on many different computers and many different operating systems. Of five different systems, it may, years later, be easy to locate only two — and these two may not be the ones that seemed to be the obvious, dominant platforms. A Windows-only option is seldom a good choice when there is a system with similar capabilities that runs on Windows, Mac, and Linux.

4.7 Keep the Whole System in Mind

If unusual hardware (a graphics tablet, for instance) is part of an e-lit work, preservation means preserving the ability to use this hardware with the system. If the piece is to run directly in the future, the platform being used will have to have a driver for this graphics tablet, and the same or similar hardware will have to be available in the future. If an emulator is used instead, it will have to provide a way to emulate the capabilities of the original graphics tablet.

Preserving the work means preserving the entire system that enables it to run. Providing alternative means for the system to operate can increase the possibilities for future readers. For a system that depends on a particular network environment, alternative means could include a cache of files of the format the system expects to find on the network, so that these may be used if the network resource becomes unavailable in the future. A work that employs live video imagery could include a set of video files of the sort the system expects, to be used in case a camera and driver that produce files of this sort are not available.

4.8 Document Early, Document Often

Documentation of a system can consist simply of a few screenshots that sketch out the experience — although hardly a good indication of how a complex system

functioned, this documentation is better than nothing. It's far easier to take such screenshots while the work still runs, and it's easiest to take them just after the work has been completed. If a work goes through many versions, it's a good idea to keep track of these and to document the original appearance of the work, and to always make clear which version is being distributed or published. Similarly, extracting the texts embedded within the system and preserving them as plain text, at the end of the revision process, is usually relatively simple and can be invaluable later. More extensive documentation — for example, of the processes of interaction and the possibilities for content generation — may not be as simple to assemble. However, such documentation is highly recommended when creating work using systems that present preservation challenges (e.g., closed, proprietary systems; systems that depend on specific network resources or special hardware). As with much interactive and performance art, documentation may be all that remains of an innovative work of electronic literature.

4.9 Retain Source Files

Many authoring environments used in e-lit creation generate "source" or "project" files in addition to the files eventually provided to readers. As discussed in the sidebar, Flash creates both ".fla" project files and ".swf" delivery files. Photoshop generates both ".psd" files (which retain layers, version history, and so on) and delivery files in formats like PNG, JPEG, and TIFF. Many software development environments actually generate three types of files: project files (which maintain history and other information particular to that environment), source files (which contain only the text of the current version of the program), and executable files (the compiled binary code of the current version for delivery to the reader). In all cases, though their file sizes are usually significantly larger than those of delivery files, authors are encouraged to retain source and project files for the components of their e-lit. Ideally, these should be archived and copied with the work itself and should be well-organized and appropriately labeled. Providing copies to publishers can guard against the accidental loss of source files. The possibility of making a small change to source code, or of exporting images in a different format, may make ongoing accessibility much easier to achieve. In addition, authors and publishers are likely to appreciate the possibility of making small changes without having to reproduce elements of the work from scratch.

4.10 Use Common Tools and Documented Capabilities

Some of the tools used by e-lit authors are in common use. These include commercial products, such as Flash and Photoshop, and open-source or community-developed tools, such as the Eclipse development environment and the Inform programming language. When authors and publishers retain the source files created by tools such as these, it is much more likely that some compatible software will be available in the future, making it possible to open and edit these files. Authors may choose to work with more obscure tools, and these may afford possibilities not offered by more widely adopted software. The use of more obscure tools places an additional burden on authors, however: in order to develop or modify their work, they have to keep a version of this software running as the technological environment shifts. This can require that authors act as preservationists to maintain access to their own development tools.

> **The Connection Muse** is a library of JavaScript code to aid in the development of sophisticated hypertext literature on the Web, using

> the principles of adaptive hypermedia. The system is open and freely available. It can be used from within Dreamweaver, using a Toolkit for Browsers, or by manually including the JavaScript code found in the library. See the paper **"Toward an Organic Hypertext"** for more information about the system. ¶ By offering an interface within Dreamweaver but not requiring any proprietary software, The Connection Muse provides high ease-of-use while also adhering to standards (JavaScript, originally a proprietary language, has been standardized as ECMAScript) and not being tied to any one proprietary system. The Connection Muse scripts have very good prospects for remaining operational.

The use of undocumented capabilities in authoring or reading programs is even more likely to reduce a work's readable life. Creating effects using capabilities not specified in a system's documentation can set one's e-lit apart, can be an interesting exploration into the system's workings, and is even a part of certain computing traditions (such as the demoscene). But those who use such techniques should be aware that platform developers are unlikely to retain undocumented capabilities in future versions, and emulated hardware is much less likely to support undocumented features of previous systems. Exploiting such features is not a move that enhances compatibility and future access to a work. It calls for greater efforts at documentation or requires the costly preservation of the original platform.

4.11 Maintain Metadata and Bibliographic Information

Authors can maintain essential information about their digital works without a great deal of extra effort. Preserving the original file creation and modification times when backing up files can help authors figure out when various e-lit works, and elements of those works, were first created and most recently changed. It can also be helpful to keep key information (author or authors, date, systems on which the work is known to run) in a plain text file in the same directory as the work. Those who use formats (such as HTML, XML, or most programming languages) that allow such information to be included in comments can keep such metadata in plain text within the e-lit files themselves, making it less likely that it will be inadvertently separated from the work. Keeping version numbers can also help to distinguish among different incarnations of an e-lit work, both works that are in progress on the author's own disk and works that are published in different places.

Authors are also advised to submit information about their works to the **Electronic Literature Directory**, which ensures that essential bibliographic information is recorded and will be maintained in the Directory.

4.12 Allow and Encourage Duplication and Republication

Authors who keep the only copy of an e-lit work to themselves are reducing the likelihood that this work will survive. Commercial publishers of electronic literature, including Eastgate Systems and the late Voyager, duplicate authors' works; one result is that many physical copies are available and it's more likely that some of them will last. Another alternative is to provide e-lit works online, for free, and to encourage downloads and mirroring (making files available on several sites). This isn't necessarily an alternative, strictly speaking, since it's possible to sell physical media commercially and also make the digital component available

for free — as Linux distributors do and as some artists and writers have done. When online literary magazines publish e-lit and host works on their site, they not only increase access today but also make it more likely that the literature they host will still be around in the future.

4.13 Keep Copies on Different, Durable Media

Archival solutions that looked promising several years ago (such as tape backups) may look less promising today (when CD-ROM drives are ubiquitous but tape drives are rare). One way to help e-lit last is to back up essential files in multiple ways. When backing up to tape, authors can also copy files to another computer's hard drive, burn them on CD-Rs, and transfer at least some of the most important files onto a network system for which someone else maintains backups (such as a university network).

5 Looking Ahead

Outlined above are several principles authors can follow to improve the future prospects for their e-lit work. Still, the systems and standards employed for electronic, literary purposes are seldom as "acid-free" as we would like. The ELO's long-term PAD project is attempting to improve the longevity of electronic literature by working at other levels. The project looks back to some of our most important early works of electronic literature, and it looks ahead to try to establish better practices, tools, and archival methods for keeping e-lit readable. *Acid-Free Bits* publicly initiates the effort.

Continuing this effort is an initiative of the ELO's PAD project called X-Lit. The X-Lit initiative involves developing a rich representation for electronic literature that is in keeping with the principles described here, one that will be human-readable and machine-playable long into the future. X-Lit will allow the representation of media elements (including text, graphics, sound, and video) as well as a description of the interactive and computational workings of an e-lit piece. The standard will also provide a way to document the physical setup and material aspects of an e-lit work. X-Lit will be an application of the open standard XML (eXtensible Markup Language), and itself will be open and community-directed. X-Lit planning (and planning for the PAD project's complementary interpreter initiative) seeks to build upon and to supplement, rather than to replicate, the products of related efforts such as Archiving the Avant-Garde, the Variable Media Network, the Open Archival Information System, and the Metadata Encoding and Transmission Standard.

Initially, X-Lit will offer a standard and efficient way for authors to document their e-lit works. It will serve as a human-readable description of the elements of a work of e-lit, and of the way these elements interact and operate, allowing a uniform way for authors to document e-lit works of all sorts so that they can be understood or even re-created in the future. (However, authors shouldn't wait for X-Lit before they look to document their work; documenting work now, while it is still running or as it is developed, will make it far easier to encode it in the standard X-Lit format in the future.) X-Lit will also provide an example for software developers who are looking to create systems for preservable e-lit. The PAD project will work with authors to ensure that X-Lit is useful and easy to work with, and will also

work with software developers so that authors will be able to save their work in X-Lit format. As development of X-Lit continues, it will become an open format that e-lit applications of many different sorts can use, one that authors can export or save to and, in many cases, directly play or run. Once fully realized, X-Lit capable systems will provide authors with not just documentation, but also a long-lasting, open format for their work — a new, powerful, and easy option for following the principles outlined here.

Authors should take the initiative now; other institutions will certainly need to be involved in the long term. Supporting the ELO as the PAD effort continues, by becoming a member and by offering feedback and participation, is one important way to help. Publishers can take steps as well, by making their Web-published works available at pages with fixed URLs, incorporating bibliographic information, listing works in the ELO's Electronic Literature Directory, and using technologies that are as long-lasting as possible. Those associated with a university can also meet with librarians and system administrators there and explain to them the artistic and scholarly motivations for keeping electronic literature accessible through the years. The people who now preserve our non-digital culture may have a hard time understanding how existing institutions can help make born-digital work accessible in the future, unless authors, publishers, scholars, and teachers of electronic literature take the time to talk with them about preservation, archiving, and dissemination issues today.

The Electronic Literature Organization
http://www.eliterature.org

Colophon · The Web edition of this document was marked up by Nick Montfort in valid XHTML 1.1 with a valid CSS2 stylesheet. It is screen-friendly and printer-friendly; a stylesheet for printer output is provided which browsers should use automatically when users print the document. To cite a specific part of this document, give the section number (such as 3.2); it's also possible to link to specific parts of this document by using the links at the top, under the heading **"Contents."** ¶ The authors of *Acid-Free Bits* thank the other members of the ELO board of directors for their numerous, detailed corrections and suggestions for revisions. Special thanks go to Jeff Ballowe, Robert Kendall, Matthew Kirschenbaum, Alan Liu, and Joseph Tabbi, who made substantial contributions and helped to shape this pamphlet, and to Rob Swigart, who supplied its name. *Acid-Free Bits* is based on the work of the ELO's PAD committee and its subcommittees, which was facilitated by a Digital Cultures Project conference at UC Santa Barbara.