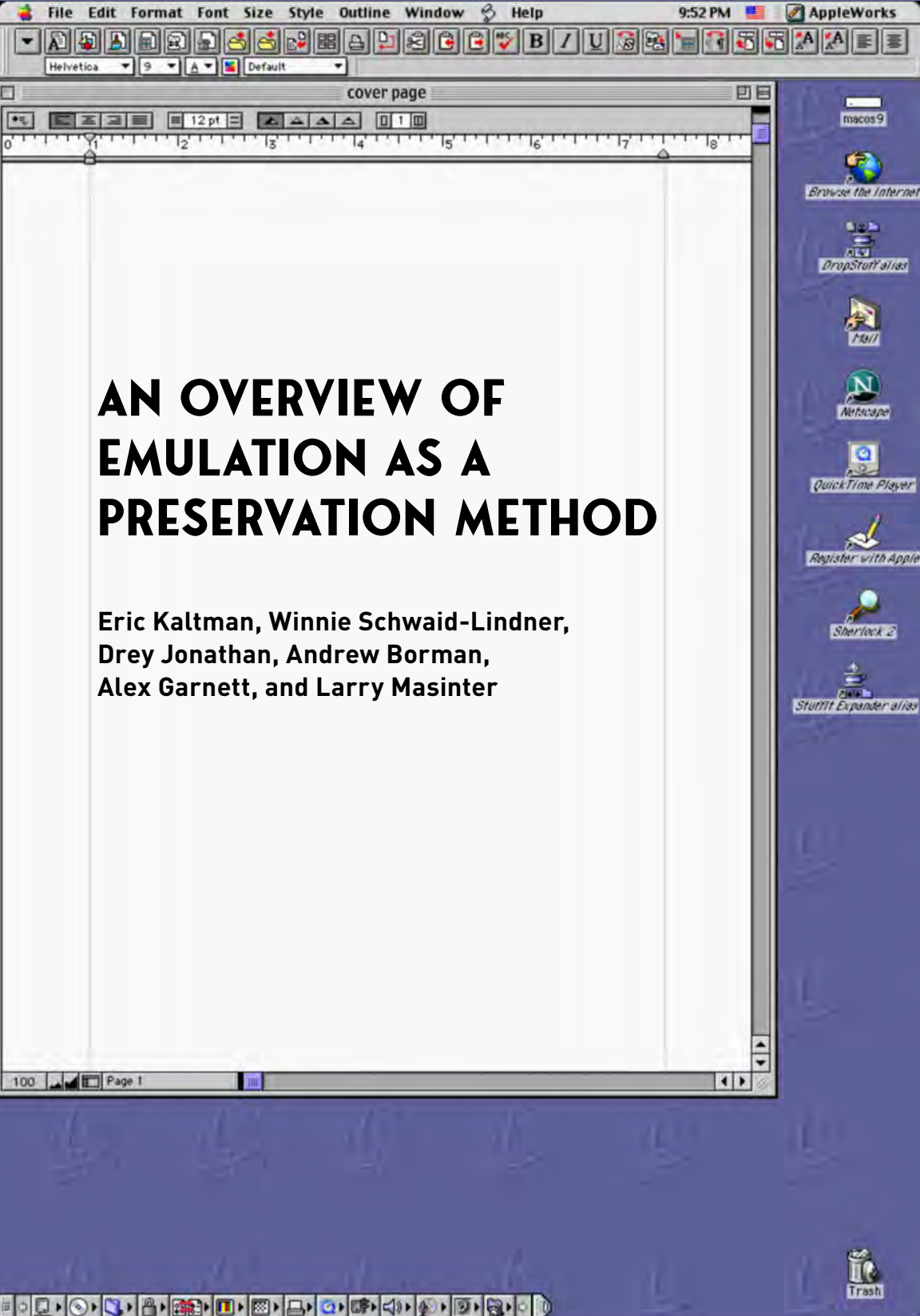




# **AN OVERVIEW OF EMULATION AS A PRESERVATION METHOD**

**Eric Kaltman, Winnie Schwaid-Lindner,  
Drey Jonathan, Andrew Borman,  
Alex Garnett, and Larry Masinter**

**July 2025**



Copyright © 2025 by Council on Library and Information Resources

CLIR Publication 194



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

Cover design by Winnie Schwaid-Lindner

Council on Library and Information Resources  
Alexandria, VA, USA



# Contents

|                                                                            |           |
|----------------------------------------------------------------------------|-----------|
| <b>Introduction .....</b>                                                  | <b>6</b>  |
| What Is Emulation?.....                                                    | 6         |
| <b>Key Concepts for Emulation Discussion .....</b>                         | <b>8</b>  |
| Interpretation.....                                                        | 8         |
| Environments.....                                                          | 9         |
| Virtualization, Containerization, and Simulation.....                      | 10        |
| Images and Imaging.....                                                    | 11        |
| <b>User Considerations for Emulation .....</b>                             | <b>14</b> |
| Experiencing Emulation.....                                                | 16        |
| Example: Emulation at The Strong National Museum of Play.....              | 17        |
| <b>Emulation Approaches and Available Solutions .....</b>                  | <b>19</b> |
| Local Emulation .....                                                      | 20        |
| Examples of Local Software Emulation: DOSBox and QEMU .....                | 21        |
| Cloud-based Emulation.....                                                 | 23        |
| Web-Browser-Based Emulation .....                                          | 25        |
| Browser-based Examples: The Internet Archive and Executable Citations...26 |           |
| Hardware-based and Specialty Device Emulation .....                        | 27        |
| Evaluating Emulation Solutions .....                                       | 28        |
| <b>Prototypical Emulation Workflow Example .....</b>                       | <b>30</b> |
| Identifying the Technical Requirements .....                               | 30        |
| Is Emulation Necessary?.....                                               | 34        |
| Which Emulator to Use? .....                                               | 36        |
| Configuring the Emulators .....                                            | 37        |
| Emulating .....                                                            | 39        |
| Documentation and Troubleshooting .....                                    | 41        |

|                                                        |           |
|--------------------------------------------------------|-----------|
| <b>Limitations of Emulation for Preservation .....</b> | <b>42</b> |
| Accuracy in Reproducing Historical Software .....      | 42        |
| Hardware Peripherals and Displays .....                | 42        |
| Context of Objects and Emulators .....                 | 43        |
| Dependencies.....                                      | 44        |
| Obsolescence of Emulation.....                         | 45        |
| Staff Resources .....                                  | 45        |
| <b>Literature Review.....</b>                          | <b>47</b> |
| History of Emulation for Preservation Projects.....    | 47        |
| Methodology.....                                       | 49        |
| Common Themes in the Literature.....                   | 50        |
| Game Studies and Computer Game Communities .....       | 50        |
| Defining and Critiquing Emulation Methods.....         | 51        |
| Differences in Representative Samples.....             | 52        |
| <b>Conclusion .....</b>                                | <b>54</b> |
| <b>References .....</b>                                | <b>57</b> |
| <b>Acknowledgments .....</b>                           | <b>60</b> |
| <b>About the Authors .....</b>                         | <b>61</b> |

## Introduction

---

Although libraries, archives, and other cultural institutions are growing more adept at the management and storage of born-digital records, providing reliable access remains a challenge. Digital objects require software for interpretation, and that requires complex maintenance of physical and digital dependencies beyond the expertise and budgets of many institutions. There is a variety of tools for removing data from legacy media formats through imaging, but once the data is captured, how can it be rendered and viewed? What can make it comprehensible to patrons and accessible for researchers? One solution is the use of emulators—software that imitates older software or hardware—to combine all the complex dependencies needed to access an object and make it interpretable for a modern computer. This Pocket Burgundy publication introduces, summarizes, and contextualizes emulation within the wider space of digital recovery practices at cultural heritage institutions.

### WHAT IS EMULATION?

Emulation, in the broadest and most inclusive sense, is any method for enabling one computational system to interpret data intended for a different (usually past) system.<sup>1</sup> Any digital object, be it a static document or a complex piece of software, is created within a particular computing context. That is, it is intended to be interpreted by a configuration of hardware and software that is temporally situated within a particular technological era. Historically, emulation developed as a pragmatic technique aimed at solving maintenance issues with continuously updating computer systems. When a new commercial or industrial machine was developed, there were usually significant programs or data still in use by an older one. Emulation allowed the new system to execute the old programs and reproduce their output, thereby avoiding expensive reimplementations costs.<sup>2</sup>

With emulation, it is possible to recreate the interactive experience of historical software, and to then use that software to interpret other historical data. The key benefit of emulation, in relation to other preservation methods like migration, is that emulators operate on the

---

<sup>1</sup> Definitions of emulation with various levels of specificity are endemic within academic and professional literature and often aligned with particular contexts of discussion. For user-facing definitions, see Acker 2021, 2023; and Hedstrom 2006. For more computational ones, see Rosenthal 2015, Rothenberg 1999, and van der Hoeven et al. 2005.

<sup>2</sup> Recent examples of commercial emulation include Nvidia's process for emulating new GPU designs, and Apple's transition away from Intel-based systems to the new Apple Silicon. In the latter case, Apple provided an emulation layer (Rosetta) that converted instructions from the x86 (Intel) instruction set to the ARM (Apple Silicon) one.

original data objects without potentially destructive conversion processes. This means that digital documents can be collected, unmodified, from their original physical carriers or computing contexts, since all their future interpretation can be handled within an emulator. Because of the lack of extensive instructional and technical documentation for the majority of software programs and digital formats, emulation is the most practical and cost-effective method for the future interpretation of all but the most well-known and standardized digital objects.

Emulation, when correctly configured, can function as an effective—almost magical—time machine for accessing past systems. It can also provide patrons and users access to historical modes of software interaction, thereby improving our collective understanding of the past. Taking an emulation approach requires consideration of both its benefits in comparison to other recovery methods and the specific access strategies envisioned for a target artifact. While some emulation solutions are essentially “plug-and-play,” many objects will resist recovery without additional knowledge of how they function within a particular computational context. Running old software objects and interpreting old data also requires interacting thoughtfully with older operating systems and other dependencies. One of the objectives of this report is to provide readers with enough information to make an informed decision about whether emulation is a suitable solution within their individual or institutional constraints and goals.

This report provides: (1) a historical overview of emulation for preservation projects; (2) a discussion of the fundamental concepts required to begin working with emulation, including considerations and criticisms for using emulation in preservation workflows; and (3) a review of the literature for emulation-related projects and publications from the past three decades.

Due to the consistent march of technological innovation in this area, it is assured that many of the details presented in this report, such as specific recommendations for emulators or emulation frameworks, will become outdated. Following this publication, we will be working to create an online community resource (maintained in collaboration with the Software Preservation Network) that will attempt to remain up-to-date on emulation-related work.

Our overriding goal is to provide both an entrée into the current state of emulation as a means for accessing past digital objects and an orientation around which a continuously maintained knowledge base of practical advice, salient examples, and useful tools can be developed to further emulation as a digital preservation approach.

## Key Concepts for Emulation Discussion

---

Many reports and publications describe emulation and its related terms and concerns, in depth. Here, we will focus only on the minimum set of concepts needed to form a basic understanding of how emulation works, and how it might be most effectively used within the context of galleries, libraries, archives, and museums (also known as GLAM organizations). We recommend that readers follow the cited references and footnotes to round out their knowledge depending on their interests.

The processes behind emulation are highly varied and technically complex; however, developing a basic understanding about how emulators function allows for smoother interactions with these systems. In this section we define key concepts that are necessary for an informed understanding of emulation-based methods.

### INTERPRETATION

As the goal of emulation is to get data created by one system to be interpretable by another, the first step is outlining how that “interpretation” occurs. Any digital document, piece of software, or other collection of data is composed of binary-encoded information transcribed onto a stable storage device. Even data “in the cloud” is still, in some capacity, stored on a physical medium while it awaits access requests from other computers or users over a network. This data is “encoded” in that it was created using a particular software according to specific standards; these standards allow for that software to understand what it created. A basic example is the encoding of ASCII-standard text, where specific collections of bytes correlate to numbers, which are then linked to a particular letter.<sup>3</sup> In order to decode (or interpret) encoded text, we need to know both what the encoding standard is and whether the data under consideration conforms to that standard. Determining the encoding without secondary contextual clues, such as associated metadata or literal written notes, can become challenging or potentially impossible for more esoteric data.

---

<sup>3</sup> To complete this example, the capital letter “A” is the numeric value “65” in the ASCII standard. Therefore, while 65 is written as “65” in base-ten (decimal) notation, it is written as 41 in base-sixteen (hexadecimal) notation, and 0100 0001 in base-two (binary) notation. Binary encoding literally means that the encoding uses a pair of values (0 and 1) to represent information. Within computing hardware, these values are physically stored through a large variety of methods, including the presence (1) and absence (0) of magnetic fields on hard drive platters, or in the presence or absence of electrical charge within the groups of transistors that make up “flash” or solid-state drives. If a document is known to be encoded in ASCII (or to contain ASCII-encoded segments), then any instance of the byte represented as 0100 0001 can be assumed to be (interpreted as) “A.”

In emulation contexts, the interpretation of data is generally not happening at the document or file level, but at the level of instructions issued to the hardware being emulated. When running a software program, a computer loads that program into memory as a series of instructions to the central processing unit (CPU) and other hardware components. An emulator reads these instructions (interprets them) and then follows them in a similar fashion to the original hardware. We say “similar” here because as long as the emulator’s output matches the expected output from the original machine, it does not matter how the emulator accomplishes its tasks. In many cases, an emulator will use optimizations or methods that did not (or could not) exist when the original instructions were written. This also means that certain “edge cases” (uncommon uses of the original software) may not be covered by the emulator unless those idiosyncrasies were documented. Emulator optimization is necessary in many cases to allow the emulated software to run at usable or historically accurate speeds.<sup>4</sup>

Interpretation appears within emulation discourse mainly in relation to hardware instructions. References to “x86”, “x64”, or “ARM”-based architectures denote instruction sets designed for specific CPUs that have standardized functions. Incompatibility between instruction sets is the foundational issue that emulation was originally designed to resolve. By enabling the interpretation of a specific instruction set, everything built on top of it—from operating systems to individual programs—becomes potentially accessible.<sup>5</sup>

## ENVIRONMENTS

To engage in emulation is to create a relationship between computing systems so that one system can understand the other. Emulation literature is rife with terminology that refers to the hierarchy and orientation of emulated systems. In most cases, the relationship involves two systems: the one running an emulator and the emulator itself. “Environment” is an umbrella term used to encapsulate all the dependencies and configurations of a computing system. A “host” environment is the one in which a “guest” environment is installed. For example, a popular emulator for early Apple Macintosh systems is Sheepshaver. This emulator can

---

<sup>4</sup> An extreme example of this is the Super Nintendo Entertainment System emulator “bsnes” or “higan,” which attempts to mimic the timing of the original system’s hardware and therefore requires processing power orders of magnitude higher than the original.

<sup>5</sup> Some emulators are not based on instruction set interpretation and operate at higher levels of the computing stack. One such example is programs that imitate operating system-level function calls, like the Proton framework that translates Windows-based audiovisual functions into a form interpretable on Linux-based systems. Proton is mainly used to enable current Windows-based games to run on Linux.

run in several host environments, including modern Microsoft Windows and Apple macOS operating systems, and it provides access to a range of guest environments, such as Apple's System 7 or Mac OS 8 operating systems. Conceptual overlap between the "environment," "computer system," and "operating system" is common when discussing emulation schemes. This is likely unavoidable due to differing definitions of what the boundaries of these terms should be within specific disciplinary contexts. Regardless, when beginning a foray into emulation-based methods, the key determination is to identify what guest environment is needed to access the objects in your collections and whether the host environments to which you have access can support that guest.

### **VIRTUALIZATION, CONTAINERIZATION, AND SIMULATION**

In the preservation literature, as well as in this report, the term "emulation" is used any time one environment interprets data intended for another one. However, because emulation developed out of specific contexts within the computing industry, and because the "use of one system to interpret data designed for another one" is a broad (and nontechnical) definition, you will encounter other terms that appear synonymous with emulation or even sources that argue that a particular process is, or is not, emulation. A common point of confusion surrounds the terms "virtualization," "containerization," and "simulation."

In computing sciences, the use of "virtual" denotes a structure that acts like something but is not that thing. Virtualization, therefore, is the process of making or creating the virtual. "Virtual" has a wide range of common uses, like "virtual reality," "virtual memory," and most importantly for this context, "virtual machine." In this last sense, an emulator is a type of virtual machine that acts like some past hardware-based system. Virtual machine (VM) is also used to describe machines that do not have physical manifestations. For instance, the Java Virtual Machine (JVM) is a program that interprets other programs written in the Java programming language. Any environment with JVM installed can then interpret any Java program. This contrasts with languages like C++, whose programs are compiled into instructions for a specific instruction set and operating system.

In modern terms, virtualization usually refers to the creation of multiple, separate VMs on a single host server. The server shares its resources with each VM; however, the VMs themselves operate as if they are independent machines. This cuts down on physical resource use (as many servers do not use all their resources at the same time) and allows

different operating systems (like Linux or Microsoft Windows Server) to run alongside one another on the same physical host. Containerization is a type of virtualization that allows for the creation of pre-configured VMs known as “containers.” These containers are minimal installations of operating systems and sets of programs dedicated to a particular purpose. There are containers for databases, analytic tools, and even emulators.<sup>6</sup>

Like virtualization, simulation can be a slippery concept depending upon who is using the term (Slats et al. 2003). In digital preservation discussions, simulation is distinguished from emulation in that to simulate something is to create an accurate replica of a thing, whereas emulation acts as a replacement for a thing. Rothenberg (2000) likens simulation to an airplane simulator used to train pilots that operates within a model of an actual cockpit detached from an actual plane, so that pilot trainees can have the experience of flying without leaving the ground. The airplane simulator is modeling the behavior of the real thing but is not intended to be used as that thing. In a computational example, an emulation of Pac-Man would try to interpret the original game’s data by creating, in software, an imitation of the original arcade hardware. A simulation of Pac-Man would recreate the arcade cabinet, its controls, and the game, but might not make any use of the original Pac-Man code (Ross and Gow 1999).

The boundaries between emulation, simulation, and virtualization are only well-defined within specific disciplinary contexts. As a result, readers will encounter significant overlap in the use and definition of these terms when investigating emulation-related resources. In general, this publication considers emulation to be a type of virtualization concerned with the recovery and reproduction of historical data. Emulation may also overlap with simulation in some cases as both are concerned with historical reproduction. Other definitions of emulation, virtualization, and simulation are equally valid (Rosenthal 2015) but will not be further elaborated on here.

## IMAGES AND IMAGING

A “disk image” is an exact, bit-perfect copy of the information stored on some physical media object or other carrier containing binary data. The carrier might be a floppy disk, optical disc, or even a computer hard drive. Because an image is a bit-for-bit copy, it may contain data that is thought

---

<sup>6</sup> The EaaSI project makes use of containers to run and configure its cloud-based emulators.

to have been deleted as well as other bit-level remnants that would be erased through the direct copying of files.<sup>7</sup> While some emulators allow users to directly run software from original storage media, such as floppy disks or CD-ROMs, it is recommended that digital images of the original carriers be used in emulation work. The advantage is in limiting the use of original media, as physical access can further degrade a physical carrier and might eventually lead to the failure of that carrier. Creating an image also means that users can treat the image like any other file-based object in a modern computing environment.

With emulators, there are two primary types of images. The first type are images of physical media carriers, such as floppy disks or optical discs, that can be loaded into the corresponding virtual drives of an emulated guest environment. From the standpoint of the guest environment, the images contain the same files and file organizational metadata as the original media and will function identically. The second type of image is that of physical storage media such as hard drives. These storage devices retain the persistent data needed for an environment to start up and contain all the files, configurations, and other data modifications made through user interaction. When configuring an emulator, it is common to load a disk image containing operating system or software installation files into the emulator, which then installs those files into a hard disk image to be used going forward.

Images come in a variety of formats corresponding to their originating emulators or environments. It should be noted that imaging is not just a preservation activity but is also used extensively in commercial settings to copy, transfer, and reproduce environments and their functionality. This includes modern systems that make use of imaged “snapshots” of physical or virtual machines for the purposes of data backup, retention, and recovery. Sometimes, the literature does not distinguish between physical media images and hard drive images as they are both technical disk images. For emulation workflows, it is important to know whether you are dealing with an image that is loaded into an environment or an image representing the current state of that environment, its files, and its configuration settings.

Creating images from physical carrier formats is a common activity in preservation workflows, and physical disks and optical drives are still

---

<sup>7</sup> A file system on a physical carrier or hard drive is a listing of the contents of that carrier organized by some standard. Different environments conform to different logical standards for arranging and storing files and folders. An image of a carrier copies all the file information as well as all the information about how those files are accessed and stored by the creating environment. In this way, an image is completely identical to its physical counterpart from the perspective of the environment accessing the image.

readily available and compatible with modern environments.<sup>8</sup> There are many proprietary and open-source applications that allow for the creation of disk images. Tools like IMGBurn, DVDDisaster, DD and Guymager, or the Aaru Digital Preservation Suite can create disk images using standard drives. More obscure formats, such as computer game cartridges, are not designed to be accessed except through their original hardware interface. In these cases, enthusiast communities may have developed custom imaging hardware.<sup>9</sup>

---

<sup>8</sup> Currently, these readers usually use a USB interface; however, some contemporary motherboards still have legacy connections for floppy and optical disc drives.

<sup>9</sup> The Retrode 2 is an example of a ROM copier that allows for the creation of backup images of some common computer game cartridge formats.

## User Considerations for Emulation

---

There are as many use cases for emulation as there are users and digital collections. Certain emulation schemes prioritize different aspects of how a user experiences software, from the look-and-feel to smooth interaction to computational accuracy. Each choice made by an emulator will impact the representation and comprehension of the underlying data (Cochrane 2012), and broadly positive adjustments often come at the expense of some aspect of the experience. For example, the authenticity of an emulated experience may come at the expense of convenience, like finding and calibrating a period-accurate display over a contemporary one (Dietrich et al. 2016). It is helpful to consider which aspects of the emulated experience should be prioritized and which are secondary or supporting preferences that can bend when needed.

Factoring in how users will interact with an emulator or emulated material is a necessity, and when used in conjunction with the specific technical requirements of an object (such as which operating system is needed for access), this factor serves as an important selection criterion for any emulation approach. Considerations for effective emulation, therefore, operate relative to both the subjective positions of users (both staff and patrons) and the technical idiosyncrasies of the digital document or software under scrutiny. The user's level of technical expertise, their experience within a particular domain, and their goals for interaction can yield a different set of choices for emulation practice. Additionally, the technical status of an object, including its dependencies, provenance, and socio-cultural context can also affect how and whether it can be effectively emulated.

Recently, Acker (2021) denoted three user classes for emulation, each with a different access perspective: preservation, archival access, and exhibition. Emulation as a support for digital preservation refers to the use of emulation within practical preservation workflows. Users in this category need to have a significant understanding of the properties of both emulators and the objects they are trying to recover. This type of access requires a large amount of time and one-on-one attention as it necessitates the preservation of the authenticity and integrity of the target documents. To maintain the most accurate representation of a digital object, the emulator needs to have high execution and experiential fidelity. This means that the emulator accurately reproduces the technical operations of the emulated object and provides an experience as close to the original's as is possible (Rosenthal 2015). Preservation users of emulation are typically

staff at collecting institutions who work with emulation as part of a larger digital preservation strategy. This may include using emulation to migrate objects from a past system to a more recent one, or literally reconstructing complex historical objects that are dependent on accurate emulation to function (e.g., mixed media or digital fine artworks).<sup>10</sup> The position of emulation in effective workflows is an area of active research, and while certain software preservation workflows encourage the use of emulation (Rios 2016), its effective uses are still being determined.

In contrast to preservationist users, archival users of emulation are more likely to focus on the ability to interact with and experience past digital materials. These users might be researchers with significant archival expertise or students working on developing their understanding of archival use. In either case, access to the material is the primary concern (Benner et al. 2022). For advanced software scholars, the fidelity and constraints of the emulated reproduction would be known and accounted for if they were germane to the object under study. Other researchers and students may be more interested in stability, speed, or usability instead of complete historical fidelity. Some beginning researchers may want to deal with the original digital materials in the archive (instead of migrated copies of files) to better learn about the interaction and computational constraints of the original environment (Slats et al. 2003).<sup>11</sup> However, authentic rendering might also prove “counter-intuitive and frustrating” to users due to how streamlined and responsive modern interfaces have become (Dietrich et al. 2016). In any event, the priority for emulation in archival access is to make the materials available and usable. After that, it is up to the user or public services staff to work out the particulars of textual or experiential interpretation.

Exhibition users, those who interact with emulated materials in curated displays or exhibitions, are mostly invested in the experiential fidelity of the emulator, in that the interactive experience matches the curated intent of the display. As long as the experience feels like a true and usable rendering of the material, the details of the implementation (including whether it is historically accurate in technical implementation) are secondary. Even the presence of an emulator, or an understanding of that concept, may not be vital or necessary to the exhibition experience.

---

<sup>10</sup> See Rinehart and Ippolito (2014) on the reconstruction of the Erl King installation and the Seeing Double Exhibition (2004). Dourish (2017) discusses the recovery of Andy Warhol’s Commodore Amiga artwork through emulation.

<sup>11</sup> Early literature refers to the use of emulation to extract “vernacular” copies of data from the emulator, effectively migrating those objects to a contemporary environment. They note that migrated vernacular copies are suitable for many types of scholarly use as long as the historical rendering or interaction contexts of the original are not important for the research.

Each user class diverges in the level of technical understanding of emulation needed for a particular use case. Preservationist users usually desire the highest transparency of all aspects of emulation, ensuring that each component of the process is clear and documented. An emphasis is placed on the visibility of each element as well as documentation of the labor of the practitioner (Acker 2023). On the other hand, archival users and especially visitors to exhibitions may prefer the invisibility of the emulation process and place a much higher value on a seamless access experience. Understanding what is going on “under-the-hood” may be of interest to some, but the more general the population interacting with an emulation, the lower the barriers to entry need to be. Many popular computer game systems and consoles now provide access to historical libraries of past games, so the understanding that older software is somehow still playable on modern systems provides a good entry point for engaging with more complex emulated interactions.

## EXPERIENCING EMULATION

Emulation allows a user to experience the computational past and interact directly with historical objects in a way that is more tangible and expressive than text, image, or video. Grappling with the idiosyncrasies of historical computational design creates more awareness of how our present software-mediated modernity developed. Curating software experiences in an exhibition or giving a scholar interactive access to an old software program requires institutional expertise and skills that are only beginning to take shape. Due to the consistent degradation of physical hardware systems, it is likely that emulation will be the primary way we experience past digital objects and software in the coming centuries. It behooves us to begin developing stabilized and standardized methodologies for emulated access and the potentially transformative access that decoupling from past physical hardware makes possible.

Since emulators are programs running within a host environment, it is possible to provide modified experiences that differ from those of the original hardware and software in significant ways. As an example, emulators can be configured to run at speeds far beyond what the original hardware is capable of, enabling both quicker processing and the ability to process more, and more varied, data than the original environment may have supported. This would make the emulated software easier to use (by speeding up loading and other user interface transitions) and potentially more useful for other contemporaneous data processing tasks. Emulation also provides the opportunity to add features not present in the original environment, such as support for additional hardware features that did

not exist when a software application was first created, modifying the emulation display to either imitate or clarify older visuals<sup>12</sup> or to add modern accessibility features. Although accessibility modifications may be inconsistent with the original intent of a software or hardware's design, the ability for previously excluded communities to gain access to legacy applications is likely worth it.

At museums and other cultural institutions, emulation for exhibition and experience may include some combination of direct use and transformative experience; for example, an emulation may offer greater reliability while also providing enhancements to make interactions with the software more enjoyable for guests or patrons.

### **Example: Emulation at The Strong National Museum of Play**

The Strong National Museum of Play in Rochester, New York, is home to the world's largest collection of physical and electronic games and toys. The museum's curators have many years of experience organizing exhibitions that include emulated environments. While most interactive computer games at the Strong are available through their original hardware, emulation has become an important tool for guest experiences. The Women in Games Exhibit (2018–2022) highlighted and celebrated women's contributions to the gaming industry. While original hardware was used extensively in the exhibition, one installation featured a transformative use of emulation.

While modern video games do include credits, older ones sometimes lacked any attribution for individual or group contributions. Through the use of emulation, the Women in Games Exhibit provided an interface that allowed patrons to look up various early game designers and play their games directly. The interface provided text-based information about each game designer and then listed games available for immediate emulation. This ability to connect each of the featured designers with their creations would not be possible with the original hardware and software. The emulations allowed for a spatial and temporal collapse of many games into a singular exploratory display (figure 1).

---

<sup>12</sup> In addition to providing greater resolutions than the original environments, one can also modify the output of the visuals with the application of graphical shader programs. The RetroArch emulator program has a large collection of effects, including shaders that imitate a range of historical cathode ray tube (CRT) displays ("CRT Shaders" 2022).



Fig. 1. Women in Games Exhibit, The Strong National Museum of Play, November 2018 (Image credit: Andrew Borman, The Strong National Museum of Play)

## Emulation Approaches and Available Solutions

---

Given our definition of emulation as the process of interpreting data intended for one environment within a different one, what are the various ways that emulations are implemented? Any emulation process is managed by some form of emulator, and emulators can be implemented either as software (through programmed code), hardware (through the creation of physical systems analogous to past systems), or some combination of the two. For the purposes of this publication, we will focus on software-based emulators as hardware emulation is not yet extensively used within the digital preservation community; however, progress has been made on this front, and details on hardware and specialty device emulation are included at the end of this section.

The most common form of emulator is one that uses a software program to reproduce legacy hardware. As noted in the discussion of interpretation, an emulator reproduces the functionality of a historical environment by reading data intended for that environment and translating it into a form that the host environment running the emulator can understand. These software-implementing-hardware emulators typically reproduce the input and output behaviors associated with low-level instructions to a particular set of historical CPUs. A prime example of this is the QEMU open-source emulator and virtualizer. QEMU supports popular hardware platforms, such as x86-based machines that can support Microsoft Windows operating systems. Other hardware features such as the devices supporting audiovisual output, virtual drives, and peripheral input are also sometimes implemented within emulators.

Emulation can also occur through more complex virtualization platforms, like the Oracle VirtualBox or VMWare Fusion hypervisors. These systems are virtual machine managers (VMMs) that support the creation, management, and replication of collections of virtual machines. Hypervisors are used in commercial contexts to run multiple VMs on a single host server; alternatively, they allow for the installation of one modern operating system within another.<sup>13</sup> Conveniently, many hypervisors also support legacy systems; this makes VirtualBox, in particular, a common solution for past MS-DOS and Windows-based environments.

---

<sup>13</sup> VMWare Fusion, for example, is specifically marketed to Apple macOS users interested in running modern Microsoft Windows operating systems within macOS.

It is also possible to write emulator programs that imitate operating system and software application interfaces directly. These programs are less common because reproducing low-level functionality, like CPU instructions, will implicitly support higher levels of abstraction such as operating system calls. However, in some cases higher-level emulation is preferred. This would be true, for example, with the DOSBox emulator, which imitates the functionality of the DOS operating system (figure 2). This emulator allows for the installation of DOS programs without the need to know the specific DOS version that is required, since DOSBox is designed to support many of them. DOSBox is frequently used to commercially re-release older computer games for modern systems.

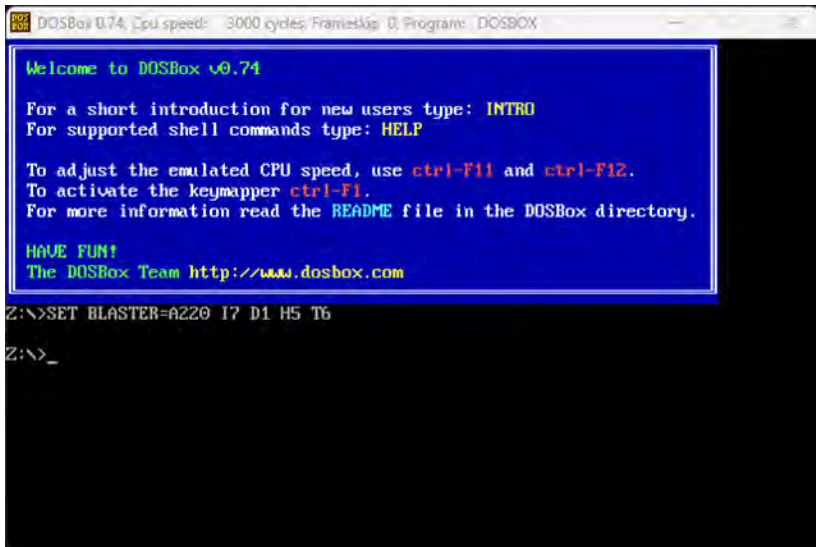


Fig. 2. DOSBox 0.74 Interface (Image credit: Andrew Borman)

Software-based emulators, being software objects themselves, can therefore run in a variety of different contexts and configurations. Below, we detail three approaches to emulation currently used in preservation work: local emulation on a host machine, remote cloud-based emulation on servers, and web-browser-based emulation.

## LOCAL EMULATION

Local emulation is among the most common forms of emulating software. It offers a wide variety of platform support and allows users to emulate hardware that they may already own, namely their computer. Local use of emulation involves downloading an emulator like any other software

program. Running an emulator locally will generally provide the fastest and most responsive emulation as there is no need for additional network or software interface layers that can slow things down. Local emulation can also take advantage of the hardware features of specific hosts. For example, if you are running a local emulator designed to reproduce Intel's x86 instruction set on a host that makes use of AMD's current x64 instruction set,<sup>14</sup> the emulator needs to do less work as x86 is a backward-compatible subset of x64.<sup>15</sup> This means instructions require less modification for interpretation by the host.

In many preservation contexts, local emulation is preferred as it keeps the data securely within one host machine and provides the best performance. The disadvantage of local emulation stems from the need for extensive documentation to reproduce local configurations across multiple host machines. Local emulation also means that the versions of each local emulator matter, as new features or optimizations may be added over time, and these may break or change the representation of previously emulated objects. Since local emulators are also programs designed to run in particular environments, it is essential to locate an emulator that can run on your host machine and support the guest environment you are hoping to emulate.

Many emulators are open source and supported by active communities that sometimes align with specific commercial entities. Both Oracle's Virtual Box and QEMU are used in industry for commercial emulation and software development. As such, they have long, stable development histories and follow software engineering best practices. Because they are open source, many useful emulators are free to download and use. Most active emulation projects also provide support through documentation and community error reporting.

### **Examples of Local Software Emulation: DOSBox and QEMU**

One example of a local emulation solution is DOSBox, which emulates a collection of DOS-based environments from the 1980s and early 1990s. The emulator provides an imitation of the DOS environment directly, which is somewhat atypical due to its ability to reimplement DOS functionality and interfaces without making use of proprietary code or data. Many operating systems are protected by copyright or depend on compiled data that is proprietary; this makes emulator development more challenging as it is

---

<sup>14</sup> This is common when running older Windows environments on newer ones.

<sup>15</sup> There are significant caveats to this example, see Rosenthal 2015, Craig 2006, and "X86-64" 2024 for more technical details.

necessary to either imitate some functionality without direct access to its implementation or to require a user to locate copyrighted data on their own, which is a legal gray area.

Managing audio and video features is one of the more confusing aspects of DOSBox, as machines of the DOS era used a variety of different devices for audio and visual output. Working with DOSBox also requires a basic understanding of its command-line interface to navigate through the file system and run programs. It is possible to pre-configure DOSBox to run a particular game or software immediately on startup or to use a third-party graphical user interface (GUI), such as D-Fend Reloaded, to streamline software configuration.

QEMU is a comprehensive emulator that supports a large range of hardware configurations.<sup>16</sup> It is mainly configured through the command line and supports many additional device drivers, video adapters, and sound cards. In QEMU's simplest use case, it is possible to quickly run a basic emulation with only a few command lines.

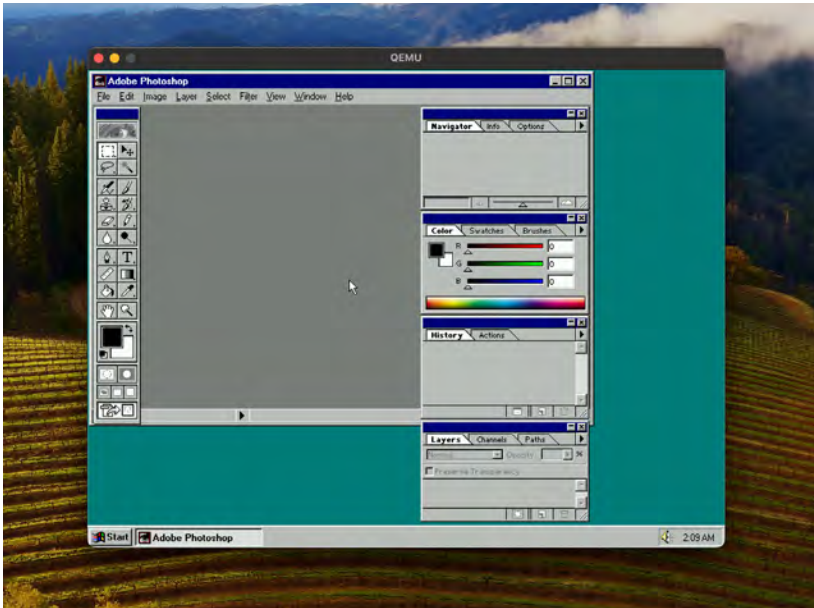


Fig. 3. Adobe Photoshop 5.5 running within a QEMU emulation of Windows 95 on Apple macOS 10.14 (Sonoma) on M3 Max Apple Silicon hardware (Image credit: Eric Kaltman)

<sup>16</sup> A full listing can be found in the QEMU documentation ("Emulation" 2024).

QEMU is used extensively in industry, and while not explicitly preservation-focused, it has been tested in some emulation workflow projects such as the Preservation and Access Frameworks for Digital Art Objects (PAFD-AO) project (Rieger et al. 2015).<sup>17</sup> It is also actively maintained and updated for modern architectures. Due to the complexity of managing a QEMU configuration, an ecosystem of tools is available to allow for more streamlined configuration through graphical user interfaces. Examples include UTM, a frontend manager for iOS and macOS devices, and Qiwi, a Python-based user interface for configuring QEMU images (Gates 2023; “UTM” n.d.).

## CLOUD-BASED EMULATION

Cloud-based emulation, in which a remote server runs an emulator instead of a user’s local machine, can offer great flexibility and convenience for users. While cloud-based emulation functions similarly to local emulation (in that the same emulator programs are being used), the primary differences are that the local environment is no longer directly executing the emulator, and the same emulator configurations can be accessed by multiple, geographically distributed users. In most cases, the remote machine presents the user with a stream of the emulator’s audiovisual output. Any input or interaction with the stream is then sent back to the remote emulation.

In cloud-based emulation, the local environment only needs the ability to interpret the streamed output and send input over the network. As some emulators are rather resource-intensive, cloud-based emulation allows for local environments without powerful hardware to access a wider range of emulators, including emulators that would not run in the local environment under normal conditions. Additionally, due to remote execution, access to the emulator can be controlled and managed without affecting the local user’s environment. Sensitive data or materials can be stored remotely in the cloud-based system and will, therefore, never be directly accessible on a user’s machine. This also means that cloud-based emulation provides significant control for managing intellectual property rights and access restrictions.

However, cloud-based emulation still requires that someone with expertise configure the emulator, which means that the environment may not be set up by the end user directly. This shifts both the labor of configuration and user control to a third party, meaning that incorrect configuration may present challenges and make other troubleshooting necessary before

---

<sup>17</sup> Rieger et al. (2015) describe this project and include a brief tutorial on the use of QEMU.

a user can gain access. On the other hand, this reduced agency can be beneficial for general usability, as a correctly configured emulator can be utilized by any user with access to the system anywhere in the world. The democratizing effect of cloud-based systems could open up access to troves of materials currently inaccessible due to resource or expertise constraints.

Unlike video streaming services, like Netflix, where users can sit back and experience a stream passively, using and controlling an emulated system requires providing input back to the video stream. A fast network connection and closer geographic proximity to the remote cloud server can make this process appear seamless; however, slower network connections can cause frustrating delays, or lags, between inputs and the video stream. Many emulation use cases, such as interactive art programs or computer games, require continuous, responsive input and feedback. Delays in input can make items effectively unusable or maddeningly frustrating to interact with. Nevertheless, as network technology and cloud-based emulation continue to improve, these issues will become less significant. In the commercial space, many companies offer modern, visually complex computer games through cloud-based streaming. Adapting the same technology to the preservation context will take time (and much more funding).

The resources needed to implement and manage cloud-based systems are additional considerations. Hosting and managing a remote emulation service has associated costs, and configuring and securing cloud-based services may be challenging for individuals or institutions that lack significant IT infrastructures. Some cloud-based emulation solutions offer institutional access through paid subscriptions, thereby offloading the IT management of a cloud-based system to a centralized organization.

The most prevalent cloud-based solution is the Emulation as a Service Infrastructure (EaaSI) project managed by Yale University Library and funded by the Mellon and Sloan Foundations. To date, EaaSI allows an institution to set up their own instances on a local network or to access Yale's hosted servers. The system runs emulators within virtual containers on a remote machine and then sends video output to a web application running in a standard web browser on a user's machine. Inputs are sent back to EaaSI through the web interface. Normal interaction with EaaSI is therefore similar to other web applications (figure 4). Users log in to the service and have their own dashboard of environments. Crucially, configured environments can also be shared back to a collective pool of environments that is then accessible by other institutions using EaaSI. In this way, it is possible to configure commonly emulated environments once and then share that work with others instead of repeating emulator installations over and over again.



Fig. 4. Microsoft Encarta 98 running in Microsoft Windows 98 within the cloud-hosted EaaS service (Image credit: Eric Kaltman)

## WEB-BROWSER-BASED EMULATION

Sitting in between local emulation and cloud-based solutions is browser-based emulation. Instead of installing and running an emulator locally or viewing the output of a remote emulation through a browser, browser-based emulation downloads an emulator directly into a webpage. This allows the emulation to run locally but be contained within a browser environment. As a result, there is no need for local configuration (as the browser handles that), nor is there any input or streaming delay (because the emulator is running locally). Browser-based emulators can often be configured to directly load specific pieces of software and require little configuration by the end-user, allowing them to quickly and easily begin emulating software, often within seconds or minutes based on the download size of the emulator.

The downside, however, is that compatibility between different browsers can limit emulator functionality, or prevent an emulator from functioning at all. Additionally, browser-based emulators do not have the same level of access to the underlying host system as the other solutions. This means that peripheral configuration, complex hardware, or other networking functionality may not be possible. Browser-based emulators are also being interpreted by the browser program, which is then interpreted by the user's local environment. This means that browser-based emulation cannot reproduce more recent systems at a speed that is acceptable for many applications. Older environments, such as game consoles from the 1980s and 1990s or desktop operating systems from 25 years ago will usually run well, but more recent environments may struggle to function correctly.

Finally, one benefit *and* deficiency of browser-based emulation is the precarity of an emulator running in a web page. These emulators are JavaScript applications, so if there is an error or issue, one simply reloads the page to get instantaneous access to a fresh emulator. However, many browser-based emulators do not save their states or storage when being refreshed; therefore, it is sometimes not possible to save one's work within these emulators. There are developing solutions to this problem (see the following example), but currently the best use case for browser-based emulation is for the end-user experience of computer games or application software.

### **Browser-based Examples: The Internet Archive and Executable Citations**

The most well-known browser-based emulator collection is hosted by the Internet Archive (IA) through their Internet Arcade and Software Library. These collections allow visitors to a software object's webpage to load an object directly in the browser by making use of JavaScript versions of the MAME and DOSBox emulators. Information is not stored across sessions, and users cannot load new data into the emulators, but they can play and experience numerous historical games and software objects through the site. User comments provide additional context for input controls and other interaction nuances. The IA collections are used frequently in courses that focus on the history of software and computer games as they are an easy way to introduce students to past computing environments.

Because browser-based emulators exist and execute completely within a containing browser, they can also be manipulated like any other HTML-based object. This means that multiple emulators can be run alongside each other and that they can be embedded within webpages like images or video. A few projects are exploring ways to cite these emulators in a way that is similar to other text-based and hyperlinked resources.<sup>18</sup> In this way, a user could configure an emulator to show a particular interface or location in a game and then save that emulator's memory and configuration for future recovery by a later user. This would solve the issue of having changes to browser-based emulators persist across multiple sessions. It should be noted that cloud-based systems, like EaaSI, also allow embedding their emulated output within web pages; however, the embedded items still require a connection to the cloud-based server to allow for input and continued interaction.

---

<sup>18</sup> The NEH-funded Game and Interactive Software Scholarship Toolkit (GISST) is a system that allows for the recovery of browser-based emulator states (Kaltman et al. 2017; Osborn and Kaltman 2024; Kaltman and Osborn 2024). Additionally, work at the University of California Santa Cruz has allowed for "playable quotes" to games that can be embedded as executable images (Franušić et al. 2023).

## HARDWARE-BASED AND SPECIALTY DEVICE EMULATION

In addition to emulators that are designed to be used on commonly available computers, there are specialty devices designed primarily for emulation, mainly to emulate and play video games. These are usually all-in-one handheld devices, complete with a control interface and buttons, display screen, and battery all built in. However, the handheld nature of these devices precludes a full keyboard, which can make it challenging to use these systems for typing-based emulation tasks. Due to the rapidly changing availability of these devices, user reviews can be a valuable resource for staying up to date on the latest advances and determining whether a given specialty device is a viable choice for particular emulation goals.

Hardware emulation similarly relies on additional physical equipment but differs from specialty devices in the components and methods by which emulation is achieved. Field Programmable Gate Arrays (FPGAs) are a type of hardware, a circuit, designed to be configured and changed according to specific hardware needs (figure 5). Where software emulation translates one existing set of instructions, or code, so that it is compatible with a computer that only understands another instruction language, FPGAs facilitate approaches to emulation from the foundational levels of how the computer works, and this can make it possible to set up a system that does not need any instructions translated to begin with.



Fig. 5. DE10-Nano FPGA Platform used in the MiSTer Project. MiSTer, first introduced in 2017, is a rapidly developing open-source project that is used to emulate multiple platforms, most commonly for video game software. (Image Credit: Andrew Borman)

Hardware emulation grants a high degree of flexibility because it allows for tailoring the hardware to precise emulation goals, but designing a hardware emulation system using FPGAs requires technical skills and investment and may not be a good fit for organizations with limited technical resources or those not relying heavily on emulation.

## EVALUATING EMULATION SOLUTIONS

There are a variety of approaches and forms of emulation, which can be as simple as downloading a program or as complex and bespoke as resources and skills allow. Each solution comes with benefits and trade-offs that factor into determining which emulation solution (or solutions) is the right choice for the project at hand.

When selecting emulators from possible options, Ponsard (2021) suggests criteria for evaluating emulation solutions for a given situation. Table 1 highlights these selection criteria, adding questions to consider when choosing a solution.

| <b>Selection Criteria</b>              | <b>Questions to Think About</b>                                                                                                                                                                                                                                                                                                                       |
|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ease of installation/<br>configuration | <ul style="list-style-type: none"> <li>• How do you set up the emulator?</li> <li>• How do you get to the point where the emulator is usable?</li> </ul>                                                                                                                                                                                              |
| Ease of use                            | <ul style="list-style-type: none"> <li>• Does the emulator's interface make sense to the user?</li> <li>• Does the emulator include features that you need for your goal (e.g., ability to take snapshots)?</li> </ul>                                                                                                                                |
| Long-term support                      | <ul style="list-style-type: none"> <li>• Is this solution an established project and actively maintained?</li> <li>• Is there good documentation?</li> <li>• How long do you need this emulated solution to work?</li> </ul>                                                                                                                          |
| Accuracy of emulator                   | <ul style="list-style-type: none"> <li>• Is the way the emulator renders the legacy software object capable of fulfilling your goal?</li> <li>• Is the emulator capable of fulfilling your goal in a way that's achievable for you?</li> </ul>                                                                                                        |
| Resources required                     | <ul style="list-style-type: none"> <li>• How much time are you able to devote to getting an emulated solution to the point that it fulfills your goals?</li> <li>• What financial resources (if any) can you devote to the emulation?</li> <li>• What technical skills and resources are required for this emulator, and do you have them?</li> </ul> |

Table 1. Criteria for evaluating emulation solutions (Ponsard 2021)

While these criteria can inform the choice of an emulator for a particular use case, some of the criteria relate more closely to the host machine's ability to run the emulator (i.e., ease of installation, resources required), while others are related to what the goal of the emulator is and the resources required to achieve that goal (i.e., ease of use, long-term support, accuracy). These factors can be at odds with one another and require making trade-offs to determine the best solution for any given situation. For example, an emulator that perfectly fulfills your project goals but is difficult to install and use may ultimately be a worse fit than an emulation solution that fulfills the goals less perfectly but is substantially easier to configure. Similarly, there will always be trade-offs between speed and accuracy and the desire to build a bespoke solution or custom reproduction. There is never only one correct solution for any given task; in many cases, you may benefit from using multiple complementary solutions in tandem to accomplish a larger goal.

Many of the trade-offs or challenges you will likely encounter when evaluating emulation solutions are similar those you face with other kinds of software design dilemmas, due to the approach and method of emulation that an individual solution takes. Local, cloud-based, and browser-based approaches often present familiar trade-offs and challenges. For example, most in-browser emulators will compare favorably on "ease of use," while emulating locally may have higher accuracy but require additional time for configuration. In this respect, these three approaches can be considered together with the criteria to delve deeper and determine which emulation solution or solutions best fits a given need.

## Prototypical Emulation Workflow Example

---

While many factors go into deciding if emulation is the right choice, it may seem daunting to even begin the process. This section describes potential workflows to follow when beginning to work with emulation. There are two primary use cases involving emulation: running a past software object on its own or accessing a data object that can only be interpreted through a legacy software program or environment. In either case, we assume in this section that the data under investigation is already in digital form (i.e., has already been imaged from physical media and downloaded or otherwise copied to the host environment). The typical “path” in getting a digital object running in emulation involves identifying the guest environment needed for the object, collecting and imaging the requirements for that environment, and then configuring them within an emulator.<sup>19</sup> Documentation of the process is also an important component, as it is likely that multiple objects from the same collection or accession will have similar access requirements. Maintaining a record of how to recover an object would therefore help in reproducing or replicating a similar workflow for a related object.

### IDENTIFYING THE TECHNICAL REQUIREMENTS

Determining the technical requirements for a target software or data object is highly dependent on the availability of contextual information about that object. In the case of commercial software programs, items like the containing packaging, instructional manuals, or other identifying marks help with identifying a suitable guest environment. Figure 6 shows the commercial packaging for the 1989 Microsoft DOS version of the popular game SimCity, photographed from the collections at The Strong National Museum of Play. This version of SimCity included both a 5.25-inch and 3.5-inch floppy disk containing the game’s data. Assuming that either was imaged, we would then need to figure out the system requirements that would allow an emulator to load the disk’s image and install the program.

---

<sup>19</sup> This notion of a workflow “path” comes from Acker (2021); however, the idea of a “view path” for a digital object is present in emulation literature dating back to the late 1990s. See Rothenberg 1999 and van der Hoeven et al. 2005.

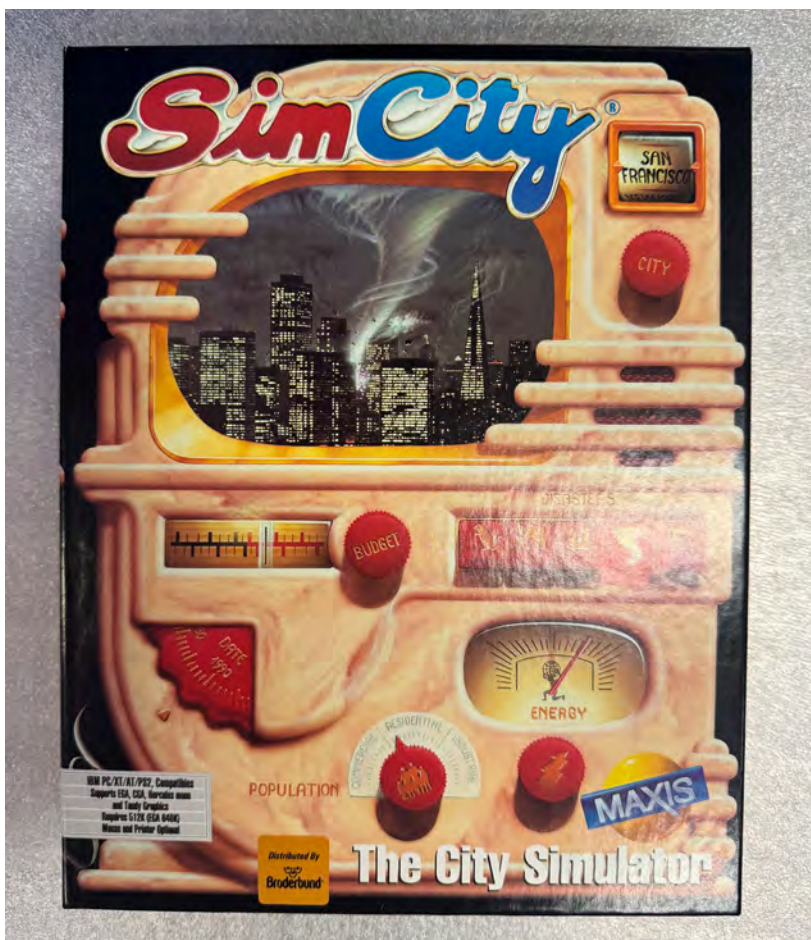


Fig. 6. Packaging for the 1989 Microsoft DOS version of SimCity, The Strong National Museum of Play (Image credit: Andrew Borman)

The situation is ideal in this case, as both the box and manual specify system requirements. In the manual, the “hardware required” describes the environmental requirements, stating that the software needs an IBM-compatible machine, 512K of RAM, and a floppy disk drive. Optional requirements include a range of video output modes, input peripherals, and audio support. The task here is to align the requested requirements with the correct emulator or emulation scheme.

Although the above example is straightforward, in most instances, the complete technical information is not likely to be available with the software or data object. Figures 7 and 8 show disks from recent emulation-focused

research by members of the report's authoring team. In figure 7, we see a disk from the Workshop West Theatre fonds at the University of Alberta Archives, and figure 8 displays a similar image from the Prisby Bottle Village materials in California State University Channel Islands' collections. Both disks have some useful contextual information. The Bottle Village Zip 100 disk is formatted for Apple Macintosh systems and ostensibly contains files relating to the QuarkXPress desktop publishing software (though the version is not specified). Based on dates marked on other items in the collection, it was determined that the Bottle Village carriers originated in the late 1990s. On the Workshop West disk is a note that the disk is the fifth of a series and that the data on the disk relates to the "8 ½ / 11" paper size. While the accession date is labeled for 2003, the materials were part of a larger collection from the late 1980s to mid-1990s. The disks are also labeled as "FORMATTED for IBM PS/2 & compatible."



Fig. 7. Disk from the Workshop West Theatre fonds (UAA-2003-076-1-5), c. 1980s–1990s, University of Alberta Archives. Disks from this collection were imaged using a Kryoflux device. The Workshop West Theatre is a well-known theater company founded in Edmonton, Alberta, in 1978 to promote and develop the work of Canadian playwrights. (Image credit: Mohsen Kardar / University of Alberta Archives)



Fig. 8. Zip 100 disk from the Prisby Bottle Village materials, c. 1990s, California State University Channel Islands University Archives. To read this disk, we had to purchase and configure a legacy Zip 100 drive. Grandma Prisby's Bottle Village, self-built and made from found materials, is a local landmark in Ventura County, California. The notation that this disk "has AutoStart Virus (Power PC)" required an investigation into how that historical virus would affect access. Personal phone numbers have been redacted. (Image credit: Adam Larson)

Usually, there will not be enough hand-labeled information to determine the requirements for a particular disk image. For the Workshop West example, we needed to further analyze the disk's contents with a hexadecimal editor (hex-editor) that visualizes the raw bytes of the disk image. Hex editors that provide similar features are available for most current platforms. Germane to this example is the hex editors' ability to show all ASCII-encoded information on a media image. The Workshop West disks included "MS-DOS 6" in the disk header information (figure 9). This led us to the realization that the disks were in fact not formatted for Macintosh and had been reformatted at some point in the past. Many times, the use of basic computer forensic techniques will be necessary to figure out what environments might be compatible with a given data object.

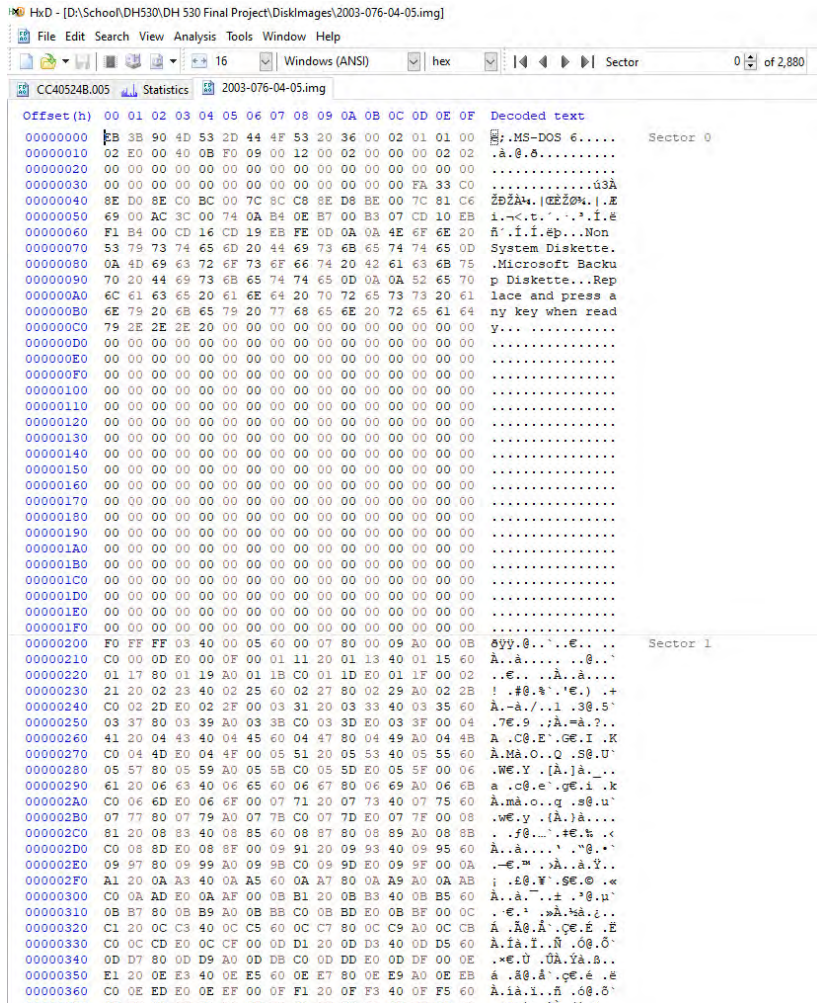


Fig. 9. A hexadecimal and ASCII-encoded view of one of the Workshop West disk images showing the “MS-DOS 6” formatting note in the header (Image credit: Logan Thienes)

## IS EMULATION NECESSARY?

With both the original software and hardware at least partially identified, it is possible to decide whether emulation is the right solution. If it is possible to use and maintain the original hardware, emulation may not be the best choice, due to any number of limiting factors described in the following section, Limitations of Emulation for Preservation. However, since working historical hardware is unlikely to be available, the decision to emulate is generally based on the intended use of the data.

In the above examples, the commercial copy of SimCity is likely intended for exhibition play or researcher use. In either case, games are expressed through play, so interaction through emulation is likely necessary. For the Workshop West and Bottle Village examples, it would depend on the specific recovery goals. The Bottle Village disks included numerous text files that were readable without an emulator albeit with potential rendering challenges, such as formatting issues (Cochrane 2012), and the Workshop West files included an old, but standardized, TIFF file. If we were only concerned with recovering the TIFF, emulation would not be necessary. If the research concerned how the image was made, then it would likely be necessary to recover the creative software used to generate it. It might also be possible that some of the obsolete files, when recovered through emulation, could be converted inside the emulator to formats that are still in use. The application of emulation techniques, however, need not be assumed for all use cases.

### WHICH EMULATOR TO USE?

The next step is to explore what emulator options exist for the identified hardware and software. Is a local emulation solution the best idea? Or would a browser-based or cloud solution be more convenient? A cursory online search will provide a selection of possible options.

Searching for both SimCity and MS-DOS emulation returns a collection of methods for recovering the game. This includes links to the same version of SimCity running within browser-based emulation on the Internet Archive. However, although many games and software objects are now immediately available through browser emulation, the legality of such sites is still an open question. Also, users have little control over the configuration of software available through web emulation on archival or enthusiast sites. If the goal is to explore or manipulate the environment of a particular game or software object, local emulation is probably the best initial option. Alternatively, if access to an emerging cloud-based solution, like EaaSI, is available, it would be worth trying to locate a pre-configured environment if one happens to be available.

As SimCity is a well-known computer game, it might also be possible to play it directly on modern computers. While this is not the case, its sequel SimCity 2000 Special Edition is currently available to play in modern Microsoft Windows environments. The sequel accomplishes this by relying on the emulator DOSBox, which is packaged with the current retail release. DOSBox is a general-purpose MS-DOS emulator that is compatible with most known DOS versions. Therefore, it is not a literal recreation of a specific DOS hardware system but more an idealized environment designed explicitly to run old DOS software. For a researcher interested in SimCity, DOSBox would be a helpful emulation tool.

The Workshop West and Bottle Village research cases are not as approachable. Because only indirect information is available, a researcher must search for contemporaneous environments and their emulators. The Bottle Village materials were labeled as associated with Mac OS 9, and the files on the Zip 100 disk and a companion optical disc were from 2001. For working with the Bottle Village objects, researchers decided to use the Sheepshaver emulator, which supports Apple Mac OS 9. For the Workshop West files, although the disks indicated MS-DOS 6, the presence of associated Aldus PageMaker (.PM4) files suggested that a graphical user interface and a version of Microsoft Windows were needed. As the disks were from the early 1990s, Windows 3.1 was the team's best guess based on the timeframe. The investigator decided to use Oracle VirtualBox to emulate Windows 3.1 due to their past familiarity with the emulator.<sup>20</sup> Additionally, due to some inherent confusion about how to navigate the legacy Windows 3.1, the investigator also installed Norton Desktop 3.0. Norton added a visual desktop that provided easier navigation for the investigator as it resembled more modern visual interfaces.

## CONFIGURING THE EMULATORS

The steps to configure each emulator vary based on the requirements of each environment. In general, environments that require operating systems and further software dependencies take more time to configure and more expertise to operate. Simpler environments without operating systems, like many computer game consoles, allow for effective load-and-play emulators. If one has the correct read-only memory (ROM) or executable data, loading it into the emulator will immediately bring up the game or program.

Running SimCity in emulation is the most straightforward of the three examples. DOSBox is a popular emulator for MS-DOS systems. With a large community and copious documentation supporting a range of common use cases, the biggest hurdle is usually a basic understanding of the DOS command line prompt. However, several front-end applications will configure DOSBox automatically, obviating the need for a deeper understanding of DOS operating system interactions. The disk image of SimCity contains the game's data files and two executable programs, INSTALL.EXE, which installs SimCity into the emulated environment, and SIMCITY.EXE, which runs the game once installed. DOS configurations also include audio and visual settings, but DOSBox will usually function correctly with an installation's default options.

---

<sup>20</sup> Microsoft Windows home versions were based on MS-DOS up until the release of Windows XP. As a result, when installing earlier Windows versions, it is sometimes necessary to install MS-DOS first. In this case, we installed MS-DOS 6.22 first and then proceeded with the installation of Windows 3.1.

The configurations for Sheepshaver and VirtualBox are more complex as each emulator requires the acquisition and installation of a guest environment's operating system. Typically, a researcher will need to locate and image original installation media or be comfortable with downloading such media from online sources.<sup>21</sup> In either case, the process involves getting the emulator to boot the operating system installation program. This involves loading the operating system disk image by selecting the correct image file in the emulator's configuration screens. When the emulator starts, the disk image will be loaded in the appropriate virtual drive (in this case a CDROM drive) and installation will proceed through user interface prompts. Figure 10 shows a Bottle Village optical disc loaded into the virtual drive of an Apple Mac OS 9 installation in the Sheepshaver emulator. Once installed, the installation media is removed from the virtual drive as the operating system can then boot from the emulator's virtual hard drive storage.



Fig. 10. The BV Booklet CD-ROM image is loaded in the virtual disc drive (circle in the top right). The image contains the folder "BVbook," which then contains the QuarkXPress files and further graphics and font folders. (Image credit: Adam Larson)

<sup>21</sup> Several enthusiast sites provide installation images for legacy operating systems, but these sites operate in dubious legal territory (at least in the US). The Internet Archive also hosts a significant number of operating system and software program images.

## EMULATING

For simpler environments, starting the emulator will also start the target program. The SimCity example is the closest to this ideal, but based on the setup above, a user will still need to navigate to the game's installation program before starting it directly on subsequent reboots. The use of DOSBox in the case of SimCity removes the need to install an operating system but still requires installing the game program itself.

Emulating the correct operating system is only the beginning for the Workshop West disk and Bottle Village Zip 100 disk examples. The next step is to install additional programs inside each emulator to read the files of interest. In both examples, researchers attempted to read files made with desktop publishing software applications: Aldus Pagemaker 4 for Workshop West, and QuarkXPress 5 for Bottle Village. Installing software into an emulated environment follows a similar method as installing the operating system. Disk images for both programs were located and then loaded into the virtual drives of each emulator.<sup>22</sup> The version of Pagemaker was apparent through the .PM4 file extension, and QuarkXPress file versions (4.0) were included in some file names. QuarkXPress 5.0, which was released shortly after the creation date of the target files, managed to interpret the previous file versions.<sup>23</sup>

The results of emulation work are shown in figures 11 and 12. In figure 11, we can see that the VirtualBox Windows 3.1 installation is running Aldus PageMaker 4.0 and shows the “Explore” logo for Workshop West’s 1995–1996 theatrical season. In figure 12, we see Sheepshaver running QuarkXPress 5.0 to recover the Bottle Village “BV Booklet,” a brochure and article describing the landmark with additional photographs. The “Explorer” logo is also present on the disk images as a .TIFF file, but the “BV Booklet” images are embedded within a Quark file and are therefore only viewable through emulation.

---

<sup>22</sup> In these cases, images of both programs’ installers were present on the Internet Archive and the abandonware site WinWorld. Each occupies dubious legal territory, with WinWorld’s policies stating that they only provide access to programs released before 2001 and no longer available for commercial purchase.

<sup>23</sup> However, it is unclear whether specific formatting issues exist. We would need to locate a version of QuarkXPress 4.0 and to do a screen-by-screen comparison to determine whether there are significant differences between versions.

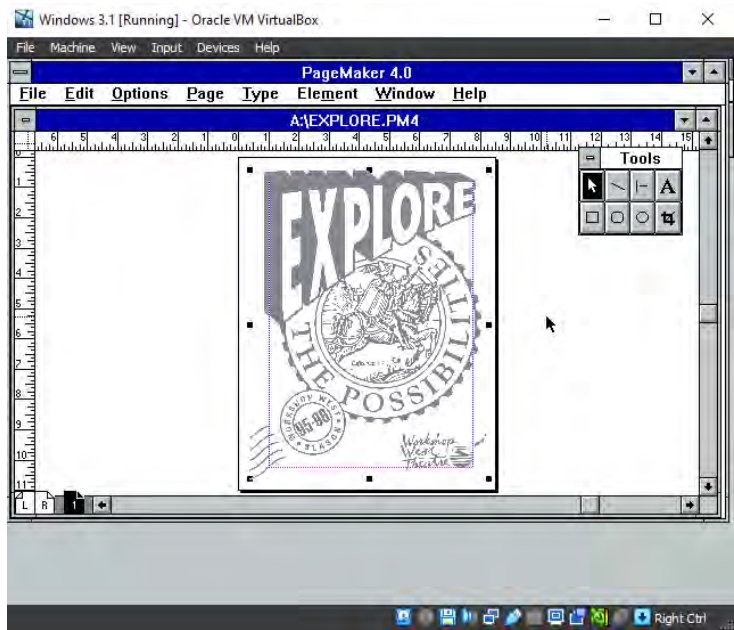


Fig. 11. A promotional image for Workshop West viewed in PageMaker 4.0 running in Windows 3.1 within Oracle VirtualBox on a Microsoft Windows 11 PC (Image credit: Logan Thienes)



Fig. 12. A page from a brochure booklet about the Bottle Village located within a QuarkXPress 4.0 file rendered in QuarkXPress 5.0 on the SheepShaver emulator running on a Microsoft Windows 11 PC. QuarkXPress 5.0 was used due to difficulties encountered with the installation of QuarkXPress 4.04. (Image credit: Adam Larson)

## DOCUMENTATION AND TROUBLESHOOTING

A final and complementary step in any emulation workflow is the generation of documentation about the configuration process. In our examples, the execution of SimCity within a DOSBox environment did not require generating significant documentation. This is due to the abundance of information online regarding the configuration of DOSBox to run older DOS-based computer games. Game emulators are generally extensively documented and represent the most widespread uses of emulation as a recovery technique. For the other two examples, however, the research teams took extensive notes about the recovery processes, including detailed information about sources for configuration information and dependencies needed for the various installation processes.

There is no general standard for the description of software recovery processes, but many detailed examples of recovery workflows are available in published literature (Dietrich et al. 2016; Rieger et al. 2015). For the above examples, an additional step involved the further configuration of the materials within a cloud-based emulation solution. Although the teams initially used local emulation, the researchers for Workshop West and Bottle Village also had access to the EaaSI cloud-based system. After configuring the emulations locally, each team reimplemented those configurations through EaaSI. This allowed the researchers to save their work, in a stable and shareable form, for future users at both institutions. One benefit of cloud-based solutions is that although some of the emulation configuration is tedious, confusing, and potentially frustrating, once a system is correctly configured, it will likely keep working indefinitely (assuming the system is maintained and that the underlying emulator is untouched).

## Limitations of Emulation for Preservation

---

While emulation can be a great solution for accessing past software and data, inherent limitations can impact how well emulation will work for specific circumstances.

### ACCURACY IN REPRODUCING HISTORICAL SOFTWARE

Emulation is a tool to access historical software, but that software was originally designed to run on specific historical hardware. As an emulator is a software program running within a host environment, that host may lack the ability to reproduce some period-accurate behaviors or interactions. Interactive software responds to input provided by a user and formulates a corresponding output back to them. If the user types a word using a keyboard or uses a mouse to click, the software reacts by updating a display to provide feedback and confirmation of user actions. The keyboard, mouse, and display showing visual output would all, historically, be individual hardware objects that may need accurate reproduction. Further, if a user wanted to access a historical feature that required additional hardware, like printing a document, the printer itself might need to be reproduced in some way. In software emulation, it may be difficult, or impossible in some cases, to approximate past physical hardware interactions or to show visual information in a fashion that is authentic to an original display. The physical context for software interactions is lost through emulation, or at best it can only be approximated.

### HARDWARE PERIPHERALS AND DISPLAYS

Some input peripherals, such as mice and keyboards, have not changed substantially over the last three decades, but more unique or complex input devices, such as joysticks, pedals, paddles, trackballs or remote controls, drawing pads, or any other discipline-specific devices create challenges for access if they are required for a particular interaction. Even if the peripheral is at hand, it might still need an obsolete connection cable or adapter to be interpretable for a modern environment, and, even then, someone might need to write a custom software interface (i.e., a device driver) to allow a host environment to interpret the peripheral's input. However, if the original input device is not available, then all the possible input actions provided by that device will need to be mapped (aligned with) actions on a different device. In many cases, a general keyboard will be used in place of an original device's buttons and other functions.

Mapping is sometimes a difficult and time-intensive addition to an already time-intensive process (Acker 2023). Both the use and imitated mapping of legacy peripherals will impact how users interact with and comprehend a historical software object. Instructions and documentation will be needed in either case to provide users with the necessary control information, adding to the burden of those providing access.

Display technologies also present challenges, as advancements in display hardware and software have rendered whole classes of formerly ubiquitous devices obsolete. In the 2000s, sales of cathode ray tube (CRT) displays were overtaken by flat screen displays, like liquid-crystal displays (LCDs). Flat screens offered many practical benefits, as they were lighter, larger, more power efficient, and much thinner than CRTs. Because many historical software objects, including all computer games, were created under the assumption that they would be represented on a CRT, they take advantage of certain color-blending features and fuzziness that the stark clarity of flat screens have since eliminated. This means that using modern displays to emulate content designed for historical ones fundamentally changes their visual appearance (Rieger et al. 2015). This distinction may not be important where visual fidelity is not an issue (e.g., reading the text of a historical document), but other, interactive uses of software may require historical display accuracy (in things like refresh rate and color representation) to be understood relative to their historical contexts.

Peripheral and display issues are fundamentally concerned with getting an emulation to accurately reproduce, as close as technically possible, the representation of a past object in a modern environment. Certain emulation use cases hinge upon rendering accuracy, where an otherwise functional (but inaccurate) emulated environment can significantly alter the representation of material, perhaps in ways a user cannot easily discern (Cochrane 2012). It is worth noting that complete accuracy is a fallacy, as the act of emulation can never recover the exact same access experience as the original material in its original hardware context (Rieger et al. 2015). Emulation is instead a proximal and ideally more accessible version of the material.

## CONTEXT OF OBJECTS AND EMULATORS

The socio-historical context of a digital object, what it means to provide a user access to that object, and what that access entails all affect the implementation of an effective emulated solution. Discussing what it means to have a functional, complete, and correct rendering (Wehrle 2014a; 2014b) is straightforward in theoretical terms, but each digital object or digital material will have a different use context, which is tied

into the purpose of emulation as well as the significant properties of the materials (Acker 2021; Rieger et al. 2015; Hedstrom et al. 2006). The materials' use contexts and properties affect decisions about the form of access as well as the logistics involved in providing it. Some materials may be emulated for purposes that differ from the software's original purpose (like opening an obsolete digital art file not to manipulate it but to simply observe it). Other emulation uses may raise ethical questions about providing or limiting the potential for the modification of past objects. When looking around Salman Rushdie's personal computer (Loftus 2010), for example, should a user be able to modify things?

Additionally, some software has set paths that users must follow to gain desired access to the digital object, while other environments contain millions of potential ways to access the digital object. Given the range of an object's significant properties (the aspects of a particular data representation that must be maintained to ensure authenticity and credibility) and how substantially emulation choices can differ, access decisions are dependent on documentation of the digital object, legacy software, and emulator. Documentation may include descriptions of the significant properties of the digital object, setup and configuration instructions, and information about potential bugs or software failures that may be triggered during emulation. Additionally, the documentation may include software requirements for the object or emulator or other dependencies, such as a specific operating system or the name and version of legacy software programs required to access the digital object in the emulated environment in a way that is accurate to the intent of the emulation experience (Wehrle 2014a; Wehrle 2014b). Incomplete or insubstantial documentation is frustrating, but it can also be a severe limitation to successful emulation.

## DEPENDENCIES

When it comes to dependencies, having complete documentation that conveys requirements is half the puzzle, while the process of acquiring and implementing the dependencies poses several limitations as well. Some dependencies are products made and sold by for-profit companies and are subject to intellectual property limitations, which may make those dependencies difficult to resolve. Specific versions of legacy software programs may be unavailable to purchase or license due to lack of market value or due to the developing company having shut down in the past. Working with modern dependencies can be challenging in other ways, as advances in digital rights management (DRM) have been designed to stop unauthorized access and software piracy, thereby restricting

emulation access and options for dependencies.<sup>24</sup> Both legacy and modern dependencies may require those searching for specific dependencies to turn to alternative or community-managed sources or to try to find workarounds that are an imperfect fit. The process of setting up an emulated system can be full of technical trial and error, and additional complexities relating to dependency access and implementation can make it difficult to determine whether issues seen during emulation are caused by the emulation process or whether they are original to the legacy software program itself.

## **OBSOLESCENCE OF EMULATION**

As we discuss the use of software emulation in interpreting historical software, it is important to consider that many emulators are also software, and that they, in time, will eventually become legacy historical software (Rieger et al. 2015) with their own limitations (Wehrle 2014a; Wehrle 2014b). As legacy software, an emulator is subject to all the limitations discussed in this section. Hardware obsolescence can be challenging for long-term emulation use when considering peripherals, adaptors, obsolete media drives, or specialty devices designed for emulation. Those additional emulated components will also need to be preserved, and as the number of obsolete hardware devices increases over time, so too will the need for their emulated counterparts. The complexity and number of emulated components will amplify challenges around documentation and dependencies, making it important to keep information on emulation preservation, even if preservation actions are not as likely to directly affect emulation activities as the previous limitations. Emulator preservation involves either taking an emulator that worked in a previous environment and reimplementing (i.e., reprogramming) it in a new environment or running that past emulator with another emulation of the emulator's original environment.<sup>25</sup> This second scenario leads to diminishing returns, as nesting emulations within each other incurs performance costs.

## **STAFF RESOURCES**

Efforts to use emulation for preservation may be severely hampered by the amount of staff resources needed. This is partially due to the complex nature of emulation, which requires sometimes significant trial

---

24 Software Preservation Network's "A Preservationist's Guide to the DMCA Exemption for Software Preservation" (Albert and Lee 2022) and "Code of Best Practices for Fair Use in Software Preservation" (Cox et al. 2019) are helpful resources for intellectual property and rights management in relation to software emulation.

25 Emulator reimplementation is also known as "porting" or "rehosting" the emulator's code in a new environment. In the literature, nesting emulations is referred to as "stacking" or "chaining."

and error in both the location and configuration of dependencies for emulated systems. Each aspect of an emulation use case—from developing emulators, researching emulation and software dependencies, setting up the emulator, troubleshooting, getting to the point of patron or visitor access, to preserving the emulator itself—requires substantial time and labor investments from cultural heritage institutions that are already overburdened and understaffed. Many demonstrated emulation use cases are bespoke and brittle since there are still no general standards for how to describe emulation environments and no convenient ways to reproduce and share environments. While EaaS and other solutions will help in this regard, those solutions will also depend on further research, education, and the maintenance of legacy documentation and collections of historical software objects.

## Literature Review

---

In preparing this report, the authors organized a wide-ranging review of literature addressing emulation within a preservation context. The initial collection of materials was assembled primarily by searching through online scholarly databases for articles related to both “digital preservation” and “emulation.” From there, we followed citation chains until a point of reasonable “saturation” was reached: that is, when we stopped uncovering new past projects or threads of inquiry. While the space of emulation for preservation is not large by academic standards, the literature review included close to 200 publications—including websites and blog posts—all related to the use of emulation for digital document recovery. This section will describe some of the key insights from that review and dive further into our methodology for analyzing references. The full review listing is available as a Zotero shared bibliography.<sup>26</sup>

### HISTORY OF EMULATION FOR PRESERVATION PROJECTS

The idea of using emulation as a preservation technique is generally attributed to Jeffrey Rothenberg, who introduced the idea in an article in *Scientific American* in 1995 (Rothenberg 1995). However, the first practical development of emulators for explicitly preservationist ends occurred nearly half a decade earlier within the hobbyist computer gaming community.<sup>27</sup> Research into emulation for preservation began in earnest in the late 1990s with major projects originating at the University of Michigan, the University of Leeds, and the National Library and Archives of the Netherlands. The leaders of these projects sought to justify the use of emulation as a solution for digital preservation in the face of the then-current focus on digital content migration. The prevailing school of thought at the time was that emulation was too technically complex, resource intensive, and impractical to function as a viable solution for the digital preservation of legacy software and formats (Bearman 1999).

Much of the work over the first decade of the twenty-first century focused on either the “significant properties” of digital objects reproduced through emulation or on new methods for addressing deficiencies in the

---

<sup>26</sup> [https://www.zotero.org/groups/5899226/overview\\_of\\_emulation\\_as\\_a\\_preservation\\_method/library](https://www.zotero.org/groups/5899226/overview_of_emulation_as_a_preservation_method/library)

<sup>27</sup> The earliest reported program is a Nintendo Family Computer (Famicom) emulator with basic game support running on a Japanese FM Towns system in 1990. This creation date is disputed within the community, but by 1995, there were an array of game emulators for multiple systems in community development and use. Ross and Gow (1999) provide an early overview of the activity in this space from the perspective of digital preservation practitioners.

creation of suitable emulator solutions. Significant properties are the aspects of a particular data representation that must be maintained to ensure authenticity and credibility in the future. These are paramount in certain contexts, like the reproduction of fine-art works, but may not be as important for more quotidian items, like computer games and desktop software. For example, Hedstrom et al. (2006) note that some users appeared to prefer emulated versions over the originals due to increased speed and the ease of using current (and therefore familiar) peripherals. Comparative work in significant properties has usually involved having users test a piece of software or computer game on both the original hardware and through an emulated reproduction.<sup>28</sup> Due to the highly variable nature of software interactions, maintaining a clear and concise set of properties can be difficult due to the different potential circumstances of each software object's intended use and use within a particular community.

Methodologies for addressing foundational issues in emulation focused on the fundamental challenge of emulator maintenance over time. Preserving an emulator is similar to preserving any other digital object; as environments change, an emulator will need to be either nested within another emulator or reimplemented for a new environment. To get around these issues, some have proposed the creation of Emulator Virtual Machines (EVM) or a Universal Virtual Computer (UVC) (Rothenberg 1999; Lorie 2001). Although each approach addresses a different technical challenge, both approaches stem from a desire to write specifications for a particular environment that can reproduce that environment in the future. They seek to create an abstraction that allows for a single virtual interface to support the implementation of an emulated environment. In this way, one would only need to reimplement that single interface in the future instead of having to implement a larger collection of emulators. Some projects, like the Dioscuri emulator and Keeping Emulation Environments Portable (KEEP), have attempted to implement some form of EVM for testing, but neither project has been updated since around 2012 (van der Hoeven 2007; Pinchbeck et al. 2009).

In the early 2010s, the possibility for cloud-based solutions to emulation began to emerge with projects like Carnegie Mellon's Olive Executable Archive and the University of Freiburg's bwFLA (Satyanarayanan et al. 2015; Rechert et al. 2012). The latter project developed into the current EaaSI framework with support from several US philanthropic organizations. Browser-based emulation had existed in a limited way on

---

<sup>28</sup> The CaMiLEON (1998–2001) and Preserving Virtual Worlds 1 and 2 (2008–2012) projects both make use of some comparative emulation research (Granger 2000; McDonough et al. 2010).

the earlier Internet, but not under a larger preservation focus. With the development of compilers that could convert formerly native code, like C++, into JavaScript, the possibility for in-browser emulation became a more practical reality. The Internet Archive launched the Internet Arcade, a collection of games playable in-browser in 2014, and saw millions of visits to the collection over the next few months (“Internet Arcade” 2014). Clearly, emulation directly in the browser was going to be a significant contribution to the preservation space.

The mid-2010s also saw the founding of the Software Preservation Network, which supported coordination of the early EaaSI network as well as the Fostering Communities of Practice (FCOP) project that supported studies into emulation use at memory institutions (Acker 2020). EaaSI also supported further research efforts into pedagogical use of emulation systems as well as continuing efforts in descriptive and technical metadata for emulated environments (Cardoso-Llach et al. 2021; Cochrane et al. 2022). Recent work focused on expanding the abilities and features of emulation frameworks and finding ways to expand the use of environment snapshots and saved states (Gieschke et al. 2023).

In general, there appears to be no doubt about the potential for emulation as a digital preservation solution. However, the work of the last two decades has not resulted in emulation being routinely included in institutional workflows. The technical complexity of emulation, paired with the time needed to develop expertise in the area, makes it difficult for institutions with already strained resources to commit to emulation work. Coupled with issues related to descriptive and technical metadata for emulator configurations, preservation concerns, and the general legal fog surrounding the use of emulation, the future is still unclear. The increasing prevalence of technologies that remove the complex setup requirements for emulation through web- and cloud-based access, along with the growing strength of communities of practice, may help turn the tide.

## METHODOLOGY

After gathering the initial articles, we reviewed their abstracts and summaries (or in some cases briefly skimmed them) to determine the contexts in which emulation was discussed. Each reference was open coded by the reviewer to allow for greater flexibility of coverage. Following the initial coding, we condensed and combined the codes to settle on topical areas to elaborate for the review.<sup>29</sup> Below, we summarize the general

---

<sup>29</sup> The codebook is available through a published Google Sheet: [https://docs.google.com/spreadsheets/d/1iITdmF\\_LK79mHd-q\\_01PDLgbYLitrC8W](https://docs.google.com/spreadsheets/d/1iITdmF_LK79mHd-q_01PDLgbYLitrC8W). Initially, 107 codes were identified; these were combined into 53 condensed codes.

topical areas that emerged from the review and include specific examples of projects or findings that could help inform current and future directions for using emulation in preservation activities.

In some cases, a reference was deemed to be “out of scope” and was not included in the final review. Each scoping decision is justified with a note in the project’s shared Zotero bibliography.<sup>30</sup> In general, a reference was out of scope if it mentioned emulation only in passing or in a context that was totally separate from preservation activities; this was mainly in cases where emulation appeared in computer science or code validation venues concerned with performance or validation testing for software systems.

Emulation has a long history in technical fields and commercial applications that predates its use in preservation. While there is some overlap in emulator use between these disciplines, it is rare for literature about commercial emulation to refer to cultural heritage preservation use cases.

## COMMON THEMES IN THE LITERATURE

This section will briefly discuss the major themes that arose through the analysis of the reviewed literature. There is not space to delve deeply into each of these topics, but a general awareness of the areas of current, defunct, and potential research into emulation should help readers follow ongoing advances in the field.

## GAME STUDIES AND COMPUTER GAME COMMUNITIES

A major area of emulation research and study exists within the game studies and computer game enthusiast communities.<sup>31</sup> Our tag related to articles that commented on or made use of computer games was the most prevalent tag in the set. This is unsurprising given that the most popular general use of emulation is the recovery of historical games and that many modern companies work on re-releasing and updating older games for sale in digital storefronts. Game emulators are also an active space of development, with enthusiast communities organized around a significant number of emulators for both well-known and relatively unknown systems. Game scholars also discuss the need for emulation as a means of studying past gameplay, and there are numerous references to the popular exhibition of games through emulation (Tucker 2012, Young 2013). As Salvador (2023) noted, 87 percent of past games are technically out of print

---

<sup>30</sup> See note 26.

<sup>31</sup> The general tag for “computer games” appeared 43 times within the review items.

and legally unavailable for purchase; this leads many to use emulation as a means of recovering past gaming experiences.

Many emulation studies or investigations also use computer games as case study objects, and case-based research is another significant pattern within the literature.<sup>32</sup> Although there is not space to enumerate them here, most of the major emulation projects of the past 25 years included some form of individuated object study as part of the research goals. This makes sense as software and computer games are used within different cultural contexts and technical environments. Individual case studies also appeared in significant properties research as a way to generate sample properties for different classes of software objects in emulation. These cases generally fall into four categories outlined by van der Hoeven and van Wijngaarden in 2005: gaming, historic computing, business virtualization, and cross-platform emulation. “Cross-platform” here implies the installation of a guest environment within a differing host environment. In 2022, Cochrane et al. expanded the set of case items to networked software and mobile applications, two software use cases that were not as prevalent in studies more than 20 years ago. Additionally, there is the consideration of web archive emulation and the reproduction of past browser environments, which were also less of a concern in the early 2000s.

## DEFINING AND CRITIQUING EMULATION METHODS

Another trend is in the definition, overview, and critique of emulation methods.<sup>33</sup> Throughout the literature, there is a constant need to redefine and reintroduce specific perspectives and critiques of emulation. This might be due to the necessity of continuously introducing emulation as a topic to international funding agencies, resource allocators, and related disciplinary and professional communities, but the result is that many research papers reiterate the same points, sometimes with 10 or 20 years of separation. Earlier emulation work tends to defend it as a method that is distinct (and more useful) than digital migration, whereas later works assume emulation is a solution and restate why it is important to have interactive, executable access to past software objects.

---

32 Fifty-two articles were tagged either “case studies” or “use cases,” indicating the presence of either a specific case study or an elaboration on a set of use cases for emulation methods.

33 Thirty-four articles featured partial (or complete) overviews of emulation as a method or the state of emulation research at the time. Other emulation-informational tags included “evaluation” (33), “critique” (32), “evangelism” (11), and “recommendations” (4). These tags were not mutually exclusive.

## COMPARATIVE METHODS

Comparative preservation methods also featured in many works, usually in comparing emulation to both migration and technology preservation, the latter being any form of physical hardware preservation.<sup>34</sup> As all the literature reviewed explicitly focused on emulation, it is unsurprising that it mostly supported emulation as a preferred method for future digital preservation workflows. Many of the comparative studies also overlapped with work investigating significant properties, as a major critique of emulation relative to the other preservation methods was the extent to which emulation did or did not accurately reproduce historical records and data.

## DIFFERENCES IN REPRESENTATIVE SAMPLES

In general, preservation in the context of libraries and archives is over-represented in our sample relative to preservation in museums and art galleries. This might be due to selection bias, in that information workers are more likely to publish about the use of preservation methods, whereas those recovering art works for exhibitions might have more practical concerns and time constraints. Additionally, while the use of enthusiast emulation is mentioned extensively (and tangentially) in the literature, there are few references to either specific investigations of hobbyist and enthusiast community preservation or dialogues between those communities and GLAM institutions. This may be due to the conflicting priorities of the communities, the questions of legality surrounding some emulation work, or a range of other factors; however, it seems that those who make use of emulation for preservation and those who make the emulators are not in contact to the extent one might expect.

Geographically, a majority of the research and projects on emulation for preservation tasks originated in Europe over the last 25 years. The Networked European deposit library (NEDLIB) and Digital Preservation Testbed projects out of the Netherlands, the Keep Emulation Environments Portable (KEEP) project, Baden-Württemberg Functional Long-Term Archiving and Access (bwFLA), and then EaaS all originally derived from funded work at European archives and research institutions. Additional projects involving emulation existed in Australia and North America as well, but not at the same volume (at least as concerns research output).

---

<sup>34</sup> There were 30 instances of comparative discussion of preservation methods, with 28 articles mentioning migration in contrast with emulation. Twenty articles featured some discussion of significant properties research.

Overall, conversations about emulation as a solution appear to occur mostly in the West, with little to no mention of similar projects developing out of the East or South Asian contexts or from South America. There is obviously a language and geographic bias within the literature review, as the authors are all native-English speakers and based at North American institutions and organizations. Thus, further work in exhausting non-Western scholarship on the topic of emulation would be beneficial.

## Conclusion

---

### A FIELD IN NEED OF GUIDANCE AND COMMUNITY

This report's goal is to provide an introduction, overview, and contextualization for the use of emulation within digital preservation workflows. Over the last 25 years, perspectives on emulation have shifted from simple curiosity and concern over its technical challenges to a general admission that emulation is likely the most sustainable means for preserving access to complex software-dependent objects and data into the future. Although the promise of widespread adoption of emulation methods is potentially significant, its use is stymied by confusing terminology and technical challenges. The reality is that most institutions do not yet have the resources and knowledge to effectively enact an emulation-based preservation program. This report was written to address these hurdles through helping to articulate and answer common foundational questions about emulation that have cropped up again and again in conversations within the digital preservation community.

Describing and defining emulation-based approaches is not new. As cited above, there are many reports illuminating the intricacies of emulation and emulators as well as describing detailed workflows related to emulation use. What has been missing, however, is coherent and consistent communication between GLAM practitioners making use of emulated methods, and the subsequent formation of a community around them. There are promising moves in this direction, such as with the Emulation as a Service Infrastructure (EaaSI) project's expansion and emergence as a full-fledged service (with the Australian Emulation Network as a leading community example). The Software Preservation Network (SPN), of whom this report's authors are volunteer members, is working to build an international community of practitioners centered on advocacy for and communication about emulated access to digital records. We may well see more international collaboration grow from these seeds over the next few years.

What is most needed is a location for conclusive, shared documentation about emulation-based workflows and methodologies. There is a lot of information available online, but it is disparate, dispersed and usually concerned with specific niches within the greater preservation and hobbyist communities. We hope this report is the beginning of a concerted, continuous effort to marshal information about use cases, projects, and emerging emulation-based solutions and to determine common criteria and practices that cut across different communities of practice. We need a

coherent, progressive narrative around emulation practices: both how they have developed and helped so far, and where they can go into the future. To that end, this report's authors aspire to create a complementary, online, community-managed resource that can act as a initial rallying space for further collaboration. Life and time constraints have pushed that goal off, but we hope to work with SPN, EaaSI, and others within the emerging community to create such a resource.

While we are still working toward that goal, in the interim we hope to see further elaboration on the topics discussed in this report. The community needs to share what it is doing with emulators, virtual machines, and all manner of virtualization schemes. Our literature review, briefly summarized above, is just scratching the surface of emulation's history within the field, and a more elaborate and detailed review will only reveal more useful trends and overlapping use cases. Even a cursory glance online reveals all manner of community wikis, enthusiast projects, and detailed (if highly technical) information about emulators and emulation approaches. A single guide cannot encompass all of the available trends and information, but we want to point out that such resources exist, and that better communication, collaboration, and organization around them will only strengthen and reinforce current trends towards virtualized access to the historical record.

## NEXT STEPS IN EMULATION

While emulation will allow for greater access to a larger swath of the digital past, there is still much work to be done in the development and adaptation of this approach to newer digital technologies. The previously common alignment of singular software objects installed on individual computers has given way to a broad array of networked, cloud-based, and mobile computing configurations that are not as amenable to current emulation approaches. Systems that require continuous network support, or that manage versions and updates remotely, will not be preservable as singular records. It is possible to emulate networked software configurations, but, given that most remote server applications are proprietary or closed-access, it is unlikely that network dependent objects will be recoverable. For these items, more work is needed on effective ways to either imitate past network traffic (with systems like Webrecorder<sup>35</sup>), or to migrate items off cloud-based services. Software objects that rely on changing web and communications standards (like GPS applications) will also require the development of novel remediation and preservation strategies, of which emulation will only be a part.

---

<sup>35</sup> <https://webrecorder.net/>

Preservation of mobile applications and games is also underdeveloped, but knowledge about these works will likely be key to understanding the history of the early 21st century. There are orders of magnitude more mobile games currently available than there are all other software objects, for all devices, that have been created since the dawn of modern computing. Simply grappling with the scale of mobile and ubiquitous computing applications will likely overwhelm memory institutions. Adding in appraisal, curation, and description tasks for these items—and then getting them running again through emulation—we are clearly approaching the need for wholly new disciplinary and technical formations to deal with the flood.

Finally, as noted in the introduction, this report and its attendant literature review are primarily based on work carried out by Western institutions and written in English, creating an inherent bias about the how, what, and why of digital record preservation practices. We want this report to be accessible to anyone, regardless of their geographic or technical context, and note that more extensive efforts need to be made to locate and adapt emulation-based practices for local communities and conditions.

## **WHERE TO GO FROM HERE**

This report is the starting point for a longer journey into emulation-based preservation methods. We recommend reviewing the report and its basic use-case examples as a preliminary guide into the use of emulation in preservation work. The past projects and publications referenced in this report can provide further, and much more nuanced, details relative to your specific disciplinary and records context. More work is needed to organize, categorize, and share emulation-aided workflows with the greater digital preservation community. We hope that your future efforts will help communicate the importance of historical software records access to your patrons, administrators, and other audiences currently unaware of emulation's potential benefits. As noted, following this report's publication, the authorship team will be working to build an online, collaborative space to share emulation workflows and use case examples. The executable past is fading more day over day, and only a concerted, community effort will allow for that past to be recoverable and accessible into the future.

## References

---

- Acker, Amelia. 2020. "Emulation Encounters: Software Preservation in Libraries, Archives, and Museums." *Proceedings of the Association for Information Science and Technology* 57, no 1: e279. <https://doi.org/10.1002/pras.2.279>.
- . 2021. "Emulation Practices for Software Preservation in Libraries, Archives, and Museums." *Journal of the Association for Information Science and Technology* 72, no 9: 1148–1160. <https://doi.org/10.1002/asi.24482>.
- . 2023. "Accessing Software: Emulation in Information Institutions." SSRN Scholarly Paper. Rochester, NY. <https://papers.ssrn.com/abstract=4450195>.
- Albert, Kendra, and Kee Young Lee. 2022. "A Preservationist's Guide to the DMCA Exemption for Software Preservation, 2nd Edition." Software Preservation Network. <https://doi.org/10.5281/zenodo.7328909>.
- Bearman, Daniel. 1999. "Reality and Chimeras in the Preservation of Electronic Records." *D-Lib Magazine* 5, no. 4. <https://www.dlib.org/dlib/april99/bearman/04bearman.html>.
- Benner, Jessica G., Seth Erickson, Wendy Hagenmaier, Monique Lassere, Christa Williford, and Lauren Work. 2022. "Supporting Software Preservation Services in Research and Memory Organizations: A White Paper from the Software Preservation Network's Research-in-Practice Working Group." Software Preservation Network and Council on Library and Information Resources. <https://doi.org/10.5281/zenodo.7272726>.
- Cardoso-Llach, Daniel, Eric Kaltman, Emek Erdolu, and Zachary Furste. 2021. "An Archive of Interfaces: Exploring the Potential of Emulation for Software Research, Pedagogy, and Design." *Proceedings of the ACM on Human-Computer Interaction* 5, no. CSCW2: 1–22. <https://doi.org/10.1145/3476035>.
- Cochrane, Euan. 2012. "Rendering Matters: Report on the Results of Research into Digital Object Rendering." New Zealand Archives. January 3, 2012. [https://assets.ctfassets.net/etfoy87f9he/5zYVqadG14U3rVP6gHP1z9/c51d5a8907be57d202d26be2e49f84e1/rendering\\_matters.pdf](https://assets.ctfassets.net/etfoy87f9he/5zYVqadG14U3rVP6gHP1z9/c51d5a8907be57d202d26be2e49f84e1/rendering_matters.pdf).
- Cochrane, Euan, Klaus Rechert, Jurek Oberhauser, Seth Anderson, Claire Fox, and Ethan Gates. 2022. "Useable Software Forever. The Emulation as a Service Infrastructure (EaaS) Program of Work." In *Proceedings of the 18th International Conference on Digital Preservation*. Glasgow, Scotland: OSF. <https://doi.org/10.17605/OSF.IO/AY48C>.
- Cox, Krista, Patricia Aufderheide, Peter Jaszi, and Brandon Butler. 2019. "Code of Best Practices for Fair Use in Software Preservation." Software Preservation Network. <https://www.softwarepreservationnetwork.org/code-of-best-practices-for-fair-use-in-software-preservation/>.
- Craig, Ian D. 2006. *Virtual Machines*. Springer-Verlag.
- "CRT Shaders." 2022. Libretto Docs. April 3, 2022. <https://docs.libretto.com/shader/crt/>.
- Dietrich, Dianne, Julia Kim, Morgan McKeethan, and Alison Rhonemus. 2016. "How to Party Like It's 1999: Emulation for Everyone." *Code4Lib Journal* 32, no. April. <https://journal.code4lib.org/articles/11386>.
- Dourish, Paul. 2017. "Finding the Material in the Virtual: The Case of Emulation." In *The Stuff of Bits: An Essay on the Materialities of Information*, 61–80. MIT Press. <https://ieeexplore-ieee-org.login.ezproxy.library.ualberta.ca/document/7932795>.
- "Emulation." 2024. QEMU Documentation. July 24, 2024. <https://www.qemu.org/docs/master/about/emulation.html>.
- Franušić, Joël, Kathleen Tuite, and Adam Smith. 2023. "Playable Quotes for Game Boy Games." In *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 1–11. Lisbon, Portugal: ACM. <https://doi.org/10.1145/3582437.3582479>.

- Gates, Ethan. 2023. "Qawi." GitLab. December 15, 2023. <https://gitlab.com/EG-tech/qawi>.
- Gieschke, Rafael, Klaus Rechert, and Euan Cochrane. 2023. "Long-Term Preservation of a Software Execution State." *iPRES 2023*. <https://hdl.handle.net/2142/121093>.
- Granger, Stewart. 2000. "Emulation as a Digital Preservation Strategy." *D-Lib Magazine* 6, no. 10. <https://doi.org/10.1045/october2000-granger>.
- Hedstrom, Margaret, Christopher Lee, Judith Olson, and Clifford Lampe. 2006. "'The Old Version Flickers More': Digital Preservation from the User's Perspective." *The American Archivist* 69, no. 1: 159–187. <https://doi.org/10.17723/aarc.69.1.1765364485n41800>.
- "Internet Arcade." 2014. Internet Archive. August 6, 2014. <https://archive.org/details/internetarcade>.
- Kaltman, Eric, and Joseph C. Osborn. 2024. "The Potential of Cited Executable State for Software Object Access, Validation, and Research: A Preview of the GISST System." In *Proceedings of the 20th International Conference on Digital Preservation (iPRES)*. doi:10.21428/5676bf2d.b1b7d24e.
- Kaltman, Eric, Joseph Osborn, Noah Wardrip-Fruin, and Michael Mateas. 2017. "Getting the GISST: A Toolkit for the Creation, Analysis and Reference of Game Studies Resources." In *Proceedings of the 12th International Conference on the Foundations of Digital Games*. Hyannis, MA. <https://doi.org/10.1145/3102071.3102092>.
- "List of Emulators." 2024. Wikipedia. [https://en.wikipedia.org/w/index.php?title=List\\_of\\_emulators](https://en.wikipedia.org/w/index.php?title=List_of_emulators)
- Loftus, Mary J. 2010. "The Author's Desktop: How Famous Writers' Computers—like Salman Rushdie's Macs in Emory's Manuscript, Archives, and Rare Book Library—and Born-Digital Content Are Creating a Revolution." *Emory Magazine*. [https://www.emory.edu/EMORY\\_MAGAZINE/2010/winter/authors.html](https://www.emory.edu/EMORY_MAGAZINE/2010/winter/authors.html).
- Lorie, Raymond A. 2001. "Long Term Preservation of Digital Information." In *Proceedings of the 1st ACM/IEEE-CS Joint Conference on Digital Libraries*, 346–52. JCDL '01. New York, NY: Association for Computing Machinery. <https://doi.org/10.1145/379437.379726>.
- McDonough, Jerome P., Robert Olendorf, Matthew Kirschenbaum, Kari Kraus, Doug Reside, Rachel Donahue, Andrew Phelps, Christopher Egert, Henry Lowood, and Susan Rojo. 2010. *Preserving Virtual Worlds: Final Report*. Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign. <https://www.ideals.illinois.edu/items/17178>.
- Osborn, Joseph, and Eric Kaltman. 2024. "Toward A Game Citation and Reference Workbench." In *Proceedings of the 19th International Conference on the Foundations of Digital Games*, 1–4. Worcester MA: Association for Computing Machinery. <https://doi.org/10.1145/3649921.3656983>.
- "UTM." UTM. Accessed October 3, 2024. <https://mac.getutm.app/>.
- Pinchbeck, Dan, David Anderson, Janet Delve, Getaneh Alemu, Antonio Ciuffreda, and Andreas Lange. 2009. "Emulation as a Strategy for the Preservation of Games: The KEEP Project." In *DiGRA 2009: Breaking New Ground: Innovation in Games, Play, Practice and Theory*.
- Ponsard, Christophe. 2021. "Meeting Digital Preservation Requirements for Software through an Emulation Strategy." In *Proceedings of the 16th International Conference on Software Technologies (virtual)*. July 6–8, 2021.
- Rechert, Klaus, Isgandar Valizada, Dirk von Suchodoletz, and Johann Latocha. 2012. "bwFLA – A Functional Approach to Digital Preservation." *PIK - Praxis Der Informationsverarbeitung Und Kommunikation* 35, no. 4: 259–67. <https://doi.org/10.1515/pik-2012-0044>.
- Rieger, Oya Y., Tim Murray, Madeleine Casad, Desiree Alexander, Dianne Dietrich, Jason Kovari, Liz Muller, Michelle Paolillo, and Danielle K. Mericle. 2015. *Preserving and Emulating Digital Art Objects*. Cornell University Library. <https://hdl.handle.net/1813/41368>.
- Rinehart, Richard, and Jon Ippolito. 2014. "Generation Emulation." In *Re-Collection: Art, New Media and Social Memory*, 115–37. Leonardo. MIT Press.
- Rios, Fernando. 2016. *Preserving and Sharing Software for Transparent and Reproducible Research: A Review*. Johns Hopkins University. <https://doi.org/10.17605/OSF.IO/D4KEF>.

- Rosenthal, David S. H. 2015. *Emulation & Virtualization as Preservation Strategies*. LOCKSS Program, Stanford University Libraries.  
[https://digital.library.unt.edu/ark:/67531/metadc799755/m2/1/high\\_res\\_d/rosenthal-emulation-2015.pdf](https://digital.library.unt.edu/ark:/67531/metadc799755/m2/1/high_res_d/rosenthal-emulation-2015.pdf).
- Ross, Seamus, and Ann Gow. 1999. *Digital Archaeology: Rescuing Neglected and Damaged Data Resources: A JISC/NPO Study within the Electronic Libraries (eLib) Programme on the Preservation of Electronic Materials*. Electronic Libraries Programme Study P2. London: JISC.
- Rothenberg, Jeff. 1995. "Ensuring the Longevity of Digital Documents." *Scientific American* 272, no. 1: 42. <https://www.scientificamerican.com/article/ensuring-the-longevity-of-digital-d/>.
- . 1999. *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation: A Report to the Council on Library and Information Resources*. Council on Library and Information Resources.
- . 2000. *Using Emulation to Preserve Digital Documents*. The Hague.
- Salvador, Phil. 2023. *Survey of the Video Game Reissue Market in the United States*.  
<https://doi.org/10.5281/zenodo.8161056>.
- Satyanarayanan, Mahadev. 2018. "Carnegie Mellon Is Saving Old Software from Oblivion – IEEE Spectrum." <https://spectrum.ieee.org/carnegie-mellon-is-saving-old-software-from-oblivion>.
- Satyanarayanan, Mahadev, Gloriana St Clair, Benjamin Gilbert, Yoshihisa Abe, Jan Harkes, Dan Ryan, Erika Linke, and Keith Webster. 2015. *One-Click Time Travel*. Carnegie Mellon University.
- Slats, Jacqueline. 2003. "Emulation: Context and Current Status." Digital Preservation Testbed White Paper. Netherlands National Archives.
- Tucker, Abigail. "The Art of Video Games." *Smithsonian Magazine*, March 2012.  
<https://www.smithsonianmag.com/arts-culture/the-art-of-video-games-101131359/>.
- Van der Hoeven, Jeffrey. 2007. "Dioscuri: Emulator for Digital Preservation." *D-Lib Magazine* 13 (11/12). <https://www.dlib.org/dlib/november07/11inbrief.html>.
- Van der Hoeven, Jeffrey, and Hilde van Wijngaarden. 2005. *Modular emulation as a long-term preservation strategy for digital objects*. The National Library of the Netherlands.
- Van der Hoeven, Jeffrey, Jacqueline Slats, Remco Verdegem, and Hilde van Wijngaarden. 2005. *Emulation – A Viable Preservation Strategy*. The National Library of the Netherlands.
- Wehrle, Dennis, Thomas Liebetraut, and Klaus Rechert. 2014. "Preservation of Web Content – An Emulation-Based Case Study." In *Proceedings of the 11th International Conference on Digital Preservation, iPRES 2014, Melbourne, Australia, October 6–10, 2014*, edited by Serena Coates, Ross King, Steve Knight, Christopher A. Lee, Peter McKinney, Erin O'Meara, and David Pearson. <http://hdl.handle.net/11353/10.378700>.
- Wehrle, Dennis, Thomas Liebetraut, Isgandar Valizada, and Klaus Rechert. 2014. "Emulation-as-a-Service – Workflows and Infrastructure to Support Recomputable Science." In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 962–67. UCC '14. USA: IEEE Computer Society.  
<https://doi.org/10.1109/UCC.2014.157>.
- "X86-64." 2024. Wikipedia. <https://en.wikipedia.org/w/index.php?title=X86-64&oldid=1244768237>.
- Young, Nora. 2013. "Game On 2.0." *CBC Spark*. Accessed March 6, 2025.  
<https://www.cbc.ca/player/play/audio/1.1538894>.

## Acknowledgments

---

The authors would like to thank the Software Preservation Network (SPN) and the EaaS Research Alliance for supporting community development around emulation practices. All authors are also members of SPN's Technological Infrastructure Working Group and are grateful for SPN's direct organizational support for this work. Thanks are also due to Jessica Farrell, Jessica Meyerson, and Chelle Sanders of Educopia (SPN's fiscal host); and Wendy Hagenmeier, Claire Fox, and Ethan Gates of Yale University Library. The emulation workflow examples were made possible by the extensive work of Adam Larson, Morgan McMurray, and Desireé Caldera with funding from California State University Channel Islands Summer Undergraduate Research Fellowship (SURF) and by Logan Thienes with support from Maryna Chernyavska at the University of Alberta Archives.

## About the Authors

---

**Eric Kaltman** is a computer scientist and an Assistant Professor of Media & Technology Studies and History at the University of Alberta. He directs the Software History Futures and Technologies (SHFT) Group and researches emerging computational methods for digital record preservation.

**Winnie Schwaid-Lindner** is the Digital Preservation Librarian at Louisiana State University Libraries and holds an MA in Moving Image Archiving and Preservation. Her research is centered in the intersection of critical librarianship and digital preservation, with an emphasis on lowering barriers to entry and making digital preservation approachable for individuals of all backgrounds.

**Drey Jonathan** is a Digital Archivist at the New York Public Library.

**Andrew Borman** serves as the Director of Digital Preservation at The Strong National Museum of Play in Rochester, New York, where he leads the museum's efforts to safeguard and maintain its digital collections related to the study of play. His role involves developing and implementing strategies for digitization, long-term storage, and accessibility of digital collections.

**Alex Garnett** is a Technical Curriculum Developer at Temporal Technologies. He has previously been a technical writer, digital archivist, and librarian, and has been documenting emulation techniques for nearly 30 years.

**Larry Masinter** is a computer scientist and leader of Interlisp.org, a nonprofit volunteer group dedicated to the revival, using emulation, of the Medley Interlisp environment. Recognized by the Association for Computing Machinery (ACM) Software System Award for Interlisp, and an ACM fellow also for contributions to the creation of Internet standards, including the URL, URI, and HTTP/1.1 protocol. He was a principal scientist and researcher at Xerox's Palo Alto Research Center, AT&T Labs, and Adobe, where he contributed to media and metadata standards. For more about his work, visit <https://larrymasinter.net/>.



Council on  
Library and  
Information  
Resources

1800 Diagonal Road, Suite 600  
Alexandria, VA 22314  
[www.clir.org](http://www.clir.org)