A Gift to Another Age:

Evaluating Virtual Machines for the Preservation of Video Games at MoMA

by

Kirk Mudle

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Arts

Moving Image Archiving and Preservation

Department of Cinema Studies

New York University

May 2023

Abstract

This preservation project investigates the use of virtual machines for the preservation of video games. From MoMA's collection, Rand and Robyn Miller's classic adventure game *Myst* (1993) is used as a sample record to evaluate the performance of four different emulation options for the Mac OS 9 operating system—SheepShaver, QEMU, Apple's "Classic Environment," and Yale's University Library's Emulation-as-a-Service-Infrastructure (EaaSI) platform. Serving as the control for the experiment, *Myst* is first documented running natively on an original PowerMac G4 and iMac G3 at MoMA. The native performance is then compared with each emulation software. Finally, a fully configured virtual machine is packaged as a single file and tested in different contemporary computing environments. More generally, this project clarifies the risks and challenges that arise when using virtual machines for the long-term preservation of computer and software-based art.

Introduction

In the classic 1993 adventure game *Myst*, the player is pulled into an enchanted book containing an island replete with esoteric puzzles and confounding mysteries. Without being given any clear objective, the player must navigate the surreal landscape, interacting with objects and picking up clues in order to solve the game's many interconnected puzzles and, eventually, uncover the secrets behind the island's origins. Solving all of the puzzles in one of the game's worlds, called "Ages," allows the player to travel to different Ages, which are filled with even more complex puzzles. While traversing each Age, the player may pursue a half dozen lengthy virtual books containing lore about the game's world and characters, as well as various clues for puzzle solutions. One of these digital journals contains the following quote:

> "I think of this age as a gift to myself that I will wrap up and open someday in the future, only to discover that it has changed so much that indeed it is a surprise."

This ominous line is written by one of the game's main characters and the creator of the island, Artus. It references the strange way that time operates on the island of Myst. Reading further, the player will learn that the characters of *Myst* have the godlike power to write worlds into existence in the form of books. These books act as gateways to their corresponding Ages, where time moves exponentially faster than in the real world, allowing entire autonomous societies to develop without the creators' knowledge. The events of the Myst take place after Artus stumbles across a rather unpleasant surprise in one of these Ages. His two sons, gifted with similar abilities as their father, have gone mad with power and set fire to most of the books in his library, rendering their Ages inaccessible.

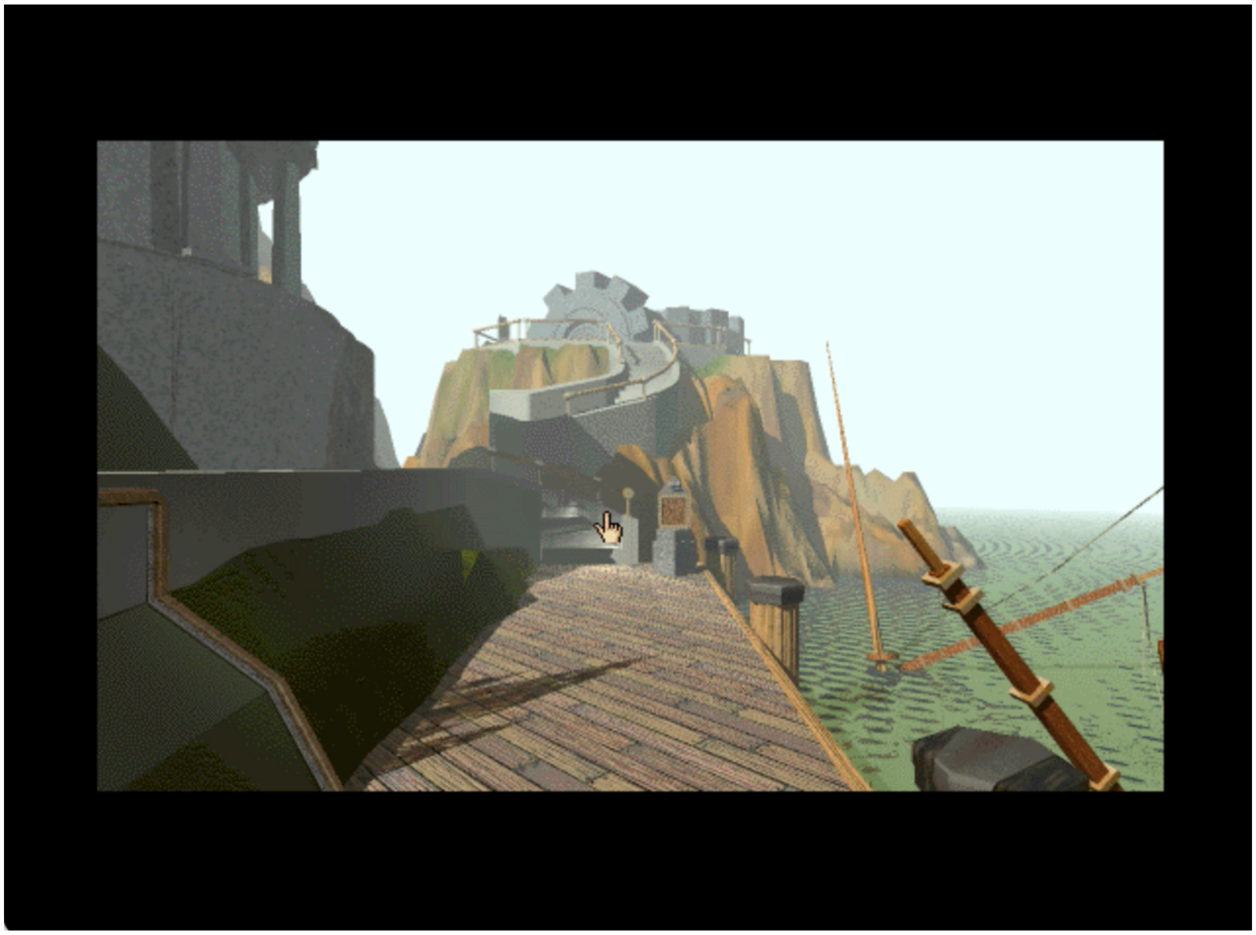


Figure 1. The opening screen from *Myst* (1993). Screen capture by author.

Myst is one of 14 titles acquired by the Museum of Modern Art (MoMA) in 2012 during the Museum's first wave of video game acquisitions. During the acquisition, the Museum received an original CD-ROM of *Myst* for Macintosh Computers, and the game's source code. In 2022, MoMA's conservators created a disk image from this CD-ROM and stored it in their secure digital repository. However, even with the original CD-ROM and disk image, MoMA currently has no way to access the game without relying on aging and vulnerable legacy hardware, namely a PowerMac G4 running Mac OS 9, an operating system that has not been supported by its original manufacturer and copyright holder, Apple, since 2001. Much like the burned books in Myst's library, these operating systems, and the proprietary hardware they run on, are now obsolete.

MoMA's Department of Architecture and Design (A&D) began collecting video games in 2011 as part of their ongoing research into "interaction design," an area of contemporary design focused on how objects influence the behaviors of their users.[1] The concept of interaction design was previously highlighted by the Museum in the 2011 exhibition *Talk to Me*. *Talk to Me* explored "the communication between people and things" and focused on objects and projects that, through direct interactions, "establish an emotional, sensual, or intellectual connection with their users."[2] In the decade since, the Museum has acquired a total of 36 video games spanning the entire history of the medium from the 1970s to the present. Just like *Myst,* the majority of MoMA's games require consoles or operating systems that are no longer in production or supported by their original manufacturers. In the coming decades, obsolescence may make it impossible to access thousands of video games without the use of digital preservation strategies such as emulation, virtualization, and migration.

To combat hardware and software obsolescence, galleries, libraries, archives, and museums (GLAMs) around the world have turned to emulation, defined by computer scientist Jeff Rothenberg as a technique "in which one computer system reproduces the behaviour of another system."[3] This process involves the recreation of either specific software applications or

[1]Paola Antonelli, "Video Games: 14 in the Collection, for Starters," *Inside/Out: A MoMA/MoMA PS1 Blog,* November 29, 2012, https://www.moma.org/magazine/articles/798.
[2]See "Talk to Me: Design and the Communication between People and Objects," The Museum of Modern Art, accessed December 17, 2022, https://www.moma.org/calendar/exhibitions/1071
[3]Jeff Rothenberg, "An Experiment in Using Emulation to Preserve Digital Publications," The Koninklijke Bibliotheek, Den Haag. https://www.semanticscholar.org/paper/An-experiment-in-using-emulation-to-preserve-Rothenberg/f95e30933c29f8b1b3b8b51c62343a8b99dc5cac, 6.

hardware architectures, or both.[4] A device or software running an emulator is referred to as the "host" system, while the device or software being emulated on the host System is referred to as the "target" or "guest" system.

Virtual Machines (VMs) are essentially another type of emulation, but instead of emulating a self-contained hardware system, like a video game console, a VM emulates an entire computer system with its own CPU, memory, network interface, and storage. A VM is a computer that "has no separate physical existence, but is part of the behavior of a physical computer…" These "mimic the instruction set and hardware configuration of some physical machine, or an abstract machine," as David Rosenthal's 2015 Mellon report on digital preservation put it.[5] VMs make it possible to run obsolete operating systems for the purpose of accessing legacy software applications.

VMs have been used widely in the Archival community for the purpose of accessing and preserving legacy software. However, few research studies have been conducted with VMs for the purpose of exhibiting and preserving computer and software-based artwork, such as video games. This type of emulation has been mostly used by hobbyists in online communities such as Reddit and E-Maculation—an online community that consists of both a Wiki and a user forum dedicated to the emulation of classic Mac computers in contemporary systems. Both E-Maculation's Wiki and forum include tutorials and boards (several of which were used for this very project) for various Mac emulators, including SheepShaver, Basilisk II, Mini vMac, QEMU and PearPC.

[4] Giovanni Carta, "Metadata and Video Games Emulation: an effective bond to achieve authentic preservation?" 193.
[5] David S. H. Rosenthal, "Emulation & Virtualization as Preservation Strategies," https://mellon.org/media/filer_public/0c/3e/0c3eee7d-4166-4ba6-a767-6b42e6a1c2a7/rosenthal-emulation-2015.pdf.

The focus of this thesis is to evaluate the use of virtual machines from two different, but related, perspectives. First, through a comparison between the performance of four different virtual machine softwares: SheepShaver, QEMU, Apple's "Classic Environment," and Yale University Library's Emulation as a Service Infrastructure platform (EaasI). And second, by building and testing "portable" virtual machines that could be used in order to access and exhibit *Myst* (and game's like it) into the future.

VMs are complicated pieces of software to configure and maintain. Downloading SheepShaver alone requires consulting at least two forum threads and cross referencing several different online tutorials to determine whether you have the correct resources at your disposal. To make things more challenging, crucial features of these VMs tend to be poorly documented and described in existing tutorials. This thesis seeks to compensate for these deficiencies by providing guides for configuring two aspects of the selected softwares: SheepShaver's portable VM packages (**appendix A**), and QEMU's "Screamer" fork (**appendix B**), both of which are discussed in more detail throughout this report.

Project Methodology, Workflow, and Structure

The workflow used for this project is adapted from the Preservation and Long-term Access through Networked Services' (PLANETS) preservation planning approach, a decision making framework for the preservation of digital objects. PLANETS was a four-year project co-funded by the European Union under the Sixth Framework Programme to address core digital preservation challenges.[6] PLANETS outlines a workflow for assessing preservation alternatives (defined as "different preservation strategies or different tools for the same strategy") based on the characteristics of the digital objects being preserved, as well as the stability, scalability,

[6] Stephan Strodl, et al. "How to Choose a Digital Preservation Strategy: Evaluating a Preservation Planning Procedure," (2007), https://www.ifs.tuwien.ac.at/~becker/pubs/strodl_choose_JCDL07.pdf.

legality, usability, configurability, and costs of each preservation alternative. The PLANETS approach has since been used in a number of digital preservation projects, including by Mark Guttenbrunner, who used the model for a set of experiments evaluating preservation alternatives for video games from several different console eras.

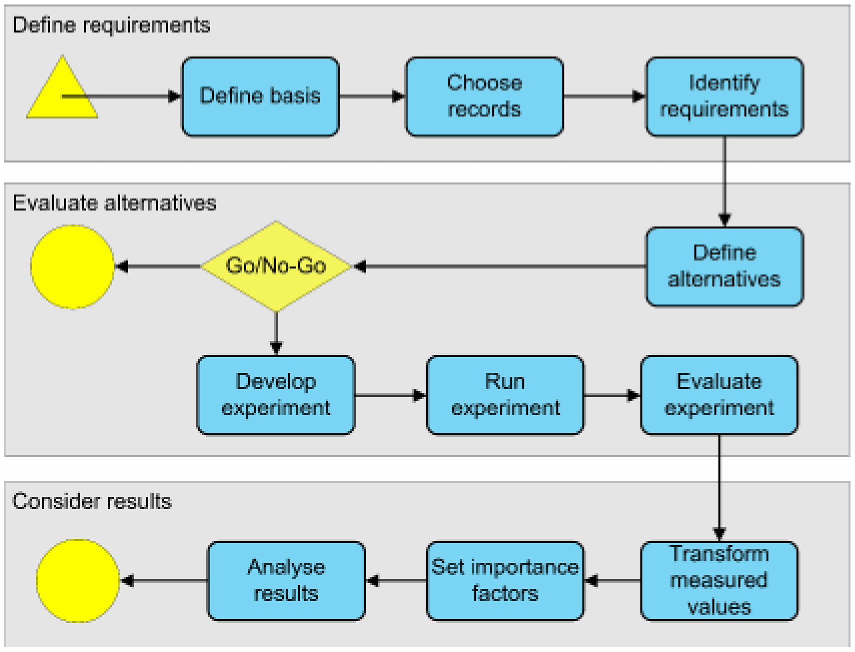


Figure 2. The PLANETS Preservation Planning workflow. Digital Preservation - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Planets-preservation-planning-approach_fig2_240321094, accessed June 20, 2023.

The PLANETS workflow has three phases consisting of a total of eleven steps. The first phase begins by “defining the preservation scenario, choosing sample records… and identifying the requirements and goals for the preservation scenario.”[7] Chapter 2 of this thesis covers this first phase, which involves identifying the required technical specifications and the significant properties (also referred to as “Object Characteristics”) that are essential for the preservation of

[7]Stephan Strodl, et al. “How to Choose,” 5, https://www.ifs.tuwien.ac.at/~becker/pubs/strodl_choose_JCDL07.pdf.

Myst, the sample record.[8] The concept of significant properties as they relate to video games is addressed in Section 2.3 of this paper.

The second phase of PLANETS consists of defining the potential preservation solutions (or preservation alternatives) based on the previously identified requirements. For this project, the preservation alternatives are four different VM softwares; SheepShaver, QEMU, Apple's "Classic Environment," and EaasI. Chapter 3 of this thesis details the basic information and background of each of the selected emulators.

Phase two of PLANETs concludes with the execution of the experiments themselves, which are covered in this thesis in Chapters 4 and 5. Chapter 4 describes the process and results of the control experiments—documenting Myst running natively on two different legacy Macintosh computers, while Chapter 5 details the results of emulation experiments in comparison to the controls. The final section of Chapter 5 (5.3) discusses the process of configuring and testing self-contained, portable VM packages in SheepShaver and QEMU.

In the final phase of PLANETS, the results of the experiments are aggregated, weighted, and each preservation alternative is ranked according to the decided metrics. [9] In lieu of this type of formal evaluation, the results and conclusions from the experiments are discussed in Chapter 6 of this paper.

[8]PLANETS uses the term "Object Characteristics" for the requirements of the digital object, and "Infrastructure" and "Process Characteristics" for the qualities of preservation strategies.
[9] The PLANETS research group later released the Planets Suite (https://planets-project.eu/software/), a set of software tools to help practitioners apply the approach.

Chapter 1. Project Context and Related Work

1.1 Video Games at the Museum of Modern Art

In a 2012 blog post celebrating MoMA's decision to collect video games, A&D's Senior Curator Paola Antonelli described four key traits of interaction design; the types of behaviors encouraged and elicited from an object's users; the quality and ingenuity of its aesthetic and visuals; the creation of a virtual, navigatable space; and the way in which time is "expressed and incorporated" into the design.[10] With this criteria in mind, MoMA's curators consulted with scholars, digital conservation and legal experts, historians, and critics to create an initial wish list of 40 video games considered both culturally significant and exceptional illustrations of interaction design.

In 2012, the Museum announced its acquisition of 14 video games, all of which were featured in the next iteration of the annual *Applied Design* exhibition; *Pac-Man* (1980), *Tetris* (1984), *Another World* (1991), *Myst* (1993), *SimCity 2000* (1994), *Vib-ribbon* (1999), *The Sims* (2000), *Katamari Damacy (2004), EVE Online (2003), Dwarf Fortress (2006), flOw (2006), Portal (2007), Passage (2008), and Canabalt (2009).* Over the next two years, MoMA acquired seven more video games and two gaming consoles, which were displayed in *A Collection of Ideas*, an exhibition that ran from February 15, 2014 to January 11, 2015.[11]

In 2021, A&D nearly doubled its game collection when they acquired an additional 14 titles. The new wave of games included genre and industry defining classics like *Chrono Trigger*

[10] Antonelli, "Video Games," https://www.moma.org/magazine/articles/798.
[11] These games were *Pong* (1972), *Asteroids* (1979), *Tempest* (1981), *Yar's Revenge* (1982), *Minecraft* (2011), *Space Invaders* (1978), and *Street Fighter II* (1991). The consoles were two creations of designer Ralph Baer, the Magnavox Odyssey (1972) and its prototype, The Brown Box (1966). See Galloway, "Video Games: Seven More Building Blocks in MoMA's Collection." and "A Collection of Ideas," The Museum of Modern Art, accessed December 17, 2022, https://www.moma.org/calendar/exhibitions/1421.

(1995), influential experimental and independently developed titles from the last five years, such as *Return of the Obra Dinn* (2018) and *Everything is Going to be OK* (2017), and games representing the perspectives and voices of Indigenous creators (*Kisima Iŋŋitchuŋa/ Never Alone* (2014)). This expansion brought MoMA's video game collection to a total of 36 works, ranging from the 1970s to the present. Many of these new acquisitions, along with nearly all the previously acquired video games, were included in the 2022-2023 exhibition *Never Alone: Video Games and Other Interactive Design*—the Museum's most expansive and comprehensive exploration of interactivity to date.[12]

As a form of computer and software-based art, video games are complex interactive systems with interconnected hardware and software dependencies. They can exist simultaneously as concrete physical objects as well as unstable digital objects in need of constant maintenance and care. So although MoMA has only acquired 36 video games, together they are represented by over 70 different hardware and software components. The collection includes 19 titles with software files of various formats (including, .exe, dmg, .swf, .fla, .app, .bin, and .zip)[13], 14 computers, 5 dedicated hardware platforms, 10 disk images of storage media (both of optical and floppy disks), 5 disk images of computer systems used in exhibitions, 6 optical discs, 3 floppy discs, 5 exhibition video files, various ancillary artist materials and physical ephemera, and several other miscellaneous electronics and equipment. All digital assets are stored in the Digital Repository for Museum Collections (DRMC), MoMA's digital repository system, while most physical components are stored in MoMA QNS, the Museum's storage facility in Long Island City, Queens.

[12] The Museum of Modern Art, "Never Alone: Video Games and Other Interactive Design," accessed December 17, 2022, https://www.moma.org/calendar/exhibitions/5453.
[13] Four of which include full or partial source code.

1.2 Related Work

Descendants of IBM's "Universal Virtual Computer"

Virtual machine-based preservation strategies can be traced back to the Universal Virtual Computer (UVC) proposed by IBM Research in 2000. IBM imagined the UVC as a general purpose computer, "complete yet basic enough as to remain relevant for a very long time."[14] The theoretical method would involve archiving a program which could decode data of any file of a specific format and return the information to a future client. The program would be written specifically for the UVC, so that in the future the only thing required to run the program would be an emulator for the UVC.

In their 2007 thesis, Mark Guttenbrunner describes a theoretical preservation strategy that combines console video game emulators with VMs that is reminiscent of IBM's UVC concept. In Guttenbrunner's model, instead of running a console video game emulator directly on the hardware of the host system (Host System 1), a VM (VM 1) is developed containing an emulator for the target system. When Host System 1 itself becomes obsolete, VM 1 is "ported" as VM 2 onto Host System 2. Gruttenbruner argues that, due to the complexity of developing emulators for console video game systems, this is the best approach for accessing preserved console video games.[15]

Around the same time as Guttenbrunner's thesis, taking inspiration from IBM's UVC, the Koninklijke Bibliotheek, National Library of the Netherlands developed "Dioscuri," an x86 computer hardware emulator written in Java.[16] The emulator was written in Java so that, in

[14]IBM Research, "UVC: A Universal Virtual Computer for Long-term Preservation of Digital Information," Technical Paper Archive, accessed May 8, 2023, https://dominoweb.draco.res.ibm.com/10229b6de0d054c585256fa900681995.html
[15]Mark Guttenbrunner, "Digital Preservation of Console Video Games," http://www.ifs.tuwien.ac.at/~becker/pubs/guttenbrunner_games2007.
[16] Dioscuri team, "Dioscuri: Ideas and Key Features," Koninklijke Bibliotheek, 2007, accessed on November 17th, 2022, http://dioscuri.sourceforge.net/dioscuri.html.

theory, it could be ported to any computer that supports the Java Virtual Machine (JVM).[17] Dioscuri worked using a "modular emulation" approach: each hardware component was emulated by a software surrogate called a module. New modules, written in Java, could be added to the software library. The Dioscuri project aimed to create an emulation system that was stabilized and standardized under Java and the JVM. The Dioscuri project was last updated in 2011 with the release of version 0.7.0.

In 2009, the Keeping Emulation Environments Portable (KEEP) project was launched, which incorporated and expanded on Dioscuri's work. Similar to the model outlined by Gurrenbrunner, KEEP was "a series of virtual machines stacked in order of complexity…"[18] It was developed as a hybrid approach between emulation and migration, where a VM is developed as a platform on which "emulators will be written (or ported) to run."[19]

KEEP eventually developed a desktop application that bundled a set of emulators together with a configuration Graphic User Interface (GUI). KEEP included a method for associating file formats with configured emulated environments. This allowed a recognized file to be automatically linked to its required computing environment. The appropriate environment

[17]According to Software Architect Matthew Tyson, the JVM is "the specification for a software program that executes code and provides the runtime environment for that code. The JVM is not a virtual machine in the traditional sense. Instead, it is the specification for the processes running on a computer or software that controls 'resource usage' for a Java application." Essentially, the JVM is what allows Java programs to run. See Matthew Tyson, "What is the JVM? Introducing the Java virtual machine," InfoWorld, October 28, 2022, https://www.infoworld.com/article/3272244/what-is-the-jvm-introducing-the-java-virtual-machine.html.

[18]Dan Pinchbeck, et al., "Emulation as a strategy for the preservation of games: the KEEP project," (paper presented at the DiGRA 2009 Breaking New Ground: Innovation in Games, Play, Practice and Theory, Brunuel University, West London, September 2009), 4, http://www.digra.org/digital-library/publications/emulation-as-a-strategy-for-the-preservation-of-games-the-keep-project/.

[19]David Anderson, Janet Delve, and Dan Pinchbeck, "Toward a workable emulation-based preservation strategy: rationale and technical metadata." New Review of Information Networking, Vol. 15 No. 2 (2010): 114. https://doi.org/10.1080/13614576.2010.530132.

could then be loaded onto the user's desktop. According to Cochrane et. al., (2019) the KEEP project's approach was "less efficient" than the UVC or Dioscuri because it "incorporated multiple emulators that would have to be supported over time," and therefore did not address the problem of emulator obsolescence. However, KEEP "did introduce the dramatic efficiency of being able to use existing file-interpreters (E.g. commercial software applications) and the ability to reuse off-the-shelf emulators developed by third parties."[20] These concepts and approaches were carried forward into future work, such as the Global Remote Access to Emulation Services (GRATE) service, a sub-project of the Planets Project, as well as Yale Libraries' Emulation as a Service Infrastructure (EaaSI).[21]

Towards a Web Browser-based Emulation Solution

The GRATE project developed a method for remotely accessing emulated environments via a web browser.[22] The team behind GRATE believed that providing remote access to emulation software through a web browser had at least two transformational benefits. First, it would lower the barrier for entry for users. Second, since software can be managed and executed remotely, browser-based emulation would offload resource-intensive emulation software. GRATE was led by a team at the University of Freiburg, which would eventually become the Baden-Württemberg Functional Long-Term Archiving (bwFLA) project.

After GRATE, the resulting bwFLA project developed a suite of tools now referred to as Emulation as a Service (EaaS). EaaS is a hybrid approach that combines the most successful aspects of both KEEP and GRATE. According to Cochrane et. al., (2019), the EaaS tools "follow

[20] Euan Cochrane et. al., "TOWARDS A UNIVERSAL VIRTUAL INTERACTOR (UVI) FOR DIGITAL OBJECTS," (paper presented at the 16th International Conference on Digital Preservation iPRES 2019, Amsterdam, The Netherlands, 2019), 3, DOI: 10.1145/nnnnnnn.nnnnn.
[21]Ibid.
[22]Ibid.

the basic approach pioneered with KEEP, but implemented with a browser-based interface, while adding features such as enabling the definition of derivative disk images …, the separation of objects, environments and emulators, and the addition of many reliability improvements."[23]

Since 2019, EaaS' engineers and developers have worked to create a Universal Virtual Interactor (UVI). The goal of the project is to develop a framework into which organizations can add their own legacy software and metadata. Within this framework, digital objects can then be automatically presented and interacted with in emulated environments using "original or representative software from a time period that is appropriate to the object."[24] Essentially, the UVI automates the process for opening legacy files in their "original" software within a web browser.

It is on the basis of the UVI that the Emulation as a Service Infrastructure (EaaSI) program of work is being built. EaaSI is currently in the process of building the "EaaSI Network," a collection of partner organizations each of whom are hosting EaaSI nodes running instances of EaaS. Along with a local team at Yale University Library, EaaSI is "configuring and documenting emulated computing environments and enabling the environments to be shared between nodes in the EaaSI network." They are also "building services, workflow interfaces, and APIs to perform various digital preservation and curation functions." One of those services is the UVI, which relies on this set of configured computing environments in order to function.[25] At present, EaaSI represents the most forward thinking and promising VM-based approach for accessing legacy software. However, it remains to be seen whether a web browser-based solution is viable for complex computer and software-based artwork, including video games.

[23]Ibid.
[24]Ibid.
[25]Ibid., 4.

Chapter 2. Identifying Requirements

2.1 Target Computing Environment: Mac OS 9

The first phase of this project involved identifying what is required to preserve a computer game. Both the basic system requirements and the significant properties of a digital object must be defined in order to make informed preservation decisions. One of the most important considerations at this stage was selecting the appropriate operating system for the emulation experiments. In addition to the User Manual included with the original release of the game, I consulted various online resources, such as Mobygames, the E-Maculation forums, and the *Myst* Reddit community, in order to determine the most appropriate version of Mac OS to emulate for the purpose of playing the original release of *Myst.*

Although the game was initially developed for Mac OS 7.0.1, I chose Mac OS 9 as the target environment for the following reasons. First, several online sources indicated that emulating *Myst* on Mac OS 9 was the most accessible and authentic way to experience the game today.[26] Secondly, targeting Mac OS 9 would allow me to compare two different open-source, Power PC emulators, SheepShaver and QEMU to Apple's own proprietary Mac OS 9 emulator, the "Classic Environment." SheepShaver and QEMU are, at the time of writing, two of the most popular and well supported VM software for PowerPC-based Mac operating systems.[27]

[26]BSS8888, "I can't believe how different the original 1993 Mac release of Myst is from every other version," Reddit, March 15, 2022, https://www.reddit.com/r/myst/comments/tf5573/i_cant_believe_how_different_the_original_1993/
[27]"Macintosh line," Emulation General Wiki, accessed October 17th, 2022, https://emulation.gametechwiki.com/index.php/Macintosh_line#Emulators

Minimum CPU Class Required	Motorola 68020
Minimum OS Class Required	Mac OS 7.0.1 or higher
Minimum RAM Required	4 MB
Disk Space Required	3 MB
Video Modes Supported	256 Colors Required
Sound Devices Supported	Sound Manager
Input Devices Supported	Keyboard, Mouse
Additional software requirements	QuickTime 1.6 Sound Manager 3.0

Table 1. *Myst* System Requirements according to video game database Mobygames.com.[28]

Before moving forward, it is necessary to provide some background information about this particular operating system. Mac OS 9 is the ninth major release of Apple's Mac operating system, and is considered the last generation of the "Classic Mac OS" line. It was introduced on October 23rd, 1999, and discontinued in late 2001.[29] Apple marketed Mac OS 9 as having "50 new features," including the Keychain, a Software Update control panel, a redesigned Sound control panel and support for USB audio devices, USB Printer Sharing, 128-bit file encryption in the finder, support for files larger than 2 GBs, CD Burning within the Finer, and more.[30]

OS 9 received approximately eight updates over the course of its short lifespan. Some of these were insignificant, like version 9.0.4, which was simply a collection of bug fixes primarily relating to USB and FireWire support. However, other updates came with very significant

[28] "Myst Attributes, Tech Specs, Ratings," MobyGames, accessed November 12, 2022, https://www.mobygames.com/game/1223/myst/specs/.
[29] The final update to the Classic Mac OS and Mac OS 9 was version 9.2.2. https://web.archive.org/web/20100418212841/http://www.youtube.com/watch?v=Cl7xQ8i3fc0
[30] "MacHelp What's New in Mac OS 9," Apple Inc., updated June 28, 2004, https://web.archive.org/web/20071013225329/http://docs.info.apple.com/article.html?artnum=50100.

changes, such as OS 9.1. Starting with OS 9.1, Apple introduced the memory management unit (MMU), a piece of hardware that primarily performs virtual memory functions.

Virtual memory is a memory management technique where secondary memory can be used as though it were a part of the computer's main memory. This technique uses both hardware and software to enable a computer to temporarily transfer data from random access memory (RAM) to disk storage. Mapping chunks of memory to disk files allows a computer to compensate for physical memory shortages, thereby improving overall system performance.[31] This makes the MMU an important part of the hardware and software architecture of versions of Mac OS 9 after 9.0.4. Unfortunately, due to its proprietary nature, the MMU also happens to be a particularly challenging piece of hardware to emulate. SheepShaver, for example, can only emulate up to OS 9.0.4 specifically because of the complexities of the MMU.[32]

Version	**Release date**	**Changes**	**Codename**	**Computer**
9.0	October 23, 1999	• Initial release	Sonata	iMac G3
9.0.2	February 2000 (Shipped with Macs)	• Bug fixes	—	PowerBook (FireWire)
9.0.3	March 2000 (Shipped with Macs)			iMac/iMac DV/iMac DV SE
9.0.4	April 4, 2000	• Improved USB and FireWire support • Other bug fixes	Minuet	iMac G3 (slot loading)

[31] Alexander S. Gillis, Sacey Peterson, and Sonia Lelii, "What is Virtual Memory?" TechTarget.com, accessed June 16, 2023. https://www.techtarget.com/searchstorage/definition/virtual-memory
[32]Apple Computer, Inc., "Mac OS 9.2.2: Document and Software," MacHelp, updated March 17, 2005, https://web.archive.org/web/20060421074349/http://docs.info.apple.com/article.html?artnum=75186.

9.1	January 9, 2001	• Integrated Disc Burning within Finder • Implementation of Finder 'Window' menu • Memory management unit required • Improved stability	Fortissimo	iBook 14 inch and 12 inch
9.2	June 18, 2001 (Shipped with Macs)	• G3 processor as minimum system requirement • Improved speed • Improved Classic Environment support	Moonlight	Power Mac G4 (QuickSilver)
9.2.1	August 21, 2001	• Minor bug fixes	Limelight	iBook (Late 2001), PowerBook G4 (Gigabit Ethernet)
9.2.2	December 5, 2001	• Bug fixes relating to Classic Environment	LU1	eMac

Table 2. List of Mac OS 9 releases. “Mac OS 9,” Wikipedia, accessed March 16, 2023 https://en.wikipedia.org/wiki/Mac_OS_9#Version_history.

2.1a PowerPC CPUs

When discussing this era of Mac operating systems, it is also important to understand the basics of PowerPC CPUs. Mac OS 9 works on the PowerPC (PPC) architecture, which is a reduced instruction set computer (RISC) instruction set architecture (ISA), originally created by the 1991 Apple–IBM–Motorola alliance, known as AIM.[33] The PPC architecture is perhaps most notably associated with several models of Macintosh computers released between 1994 and 2005. Different generations of PPC are distinguished by the letter G. Apple popularized the term “G3” when they released the Power Mac G3 and PowerBook G3 in 1997, and Motorola and

[33]“PowerPC,” Wikipedia, accessed March 16, 2023, https://en.wikipedia.org/wiki/PowerPC#Operating_systems_2.

Apple later used the moniker “G4” for the 7400 family of processors and the Power Mac G4 in 1998 and 1999, respectively.[34] Versions of PPC exist in both 32-bit and 64-bit implementations.

The first version of classic Mac OS to work on the PPC architecture was Mac OS 7 (also known as “System 7” and codenamed “Big Bang”).[35] In 2005 Apple migrated the CPUs of their Mac Computers to Intel Processors, specifically Intel’s x86 architecture.[36] Other notable uses of PowerPC CPUs were in some of the 7th generation (approximately from 2005 to 2014) video game consoles, such as Microsoft’s Xbox 360 and the Nintendo Wii.

Emulating PPC-based operating systems presents a number of unique challenges compared to other computer architectures. In comparison to 68k emulators (used for classic Mac OS prior to OS 8.2), PPC emulation is more complex and requires more processing power. Many of these issues can be traced back to the difficulty of emulating the Memory Management Unit (MMU). According to Reddit user DivingKataeGuru, unlike other architectures, crucial components such as the audio chips and hard drive controllers are proprietary (including the video driver in versions prior to Mac OS 9.1). This is problematic because little information is publically available from Apple regarding these legacy pieces of hardware.[37]

2.2 Significant Properties and Authenticity

In order to properly identify the requirements of preserving a computer game like *Myst,* we must also identify its significant properties. Margaret Hedstrom and Christopher A. Lee wrote

[34]Susan Seale, “PowerPC G4 Architecture White Paper,” Freescale Semiconductor, Inc., 2001, https://www.nxp.com/docs/en/white-paper/G4WP.pdf
[35]Jim Gaynor, “System 7.0 - Will it be on apple.com?” Mac GUI, February 8, 1991, https://web.archive.org/web/20160401212219/http://macgui.com/usenet/?group=53&id=83125
[36]Apple Computer, Inc., “MacHelp What's New in Mac OS 9,” MacHelp, updated June 28, 2004, https://web.archive.org/web/20180130185804/https://www.apple.com/newsroom/2005/06/06Apple-to-Use-Intel-Microprocessors-Beginning-in-2006/
[37] DivingKataeGuru, Why emulation of PowerPC Macs was underwhelming,” *Reddit,* September 4, 2018, https://web.archive.org/web/20211114062934/https://www.reddit.com/r/emulation/comments/9d2sl9/why_emulation_of_powerpc_macs_was_underwhelming/

in 2002 that the significant properties of a digital object refer to those properties which impact the object's "quality, usability, rendering, and behavior."[38] The concept of significant properties is an established principle in both the fields of digital preservation and time-based media conservation.[39] As such, before discussing how this concept applies to computer games in the context of museum installations, it is worth briefly reviewing how the concept of significant properties applies to the conservation and preservation of both digital objects and time-based media installations.

"Work-defining" Properties of Time-Based Media Installations

Collecting, preserving, and exhibiting time-based media poses many complex technical and ethical challenges for conservators, particularly around the issue of authenticity. In 2006 Professor Pip Laurenson wrote an influential paper addressing the need for different conceptual frameworks from traditional conservation ethics in order to account for the unique properties of time-based media installations. In contrast to traditional artworks, where authenticity can be determined through scientific analysis, for time-based media, authenticity is related to the "identity" of the artwork.[40] Working from philosopher Stephan Davies' idea that the identity of musical works can be "thinly or "thickly" specified, Laurenson applies this concept to

[38] Margaret Hedstrom and Christopher A. Lee, "Significant Properties of Digital Objects: Definitions, Applications, Implications," (paper presented at Proceedings of the DLM-Forum, Barcelona, 2002): 218.
[39] When put into the context of art museums, video games should be treated by curators and conservators as both digital objects and types of time-based media installations. Contemporary artworks that incorporate video film, slide, audio, or computer technologies are referred to as time-based media works because they have duration as a dimension and they "unfold to the viewer over time." "Time-Based Media," THE SOLOMON R. GUGGENHEIM FOUNDATION, accessed June 24, 2023, https://www.guggenheim.org/conservation/time-based-media.
[40] Pip Laurenson, "Authenticity, Change and Loss in the Conservation of Time-Based Media Installations" in *Tate Papers no.6*, (Fall 2006): 4, https://www.tate.org.uk/research/tate-papers/06/authenticity-change-and-loss-conservation-of-time-based-media-installations.

time-based media: “In time-based media installations, 'thickly' specified works are works where the artist has specified the qualities of the work and its presentation as precisely as possible.”[41] “Thinly” defined works are then those where there are relatively few “determinative properties.” Work-defining (or significant) properties are a useful tool for determining the identity, and by extension the installation specifications of time-based media installations. According to Laurenson, the work-defining properties of a time-based media installation may include: “plans and specifications demarcating the parameters of possible change, display equipment, acoustic and aural properties, light levels, the way the public encounters the work and the means by which the time-base media element is played back.”[42]

Significant Properties of Video Games

Turning to the fields of digital and video game preservation, Jerome McDonough, the Principal Investigator for the Preserving Virtual Worlds project, argued that the survival of video games will require librarians, archivists, and curators to adopt new metadata systems.[43] McDonough argues that such systems need to efficiently document information about both the “technical interpretability” and the “social interpretability” of these complex digital objects. For McDonough, the technical interpretability of a video game refers to a user’s ability to operate a game itself, while its social interpretability refers to the ability of future scholars to ascertain the “significance and meaning” of a game within its original context and use.[44]

However, determining what metadata is significant for the technical and social properties of video games is complicated due to their poorly defined boundaries as both digital and cultural

[41]Ibid., 5.
[42]Ibid., 6.
[43] Jerome McDonough, “A Tangled Web: Metadata and Problems in Game Preservation.” The Preservation of Complex Objects 3, (2013): 62.
[44]Ibid., 51.

objects. From the technical perspective, the operation of computer games requires interconnected hardware and software systems—including dedicated consoles, operating systems, display devices, and dynamic software libraries. To access some historical games, institutions may need access to (or the ability to emulate) legacy operating systems in addition to detailed representation information and documentation about obsolete file formats, software, and coding languages.[45]

Documenting relevant data about a game's social significance can be equally convoluted: screenshots and user videos of time-sensitive in-game events, fan websites, online forums, and scholarly writings about games could all be crucial sources of contextual information for future scholars attempting to understand gaming culture. Even a game's original packaging and marketing artifacts could provide vital information about the history and evolution of the medium of games from a socio-historical perspective.[46]

The issue of defining video games' boundaries becomes even more complex when the focus is expanded beyond academia to other stakeholders in the video game community. In her analysis of the data collected from the Preserving Virtual Worlds II project, Rhiannon Bettivia found that significant properties of games varied greatly across user communities. Interviews with game developers, educators, and enthusiasts suggested that significance often laid outside the archival objects themselves—including the original historical context in which the game was designed and played, peripheral devices and ephemera included with certain game releases, the "social performative aspects of play," and natively or externally captured playthroughs.[47]

[45]Ibid.
[46]Ibid., 52.
[47] Rhiannon Bettivia, "Where Does Significance Lie: Locating the Significant Properties of Video Games in Preserving Virtual Worlds II Data," *International Journal of Digital Curation* 11, no. 1 (2016): 25, DOI: 10.2218/ijdc.v11i1.339.

Significant Properties as Emulation Metrics

In their 2007 thesis, Mark Guttenbrunner defined a set of “Object Characteristics” of console video games. Most significantly, Guttenbrunner outlined a framework for evaluating the accuracy of emulation configurations based on five categories of console video games: speed, interactivity, sound, graphics, and network support. These five object characteristics are akin to significant properties. Guttenbruner asserts that these five characteristics must be accurately represented in order to preserve the “look and feel” of a video game. In his own thesis on the benefits of hardware emulation, Mike Stetz, working from Guttenbrunner’s framework, further distilled the significant properties of console video games into three categories: Control (or “game feel”), graphics, and sound.[48]

Although Guttenbrunner and Stetz focused on console video games, these metrics could be adapted for computer-based video games like *Myst* by accounting for the game’s distinct qualities (such as mouse and keyboard support) and additional proprietary software and hardware dependencies. For this experiment, I will slightly modify Guttenbruner and Stetz’ categories and sub-categories, selecting only those that are most-relevant to a single player computer game like *Myst*: Speed, Sound, Graphics, and Interactivity/Input.[49]

2.2a Speed:

This factor looks at how accurately a preservation solution recreates the speed of the original game on legacy hardware.[50] For this experiment, speed refers to how much faster or

[48]Michael Stetz, “Leveling Up Emulation: The Benefits of Field-Programmable Gate Arrays in Video Game Preservation,” (New York University, 2022), 19-26. For example, while the graphics, sound, and controls of *Myst* were considered immersive in 1993, today the graphics and controls would likely be perceived by a contemporary player as being archaic compared to the cinematic capabilities of video games today.
[49]Network Support was dropped due to the fact that *Myst* did not operate with any internet capabilities.
[50]Guttenbrunner, 56.

slower the virtualization software plays the computer game. To measure this factor, I will note the speed of the game's loading times, the speed at which movies and animations play, and the time it takes for both the physical and virtual machines to boot their operating systems.

One of the most surprisingly important features of Myst that is impacted by the game's speed are the transition animations that play when the player travels from one screen to the next. A subtle gradient dissolve appears as the player clicks to advance forward, slightly delaying the speed of the action. When turning to the right or left, a horizontal slide transition animation is played, quickly panning between the current screen and the adjacent screen. This has the intended effect of simulating turning in place in order to help orient the player in the virtual space. While seemingly archaic today, these transition animations serve an important purpose; they help orient the player so that they can more easily keep track of their location in the game's (at times) complex and confusing maps.[51] These transitions contribute to *Myst's* immersive quality, one of the game's key aspects MoMA's curators singled out when they acquired the game.

2.2b Sound:

There are two aspects of sound to consider when evaluating the performance of video games: sound effects and music. It is important to measure whether the quality of the sound is lower or higher than the original, as well as whether or not the sound is played in sync with the visuals. Sound in Mac OS 9 environments can be particularly problematic for emulation softwares. Common issues with different versions of SheepShaver and QEMU include sound

[51]BSS8888, "I can't believe how different…" https://www.reddit.com/r/myst/comments/tf5573/i_cant_believe_how_different_the_original_1993/

"crackles," inconsistent volume, or lack of sound altogether.[52] Depending on the specific version of Mac OS 9, special steps must be followed with both SheepShaver and QEMU in order to enable sound after installing and booting the operating systems onto the VMs.

One variable that is challenging to control for is the sound output hardware used across the control and emulation experiments. Both the legacy Macintosh computers used for the control experiments, as well as several of the emulator hosts, have their own built-in speakers and are pre-installed with sound software and drivers. The quality of the sound will inevitably be impacted by the built-in hardware. These discrepancies will need to be accounted for when analyzing the sound quality for the controls compared to the emulators.

2.2c Graphics:

According to Guttenbrunner, graphics refer to the game's "reproduced visual images." [53] The primary graphic characteristics are the image quality and frame rate. This category can be divided into several other characteristics, depending on whether the game relies primarily on two dimensional or three dimensional graphics, or both. For example, for 2D graphics, aspects such as the appropriate layering of objects on top of one another, background scrolling, collision detection, all contribute to the authentic look of the game. Different factors are relevant for 3D graphics, such as texture quality, object clipping, and so on.

In the case of Myst, the game's graphics were created using a combination of 2D and 3D. The game's maps were modeled entirely in 3D using the StrataVision 3D and Macromedia software. The artists then maneuvered a virtual camera around the environments in order to capture images that would serve as the player's perspective. So although 3D graphics were used,

[52]"Running Qemu-system-ppc with Mac OS/OSX guests in macOS," Emaculation.com, updated January 10, 2021, https://www.emaculation.com/doku.php/ppc-osx-on-qemu-for-osx#introduction
[53]Guttenbrunner, 56.

every 3D environment, object, or animation seen in the game was captured either as a still image or a quicktime video. These videos and images were then compressed using quicktime in order to fit them on the CD-ROM. Overall *Myst* contains approximately 2,500 rendered images, one for each possible screen the player can see, and 66 minutes of quicktime animation.[54] If *Myst*'s graphic files are not properly copied to the hard disk during installation, all of the videos will display as white squares.

Another important graphical element of *Myst* are the 2D sprites that replace the user's mouse cursor. The traditional arrow cursor is replaced in *Myst* by an animated human hand which changes appearance depending on where the cursor is placed on the screen.

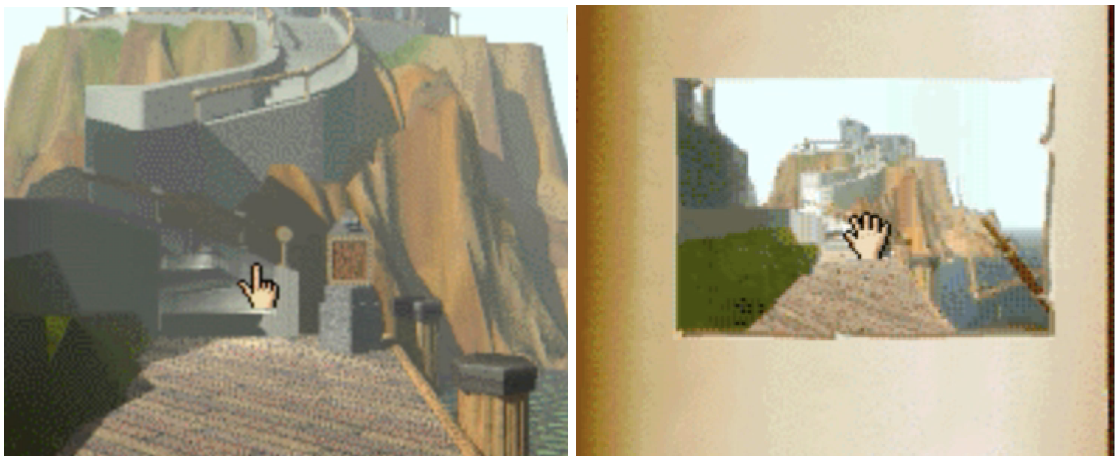


Figure 3. Close-ups of the Hand cursor from *Myst* (1993). Screen capture by author.

While image quality is a relatively subjective aspect to measure, as Michael Stetz notes, "it is easy to note if there are differences between the original image and emulated one."[55] Stetz expanded on this particular graphical characteristic in his own experiments. While looking at image quality, Stetz considered both the color and the aspect ratio of the original Sega Genesis console compared to the performance of the MiSTer hardware emulator. Aspect ratio "considers

[54]NeoGamer - The Video Game Archive, "Making of - Myst (1993)," Youtube Video, March 18, 2020 https://www.youtube.com/watch?v=c5af74KuJZE.
[55]Stetz, 22.

if the image is squished, stretched, or adheres to the original output,”[56] while evaluating color means looking at whether the hues are correct (blues appearing as greens, etc.), and whether or not the levels of saturation and brightness are consistent with the control.

Frame rate is the frequency (rate) at which consecutive images (frames) are captured or displayed. Frame rate is expressed in frames per second, or FPS. In the context of computer graphics, FPS is the rate at which a system, such as a graphic processing unit (GPU), is able to generate frames, while the refresh rate is the “frequency at which a display shows completed frames.”[57] This is why the refresh rate is typically associated with the hardware and software of a computer’s display, while FPS is associated with the machine’s CPU and GPU. If the FPS is higher than the original game, animations and movies may not display properly or even disappear entirely. An artificially low FPS can give the appearance that animations or movies are “hitching” or “lagging.”[58]

Measuring the frame rate of computer games can be a complicated process, both on legacy and modern machines. While there are many tools capable of measuring the frame rate of captured video, there are few reliable tools for measuring the native frame rate of applications running on legacy hardware, such as a PowerMac G4 or iMac G3. Due to these restrictions, frame rate will not be measured as a category during these experiments. Instead, I will use the more subjective “Speed” category discussed in the previous sections in place of frame rate.

Just like with sound, display equipment/hardware plays an important role in the appearance of computer game graphics. Control A is a PowerMac G4 tower circa 1999 hooked up to a HP Compaq LA2405wg 24-inch Widescreen LCD Monitor, which was first released in

[56]Ibid., 39.
[57]“Frame rate,” Wikipedia, accessed March 17, 2023, https://en.wikipedia.org/wiki/Frame_rate.
[58]Michael Klappenbach, "How to Optimize and Improve Graphics Performance and Frame Rates," Lifewire, accessed March 25th, 2023. https://www.lifewire.com/optimizing-video-game-frame-rates-811784

2009. Control B, an iMac G3 circa 1998, on the other hand, has a built-in CRT display. Control A’s monitor has a 60 Hz refresh rate, while Control B’s screen has a refresh rate of 75 Hz. Due to these discrepancies between display equipment, another limitation of this project is that there will not be perfect parity between the displays used in the control and emulation experiments.

2.2d Interactivity and Input:

This category describes the evaluation of the “feel aspect” of the video game, based on how the player interacts with the game. When emulating console video games, there are several different alternatives for recreating the original game’s feel, including standard computer hardware (keyboard and mouse), and using modern equivalents to classic controllers that are recreated with USB-support. Since this project is exclusively looking at a personal computer game, only keyboard and mouse support will be considered. I will evaluate how well the controls used during emulation resemble the original controls, and whether or not there is any response delay between the user’s input and the game’s response.[59]

Stetz makes an important distinction that while “true game feel” consists of “precise control, perception of space, and polish,” from an archival and museum perspective, it is important that a preservation solution recreates the exact feel of the game, which may or may not be polished or precise depending on the quality of the game and the subjective perception of the player.[60]

As previously noted, *Myst* controls entirely through keyboard and mouse. The mouse is used to interact with objects and navigate the 2D/3D environment, while the keyboard is used to enter various commands. Several of these keyboard commands are used for essential game operations, such as saving, changing the speed of transitions, dropping items, and engaging “Zip

[59]Response delay will be measured based on the subjective feeling of the author.
[60]Stetz, 21-22

Mode," which enables the player to move faster through previously explored areas. To fully evaluate the interactivity of the game in each of the emulated environments, It will be important to test each of these commands.

Function	**Command**
Menu Options	⌘ Space
New Game	⌘ N
Restore Game	⌘ R
Save Game	⌘ S
Quit Myst	⌘ Q
Fastest Transitions	⌘ 1
Fast Transitions	⌘ 2
Better Transitions	⌘ 3
Best Transitions	⌘ 4
Environmental Sound	⌘ E
Zip Mode	⌘ Z
Drop Page	⌘ D
Turn Volume Up	⌘]
Turn Volume Down	⌘ [

Table 3. List of keyboard commands in *Myst,* as described in the game's original User Manual.

Chapter 3. Defining Preservation Alternatives

This chapter provides basic overviews of each of the selected preservation alternatives, or VM Softwares, used during these experiments. While selecting PowerPC emulators, I decided to only look into projects that were still in active development and had active online communities. Due to the previously mentioned complexities of Apple's PowerPCs CPUs, there are relatively few options of different emulators that are still supported by their developers. This led me to choose SheepShaver and QEMU as the two main emulators. Apples' proprietary OS 9 emulator, "The Classic Environment " was added as an additional branch of this experiment after MoMA's Media Conservators allowed me access to one of their iMac G3 units installed with a version of Mac OS X, the host operating system for The Classic Environment. Finally, after contacting Ethan Gates of Yale University Library's Digital Preservation team, he generously allowed me access to their EaaSI portal as an additional branch of the experiment.

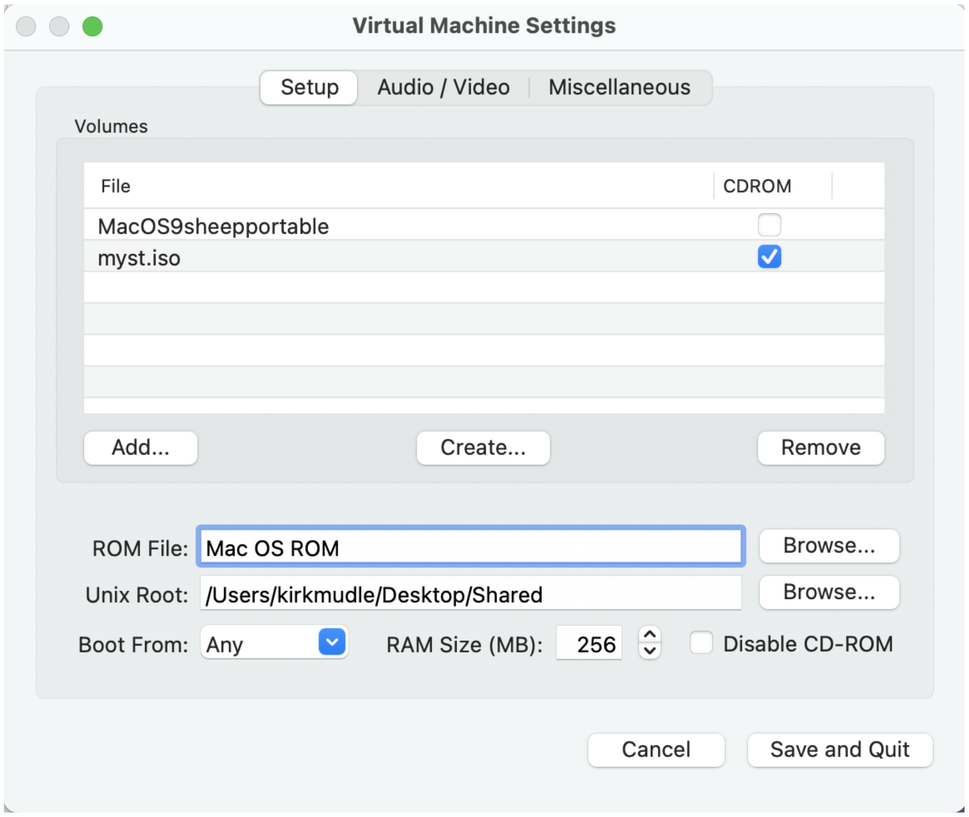
3.1 Emulators

3.1a Emulator A: SheepShaver v.2.5, (September 13, 2022)

SheepShaver is an open source PowerPC Apple Macintosh emulator that was originally developed as commercial software, but became open-source in 2002. According to its developers, the name is a pun on "ShapeShifter," a Macintosh II emulator for AmigaOS.[61] The emulator was originally conceived and developed by Christian Bauer, but today the project is maintained solely by Gwenolé Beauchesne.[62]

SheepShaver is generally more user-friendly than other comparable emulators (such as QEMU, and has several quality of life features, including the ability to exchange of files between the host and guest operating systems through a Unix folder, and a GUI for configuring its settings after completing the initial set up process. As previously noted, due to the changes between each version of Apple's operating systems, SheepShaver can only emulate Mac OS 7.5.2 to Mac OS 9.0.4. Configuring a virtual operating system requires both a copy of a Mac OS install disk and a compatible Power Mac ROM image.

[61]Christian Bauer, "SheepShaver," accessed February 12, 2023, http://sheepshaver.cebix.net
[62]"SeepShaver," Wikipedia, accessed February 12th, 2023, https://en.wikipedia.org/wiki/SheepShaver#cite_note-sheepshaver-1



Volumes

File	CDROM
MacOS9sheepportable	☐
myst.iso	✓

ROM File: Mac OS ROM

Unix Root: /Users/kirkmudle/Desktop/Shared

Boot From: Any   RAM Size (MB): 256   ☐ Disable CD-ROM

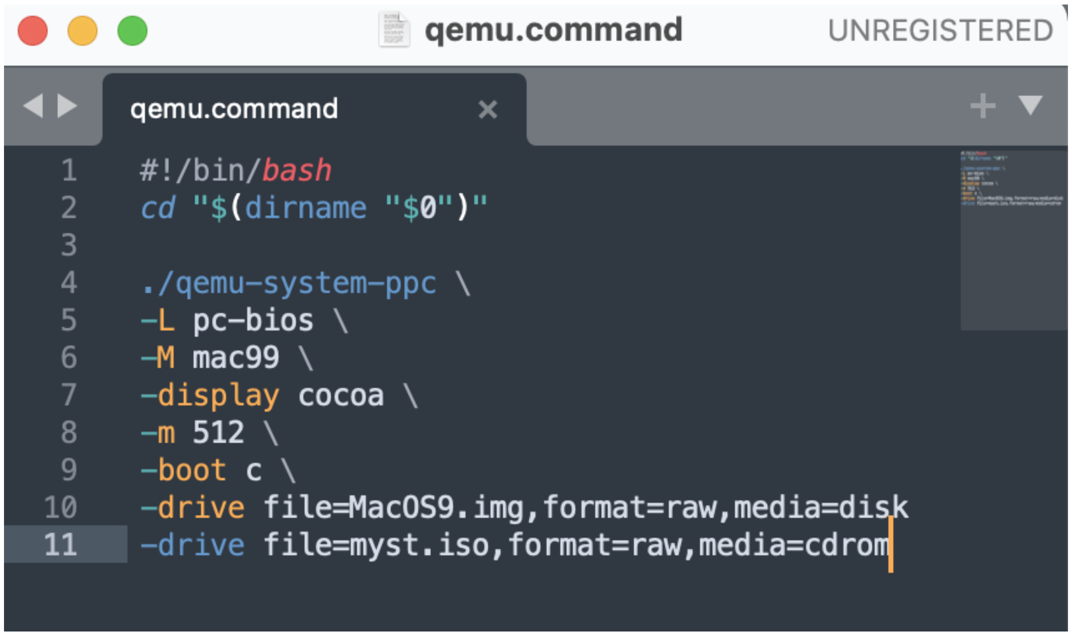
Figure 4. SheepShaver Preferences/Settings GUI. Screen capture by author.

3.1b Emulator B: QEMU v.7.2.0 (December 13, 2022)

QEMU is a versatile open source emulator and virtualizer for Mac OS 7.1 to Mac OS X 10.5 (as well as several other OSs). The project was originally developed by Fabrice Bellard in 2003.[63] In contrast to SheepShaver, QEMU is a large collaborative development project, with the latest release including contributions from 238 authors.[64] Unlike Sheepshaver, QEMU is set up through the command line and managed through a configuration file. Instead of requiring both an OS install disk and a compatible Power Mac ROM, QEMU requires an installation disk image

[63]"QEMU," Emulation General Wiki, accessed February 12, 2023, https://emulation.gametechwiki.com/index.php/QEMU
[64]"QEMU version 8.0.0 released," April 20, 2023, QEMU, https://www.qemu.org/2023/04/20/qemu-8-0-0/.

with the ROM already built in, such as the Power Mac G4 Cube, a model of personal computer between 2000 and 2001.[65]



```
#!/bin/bash
cd "$(dirname "$0")"

./qemu-system-ppc \
-L pc-bios \
-M mac99 \
-display cocoa \
-m 512 \
-boot c \
-drive file=MacOS9.img,format=raw,media=disk
-drive file=myst.iso,format=raw,media=cdrom
```

Figure 5. Contents of a qemu.command file, the configuration file for the QEMU VM. Screen capture by author.

3.1c Emulator C: The "Classic Environment" (Mac OS X 10.3.9)

In order to ease the transition from the Classic Mac OS to Mac OS X, in 2005 Apple included their own proprietary OS 9 emulator in all PowerPC-based Macintosh computers running versions of Mac OS X up to 10.4., nicknamed "Tiger"[66] This emulator was known as "The Classic Environment" or simply "Classic." The Classic Environment was a "hardware and software abstraction layer" that made it possible to run legacy applications compatible with OS 9 on computers running Mac OS X.[67] The Classic Environment can be loaded at login, on

[65]Power Mac G4 Cube," Wikipedia, accessed March 15th, 2023, https://en.wikipedia.org/wiki/Power_Mac_G4_Cube
[66]OS X 10.4 was nicknamed "Tiger."
[67]"List of macOS built-in apps," Wikipedia, accessed May 1, 2023, https://en.wikipedia.org/wiki/List_of_macOS_built-in_apps#Classic

command, or automatically whenever a compatible legacy software application is loaded in the Host Mac OS X environment.

The Classic emulator works by loading the classic OS in a sandbox environment inside Mac OS X. Upon launch, the sandbox environment displays the classic Mac OS boot process, and then recedes into a background operation. When a "classic" application is in the foreground, the menu bar at the top of the screen changes to resemble the older Mac OS system menu.[68] As previously noted, the iMac G3 that was used for the Control B experiment had a partitioned hard drive with both OS 9.2.1 and Mac OS X 10.3.9 installed. This version of Mac OS X includes the Classic Environment emulator, which virtualizes OS 9.2.1 in a sandbox environment.[69]

3.2 Remote Emulation

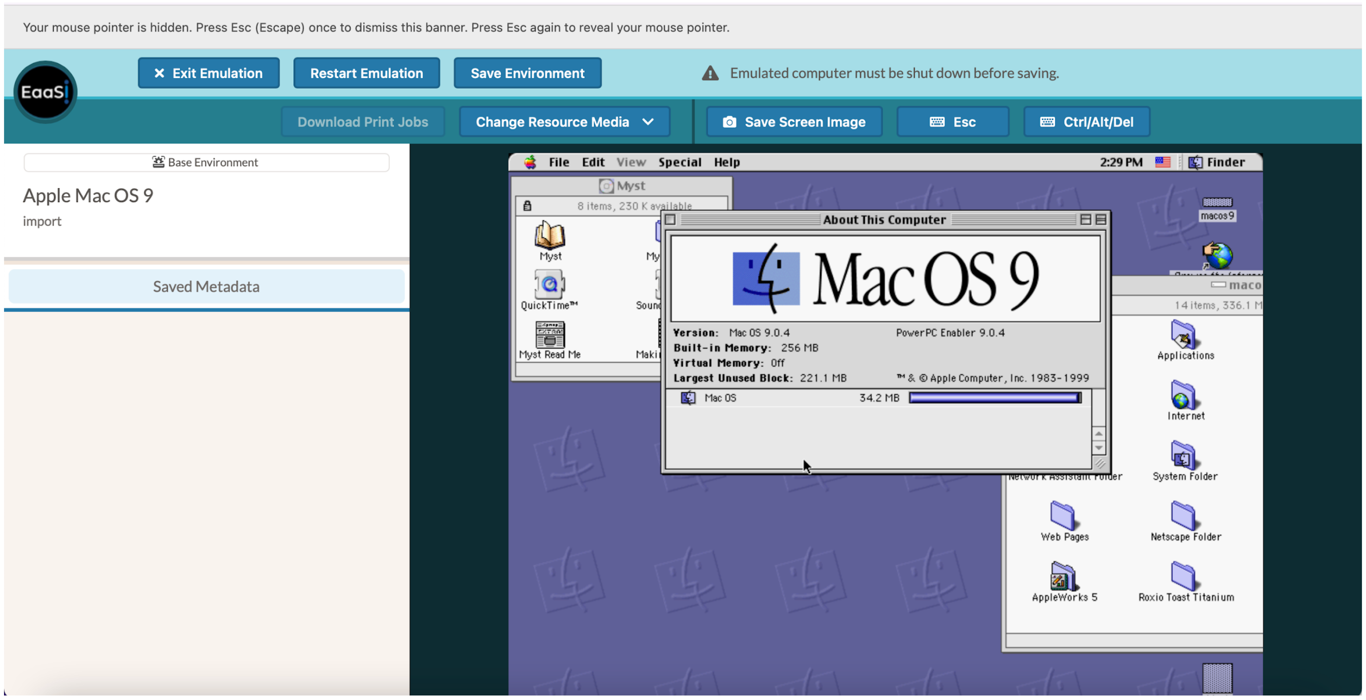
3.2a Remote A: EaaSI (SheepShaver v. 01032019)

EaaSI is an ongoing program of work developed by the Digital Preservation team at Yale University Library. EaaSI allows users to access emulated computing environments through a web browser. At the time of writing, the North American network consists of 16 member organizations, most of which are research universities. In the EaaSI portal, emulated environments are created by staff-users within the member organizations and shared over the network. EaaSI is still experimental and, because it operates over the internet, the performance of software is impacted by the internet connection itself.[70]

Within EaaSI, users can search for preconfigured environments, add their own software, and then save those edited environments as derivatives. For these experiments, I used a Mac OS

[68]Ibid.
[69] For the purposes of these emulation experiments, this instance of the Classic Environment emulator was used along with MoMA's CD-ROM of Myst, instead of a disk image of the game.
[70]Cochrane, et. al., "USABLE SOFTWARE FOREVER," 7-8.

9 environment that was created using an earlier version of SheepShaver, and added MoMA's Myst disk image to it.



Your mouse pointer is hidden. Press Esc (Escape) once to dismiss this banner. Press Esc again to reveal your mouse pointer.

Emulated computer must be shut down before saving.

Version: Mac OS 9.0.4
PowerPC Enabler 9.0.4
Built-in Memory: 256 MB
Virtual Memory: Off
Largest Unused Block: 221.1 MB
™ & © Apple Computer, Inc. 1983-1999

Figure 6. The EaaSI interface. Screen capture by author.

Chapter 4. Control Experiments: *Myst* Native Performance Documentation

This chapter describes the set up and documentation process for the two control experiments. As previously mentioned, in terms of physical components, MoMA only has in its collection a copy of the original CD-ROM of *Myst* released for Macintosh computers. Since no computer or operating system was acquired during the acquisition, for the control experiments I used two different legacy Macintosh computers stored in MoMA's Media Conservation lab. The first is a Power Mac G4 tower (Control A) hooked up to a HP Compaq LA2405wg 24-inch Widescreen LCD Monitor, and the second is an iMac G3 (Control B) with a built-in CRT Screen. MoMA's Media Conservation team previously used these two computers in order to access and view other Mac-based artworks, such as John Maeda's *Reactive Books* series. The Power Mac G4 model has a PowerPC G4 processor running Mac OS 9.2.2., the very last version of Mac OS 9 released, while the iMac has a PowerPC G3 processor with both Mac OS 9.2.1 and Mac OS X 10.3.9 installed on a partitioned hard drive. I used a GoPro camera fixed on a tripod to document the game running on both of these pieces of legacy hardware. For each test, I captured about 30 minutes of footage, starting from the OS' boot process through the completion of the game's first puzzle. The video captures for both controls are linked in appendix C.

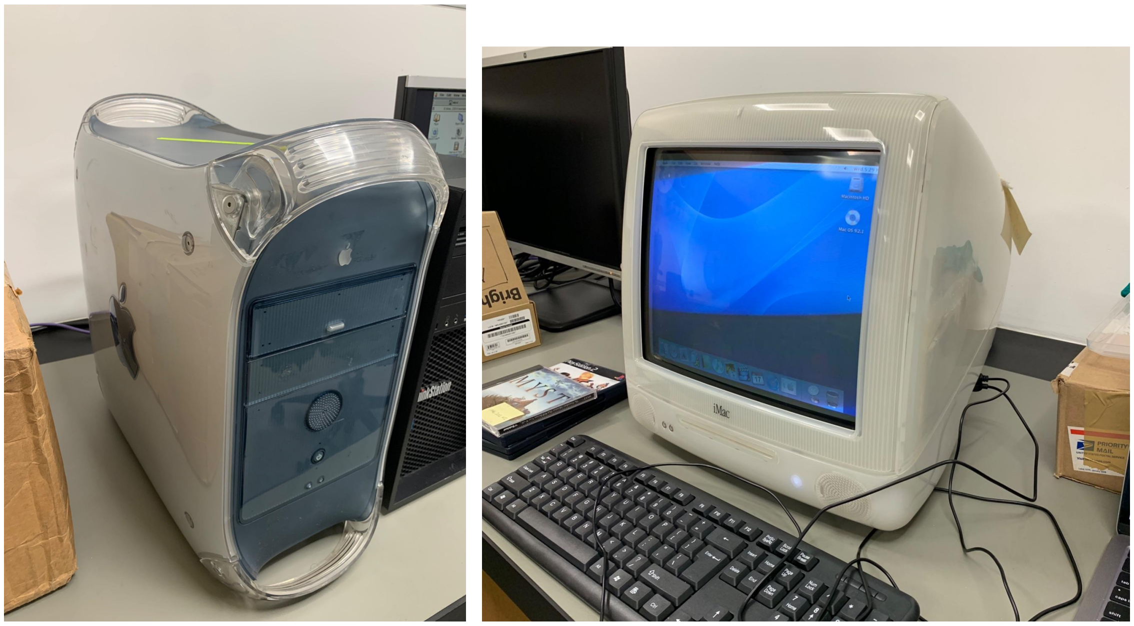


Figure 7. Left: PowerMac G4 (Control A) Right: iMac G3 (Control B). Photos by author.

	Control A	**Control B**
Model:	Power Mac G4	iMac
OS Version:	Mac OS 9.2.2	Mac OS 9.2.1
Mac OS ROM:	9.0.1	?

Processor:	PowerPC G4	PowerPC (750) G3
CPU Speed:	350 MHz	600 MHz
Quicktime Version:	5.0.2	5.0.2
Built-in Memory:	256 MB	384 MB SDRAM
Virtual Memory:	Off	385 MB used on HD
Display	HP Compaq LA2405wg 24-inch Widescreen LCD Monitor Resolution: 1920 x 1200 Refresh Rate: 60 hz	Built-in CRT Screen Resolution: 1024 x 768 Refresh Rate: 75 hz 16-bit color

Table 4. Control Experiments: Power Mac G4 and iMac G3 Specifications

After booting each computer, I inserted MoMA's CD-ROM copy of the original release of *Myst* into the Mac's built-in disk drive, and followed the installation instructions in the User Manual. After Dragging the "Myst" application and "Myst Files' ' folder into the hard disk, I launched the Myst application from the hard disk. Upon starting the game, a warning is displayed stating that the machine's monitor should be set to 256 colors, asking the player whether they want to switch the color settings. After agreeing by clicking "Okay," or declining by clicking "Continue," the game's opening movie plays.[71] In the pre-rendered video, the player character is shown falling through a crevice and into the enchanted book containing the island of Myst. The screen then hangs on the book until receiving an input from either the keyboard or mouse.

After taking control of the player character, I then proceeded to walk around the game's first area. While navigating the virtual island, I tested each of the game's four transition options:

[71]No differences were found between the two options.

Fastest, Fast, Good, and Best.[72] These options control how the game's transitions are displayed. With both Good and Best, the gradient dissolve and turn transition animation display as previously described. When Fast and Fastest are selected, neither the gradient or turning animations are visible. Instead, one screen abruptly jumps to the next when the player clicks, and no turning transition is visible.

During this documentation process, I reviewed all the menu options, tested each of the game's keyboard commands and found that all operated as described in the User Manual. I also interacted with various objects—buttons, switches, control panels, valves, books—to get an understanding of how the game's inputs and interactivity functioned—their speed and their associated sound effects. I then set about solving the game's first puzzle, which is initiated when the player finds a paper on the ground written by the mysterious Artus. This puzzle asks the player to identify the number of "Marker Switches" on the Island and enter that number into the "fore-chamber," without directly explaining what the "marker switches" look like or what exactly the "fore-chamber" is.[73]

After entering the fore-chamber, A melancholy tune plays as the player descends the stairs. This chamber contains a futuristic hologram device and a wall panel where a two digit number can be entered. When the correct number is entered, a recorded message (in the form of a quicktime movie) from Artus plays. After completing these steps, I saved and quit out of the game in order to see how its saving and loading functions worked. Every time the game is quit, the credit sequence is played, which automatically scrolls until the player clicks the mouse button to advance them faster. I used the above section of the game because it captured the

[72] The Default setting is "Good Transitions."
[73] If the player turns to the left from the game's starting screen, the fore-chamber room is directly through a gray door and down a set of stairs.

majority of its basic features and functions, from its speed, sound, graphics, to its interactivity and inputs.

I repeated this series of steps on both the PowerMac G4 (Control A) and the iMac G3 (Control B). Despite the differences between the two machines' CPUs, there were very few differences between Control A and Control B. When comparing both video captures, the speed and graphics appeared to be the exact same on both machines. The interactivity (mouse movement, keyboard commands, etc.) also matched exactly.

The quality of the sound, on the other hand, varied quite drastically between the Control machines. Both Control A and Control B have built in speakers, but the sound that exuded from Control A was much lower quality than Control B. Control A’s speakers had rather dramatic crackling noises, and the quality was so poor that any point (such as the opening movie) when speech was accompanied by music, the narration was very difficult to understand. In order to determine whether this was an error with Control A’s speakers themselves (as opposed to a software issue), I plugged a pair of wired headphones into the computer’s headphone jack. Through the headphones, the sound was much clearer and matched the quality of Control B.

With both controls machines, the sync of certain sound effects was slightly off; there was a subtle delay between on screen actions (such as opening and closing of a door) and the paired sound effect. Since this delay was consistent between both controls, it is suspected that it is a quality of the game itself, rather than an issue with the hardware or software.

Chapter 5. Emulation Results

5.1 Emulating *Myst* with Mac OS 9 Virtual Machines

My original intention was to use the exact same OS and game disk images for each emulation test. Unfortunately, due to QEMU and SheepShaver being developed independently and having entirely distinct configuration processes, the two softwares did not support the same OS image files. Two different versions of Mac OS 9 had to be used, 9.0 in SheepShaver, and 9.0.4 in QEMU. Additionally, Apple's proprietary emulator, "The Classic Environment," emulates the last release of OS 9, 9.2.2. While this should be considered a limitation of these experiments, it is also an opportunity to compare the potential differences between these different versions of OS 9. This chapter details the differences between each of the emulator performances' compared to the control experiments, using the four metrics previously outlined in Chapter 2: speed, graphics, sound, and interactivity.

Name	Control A: Power Mac G4	Control B: iMac G3	Emulator A: SheepShaver	Emulator B: QEMU	Emulator C: The "Classic Environment"[74]	Remote A: (Pre-built EaaSI SheepShaver v. 01032019)
OS Version:	Mac OS 9.2.2	Mac OS 9.2.1	Mac OS 9.0	Mac OS 9.0.4	9.2.2	9.0.4
Mac OS ROM:	9.0.1	5.0.2	New World	5.2.1	4.1.9f1 (Host)	New World
Processor	PowerPC G4	PowerPC (750) G3	PowerPC G4	PowerPC G4	PowerPC (750) G3	PowerPC G4
Quicktime Version	5.0.2		4.0.3	?	6.4	4.1
Built-in Memory:	256 MB	384 MB SDRAM	256 MB	512 MB	384 MB SDRAM	256 MB
Virtual Memory:	Off	Off	Off	513 MB used	Off	Off
Largest Unused Block:	225.2 MB	385 MB	223.9 MB	487.5 MB		221.1 MB

Table 5. Target (Guest) Computing Environments.

5.2 Virtual vs. Native Performance Comparison

5.2a Emulator Speed:

The first aspect of the speed factor that I examined was the length of time it took for each emulated OS to boot. While the control machines both averaged around 60 seconds, SheepShaver (Emulator A) was much quicker to boot, averaging about 10-20 seconds. QEMU (Emulator B) was more variable between each boot, but was slightly closer to the controls, averaging between 30 and 60 seconds. EaaSI (Remote A) was approximately twice as fast,

[74]This instance of the Classic Environment is hosted on the same iMac G4 that was used for Control B. The classic environment is hosted on Mac OS X 10.3.9.

averaging between 25 to 30 seconds. Like SheepShaver, the Classic Environment's boot time was faster than the control, clocking in at a very consistent 18 seconds each launch.

When it came to the overall speed of the game itself, The Classic Environment and (at first) SheepShaver were noticeably faster than the game's performance during the control experiments. The speed was easy to observe thanks to the game's transition animations, which do not display properly when the game speed is artificially fast. At first, SheepShaver displayed these animations artificially fast, rendering them nearly invisible, even when the options were set at "Good" or "Best" transitions. I was able to correct this issue by disabling the emulator's Just-In-Time (JIT) Compiler. While this did not change the artificially fast boot speed of the emulated OS, it did correct the speed of the transitions, making them display identically to the control experiments.

The EaaSI environment, on the other hand, was much more "laggy" and slower, making the game very frustrating to play. Overall, QEMU was the most accurate compared to every other emulator; direct comparisons between the captured videos of the control and QEMU showed that the loading and response times were nearly identical.

Regarding The Classic Environment, the performance was noticeably laggy at several points during both the game's cutscenes and gameplay. Animations would often hitch while the music and sound effects continued to play, giving the appearance that the animations were missing several frames.

	Control A	**Control B**	**SheepShaver (Emulator A)**	**QEMU (Emulator B)**	**Classic Environment (Emulator C)**	**EaaSI (Remote A)**
OS Boot Time	50-60 seconds	45-60 seconds	10-20 seconds	30-60 seconds	18 seconds	25-30 seconds
Transition Animations	Control	Control	Accurate[75]	Accurate	Inaccurate - Faster	Inaccurate - Slower
Overall Speed	Control	Control	Accurate	Accurate	Inaccurate - Faster	Inaccurate - Slower

Table 6. Evaluation of each emulator's speed compared to the two controls.

5.2b Emulator Sound:

Emulator:	**SheepShaver**	**QEMU**	**Classic Environment**	**EaaSI**
Sound effect quality	Accurate	No sound	Accurate	No sound
Sound effect sync	Accurate	No sound	Accurate	No sound
Music quality	Accurate	No sound	Accurate	No sound
Music sync	Accurate	No sound	Accurate	No sound

Table 7. Evaluation of each emulator's sound compared to the two controls.

Only SheepShaver and the Classic Environment had any kind of sound output during emulation. In these two emulators, the sync and the quality of the sound effects and music appeared to be identical to the control experiments. However, neither QEMU or EaaSI output sound at all. It has not yet been determined why the EaaSI environment did not support sound,

[75]After the JIT Compiler was disabled.

but it could be related to the fact that the pre-built emulation environment uploaded to EaaSI was created using an earlier version of SheepShaver than Emulator A.

For QEMU, in order to output sound when emulating Mac OS 9 environments, a special fork of the emulator called “Screamer” is required. The Screamer fork was developed by GitHub user mcayland.[76] Installing and running Screamer involves a process that is separate but similar to the standard QEMU set up. For a detailed description and guide on how to configure QEMU’s Screamer Fork, see **appendix B**.

5.2c Emulator Graphics:

Emulators A (SheepShaver) and B (QEMU), and Remote A (EaaSI) were mostly accurate at recreating the graphics of the original game. For image quality, the aspect ratio of all three emulators was accurate, with no stretching or shrinking. The hues were all rendered correctly; reds appeared as reds, blues appeared as blues, and so on. The saturation and brightness of the emulators also matched those of Control A (PowerMac G4). However, the image quality of the SheepShaver and QEMU were different from the appearance of Control B (iMac G3). As previously mentioned, the saturation and brightness differed between Control A and B. These discrepancies were expected, due to the fact that Control A was hooked up to a LCD monitor, while Control B has a built-in CRT Screen. Likewise, the image quality of SheepShaver and QEMU and EaaSI were also crisper and sharper than Control B. This too is not surprising, as these three emulators were hosted on a 2020 MacBook Pro with a contemporary Retina display.

There were no other graphical errors detected with either Emulator A (SheepShaver), Emulator B (QEMU), or Remote A (EaaSI). However, The Classic Environment (Emulator C) had one very noticeable error. The hand sprite used as the mouse cursor constantly flickers

[76] https://github.com/mcayland

during the entirety of the experiment. Since this sprite flickering was not apparent during the control experiment on the very same iMac G3, we can be sure that the error is a quality of the Classic emulation software, rather than the machine's hardware.

5.2d Emulator Interactivity/Input:

When it came to interactivity and input, all four emulators supported keyboard and mouse. SheepShaver and EaaSI automatically support these devices, but QEMU requires a specific argument in its command file in order to enable keyboard and mouse support.

Based on the subjective evaluation of the author, there was no detectable input lag in SheepShaver, QEMU, or The Classic Environment. The mouse movement in EaaSI, on the other hand, was quite laggy and even "slippery." This issue was so severe that it was even difficult to drag the installation files for *Myst* into the hard disk without accidentally dropping them onto the virtual desktop.

Regarding the game's keyboard commands the majority were supported by SheepShaver and QEMU, with the exception of ⌘ + Space command which is intended to bring up the game's menu bar. Instead, pressing ⌘ + Space brings up the host machine's "Spotlight Search" bar, overriding the emulation software. With EaaSI several of the keyboard commands did not work properly, as they were overrode by the web-browser. For example, when pressing ⌘ + Q, instead of quitting out of the game, it would close the entire web-browser. EaaSI's developers are already aware of these issues and are planning updates to fix them in the future.

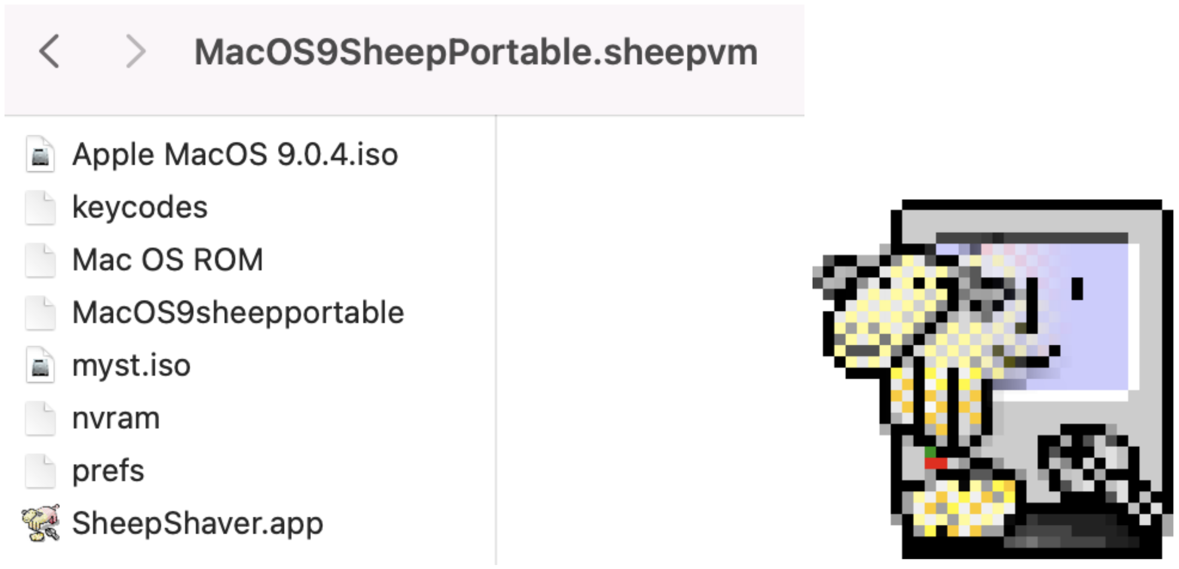
5.3 Creating "Portable" Virtual Machine Packages

This section discusses the process of creating "portable" VMs that can be transferred as self-contained VM packages without requiring installation or set up on the new host machine. In SheepShaver, this feature is actively supported by the development team through the ".sheepvm" file extension. The process is rather simple; first the user must create a folder with the desired name for the VM. Inside the folder, the user must place:

- a compatible Mac ROM file named "Mac OS ROM" (no extension)
- a keycodes file (if you use other than US-English QWERTY keyboard layouts)
- an empty file named "prefs" (no extension)[77]
- A copy of the SheepShaver.app
- Add a .sheepvm extension to the folder name

So long as the Mac ROM is compatible with SheepShaver, as soon as the .sheepvm extension is added to the folder, the folder will change into a package with a SheepShaver icon. From there, the VM can be configured using the same setup process as a typical SheepShaver VM, as described in Appendix B. Including a copy of the SheepShaver.app allows the VM to work on another host machine without needing to install SheepShaver on the new host.

[77]It should be noted that you cannot use any plain text editor, such as Text Edit, Mac's built-in text editor software. Text Edit automatically saves plain text files with an rtf header. This adds additional lines of code to the text file that makes it incompatible with the emulation software.


MacOS9SheepPortable.sheepvm

- Apple MacOS 9.0.4.iso
- keycodes
- Mac OS ROM
- MacOS9sheepportable
- myst.iso
- nvram
- prefs
- SheepShaver.app

Figure 8. Contents of the .sheepvm package. Screen capture by author.

I tested this feature by configuring one of these .sheepvms on the same 2020 MacBook Pro host that was used to configure the other emulators used in this project. After configuring the VM and confirming that it ran identically to the original SheepShaver environment (Emulator A), I zipped the .sheepvm package using the host's built-in compression software. I then uploaded the resulting .Zip file to the Internet Archive. From the Internet Archive page, I downloaded the VM package onto two other Apple computers, the first was a 2019 Mac Pro running macOS 13.1 in MoMA's Media Conservation lab, and the second was a MacBook from early 2015 running macOS 11.7.4.

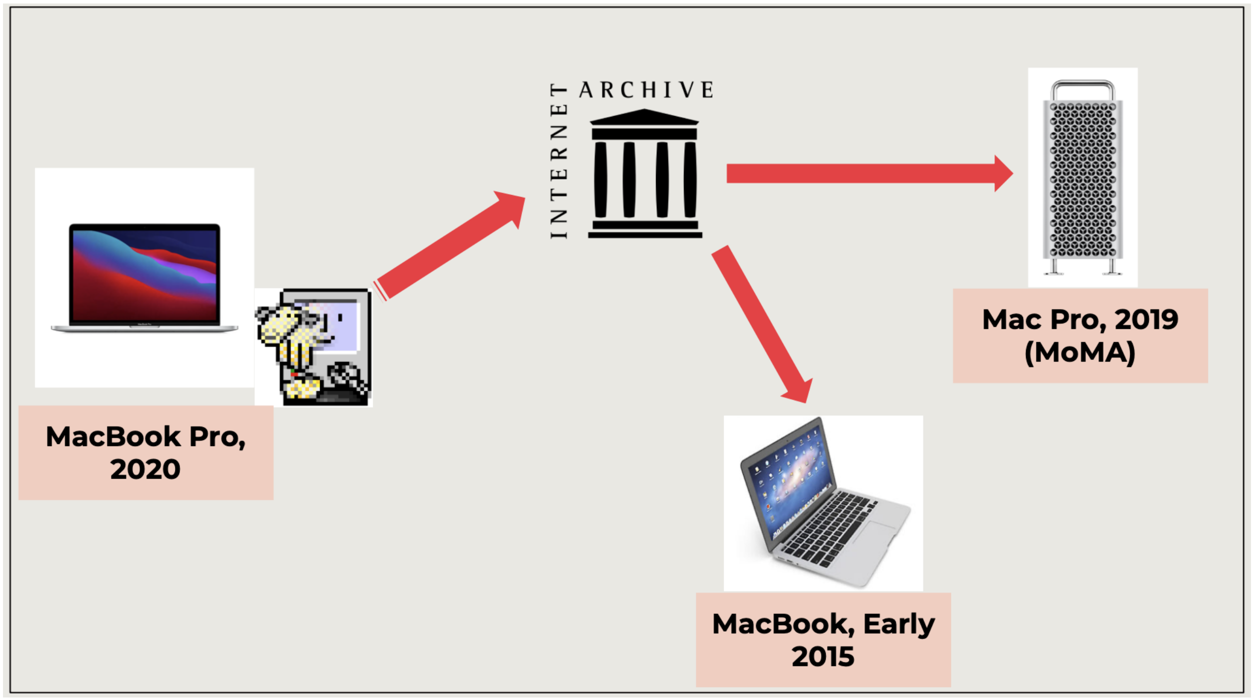


Figure 9. .sheepvm transfer diagram. Diagram created by author.

Name	**MacBook Pro, 2020**	**Mac Pro, 2019 (MoMA)**	**MacBook, Early 2015**	**iMac ca. 2002.**
Role	Used to configure all VMs	Used to test portable VMs	Used to test portable VMs	Host for “Mac Classic Environment” (Emulator C)
OS Version:	macOS Monterey Version 12.1	macOS 13.1 (22C65)	macOS Big Sur Version 11.7.4	Mac OS X 10.3.9
Processor	2 GHz Quad-Core Intel Core i5	3.5 GHz 8-Core Intel Xeon W	1.1 GHz Dual-Core Intel Core M	PowerPC (750) G3
Memory:	32 GB 3733 MHz LPDDR4X	32 GB 2666 MHz DDR4	8 GB 1600 MHz DDR3	384 MB 600 MHz
Graphics	Intel Iris Plus Graphics 1536 MB	AMD Radeon Pro 580X 8 GB	Intel HD Graphics 5300 1536 MB	Quicktime 6.4

Table 8. Specs for each host machine.

In both of these new host environments the VM appeared to perform identically as it did on my original host environment without needing any additional configuration. The portable .sheepvm package's performance was identical to the performance of Emulator A on the original host machine; the speed, sound, graphics, and interactivity all conformed to the original SheepShaver VM. The settings in the VM were transferred just as they were configured on the original host, including the vital JIT Compiler fix.

One notable difference between the .sheepvm on the two new hosts was the accessibility of its settings. Surprisingly, the settings were able to be accessed in the 2015 macbook, but not on the 2019 Mac Pro. This is an area of this project that warrants further research. For a detailed guide on how this type of portable VM can be configured, see **appendix A** of this document.

Chapter 6. Conclusion

Based on the results of the emulation tests, in terms of both authenticity and accessibility, SheepShaver (Emulator A) appears to be the optimal choice for Mac OS 9 emulation for the purposes of accessing classic Macintosh computer games. Both QEMU and SheepShaver (after adjusting its settings) accurately recreated the graphics and speed of *Myst.* However, when it came to sound, only SheepShaver and The Classic Environment supported sound output "out of the box." As discussed in the previous chapter, QEMU requires the Screamer fork of its development in order to support sound specifically for Mac OS 9 environments. This is less ideal from a longevity standpoint, because while it is true that QEMU is supported by a large online community of developers (and will likely continue to be in the future), it is uncertain whether this peculiar fork of the emulator will enjoy the same support. Future research should be conducted into the vulnerabilities of QEMU's Screamer fork from an obsolescence perspective.

Furthermore, QEMU—by virtue of being configured through the command line—has a far steeper learning curve than SheepShaver. While neither emulator has a simple configuration process, SheepShaver at least has a GUI for configuring its settings, while QEMU must be managed through a command file. On the other hand, QEMU is overall a more versatile and flexible program when compared to the relatively limited options available within SheepShaver.

Although the Classic Environment (Emulator C) was interesting to include for research purposes, due to the fact that it is only accessible through a different obsolete proprietary operating system (Mac OS X 10.4), it cannot be recommended. Additionally, as discussed in section 5.2 of this paper, the Classic Environment had more noticeable errors in both the speed and graphics categories than either of the two contemporary, open-source emulator options.

Regarding the EaaSI program of work, there is no doubt that it is an incredibly promising project with a great deal of potential to lower the barrier for entry for using virtual machines for the purposes of preserving and accessing legacy software. However, at least as of May 2023, the technology simply is not up to supporting the playback of even classic computer and software-based artwork. Since the network latency is too high for even a 30 year old computer game, the performance would likely be just as poor with more graphically demanding games.

Moving beyond the direct results of these experiments, there are some wider implications to consider about the use of VMs for the exhibition and preservation of computer/software-based artwork and computer games. Virtual machines are powerful, versatile, and complex tools. Based on the results of these experiments and on the work of projects such as EaaSI, I am confident that VMs will become essential for the preservation and exhibition of legacy software. Memory Institutions and GLAMs that have the infrastructure and resources to implement VM-based preservation strategies should do so as transparently as possible.

However, there are at least two major issues that stand in the way of the wider adoption of VMs in our field; copyright law and emulator obsolescence. It was out of the scope of this paper to address the complicated legal issues that underlie emulation in general. However, thanks to the efforts of groups such as the Software Preservation Network[78] and the DMCA Video Game Brain Trust, there have been strategic developments reframing the conversation around the issue of fair use. At the risk of oversimplifying the issue, in the GLAM context, emulation is preservation and therefore should fall under the umbrella of fair use. When corporations lose interest in supporting their own creations, conservators and archivists (many of whom are hobbyists by nature) step in to ensure that future generations will be able to access, research, and enjoy legacy software.

The second greatest impediment to the wider adoption of VMs is the issue of emulator obsolescence. Just like any other software, emulators are susceptible to obsolescence. For example, during MoMA's 2015 exhibition *This Is for Everyone*, the open-source, multi-purpose emulation platform MAME version 0.166 was used to play ROMs—digital copies of software that have been extracted from physical carriers, such cartridges, optical disks, or hard drives—of several classic arcade games, including *Space Invaders* (1978), and *Pac-Man* (1980) and *Tempest* (1981). Version 0.166 of MAME was originally developed for Windows XP and Vista in 2015, and therefore is no longer compatible with current operating systems. Although MoMA's conservators subsequently created raw disk images of each exhibition computers' hard drive, today these disk images of previous exhibition computers and emulators are useful only as documentation of past preservation and exhibition decisions. As software and hardware evolve, conservators essentially have to start from scratch, reconfiguring emulators and remapping controllers, to prepare games for each exhibition. This problem is confounded and compounded

[78] https://www.softwarepreservationnetwork.org

by the fact that most emulation and VM software is developed by hobbyists as passion projects, meaning that there is no guarantee that a particular emulator will be supported into the future.

It is promising that the portable SheepShaver VM configured for this project was compatible with a 2015 MacBook and a 2019 MacPro, but it remains to be seen how the VM will perform on future operating systems, two, five, and ten years down the line. It is equally promising that both SheepShaver and QEMU enjoy the support of sizable unofficial communities of like-minded hobbyists, one of the largest and most active being E-Maculation. Future research projects should investigate each of these VM software's susceptibility to obsolescence, possibly by testing older versions of the VM on contemporary operating systems. While the VMs created for this project should serve MoMA well the next time it decides to access *Myst* for exhibition or research purposes, in another age, it remains to be seen how these emulators will stand up to the march of time.

Appendix A. SheepShaver and Self-contained VM Configuration Guide (macOS)

This guide is specifically for configuring self-contained virtual machines with SheepShaver on OSX and macOS. These VMs are packages containing all necessary files, and they are portable in the sense that they can be transferred as single file packages to any compatible OSX/macOS system. Multiple VMs can be created, each with its own settings, and they can coexist on the same host. If the guide is followed properly, the new host should not need a copy of the SheepShaver app installed locally.

The following guide is based on two existing tutorials. The first is “Setting up SheepShaver for OSX/macOS” from the E-Maculation wiki, and the second is Redundant Robot’s SheepShaver tutorial.[79] The goal of this guide is to consolidate and simplify the information from these sources in order to make them more accessible for beginners.

Two important notes before you start:

1. **You will need a Mac OS install CD**. SheepShaver can only emulate Mac OS 7.5.3 through Mac OS 9.0.4. SheepShaver cannot run OS 9.1 or later. You will need a compatible Mac OS install CD, or a disk image created from a compatible Mac OS install CD, to be able to install a Mac OS in SheepShaver. The version of the OS installed must correspond to the Mac ROM file you are using.
2. **You will need a compatible Mac ROM file.** SheepShaver will not run without a compatible ROM file. If SheepShaver does not find a compatible ROM file, it will immediately quit on launch. According to the E-Maculation tutorial, “ROM files that will work with SheepShaver for OSX/macOS are either an ‘old world’ rom image grabbed

[79]E-Maculation Wiki: https://www.emaculation.com/doku.php/sheepshaver_mac_os_x_setup
Redundant Robot: https://www.redundantrobot.com/sheepshaver. All images used in this guide are sourced from the Emaculation Wiki.

from an actual hardware PowerMacintosh ROM, or the ‘new world’ rom file extracted from the ‘Mac OS ROM Update 1.0’ using TomeViewer.”[80]

- **Old world rom file**: can run System 7.5.3 through Mac OS 9.0.4.
- **New world rom file**: can run Mac OS 8.5 through Mac OS 9.0.4.

Note that, due to the implementation of the Memory Management Unit, the ROM file from a 9.0.4 OS CD will not work with any version of SheepShaver. For more information about ROM files, SheepShaver, and download links, visit Redundant Robot’s site: https://www.redundantrobot.com/sheepshaver.

Part 1. Setting Up SheepShaver and the Portable VM

1. **Downloaded the most recent build of SheepShaver and the “sheep shaver folder”**

   Current and past SheepShaver builds are available for download from the Emaculation forums: https://www.emaculation.com/forum/viewtopic.php?t=7360&sid=9a50ba8b977800b78b26b8486f189705 [81]

2. **Create the .sheepvm package**

   In the Finder:

   - Create a folder with the desired name for the virtual machine you want to set up.
   - Add a compatible Mac ROM file named “Mac OS ROM” (no extension) to the folder.

[80] https://www.emaculation.com/doku.php/sheepshaver_mac_os_x_setup
[81] At the time of writing , the most recent version of SheepShaver is Version 2.5 (22 December 2022).

- Add an empty file named “prefs” (no extension) to the folder.[82]
- Add a copy of the SheepShaver.app to the folder.[83]
- Optional: Add a keycodes file. This file is not needed (but does no harm either) if only a US-English keyboard will be used.
- Add a .sheepvm extension to the folder name

If the Mac ROM is compatible with SheepShaver, as soon as the .sheepvm extension is added to the folder, the folder will change into a package with a SheepShaver icon.

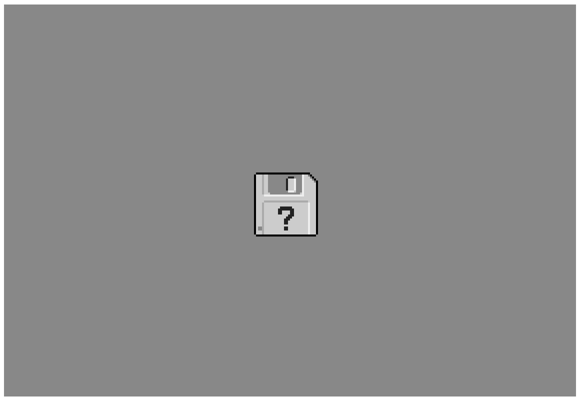
3. **Create the “Shared” folder inside the .sheepvm package**

   Add a folder named “Shared” to the .sheepvm package. This folder will appear on the emulated desktop within SheepShaver as a volume named “Unix”. Files copied or saved in SheepShaver to the “Unix” disk will appear in the shared folder on the OSX/macOS side. Likewise, files placed in the shared folder on the OSX/macOS side will appear in the “Unix” disk in SheepShaver.

4. **Double-click the VM icon.**

   Provided that a compatible ROM file with the name “Mac OS ROM” or “ROM” is present in the folder, SheepShaver will launch and show in its window a gray floppy disk icon with a blinking question mark “?”, indicating that the emulated Mac has not found a startup volume.

[82]It should be noted that you cannot use any plain text editor, such as Text Edit (Mac’s built-in text editor software). Text Edit automatically saves plain text files with an rtf header. This adds additional lines of code to the text file that interferes with the emulation software.
[83]Including the Sheepshaver .app in the .sheepvm package is not required, but will make it possible to run the VM on the new host without having to install SheepShaver again.



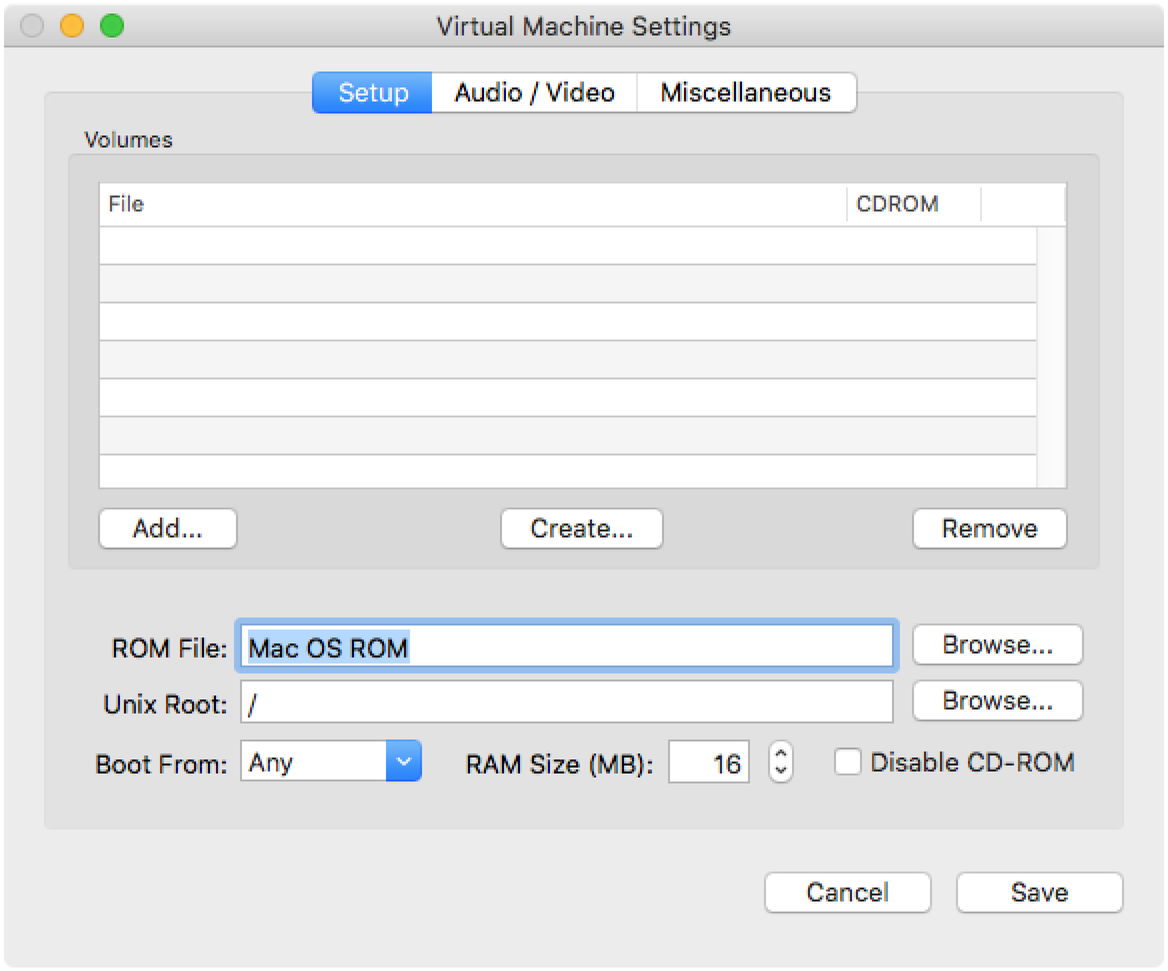
Note: At this stage, you can only turn off SheepShaver through a hard shut down (pressing Control + Escape).

Part 2. Configuring Sheepshaver's Preferences/Settings

1. **Open Preferences (Settings in macOS 13 Ventura and later) from the SheepShaver application menu**
   - Choose Preferences or Settings from the SheepShaver menu to open the Virtual Machine Settings window.
2. **Configure settings in Setup tab**

   <u>**ROM File**</u>

   - Add the ROM file using the "**Browse…**" button to navigate to and "Open" the ROM file.



ROM File: Mac OS ROM

Unix Root: /

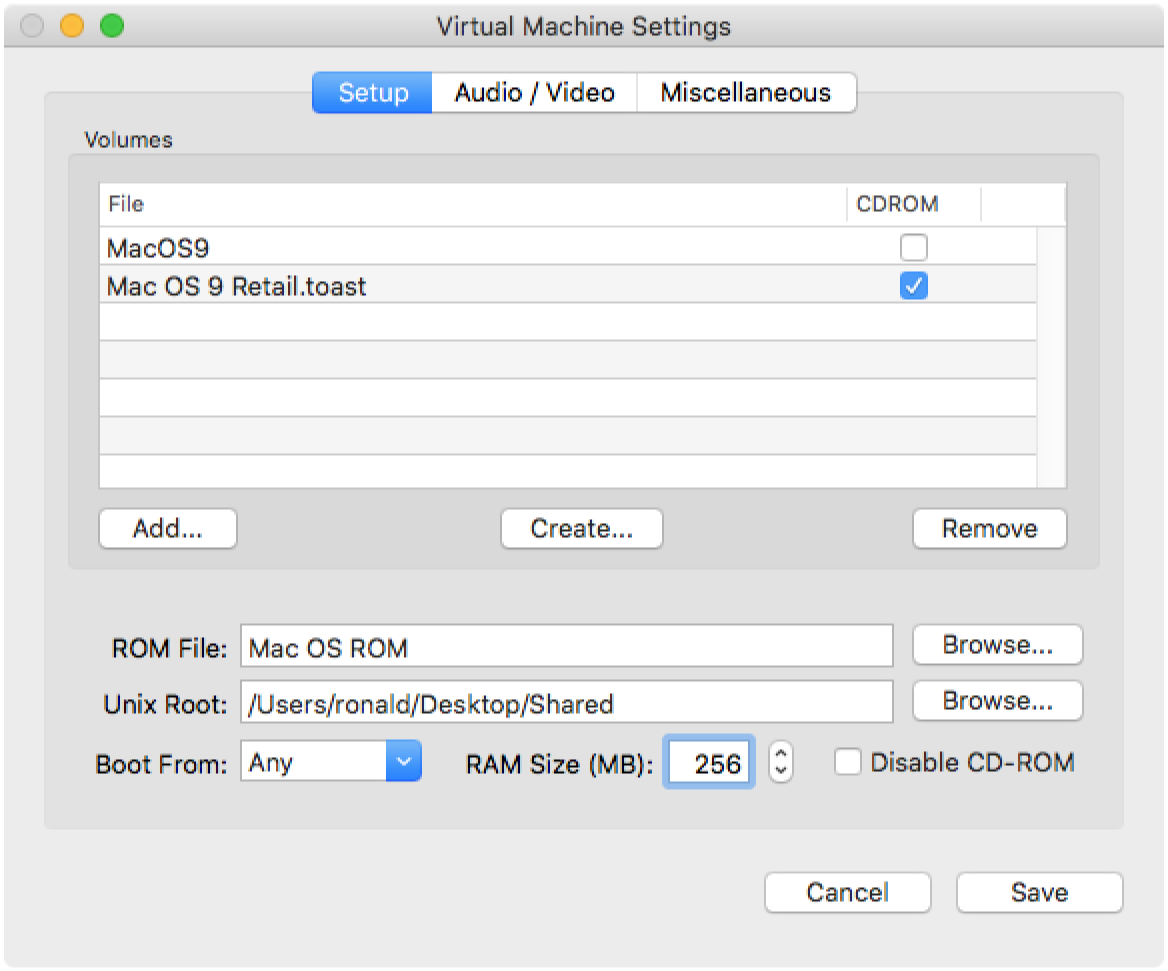
Boot From: Any

RAM Size (MB): 16

Disable CD-ROM ☐

Volumes

- You can now create the disk image file that will serve as the virtual hard disk on the emulated Mac. First, click the "**Create…**" button.
- Next, choose an appropriate name for the image file, such as "MacOS9". Then select the Volume Size (I recommend between 500 and 2000 MB for Mac OS 9 systems). Then, save the file inside the .sheepvm package.
- Then, to add a Mac OS install CD image file, use the "**Add…**" button to navigate to and "Open" the CD image file. The image file will appear in the Volumes list below the volume you created. Make sure to check the "**CDROM**" box for the CD image file to make it behave inside SheepShaver as an actual CD-ROM.



Volumes

File	CDROM
MacOS9	☐
Mac OS 9 Retail.toast	✓

ROM File: Mac OS ROM

Unix Root: /Users/ronald/Desktop/Shared

Boot From: Any

RAM Size (MB): 256

☐ Disable CD-ROM

Unix Root

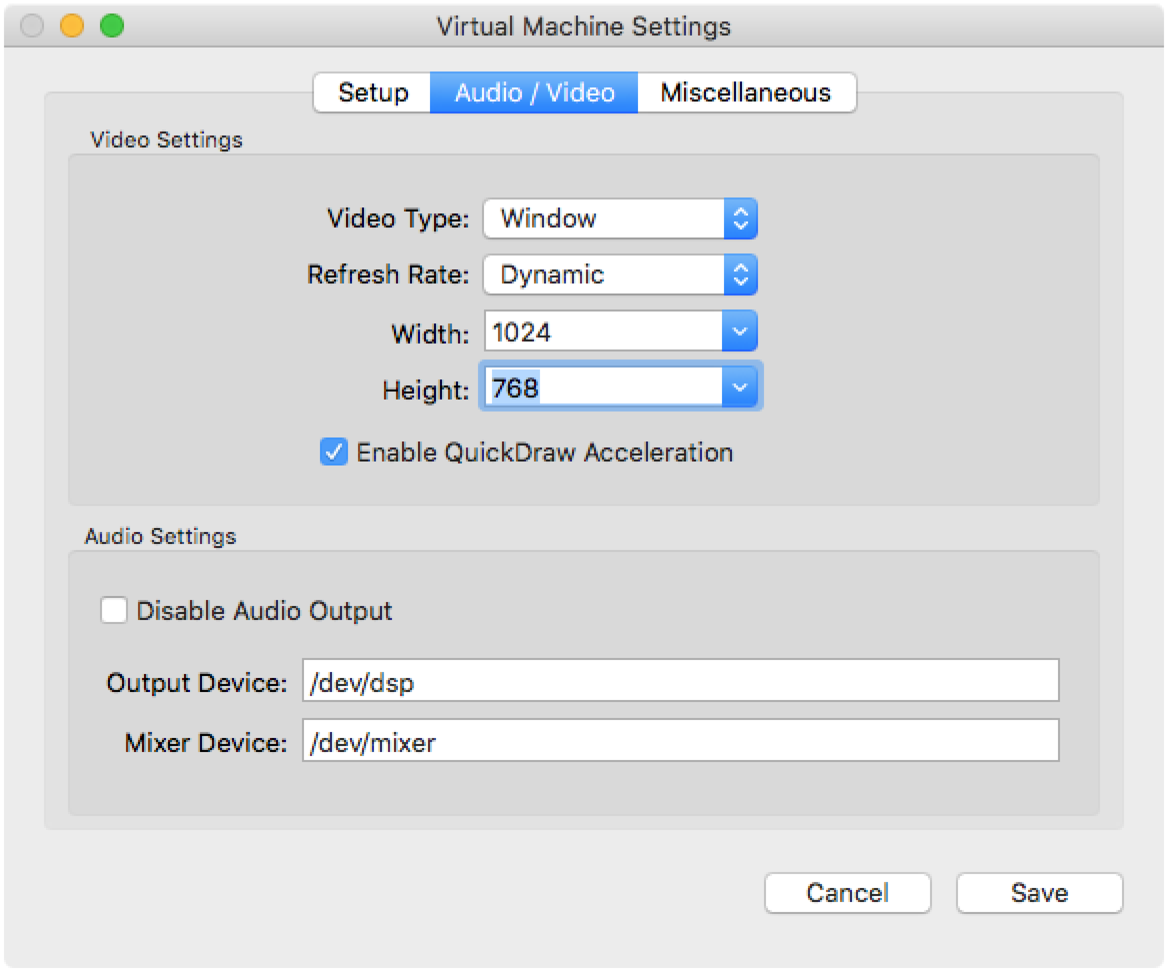
- Add the path to your Shared folder using the second "**Browse…**" button to navigate to and "Open" the folder you created to serve as the shared folder.

Note: Do not keep the default value ”/“, as that will make your entire hard disk the shared folder.

RAM Size (MB)

- Enter a value for the amount of RAM that the virtual Mac will have, such as 128, 256 or 512.
- Leave **Boot From** set to "Any".
- Leave **Disable CD-ROM** unchecked.

3. Configure settings in Audio / Video tab



Video Settings

Video Type: Window

Refresh Rate: Dynamic

Width: 1024

Height: 768

[✓] Enable QuickDraw Acceleration

Audio Settings

[ ] Disable Audio Output

Output Device: /dev/dsp

Mixer Device: /dev/mixer

<u>Video Type</u>

- Set Video Type to “Window.” Using “Fullscreen” at initial setup is not recommended.

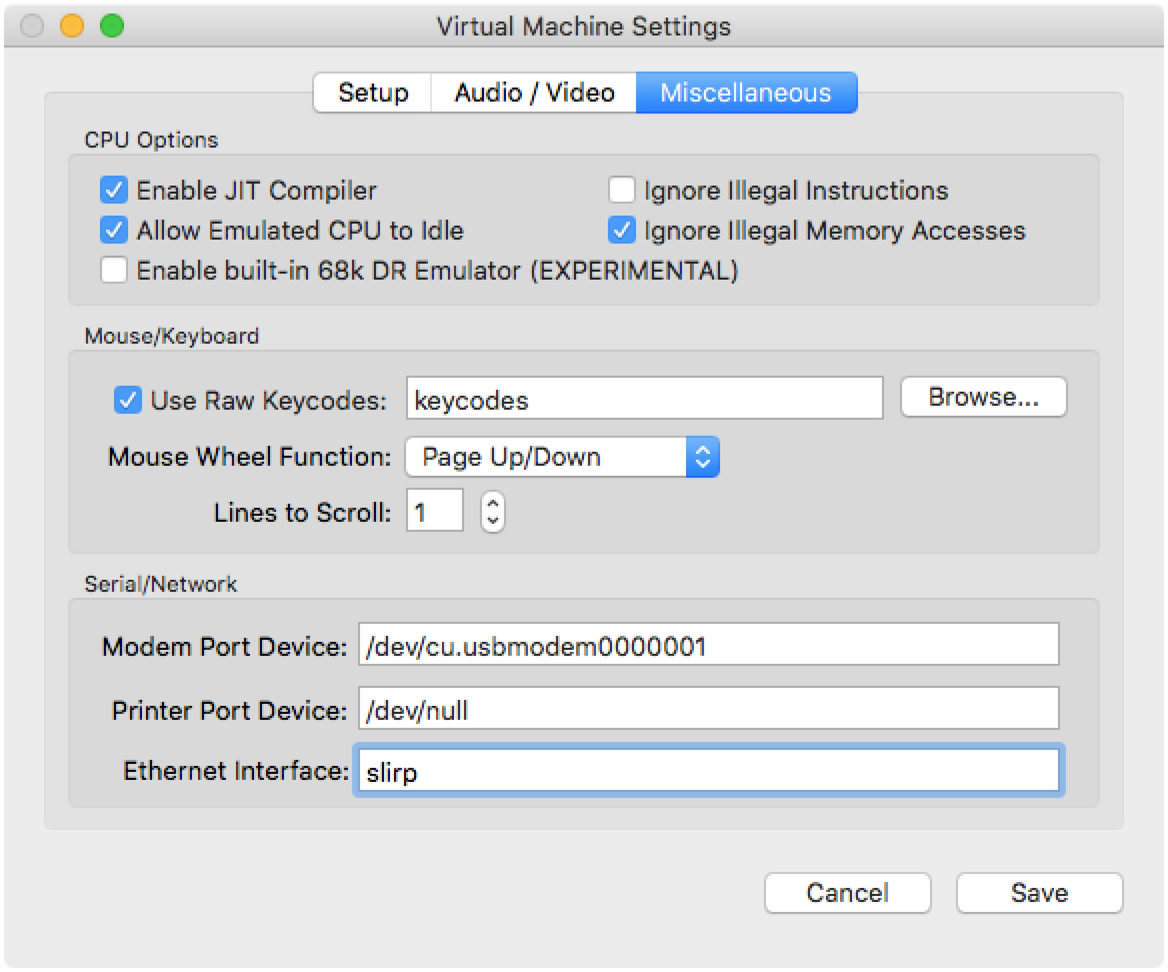
<u>Refresh Rate</u>

- The Emaculation editors suggest that “Dynamic” should always be selected on “fast Intel machines…Lower settings will make the emulated machine appear sluggish.”

Width and Height

- Set Width and Height for the window to lower values than the width and height of your host screen. If you have a large screen, you may enter higher values than the presets in the menu, such as 1280x800.
- Check **Enable QuickDraw Acceleration**.
- **Audio Settings** can be ignored at this time.

4. Configure settings in Miscellaneous tab



CPU Options

- [✓] Enable JIT Compiler
- [ ] Ignore Illegal Instructions
- [✓] Allow Emulated CPU to Idle
- [✓] Ignore Illegal Memory Accesses
- [ ] Enable built-in 68k DR Emulator (EXPERIMENTAL)

Mouse/Keyboard

- [✓] Use Raw Keycodes: keycodes
- Mouse Wheel Function: Page Up/Down
- Lines to Scroll: 1

Serial/Network

- Modem Port Device: /dev/cu.usbmodem0000001
- Printer Port Device: /dev/null
- Ethernet Interface: slirp

CPU Options

- Check "Enable JIT Compiler", "Allow Emulated CPU to Idle", and "Ignore Illegal Memory Accesses".

- **Note:** Disabling the JIT Compiler is optional, but is necessary for running certain games (such as *Myst*) at the authentic speed.

Mouse/Keyboard

- If your keyboard lay-out is not US-English, check "Use Raw Keycodes" and browse for the keycodes file in the SheepShaver folder.
- "Mouse Wheel Function" and "Lines to Scroll" are a matter of personal preference.

Serial/Network

- You can ignore "Modem Port Device" and "Printer Port Device".
- For "Ethernet Interface" enter "slirp."

5. **Save your settings**
    - Click "Save and Quit" to save the settings and to close the Settings window.
    - **Note:** SheepShaver must be quit and shut down to save changes made to its Settings.
6. **Quit SheepShaver**
    - Use Control+Escape to quit SheepShaver.

Part 3. Booting SheepShaver and installing the system

1. **Launch SheepShaver and initialize the volume**
    - During startup, the system will encounter the newly created volume and will offer to initialize (erase) it. Give the volume a name and choose **Mac OS Extended** for the format. Click to proceed with initializing the volume.

- You should see the desktop with icons for the startup volume (the CD image or the CD), for the new volume that you will install the system onto, and for the "Unix" volume that gives access to the shared folder on the OSX/macOS side.

2. **Install the OS**
    - Proceed with the system installation by double-clicking the installer "**Mac OS Install.**" When the installation is completed, quit out of the installer.
3. **Remove OS Install Disk and SheepShaver**
    - Open SheepShaver Preferences/Settings and remove the CD disk image file from the volumes list by selecting it and clicking the **Remove** button.
4. **Save the changes and shut down the VM**
    - Save the changes and shut down the emulated Mac to quit SheepShaver.

Part 4. Notes on Running Mac OS in SheepShaver

- When you launch SheepShaver, the emulated machine will boot from the installed system. The Configuration Assistant will guide you through the configuration of the new system.
- **In SheepShaver the Configuration Assistant will lock-up while configuring network settings.** It is recommended that the user quit the Assistant before it arrives at the network settings. The remaining configuration may be done manually in the various control panels.
- **WARNING**: Restarting the VM may cause SheepShaver to freeze or crash. Instead, shut down and start again.
- The Startup Disk control panel is not functional in SheepShaver

- SheepShaver will startup from the first bootable volume in the Volumes list, or from a bootable CD if no bootable volume is in the list. When “Boot From” is set to CD-ROM in Preferences/Settings, the emulator will always boot from a bootable CD. **Trying to use the Startup Disk control panel may cause SheepShaver to crash.**
- **Enabling sound in SheepShaver**

In the “**Sound**” control panel, select “**Built-in**” for the output device. Click “Built-in” to highlight the option. In Mac OS 9 the Sound control panel is installed with the other control panels and can be accessed from the Apple menu. If the “Apple Audio Extension” is installed in the Extensions folder, remove it.

- Note that after any change in settings SheepShaver needs to be quit and launched again in order for the changes to take effect.

Appendix B. QEMU Screamer Configuration Guide (macOS)

This guide is written only for macOS hosts. The user must already have installed Homebrew, a package manager for installing open-source software on macOS.[84] This guide also assumes the user has some familiarity with the command line/terminal.[85]

<u>Note</u>:

- Text written in `courier new` indicates terminal commands
- Brackets "`[...]`" around sections of terminal commands indicate where the user needs to enter paths or file names from their own machine.

1. **Clone the repository.**

```
git clone --recursive -b screamer
https://github.com/mcayland/qemu.git qemu-screamer
```

This step may take a few minutes. Once its finished, navigate to the new folder containing Screamer:

```
cd qemu-screamer/
```

2. **Install the dependencies via homebrew:**

```
brew install libffi gettext glib pkg-config autoconf
automake pixman ninja meson gnutls jpeg libpng
libslirp libssh libusb lzo ncurses nettle snappy vde
```

3. **Configure the build:**

[84] https://brew.sh

[85] This guide is adapted from Greg Gant's tutorial for QEMU Screamer. See Grant "Emulating Mac OS 9.2 with sound on Apple Silicon and Intel" http://blog.greggant.com/posts/2021/12/18/ppc-qemu-mac-os-9-with-sound-on-apple-silicon-intel-mac.html

```
PKG_CONFIG_PATH="$(brew
--prefix)/opt/ncurses/lib/pkgconfig" ./configure
--target-list="ppc-softmmu"
```

4. **Navigate to the build folder:**

```
cd build/
```

4a. Build the app:

```
make -j$(sysctl -n hw.ncpu)
```

5. **Create the virtual machine:**

```
qemu-img create -f qcow2 [myos].img 2G
```

Replace `[myos]` in the above command with the desired name of your virtual machine.

This command will create a virtual machine disk image with a maximum size of 2 gigabytes.

From here you can now run your disk images using this copy of QEMU screamer from the build folder (or wherever you placed it).

NOTE: You will need a compatible Mac OS 9 install disk for your emulator. I recommend the Mac OS 9.2.2 Universal Install Image from Mac OS 9 Lives: http://macos9lives.com/smforum/index.php?topic=2109.0. This image has been tested and confirmed to work with this version of QEMU Screamer.

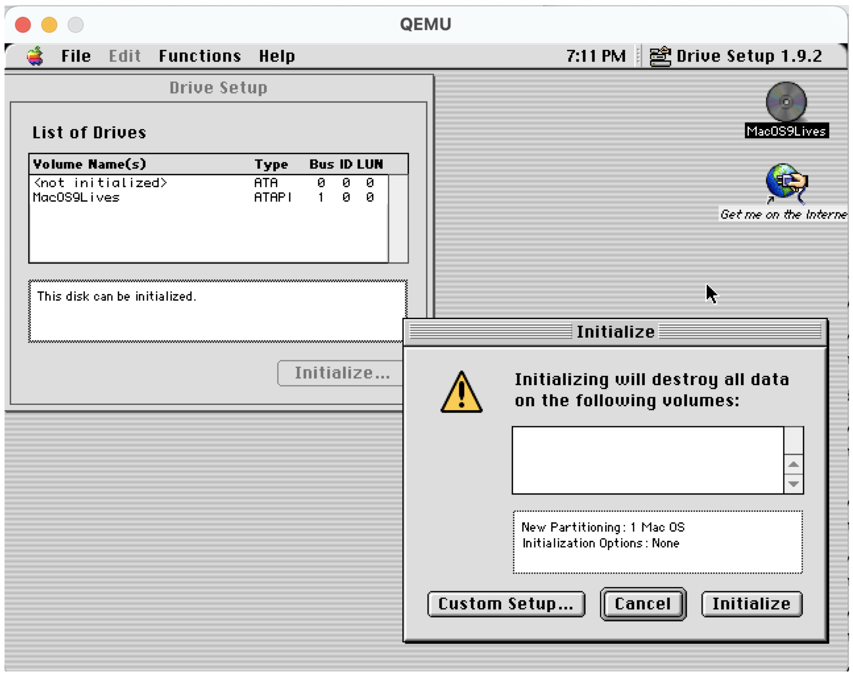
6. **Install the OS on the virtual machine:**

```
./qemu-system-ppc -L pc-bios -boot d -M mac99 -m 512
-hda [myos].img -cdrom [path to install.iso]
```

Note that you will need to change `[myos]` to the name of your own virtual machine and `[path to install.iso]` to the path of the install disk on your local machine.

6a. Installing the OS on your virtual machine (if using the above OS 9.2.2 install disk from the Mac OS 9 Lives forums):

Partition and Format the Hard Drive using the Mac OS **Drive Setup** Utility. Drive Setup. should start automatically, but if it does not, this utility is located in the applications folder of the installation CD.



List of Drives

Volume Name(s)	Type	Bus	ID	LUN
<not initialized>	ATA	0	0	0
MacOS9Lives	ATAPI	1	0	0

This disk can be initialized.

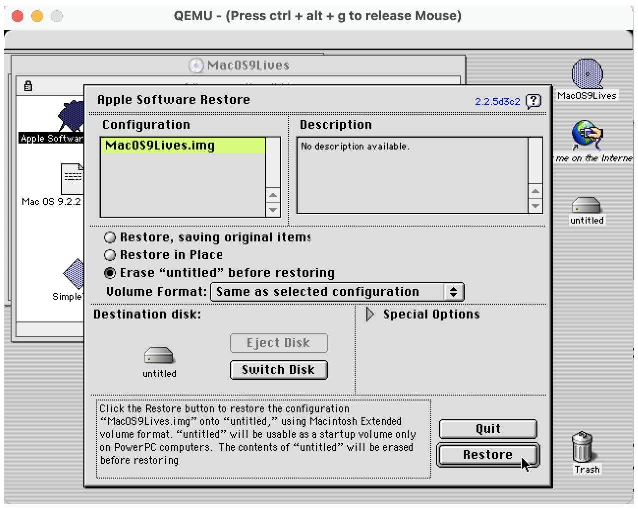
Initializing will destroy all data on the following volumes:

New Partitioning: 1 Mac OS
Initialization Options: None

Drive Setup will **Initialize** (erase) the virtual hard drive. The new hard drive will then appear on the desktop.

6b. Launch the **Apple Software Restore** application and make sure the "Destination disk" is the newly formatted Hard Drive or a Volume you wish to restore to.

6c. Click the "**Restore**" button and answer any prompts:



Configuration: MacOS9Lives.img

Description: No description available.

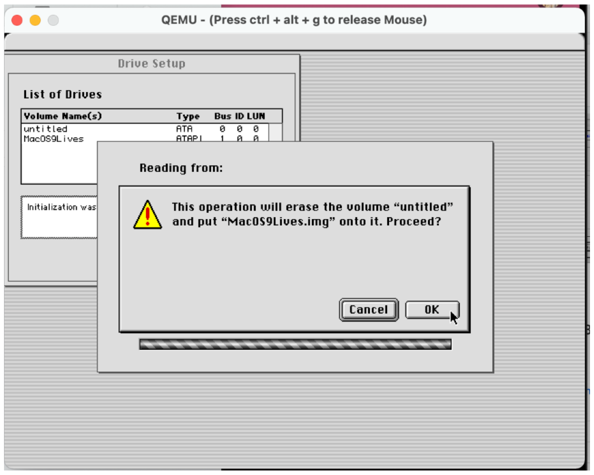
Restore, saving original items
Restore in Place
[✓ Erase “untitled” before restoring]

Volume Format: Same as selected configuration

Destination disk: untitled

Special Options

Click the Restore button to restore the configuration “MacOS9Lives.img” onto “untitled,” using Macintosh Extended volume format. “untitled” will be usable as a startup volume only on PowerPC computers. The contents of “untitled” will be erased before restoring



Volume Name(s)	Type	Bus	ID	LUN
untitled	ATA	0	0	0
MacOS9Lives	ATAPI	1	0	0

Initialization was

Reading from:

This operation will erase the volume “untitled” and put “MacOS9Lives.img” onto it. Proceed?

6d. Shut down the virtual machine.

7. **Boot the virtual machine with Mac OS installed:**

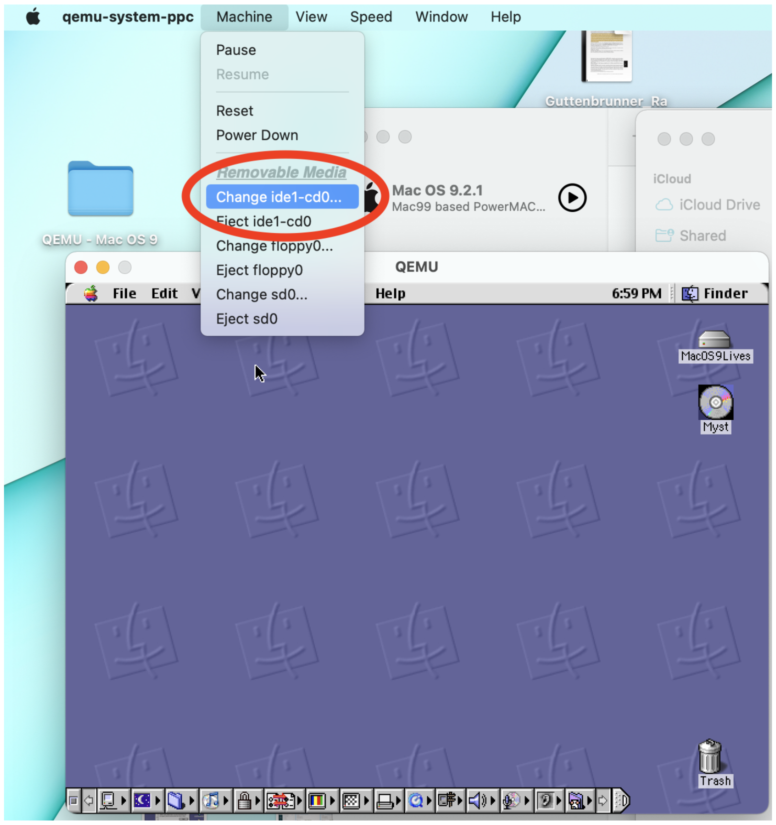
```
./qemu-system-ppc -L pc-bios -boot c -M mac99 -m 512
-hda [path to myos.img]
```

Use this command each time you want to boot your configured QEMU virtual machine.

8. **Loading other apps/software onto your virtual QEMU Mac:**

8a. Option 1 (QEMU GUI):

While the virtual machine is running, select **Machine** from the task bar and under the **Removable Media** list, select **Change ide1-cd0…**



Note: If you use this option, you will have to repeat this process each time you boot your QEMU mac.

8b. Option 2 (Terminal):

```
./qemu-system-ppc -L pc-bios -boot c -M mac99 -m 512
-hda myos.img -cdrom [path to disk image]
```

This command should launch the virtual machine with your chosen application on the desktop (as though a CD-ROM was inserted into the disk drive).

Appendix C. Links to Experiment Recordings

Google Drive Folder:

https://drive.google.com/drive/folders/1YcaGNqldA10qGx2IXKsAfdAnyoLuXxu4?usp=sharing

Control Experiments

- Control A. PowerMac G4:

  https://drive.google.com/drive/folders/1Tl3eecHcDg-_k__A0HPVygZgNKTQ-Zc3?usp=
  share_link

- Control B. iMac G3:

  https://drive.google.com/drive/folders/1VGnci3i4LTCLw7S5mg416jdwHutWT_Nx?usp
  =share_link

Emulation Experiments

- Emulator A. SheepShaver:

  https://drive.google.com/drive/folders/15w4JuEZSQk5t49UPrMp5BVlKugKPozrn?usp=
  share_link

- Portable A. Sheepvm:

  https://drive.google.com/drive/folders/1iwdisqzyh5GOPC4oqTxh26Npip6hcbaP?usp=sh
  are_link

- Emulator B. QEMU:

  https://drive.google.com/drive/folders/1I1J8ozenRmoNKXqGD5mQTgjxvDzthger?usp=
  share_link

- Remote A. EaaSI:

  https://drive.google.com/drive/folders/1zzFHij7CzVazRl2zhNeLVkay9u6NokYf?usp=sh
  are_link

Works Consulted

Anderson, David, Janet Delve, and Dan Pinchbeck, "Toward a workable emulation-based preservation strategy: rationale and technical metadata." *New Review of Information Networking*, Vol. 15 No. 2 (2010): 110-131. https://doi.org/10.1080/13614576.2010.530132

Antonelli, Paola. "Video Games: 14 in the Collection, for Starters." *Inside/Out: A MoMA/MoMA PS1 Blog.* November 29, 2012. https://www.moma.org/magazine/articles/798.

Bauer, Christian. "SheepShaver." Accessed February 12, 2023, http://sheepshaver.cebix.net

Bettivia, Rhiannon. "Where Does Significance Lie: Locating the Significant Properties of Video Games in Preserving Virtual Worlds II Data." *International Journal of Digital Curation* 11, no. 1 (2016): 17-32. DOI: 10.2218/ijdc.v11i1.339

BSS8888. "I can't believe how different the original 1993 Mac release of Myst is from every other version." *Reddit,* March 15, 2022. https://www.reddit.com/r/myst/comments/tf5573/i_cant_believe_how_different_the_original_1993/

Carta, Giovanni. "Metadata and Video Games Emulation: an effective bond to achieve authentic preservation?" *Records Management Journal* 27, no. 2 (2017): 192–204. DOI 10.1108/RMJ-10-2016-0037

Cochrane, Euan, Klaus Rechert, Seth Anderson, Jessica Meyerson, and Ethan Gates. "TOWARDS A UNIVERSAL VIRTUAL INTERACTOR (UVI) FOR DIGITAL OBJECTS." Paper presented at the 16th International Conference on Digital Preservation iPRES 2019, Amsterdam, The Netherlands, 2019. DOI: 10.1145/nnnnnnn.nnnnnnn

Cochrane, Euan, Klaus Rechert, Jurek Oberhauser, Seth Anderson, Claire Fox, and Ethan Gates. "USEABLE SOFTWARE FOREVER: The Emulation as a Service Infrastructure (EaaSI) Program of Work." Paper presented at the 18th International Conference on Digital Preservation iPres 2022, Glasgow, Scotland, 2022.

Dioscuri team. "Dioscuri: Ideas and Key Features." Koninklijke Bibliotheek, 2007. Accessed on November 17th, 2022. http://dioscuri.sourceforge.net/dioscuri.html.

DivingKataeGuru, Why emulation of PowerPC Macs was underwhelming," *Reddit,* September 4, 2018,
https://web.archive.org/web/20211114062934/https://www.reddit.com/r/emulation/comments/9d2sl9/why_emulation_of_powerpc_macs_was_underwhelming/

Galloway, Paul. "Video Games: Seven More Building Blocks in MoMA's Collection."

Inside/Out: A MoMA/MoMA PS1 Blog. June 28, 2013.
https://www.moma.org/explore/inside_out/2013/06/28/video-games-seven-more-building-blocks-in-momas-collection/.

Gaynor, Jim. "System 7.0 - Will it be on apple.com?" February 8, 1991.
https://web.archive.org/web/20160401212219/http://macgui.com/usenet/?group=53&id=83125

Gillis, Alexander S., Sacey Peterson, and Sonia Lelii. "What is Virtual Memory?" TechTarget.com. Accessed June 16, 2023.
https://www.techtarget.com/searchstorage/definition/virtual-memory

Guttenbrunner, Mark. "Digital Preservation of Console Video Games." Vienna University of Technology. 2007. http://www.ifs.tuwien.ac.at/~becker/pubs/guttenbrunner_games2007

Guttenbrunner, Mark and Andreas Rauber. "A measurement framework for evaluating emulators for digital preservation." ACM Transactions on Information Systems, Vol. 30, No. 2, Article 14 (May 2012). http://doi.acm.org/10.1145/2180868.2180876

Hedstrom, Margaret, and Christopher A. Lee. "Significant Properties of Digital Objects: Definitions, Applications, Implications." Paper presented at the Proceedings of the DLM-Forum, Barcelona 2002: 218–23.

IBM Research. "UVC: A Universal Virtual Computer for Long-term Preservation of Digital Information." Technical Paper Archive. Accessed May 8, 2023. https://dominoweb.draco.res.ibm.com/10229b6de0d054c585256fa900681995.html

Klappenbach, Michael. "How to Optimize and Improve Graphics Performance and Frame Rates." Lifewire. Accessed March 25th, 2023. https://www.lifewire.com/optimizing-video-game-frame-rates-811784

Laurenson, Pip. "Authenticity, Change and Loss in the Conservation of Time-Based Media Installations", in *Tate Papers no.6* (Fall 2006). https://www.tate.org.uk/research/tate-papers/06/authenticity-change-and-loss-conservation-of-time-based-media-installations.

MacHelp. "Mac OS 9.2.2: Document and Software." Apple Computer, Inc. Updated March 17, 2005. https://web.archive.org/web/20060421074349/http://docs.info.apple.com/article.html?artnum=75186.

—. "What's New in Mac OS 9," Apple Computer, Inc. Updated June 28, 2004, https://web.archive.org/web/20180130185804/https://www.apple.com/newsroom/2005/06/06Apple-to-Use-Intel-Microprocessors-Beginning-in-2006/.

The Museum of Modern Art. "Talk to Me: Design and the Communication between People and Objects." Accessed August 18, 2021. https://www.moma.org/calendar/exhibitions/1071.

—. "A Collection of Ideas." Accessed August 18, 2021. https://www.moma.org/calendar/exhibitions/1421.

—. "Never Alone: Video Games and Other Interactive Design." Accessed December 17, 2022. https://www.moma.org/calendar/exhibitions/5453.

NeoGamer - The Video Game Archive. "Making of - Myst (1993)." Youtube Video, March 18, 2020. https://www.youtube.com/watch?v=c5af74KuJZE.

Pinchbeck, Dan, David Anderson, Janet Delve, Antonio, Ciuffreda, Getaneh Otemu and Andreas Lange. "Emulation as a strategy for the preservation of games: the KEEP project." Paper presented at the DiGRA 2009 Breaking New Ground: Innovation in Games, Play, Practice and Theory, Brunuel University, West London, September 2009. http://www.digra.org/digital-library/publications/emulation-as-a-strategy-for-the-preservation-of-games-the-keep-project/

PLANETS. "PLANETS Software." Planets-project.edu. Accessed October 21, 2022. https://planets-project.eu/software/.

"QEMU version 8.0.0 released." April 20, 2023. QEMU. https://www.qemu.org/2023/04/20/qemu-8-0-0/.

Rosenthal, David, S. H. "Emulation & Virtualization as Preservation Strategies." LOCKSS Program, Stanford University Libraries, Andrew W. Mellon Foundation, 2015. https://mellon.org/media/filer_public/0c/3e/0c3eee7d-4166-4ba6-a767-6b42e6a1c2a7/rosenthal-emulation-2015.pdf

Rothenberg, Jeff. “An Experiment Using Emulation to Preserve Digital Documents,” The Koninklijke Bibliotheek, Den Haag. https://www.semanticscholar.org/paper/An-experiment-in-using-emulation-to-preserve-Rothenberg/f95e30933c29f8b1b3b8b51c62343a8b99dc5cac.

“Running Qemu-system-ppc with Mac OS/OSX guests in macOS.” Emaculation.com. Updated January 10, 2021. https://www.emaculation.com/doku.php/ppc-osx-on-qemu-for-osx#introduction

Seale, Susan “PowerPC G4 Architecture White Paper.” Freescale Semiconductor, Inc. 2001. https://www.nxp.com/docs/en/white-paper/G4WP.pdf

Strodl, Stephan, Christoph Becker, Robert Neumayer, and Andreas Rauber. “How to Choose a Digital Preservation Strategy: Evaluating a Preservation Planning Procedure.” Vienna, Austria: Vienna University of Technology. 2007. https://www.ifs.tuwien.ac.at/~becker/pubs/strodl_choose_JCDL07.pdf.

Stetz, Michael. “Leveling Up Emulation: The Benefits of Field-Programmable Gate Arrays in Video Game Preservation.” New York University, 2022

Tyson, Matthew. “What is the JVM? Introducing the Java virtual machine.” InfoWorld. October 28, 2022. https://www.infoworld.com/article/3272244/what-is-the-jvm-introducing-the-java-virtual-machine.html.

“Macintosh line,” Emulation General Wiki, accessed October 17th, 2022, https://emulation.gametechwiki.com/index.php/Macintosh_line#Emulators

McDonough, Jerome. “A Tangled Web: Metadata and Problems in Game Preservation.” *The Preservation of Complex Objects* 3, (2013): 49-62.

Wikipedia. “Mac OS 9.” Accessed March 16, 2023.

https://en.wikipedia.org/wiki/Mac_OS_9#Version_history.

—.“Frame rate.” Accessed March 17, 2023. https://en.wikipedia.org/wiki/Frame_rate.

—. “SeepShaver.” Accessed February 12th, 2023.

https://en.wikipedia.org/wiki/SheepShaver#cite_note-sheepshaver-1

—.“List of macOS built-in apps.” Accessed May 1, 2023,

https://en.wikipedia.org/wiki/List_of_macOS_built-in_apps#Classic