https://doi.org/10.2352/issn.2168-3204.2021.1.0.11

Emulation of Historical Software as a Tool for Research and Pedagogy: A Case Study in the History of CAD

Daniel Cardoso Llach; School of Architecture, Carnegie Mellon University; Pittsburgh PA

Eric Kaltman; Computer Science, California State University Channel Islands; Camarillo, CA

Abstract

This paper presents an abbreviated summary of previous work using a distributed emulation network (EaaSI) to allow for the analysis of computer assisted design (CAD) tools including multiple versions of the popular AutoCAD system. It elaborates on the use of EaaSI in a graduate seminar on the history of computational design, presenting a design pedagogy use case for archived software objects and showing how their remediation through emulation can lead to new historical and design insights into contemporary software. It includes further clarification on the relevance of emulation to the archival community and highlights extended use cases not found in the original publication.

Introduction and Motivation

Examining historical software systems is a difficult task for researchers and educators. Due to a variety of issues, like obsolescence and complex networks of dependencies, running legacy code is often a significant technical challenge before its study can become a historical one. Secondary archival sources including software manuals and recordings of software use can provide significant context, however, many aspects of historical systems are only interpretable during run-time interactions with users. As archives build collections of legacy software code, executables and other computational objects, questions arise regarding how to provide access to those records, and reciprocally, what patrons can do with that access. The published study at the heart of this abstract aligns with these issues of access and use [1]. By utilizing a process known as "emulation", the creation of modern software that is able to run legacy software, it is possible to run older software inside modern systems.

In this case study, we used the Emulation as a Service Infrastructure (EaaSI) distributed emulation network to share multiple historical versions of various computer-aided design (CAD) programs with students in a history of CAD seminar at the Carnegie Mellon University School of Architecture. Our study primarily focused on versions of AutoCAD, a popular CAD program used in architecture and engineering. The course explored three interrogative methods of software historical study: reconstruction, emulation, and speculation, with three course projects aligned with each method. As proposed here [2], *reconstruction* is a process through which a researcher uses archival materials to approximate a legacy software system using modern software development tools. The cited example involved recreating historical computer graphics systems as web-based JavaScript applications. *Emulation*, as described, involves running archived software inside another program that imitates the original archived software objects' technical context. *Speculation* involves gathering insights from reconstructed and emulated investigations and marshalling them in the service of new design ideas.

The work here elaborates on the emulation portion of the course and shows how insights gleaned from the historical emulations influenced students' speculative designs for new systems. To our knowledge, this is the first use of emulation in the context of design pedagogies. It is also an argument for further use of archived software as primary historical source materials. The following sections of this paper will briefly describe the emulation environment and its configuration; summarize the process and results of the published emulation study; and outline some implications of this work for both software design and archival use of software as a subject of historical analysis. Recent work by Acker has also highlighted the challenges and potential of emulation for library, archive, and museum workflows. Our pedagogical use case could therefore expand on Acker's designation of "emulation encounters" [3] and we will focus some of the conclusion on aligning this work with emerging considerations on the use of emulation in LAMs.

Emulation as a Service Infrastructure

The EaaSI project is a joint collaboration between the Mellon Foundation, the Sloan Foundation, and an international consortium of university libraries aimed at providing an infrastructure to share pre-configured computing environments to enable native access to archived software programs and other digital objects. The system allows for a library to configure a suite of emulations that, once cohered, can be easily replicated across different "nodes" in the network. For example, a librarian at one institution can set up a Microsoft Windows 98 environment and install a contemporaneous copy of Microsoft Word in order to read an old Word document for local use. If another librarian at a different institution also needs to read a Word document from the same time period (and with the same version dependency), they can simply copy the initial environment to their local node without needing to repeat the Windows 98 and Word installations. EaaSI provides access to these environments through a standard web-browser window so there is no local installation or configuration needed for an end user.

In our case, a number of different CAD software programs were configured in a combination of 11 different versions and operating systems by a CMU librarian for use in the course. Each installation required imaging the original software from library

owned copies or cloning the installations from others in the network. The cited study looked at the use of two specific systems: AutoCAD R12, released for Microsoft DOS in 1992, and AutoCAD 2000, released for Microsoft Windows 98 in 1999. Two other software programs, VistaPro 4.1, a landscape generation application, and Autodesk Maya 2010, a 3D modeling and animation program, were also examined in the course.

Case Study Design and Methodology

For the emulated component of the course, students were invited to choose one of the CAD systems provided within the EaaSI environment and to scrutinize it both as users and through documentary historical research. The purpose was to explore both the sociohistorical context of the software, and its visual and interactive components. Unlike the reconstruction module, in which students read static archival materials to create an interactive version of the system, the emulations allowed for a more direct engagement with the software as a primary research document. As this was a graduate course, many of the students were professional level users familiar with modern CAD systems. As we discuss in more detail in [1], this approach provided the class with an entry point into a more nuanced analysis of each system's design since the students could readily apply processes and literacies developed in the modern systems to their earlier counterparts. Two students in particular, "Josie" and "Jer", were both trained professional architects and considered themselves expert users of modern versions of AutoCAD. Therefore, they each chose a previous version of the same software for analysis. What followed was something akin to an interface "close reading" of each legacy AutoCAD system.

We asked students to reflect in class and to document their engagement with the emulations through written assignments and a final paper. Students had two weeks to complete the analysis assignment and spent around 10 hours experimenting with the emulations. Following that reflective practice, the students were asked to develop insights gained from the emulations into new design ideas for software system interactions.

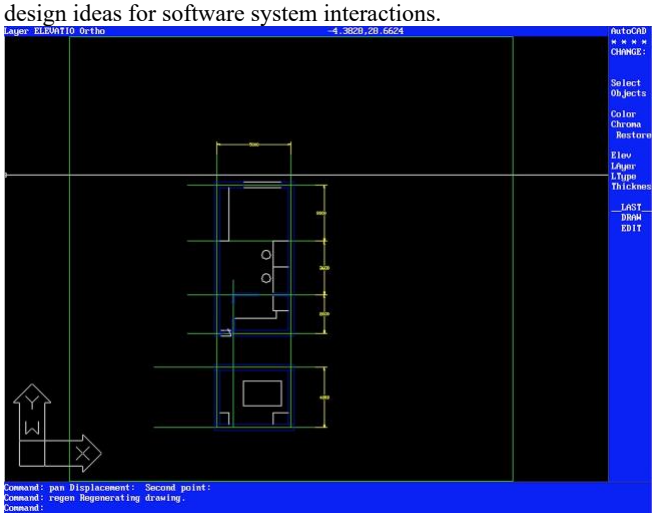

Figure 1: Simple floorplan in AutoCAD R12 running on EaaSI system

Close Readings and Speculations

The students in the course produced in-depth comparative analysis essays reflecting on their use of the emulated software systems. Each student engaged in a unique and personal interaction with their chosen software artifact. In the published study, Josie decided to see how his current understanding of AutoCAD's current shortcuts and key features mapped onto AutoCAD R12. He found that the six commands he considered to be "most essential" to AutoCAD, "line, erase, trim, copy, move and zoom," were present in the older version. This enabled him to sketch out the floorplan of a house (Figure 1). The major hurdle here was adapting to the reduced interface of the DOS system, which lacked multiple windows and toolbars that are commonly provided in most modern graphical user interfaces. Luckily, the basic commands and abstractions (specifically the ability to use different "layers" for different parts of the drawing) functioned similarly to their modern counterparts, including their specific keyboard shortcuts. This led Josie to reflect on the consistency of AutoCAD's interface and whether there were other options for issuing common commands to the system. As a result, they developed a prototypical gaze and gesture interface for AutoCAD that mapped the core functionalities found to be consistent across versions onto embodied actions of the user (Figure 2).

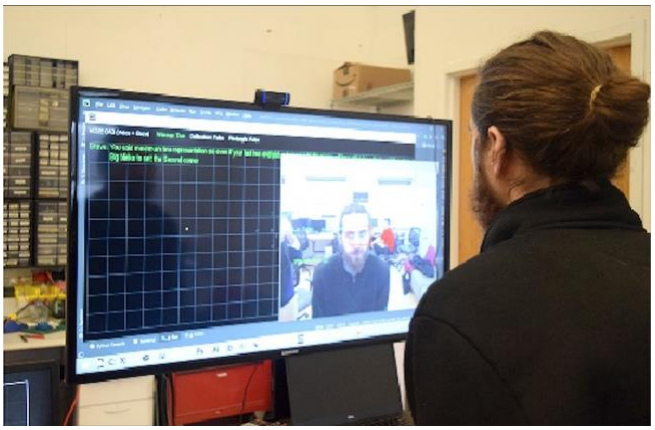
Figure 2: Speculative "gaze" interface

Jer, the other expert user, tackled the more recent AutoCAD 2000. Instead of creating a new drawing from scratch, he decided to look at contemporaneous example files provided with the software (Figure 3). AutoCAD 2000 was similar to the version of AutoCAD Jer had used when starting his career as an architect. Similar to Josie, Jer felt that since AutoCAD introduced a full graphical interface the following twenty years of development had not altered much of the previously established mouse and keyboard interactions. Additionally, AutoCAD 2000 included features essential to modern architectural practice, like the abilities to reference external files, so called "x-refs", and to organize a master drawing that stitched together multiple other CAD drawings. This latter feature was important for larger projects, where teams might be responsible for only a part of the whole project. Due to the consistency of the interface, Jer's speculation reflected on how the routinized and unchanged mouse and keyboard interface of AutoCAD shaped architectural drafting practices, and whether newer input paradigms might provide new avenues for design. He developed a custom

tangible interface that mapped six common commands, “polyline”, “curve”, “trim”, “copy”, “mirror”, and “offset”, onto a series of large, movable blocks (Figure 4). This provided a new set of possibilities for gesture interface design.

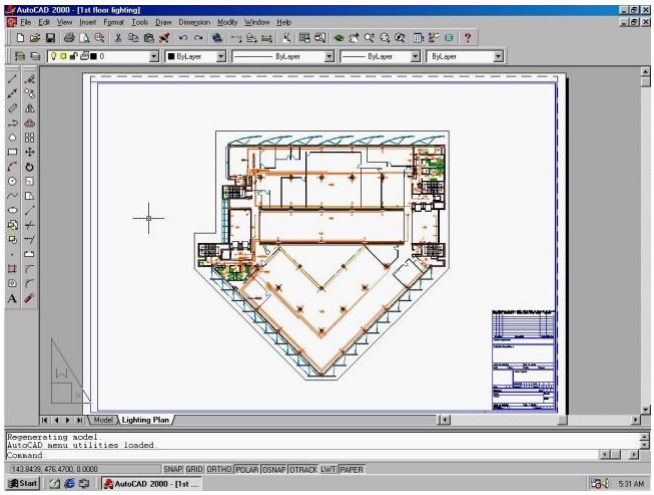

Figure 3: Sample file rendered in AutoCAD 2000 running on EaaSI system

The other two students in the course conducted similar reflective work, though they did not have previous experience with their chosen CAD programs. One student’s analysis of the Vistapro landscape generation software led to many experiments in constructing rule based, procedurally articulated landscapes. Vistapro was used in the 1990s through early 2000s to render potential real (national parks) and imagined (science fiction book covers) landscapes with a simple configuration interface (Figure 5). The rule-based structure of the program caused the student to compare Vistapro’s approach to more recent machine learning methods, like generative adversarial networks (GAN), which has also been used to generate artificial landscape renderings.

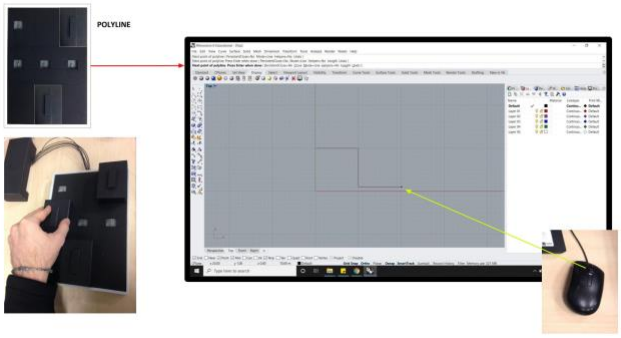

Figure 4: Speculative tangible interface (figure taken from [1])

The final student explored Autodesk’s Maya 2010 and spent time modeling various 3D characters. Her reflections commented on the lack of substantial change to the Maya interface in the last decade and the unavailability of tutorial documentation for a software program even a decade old. This highlights the advantage that the more experienced students had in working with legacy software, their previous tacit experience and literacies allowed them to fill in gaps that would normally require explicit documentation.

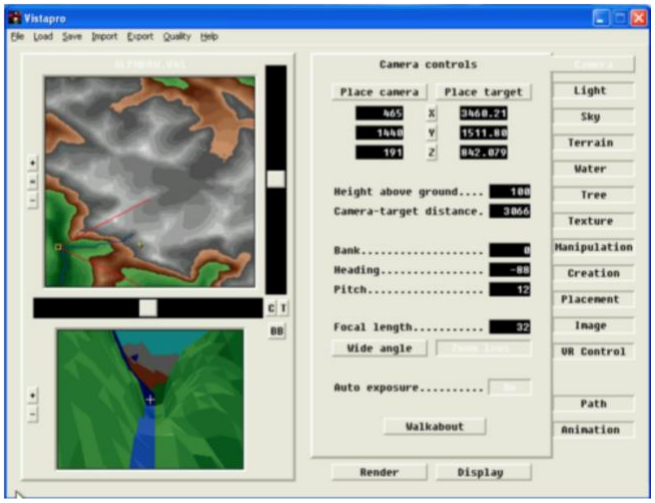

Figure 5: Vistapro 4.1 example landscape running on EaaSI system

Results

The study revealed many salient observations about historical CAD design that would likely not have been possible without access to the emulated systems. Two particular patterns emerged through the professional comparative work of Jer and Josie. First, there appeared to be a core functional literacy underlying the evolution of AutoCAD’s design. Both users were able to pick up and use the antiquated versions quite quickly and thoroughly. Second, both users found that the earlier versions made certain tasks simpler due to the lower level of complexity present in the interfaces and feature sets. The resulting analyses, combining historical context with insights gained via direct interaction with the software, was particularly enriching, and led students to draw hypotheses about how the software’s design responded to architects’ changing professional culture and about how, in a converse manner, changes in the software — for example, the introduction of “x-refs” — fostered changes in the social organization and subdivision of architectural work.

Conclusion and Next Steps

As summarized above, this work highlights the potential for the use of emulation of archived software as a resource for both pedagogy and design activities. Crucially, the direct engagement with the software afforded by the emulations allowed students to think comparatively and to draw contrasts and connections between the emulated historical systems and their contemporary counterparts. This line of inquiry became one of the important areas of our analysis. The students in the course engaged with the interfaces of each CAD system not simply through documentary analysis, but through direct interaction. This revealed visual and operational aspects of the systems that would not be possible to intuit without emulation as an access strategy. We noted in the publication [1] that, “software operations manifest not only as logical possibilities, but also as phenomenal experiences. Differences in how to save a document, or activate a particular

command, illustrate how the use of an interface can be a deeply personal affair, inter-twined with a user's disciplinary inclinations and personal preferences … Enriched by emulations, historical studies of software-based practices can examine not just the logical structures of the source code, but also the sensory and affective experiences of users." While emulation is not an exact replication of a system's historical context, it does provide a means to explore historical system interactions. As shown, even the browser-based remediations of the AutoCAD systems revealed new insights for design and provided students with a means of reflecting on their own software practices and literacies.

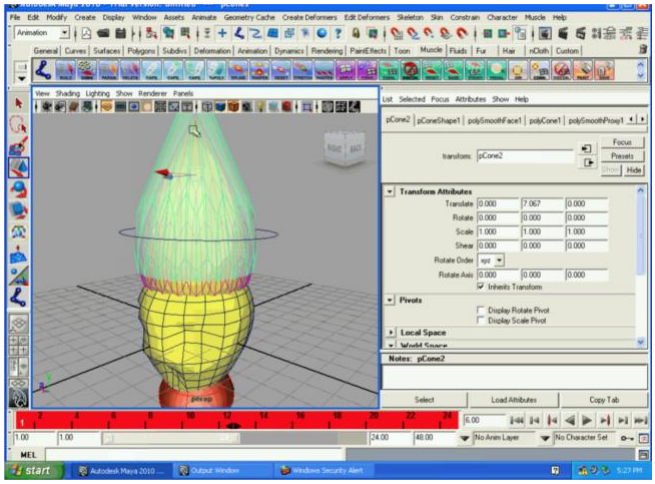

Figure 6: Student character modeling in Autodesk Maya 2010 running on EaaSI system

This study further functions as a proof-of-concept for the benefit of an accessible emulation solution like EaaSI. Students in the course did not have to spend time figuring out how to get the historical software running and could just dive right into course directed analysis assignments. The students work falls into a category of "emulation encounters" noted by Acker [3] in her study of other EaaSI emulation use cases. This work falls into the category of "emulation for archival access" with the students acting as archival users of emulation. Further, Acker notes that "user demand for accessing obsolete software from collections is unproven…[and] the challenges of representing emulation as an access point and information service to users remain." We feel this study legitimates the use of emulation as an archivally supported pedagogical aid, and therefore adds a new category of archival use cases for emulation-based methodologies. It provides an example of a specific user demand for emulation, and how the emulation service was successfully integrated into an experimental pedagogical activity.

There is a common need to justify the storage of software objects in archives, and to provide means of access that can make those objects speak from and for their pasts. Using emulation provides access to a "space of possible actions" that might not be perceptible without allowing a user to literally engage with a running system [4]. It also legitimates the efforts to save the executable components of historical software systems, as their use and analysis can certainly provide new historical insights. Lastly, as noted in the case study above, certain interaction literacies are at play when engaging with a software system. These literacies are based in the design decisions made in the construction of software systems and enacted by the system's users. Emulation allows for reflection on how software is used in practice, what commitments its design enforces on users, and how those design constraints have evolved over time, both in shaping the disciplines making use of the software and in constraining the possibilities for user expression. We are planning on adapting the work of the above study to many other use cases involving different types of software use. What other literacies are at work in other types of software systems, and how have they changed over time? We can only pose (and answer) these questions with access to the historical systems and the ability to run them.

References

[1] Cardoso et al. "An Archive of Interfaces: Exploring the Potential of Emulation for Software Research, Pedagogy, and Design," CSCW '21 (Forthcoming.)

[2] Cardoso Llach, Daniel, and Scott Donaldson. 2019. "An Experimental Archaeology of CAD: Using Software Reconstruction to Explore the Past and Future of Computer-Aided Design." In *Computer-Aided Architectural Design. "Hello, Culture,"* edited by Ji-Hyun Lee, 105–19. Communications in Computer and Information Science. Springer.

[3] Acker, Amelia. 2021. "Emulation Practices for Software Preservation in Libraries, Archives, and Museums." *J Assoc Inf Sci Technol.* 1-13 https://doi.org/10.1002/asi.24482

[4] Janlert, Lars-Erik and Erik Stolterman. 2017. "The Meaning of Interactivity: Some Proposals for Definitions and Measures." *Human Computer Interaction* 32, 103–138. https://doi.org/10.1080/07370024.2016.1226139

Author Biography

Dr. Daniel Cardoso Llach is an Associate Professor in the School of Architecture at Carnegie Mellon University, where he chairs the Master of Science in Computational Design program and co-directs CodeLab. He is the author of the book Builders of the Vision: Software and the Imagination of Design (Routledge 2015), a sociotechnical study of the history and contemporary practice of computer-aided design.

Dr. Eric Kaltman is a former CLIR Fellow for Data Curation in the Sciences at Carnegie Mellon University Libraries. Currently, he is an Assistant Professor of Computer Science at California State University Channel Islands where he is founding a Software History Futures and Technologies (SHFT) research group.